Ulrich Scholten

Dynamic Network Notation

A Graphical Modeling Language to Support the Visualization and Management
of Network Effects in Service Platforms

Dynamic Network Notation

A Graphical Modeling Language to Support the Visualization and Management of Network Effects in Service Platforms

Zur Erlangung des akademischen Grades eines

Doktors der Wirtschaftswissenschaften (Dr. rer. pol.)

von der Fakultät für Wirtschaftswissenschaften des

Karlsruher Instituts für Technologie (KIT)

genehmigte

DISSERTATION

von

Dipl.-Ing. (FH) Ulrich Scholten, MBA

Tag der mündlichen Prüfung: 14.06.2013
Referent: Prof. Dr. Stefan Tai
Korreferent: Prof. Dr. Orestis Terzidis
Karlsruhe, 2013

Impressum

Karlsruher Institut für Technologie (KIT)
KIT Scientific Publishing
Straße am Forum 2
D-76131 Karlsruhe
www.ksp.kit.edu

KIT – Universität des Landes Baden-Württemberg und nationales
Forschungszentrum in der Helmholtz-Gemeinschaft

KIT Scientific Publishing 2013
Print on Demand

ISSN: XXXX-XXXX
ISBN: 978-3-86644-XXX-X

Abstract

Service platforms have moved into the center of interest in both academic research and the IT industry due to their economic and technical impact. These multitenant platforms provide own or third party software as metered, on-demand services. Corresponding service offers exhibit network effects. Network effects describe the interdependency of service or platform value, service consumption and third party service provisioning. The present work introduces a graphical modeling language to support service platform design with focus on the exploitation of network effects. Several notations exist allowing graphical modeling of collaborative networks around service platforms from a platform operators' perspective. All these solutions model distinct value flows. The representation or control of network effects, including more implicit relationships of the platform operator to groups of ecosystem participants, however, does not find consideration.

The artifacts developed in this thesis build on a conceptual model of these effects including ways to manipulate them. This conceptual model has been derived through surveys, enhanced by fundamental work from system theory as well as theory of control, applied to service platforms. The central artifact in this thesis is the Dynamic Network Notation, a graphical modeling language. This language is able to represent platforms as well as platform ecosystems together with their competitive environment and, most importantly, immanent network effects. It further allocates categories of control mechanisms to manipulate these effects. In addition, the thesis suggests a pattern language to formulate building blocks of structured experience, to be used within the models. Lastly, the present work instantiates the Dynamic Network Notation within an editor.

Through several use case studies, the present work evaluates and confirms the expressiveness of the Dynamic Network Notation and its ability to produce models which represent network effects in service platforms. The original research contributions of this work and its resulting artifacts comprise (a) the representation of the platform operator's staged areas of authority with respect to ecosystem participants, (b) the representation of and differentiation between distinct flows of value exchange and exercise of influence in and around the platform, (c) the allocation and conceptualized interplay of enforcing- and incentivizing mechanisms to manage ecosystem participants and (d) the focus on the engineering and exploitation of network effects.

Acknowledgement

I would like to offer my sincere thanks to my PhD supervisor Professor Dr. Stefan Tai, for the guidance, support, motivation and for setting consistently high goals. His knowledge, patience and strength of purpose guided me through my research and the writing of this thesis. Also, I would like to express my great appreciation to the second reviewer of my thesis Professor Dr. Orestis Terzidis for the many valuable contributions and suggestions made during the preparation of this work. I would like to thank the thesis committee, Professor Dr. Ute Werner, Professor Dr. Rudi Studer, along with Professors Tai and Terzidis for the professional, warm and yet challenging reception of my thesis defense. Sincere thanks also to Professor Dr. Lutz Heuser, who – in his former position as Vice President of SAP Research of SAP Research – initiated the Strategic Value Net (SVN) Project which set the frame for my dissertation project.

I would like to thank my fellow researchers and co-publishers, in particular my wife Dr. Simone Scholten, who cooperated with me in her position as Senior Researcher at SAP Research. Also deepest appreciation to Robin Fischer, my colleague and PhD candidate in the SVN team, Dr. Nelly Schuster, Project Leader of the MoSaiC project at KIT, Dr. Christian Janiesch, Senior Research Associate at the AIFB Institute of KIT, Dr. Christian Ziripins, Senior Scientist at Seeburger AG as well as Dr. Norman May, Project Leader of the Theseus/Texo project at SAP. Dr. Zirpins and Dr. Janiesch deserve additional tribute for their professional advice in the preparation of my doctoral thesis. Particular thanks also go to my Graduate Assistant Matthias Reimchen and to Dr. Andreas Harth, Senior Research Associate at AIFB for sharing his expertise with me on conducting academic research. I would like to express specific appreciation to the administrative support in the Institute for Applied Informatics and Formal Description Methods AIFB, in particular Mrs. Rita Schmidt.

I would further like to express particular thanks to Hazel Skeet at VentureSkies and Eric Moskwa at M-Engineering for continuous feedback and inspiration throughout the whole research project. Particular thanks are also due to the companies SAP AG, CAS Software AG, S.Chand Group and M-Engineering who volunteered for testing and evaluation of the Dynamic Network Notation and provided helpful feedback and ideas.

The support, inspiration and friendship offered to me by collegues and friends in the research group of Economics and Technology of eOrganizations in Berlin and Karlsruhe and my gratitude to them must also be recorded.

Sincere gratitude and love to my wife Simone and my sons Yann and Yves.

Karlsruhe,
April 2013

Ulrich Scholten
Karlsruhe Institute of Technology (KIT)

This thesis is dedicated to my parents Inge and Heinz Scholten

Table of Contents

A Foundations

1 Motivation

Service platforms have moved into the center of interest in both academic research and the IT industry due to novelty and economic impact. Multitenant platforms now provide own or third party software as metered on demand services (SaaS). The consumer benefits from increased agility and flexibility in using configurable applications from various client devices that run over a cloud infrastructure using virtualized network, server and storage capabilities [1-3]. Driving factors behind the rapid growth of these SaaS-offerings are the immanent rapid deployment times, reduced upfront implementation and minimized long-term overheads as compared with traditional on-premise software [4]. Active participation of consumers and external suppliers in the value creation process additionally accelerates the concept's success [5]. Salesforce.com for example, created a Customer Relationship Management product on demand as a first step. It then added AppExchange as an online marketplace for customer applications and then extended the open platform concept with Force.com, a development and deployment environment for third party service providers. Similarly, companies like NetSuite opened their infrastructure to host outside applications as well as their own services.

These service platforms exhibit network effects. Network effects are self-enforcing effects of value generation, generated by causal loops of reciprocal interdependency between platform attractiveness and third party value provisioning. As an example, a platform's attractiveness to third party service providers to provide services depends on the quantity of consumers subscribed to the platform. From the perspective of the service providers, the subscribed consumers represent a value. Consumers on the other hand are only attracted, if the platform deploys a suitable amount of third party services. Their value is the quantity of services, provided by the service providers. However, once opening the platform to third party service provisioning, the platform operator has to accept a certain degree of self-organization of the providers while giving away a certain degree of control over service quality [6].

Attaining critical mass is another relevant factor. It describes the fact of a limiting threshold, e.g., a minimal quantity of services or subscribed consumers required to incite a network effect. Once a network effect is unlocked, it is the interplay of network participants, which further evolves the value of the platform. In such a situation, the platform operator's tasks focuses on service management in pursuit of beneficially sustaining and growing the network effect [7]. However, network effects do not respond to simple linear relationships. They are characterized by complex differential interdependencies of the various participants and activities in and around the service platform [8].

To be able to incite and control network effects, the platform operator needs to be able to manage the flows of service provisioning and consumption. This requires a platform design that allows for control without squelching self-organization. Self-organization is a necessary prerequisite to network effects [9]. It also needs deployment and consumption environments that are – where network effects can occur – able to respond to increased platform activity with suitable scalability. Such a design is challenging due to the broad range of parameters, which need to be considered and to be successfully realized. A market study has shown that up to now, only a small set of platform operators, offering selective cross-industry-applications have successfully mastered this concept [10].

One way of improving this situation is to guide platform solution managers and platform architects in conceiving a suitable architecture through an appropriate modeling language. Such a language can support by providing

(a) a common base of representation to exchange and discuss ideas, thoughts, opinions or objectives around an intended solution [11, 12];
(b) abstraction from the complete picture, retaining only the relevant parameters [13], i.e. on platform conception, focused on harnessing network effects through appropriate service management and hence improving the overall cognition process [14];
(c) active tool-based user guidance to improve efficiency in modeling [15];
(d) a pattern language representing reusable building blocks of structured experience [16, 17];
(e) computer-based model analysis and suggestion schemes for design improvement [18].

As of today, several modeling languages exist, which consider service platforms. None of them gives exhaustive representation of those networks. In particular, none of them is able to abstract platform design in a way that guides platform operators towards service manageability through the modeling of processes, structures and mechanisms targeting the exploitation of network effects in platforms and their surrounding ecosystems. Such a language needs to comprise procedural elements, e.g., for the representation of the interplay between consumers, platform and service providers as well as for structural elements, e.g., for demarking areas of influence on services and involved players. The graphical modeling languages Service Network Notation (SNN), Service Modeling Notation (SNMN) and e3value provide the closest existing rapprochement to the requirement of a modeling language for network effects around service platforms. All three languages model and analyze concrete relationships with specified service partners. Network relations are understood in a very explicit way, focusing on aspects such as data or value flows, relationships and business protocols. However, in the dynamic context of Cloud computing, network effects originate in more indirect (i.e., implicit) patterns and relationships. They cannot be directly modeled through service choreographies or process orchestration. Aspects of infrastructure design or mechanisms of control, which foster self-organization of service platforms and their ecosystems lack consideration. Responding to these requirements to support effective platform design in view of enhanced service management for the harnessing of network effects is the predominant goal in this thesis.

The present work responds to this requirement with the introduction of a graphical modeling language supporting the design of service platforms. This language creates abstraction of planned or existing platforms, retaining parameters relevant to service management. It highlights where control or scalability is required and provides support in process design e.g., for the right placement and parameterization of network effects. It provides further guidance throughout the process of structural platform conception. It specifically focuses on the self-organization of suppliers and consumers around service platforms, while maintaining suitable quality of service.

1.1 Related Research

This chapter gives a short overview of related research. There are many different approaches to service networks around service platforms. Research by Eisenmann, Parker and Alstyne for instance has a specific service platform focus, considering aspects of control and network effects. However their research is oriented towards business models and service innovation with only little or no consideration of service management from a technical point of view [19-22].

More technically-oriented research groups propagate the computational modeling of networks ‘as is’. Those concepts extract models in a bottom-up approach through the access, retrieval and combination of globally distributed information. Examples are Open Semantic Service Relationship approach (OSSR) or the Open Semantic Service Networks approach (OSSN). The objective of OSSR is to model in a computer-understandable way various types of relationships within a network [23]. OSSN complements OSSR through nodes to model the network itself [24, 25].

None of the above described research pursues the graphical modeling of service networks in the context of service management. The closest in this specific research orientation are:

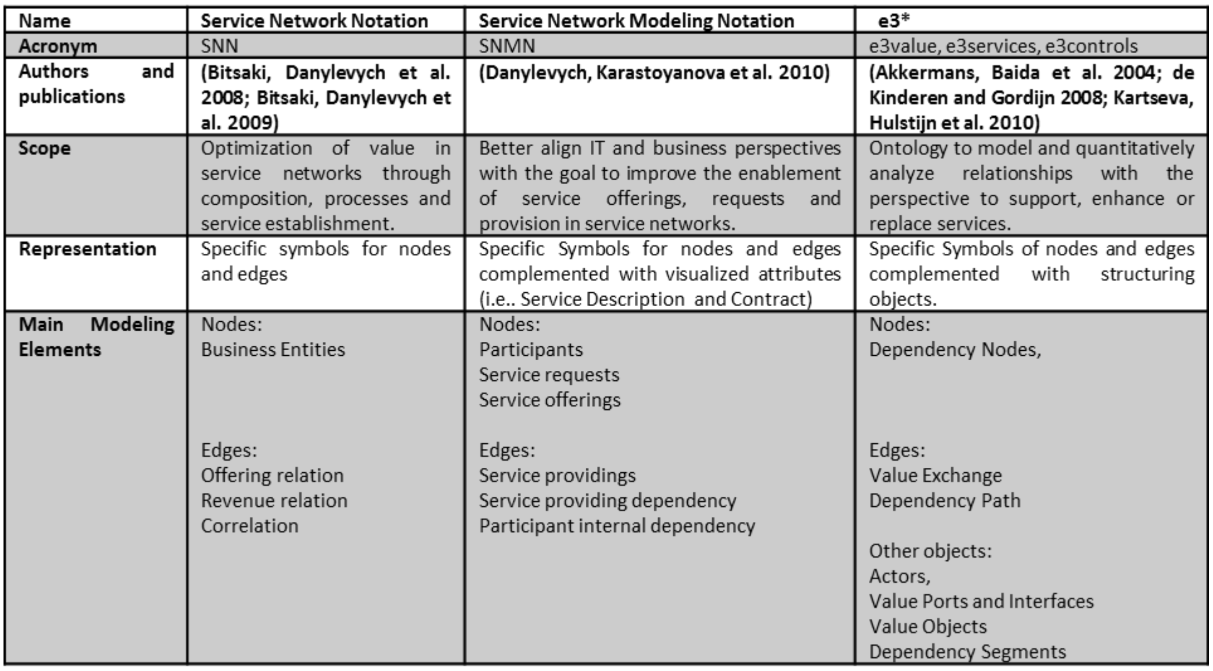
- the Service Network Notation with focus on the optimization of value in service networks [26, 27];
- the Service Network Modeling Notation (SNMN) concentrating on the enablement of service offerings, requests and provision (called *service providings*) between and inside organizations [28];
- e3* (e3value / e3services / e3controls) focusing on qualitative analysis of value flows in networks [29].

All three have in common that they model service networks as a set of nodes and edges in a *to be* approach. This top-down consideration has the objective to investigate (local) optimums through a comparison of *what-if* scenarios [28]. Second, their orientation is procedural rather than explanatory. Third, all models represent the biased viewpoint of a modeling company (*protagonist*). In other words, all models search for the protagonist's local optimum. They also have in common their goal to assist the enhancement of value generation in a network. All three models consider and focus on enhancing value chain relationships of economic value e.g., the economic value of a service or of financial transactions. The notations enable selective modeling restricted to the relationship aspects of economic value. In comparison to intended complete information aggregation of OSSN and OSSR, those are *weak relationships* [23]. The focal points of action for value enhancement in the three approaches are different.

The Service Network Notation (SNN) models abstract business relations within service networks through specific symbols, i.e. nodes representing business relationships. The relationships (edges) are abstracted to the limited view of offerings and revenues, complemented by correlations, which relate edges of the same business process. Service networks can be mapped to Business Process Models in a semi-automatic process. For that, the authors extended the BPMN stack by an additional top layer of Service Networks, related sets of Business Processes. SNN focuses the composition of new services through networks of existing services from a SOA perspective [23, 26, 27].

The Service Network Modeling Notation (SNMN) builds on SNN. Its overall orientation is to better align IT and business perspectives with the goal of improving the enablement of service offerings, requests and provision in service networks [28]. SNMN provides direct integration into the business process layer (BPMN). The notation provides network visualization with focus on contractual relationship and service offerings. Focal attributes like *service description* and *contract* (describing the SLA) have explicit representation in the model. Through a hypergraph-based representation, SNMN also depicts participant internal relationships, i.e. offerings and requests of the same service and participant. In addition it represents service providing dependencies, relating all edges of the same value flow. SNMN offers alternative role-based or participant-based views. The use of nodes and edges finds further discussion in the context of the design process of the Dynamic Network Notation in Part B.

The e3* theory and tool-suite is the basis and the umbrella for a family of modeling notations for qualitative modeling and analysis of relationships within networks [30, 31]. The basic language e3value focuses on the visualization of flows of value objects (goods, services, payments) between actors. E3control has specific focus on modeling, preventing and correcting violations of contractual obligations and models those through explicit representation [29]. A specific complement for services provided online is e3service. It is conceived to match customer needs and services through reasoning technologies [32]. Part B of this thesis takes on and discusses further representation of control mechanisms in e3control, when comparing the implementation and visualization of control mechanisms for the Dynamic Network Notation.



Name	Service Network Notation	Service Network Modeling Notation	e3*
Acronym	SNN	SNMN	e3value, e3services, e3controls
Authors and publications	**(Bitsaki, Danylevych et al. 2008; Bitsaki, Danylevych et al. 2009)**	**(Danylevych, Karastoyanova et al. 2010)**	**(Akkermans, Baida et al. 2004; de Kinderen and Gordijn 2008; Kartseva, Hulstijn et al. 2010)**
Scope	Optimization of value in service networks through composition, processes and service establishment.	Better align IT and business perspectives with the goal to improve the enablement of service offerings, requests and provision in service networks.	Ontology to model and quantitatively analyze relationships with the perspective to support, enhance or replace services.
Representation	Specific symbols for nodes and edges	Specific Symbols for nodes and edges complemented with visualized attributes (i.e.. Service Description and Contract)	Specific Symbols of nodes and edges complemented with structuring objects.
Main Modeling Elements	Nodes: Business Entities Edges: Offering relation Revenue relation Correlation	Nodes: Participants Service requests Service offerings Edges: Service providings Service providing dependency Participant internal dependency	Nodes: Dependency Nodes, Edges: Value Exchange Dependency Path Other objects: Actors, Value Ports and Interfaces Value Objects Dependency Segments

Table 1: Comparison of related work

Table 1 gives comparative overview of the three notations, focusing on scope, representation and main modeling elements. All three graphical network notations support the modeler in the design of service platforms. SNN, SNMN and e3* allow for enlightenment of different aspects in the context of locally optimized network modeling. The aspect of network effects however does not find consideration. Given the importance of those effects to the success of service platforms, an additional complementing notation is important, which abstracts respectively relevant parameters.

1.2 Main Contributions

Solution managers and platform architects need to be able to model the above described features and phenomena. In the present work, the word *harnessing* stands for exploiting outcomes through directed manipulation. Currently, this is not possible, as academia lacks a dedicated language that allows for the modeling of network effects and specific mechanisms to harness them. The contributions in the present work give an answer to the following question:

How can a modeling language support the design of service platforms, targeted at harnessing network effects?

As main contributions, the present work provides the following:

- *Conceptual model on service management in platforms* (constructs and model) – The work provides and conceptualizes structures and processes that represent the multiple relationships in and around platforms, leading to network effects as well as control mechanisms enabling supportive service management.
- *Dynamic Network Notation (model)* - Dyno is a graphical modeling language, seamlessly integrating the conceptual model for modeling service platforms, focusing on creating, stimulating and harnessing network effects. It models relevant structural and procedural aspects. Structural aspects comprise areas of staged authority or environments with scalability attributes. Procedural aspects include influences and transactions between participants of platform ecosystems and (cooperative) activities or causal loops. In response to the goal to create grounds for service management and depending on contextual feasibility, the language visualizes and allocates specific control mechanisms.
- *Service platform pattern language and repository (model)* – The language allows for the formation of a reusable base of expertise and provides a common vocabulary to communicate concepts as well as to explore design alternatives. The contribution exemplifies drafts of patterns. In addition, this contribution suggests a coordinated community-driven process to develop and evolve a repository for these patterns.
- *Dyno editor, implementing the Dynamic Network Notation (instantiation)* - This contribution provides exemplary implementation of the editor. In addition the contribution includes an analyzer with exemplary analyses. This contribution shows how the expressiveness of the Dynamic Network Notation allows for further ongoing analyses and thus for the additional guidance of the modeler.

1.3 Research Method and Thesis Structure

The present work conducts research based on the Design Science methodology as it appears to be notably appropriate to achieve the stated contributions. The key outputs in Design Science are artifacts. Artifacts as understood in Design Science refer to the entirety of constructs (terminology and symbols), models (abstractions and representations), methods (algorithms and practices) and instantiations (implementations and prototype systems) produced as outcome of a research process. By providing design concepts as well as evaluation methods, Design Science allows for the building and evaluation of artifacts conceived for clearly identified and defined purposes. Through instantiation, the methodology provides proof of concept by construction [33-36].

The remainder of this section first describes the applied research method (Subsection 1.3.1) and then outlines the thesis' structure (Subsection 1.3.2).

1.3.1 Research Method

Whereas the presentation of results in the present work follows the consecutive steps of Design Science, the research process, which led to these results, was an iterative process of building artifacts, intervention and learning and enhancement, as shown in Design Thinking [37] and as suggested by Sein, Henfridsson et al. [38] in the Action Design Research approach. Scholten revealed the first version of control mechanisms through data elicitation, surveys and experiments (Section 2.1). They evolved over a research period between 2009 and 2013 through a cyclical research process. This cyclic research process consists of the following steps [39]:

- Data collection (diagnosis) through the 5 study designs introduced in Chapter 2.
- Suggestion of categorizations (action planning); those categorizations are the result of a process of cyclical discussion with researchers and with representatives of target groups (solution managers respectively platform architects).
- Tests through modeling (intervention / action taking); in this phase the modeling of sample cases is used to reveal whether all known mechanisms can be represented.
- Evaluation and reflection (assessment and learning) through critical discussions at conferences and workshops as well as with representatives of target groups.
- Restart or termination, when the results are satisfactory.

Scholten presented an initial categorization at several conferences and workshops in 2010 and 2011 [40, 41]. Critical discussion on an evolved solution [42] at the occasion of the INFORMATIK 2011, Leipzig led to a correlation with the control modes suggested by Kirsch [43]. The categorization of control mechanisms as suggested in the present work were first submitted and critically discussed at SOCA 2012 [44]. Further developing those, the present work correlates control mechanisms, modeling elements and further details on each element. In addition it introduces the service platform provisions, an evolution of the concept of co-regulative control from the initial set of suggested mechanisms [40]. The appendix (Part E) includes an exhaustiverelated list of publications, authored or coauthored by Scholten.

1.3.2 Thesis Structure

The thesis' structure reflects Peffers, Tuunanen et al.'s [45] *Design Science Research method* on how to apply and sequence Design Science. The Design Science Research method proposes the following structure:

- Problem identification and motivation;
- Definition of the objectives for a solution;
- Design and development of a solution;
- Demonstration and evaluation (instantiation).

Part A of this thesis opens with the present motivation, outlining the reasons for this research work, insights into related research, an overview of its contributions and an explanation of the research methodology. Chapter 2 addresses the first stage of the Design Science Research method - problem identification and motivation - uncovering concrete research requirements. It closes with the formulation of respective research questions and goals. To do so, the author conducts a set of qualitative and quantitative surveys targeted at service management on service platforms, which provide a holistic impression of network-related challenges. Quantitative data, extracted from a longitudinal study on web-service intermediaries, allows the discovery of relationships that are not obvious to the researcher's perception at first glance. Qualitative data from case studies, on the other hand, is helpful to understand "the rationale or theory underlying relationships revealed in the quantitative data" [46].

Prior to addressing the actual formulation of objectives for a solution, the thesis lays the foundations of research. Correspondingly, Chapter 3 introduces and structures language engineering as well as graphical language engineering specifically and gives an introduction to the theory related to network effects and control.

Part B is dedicated to the development of the Dynamic Network Notation and its pattern language. It starts in Chapter 4 with structuring gained knowledge into a conceptual model, which then lead to the elicitation of functional design requirements for the graphical modeling language. The subsequent Chapter 5 produces models and methods as artifacts. Explicitly, it develops the thesis' central artifact with an abstract grammar (morphology and syntax) and the corresponding non-formalized semantics for the Dynamic Network Notation. The illustrating Dyno-models developed within this chapter make use of this graphical modeling language to represent real world scenarios. The chapter continues with the complementing pattern language and repository as an evolving and reusable base of experience that allows modelers to learn from and to share knowledge on best practices.

Part C is targeted at instantiating and evaluating the thesis' artifacts. Instantiation stands for the actual system implementation, allowing for demonstration and evaluation of feasibility and suitability. In this regard Chapter 6 instantiates the Dynamic Network Notation by building an exemplary editor. This editor allows solution managers and platform architects to model and analyze service platforms and their surrounding dynamic networks. In Chapter 7, various experts evaluate quality and usefulness of the Dynamic Network Notation and its editor, as well as their ability to respond to the research requirements and research hypotheses, formulated in Part A.

Part D closes the present work by critically discussing the research results (Section 8.1), and providing an outlook on future research work (Section 8.2).

Figure 1 gives a graphical overview on the present work, representing chapters, contributed artifacts and thesis structure.

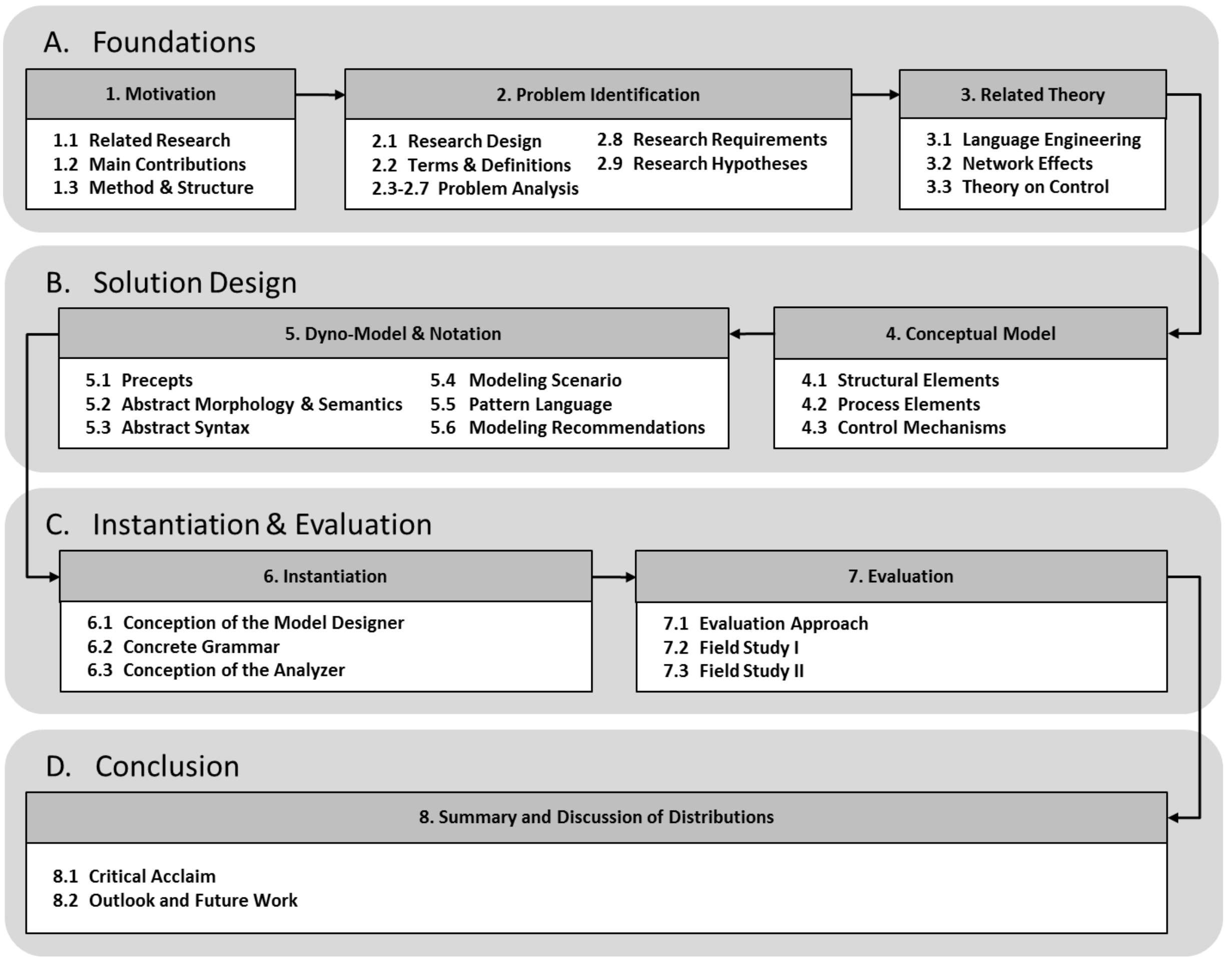


Figure 1: Outline of the present work

2 Problem Identification

This chapter analyses service management on platforms with specific focus on harnessing network effects. It first presents the research design for data acquisition (Section 2.1) and introduces basic terms and definitions (Section 2.2). The Sections 2.3 to 2.7 identify relevant problems for this thesis. It consolidates the findings through the formulation of research requirements (Section 2.8) with respect to the design of a graphical modeling language, specific to service management on platforms in view of network effects. These research requirements lead to the articulation of a research hypothesis (Section 2.9) around the requirements of such a language.

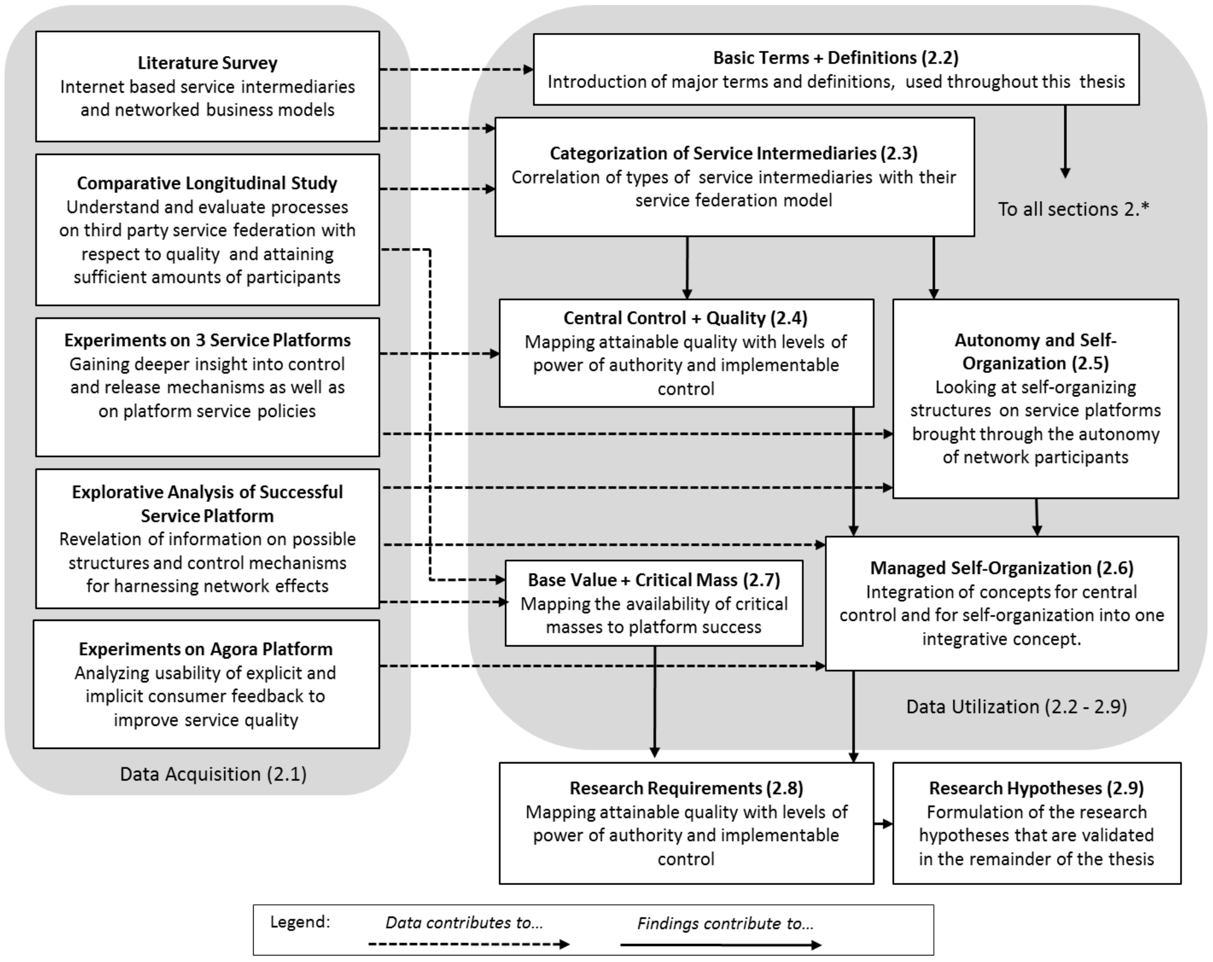

Literature Survey
Internet based service intermediaries and networked business models
Comparative Longitudinal Study
Understand and evaluate processes on third party service federation with respect to quality and attaining sufficient amounts of participants
Experiments on 3 Service Platforms
Gaining deeper insight into control and release mechanisms as well as on platform service policies
Explorative Analysis of Successful Service Platform
Revelation of information on possible structures and control mechanisms for harnessing network effects
Experiments on Agora Platform
Analyzing usability of explicit and implicit consumer feedback to improve service quality
Data Acquisition (2.1)
Basic Terms + Definitions (2.2)
Introduction of major terms and definitions, used throughout this thesis
To all sections 2.*
Categorization of Service Intermediaries (2.3)
Correlation of types of service intermediaries with their service federation model
Central Control + Quality (2.4)
Mapping attainable quality with levels of power of authority and implementable control
Autonomy and Self-Organization (2.5)
Looking at self-organizing structures on service platforms brought through the autonomy of network participants
Base Value + Critical Mass (2.7)
Mapping the availability of critical masses to platform success
Managed Self-Organization (2.6)
Integration of concepts for central control and for self-organization into one integrative concept.
Data Utilization (2.2 - 2.9)
Research Requirements (2.8)
Mapping attainable quality with levels of power of authority and implementable control
Research Hypotheses (2.9)
Formulation of the research hypotheses that are validated in the remainder of the thesis
Legend:
Data contributes to...
Findings contribute to...

Figure 2: Structure of the chapter on problem identification

Figure 2 gives an overview of the detailed workflow and revealed research requirements in the present chapter. It starts with a definition of basic terms required for the subsequent problem identification (Section 2.2). It then continues with the categorization of the various forms of service intermediary concepts into groups (Section 2.3). The analysis discovers three zones of staged authority as relevant factors of differentiation within various intermediation concepts. The subsequent section investigates aspects of control through the platform operator and resulting implications on quality within the different service intermediary types (Section 2.4). Then the work looks at concepts of autonomy of network participants and resulting self-organization (Section 2.5). It then continues researching the resulting service management dilemma immanent to service intermediaries: Service intermediaries are on the one hand tempted to favor self-organization of service providers, allowing for rapid growth and adaptability without the bottleneck of central control and organization (Section 2.5). This is countered by a need for central control to guarantee service quality. Section 2.6 consolidates both into one integrated concept of managed self-organization. Then the chapter sheds light on the question of what is required to start off a network effect (Section 2.7). Section 2.8 and Section 2.9 formulate the resulting research requirements and research hypotheses.

2.1 Research Design for Data Acquisition

For the identification of necessary data, the author conducted five different kinds of research design:

Literature Review

Scholten conducted a literature review focused on Internet based service intermediaries and networked business models. This survey provided general insight into research, related to service platforms and provided terminology. It enriches the remainder of this chapter with related theory and examples.

Comparative Longitudinal Study

In a second survey, the author conducted a comparative longitudinal analysis on service intermediaries, analyzing and evaluating their processes with respect to third party service deployment and federation as well as their performance with respect to attaining sufficient numbers of third party services and insuring quality of service. The study compared the intermediary operators *SeekDa*, *WebServiceList*, *Xmethods*, *RemoteMethods*, *eSigma*, and *StrikeIron Market Place*, backed with a longitudinal comparison of quality of service intermediated by SeekDa and StrikeIron Market place. The survey deliberately used the sample group that was used in a related publication by Legner [47] to be able to consolidate findings from literature and from this study. The sample group was extended by the intermediary SeekDa, as it provided an additional intermediary design. The study design has a qualitative and a quantitative dimension. In the qualitative dimension, the survey explores the companies' websites (i.e. the terms and conditions) to gain information on the intermediaries' architectures and service management approach. The quantitative part had a longitudinal research design. It was based on 170 log-files, acquired from an online database of monitoring data. The chosen set of log-files documented availability per service over a minimum of 6 months. Grouping service set of logs per service intermediary allowed correlations of quality of service to be deduced, for example of availability with specific service intermediation concepts. The results support Section 2.3 with respect to the categorization of service intermediaries. They further serve Section 2.4, when relating service quality, the power of monitoring quality of service and the power to control supplied quality of service of third party services. The results also find consideration in Section 2.7, when discussing base value and critical mass. For further reading on this survey, please refer to Scholten, Fischer et al. [48]

Experiments on Three Selected Service Platforms

Scholten experimented on the platforms *Force.com* by Salesforce.com, *SuiteApp* by Netsuite and *Facebook Platform* by Facebook. In the study design, experimentation meant to deploy an own sample service on the platform to gain a deeper insight into the control and release mechanisms as well as on the platforms' provisions. These experimental findings were complemented through analysis of the platforms' terms and conditions. The findings of this survey find exploitation in Sections 2.4 and 2.5, when discussing self-organization and control as well as in Section 2.6 when integrating both.

Explorative Analysis of Successful Service Platforms

Scholten revealed information on possible structures and control mechanisms with respect to harnessing network effects, through an extensive analysis of successful service platforms, conducted between 2009 and 2013. Two alternative factors justified the attribute *successful*. Either their financial success in the service platform area, substantiated through published investigations [4, 10] or mechanisms or structures which successful accomplish a specific goal e.g., the collaboration software as a service provider Trello [49], who applies specific structure and mechanisms to achieve network effects even with a small number of users. In particular, the present work refers to the following service platform operators (in alphabetical order): Appirio Inc.[1], BOINC[2], Dropbox Inc.[3] Google Inc[4].; Facebook Inc.[5], Intensify Inc., Intuit Inc.[6], LongJump Inc.[7], Netsuite

[1] http://appirio.com/ retrieved 16.02.2013;
[2] http://boinc.berkeley.edu/, retrieved 25.03.2013
[3] https://www.dropbox.com/ retrieved 16.02.2013;
[4] http://www.google.com/ retrieved 16.02.2013;
[5] http://www.facebook.com/ retrieved 16.02.2013;
[6] http://www.intuit.com/ retrieved 16.02.2013;
[7] http://www.longjump.com/ retrieved 16.02.2013;

Inc.[8], Salesforce.com Inc.[9], SAP AG[10], S.Chand Edutech Inc.[11], Trello Inc.[12]. Some of the quoted companies, e.g., SAP AG or Google operate several business models, including non-Cloud oriented activities. When speaking about such an operator, the present work always names the specific service platform. Definition 20 provides a definition of the term *service platform*, embracing all above described companies' approaches of intermediation. This definition is substantiated with findings in Sections 2.2 and 2.3. As a working definition, service platforms are providers of metered services on demand over the Internet.

Experiments on the Research Platform Agora

In the frame of the TEXO / Theseus project[13] i.e. the subproject SVN[14], May, Scholten et al. [50] analyzed the usability of explicit and implicit feedback to improve service quality (lead: Norman May). The team experimented on the research platform and e-market pilot *AGORA*. The pilot platform instantiates processes of discovery, configuring, ordering and delivering services including monitoring of pre- in- and post-delivery activities. The monitoring of experimental test consumer behavior allowed retrieving necessary feedback on consumer self-organization guided through feedback (Section 2.6). The team conducted analysis and adaptation of the services and service portfolio manually based on the information provided in analytic reports. Even though manually accomplished, comparison proofed a modification of service quality in test loops based on implicit and explicit feedback provided to service providers. At the time of write-up of this thesis, the integration of tools for a more automated process of service portfolio optimization is still under development For further reading on experimental setup and analysis results, please refer to May, Scholten et al. [50].

8 http://www.netsuite.com/ retrieved 16.02.2013;
9 http://www.salesforce.com/ retrieved 16.02.2013;
10 http://www.sap.com/ retrieved 16.02.2013;
11 http://www.schandedutech.com/ retrieved 16.02.2013;
12 https://trello.com/ retrieved 16.02.2013;
13 http://www.igd.fraunhofer.de/Institut/Abteilungen/IVA/Projekte/THESEUS-TEXO
14 Cooperation between Karlsruhe Institute of Technology and SAP AG, http://www.aifb.kit.edu/web/SVN/en, retrieved 16.02.2013;

2.2 Basic Terms and Definitions

This Section introduces major terms and definitions, which are generally used throughout this thesis. The definitions are grouped into three subsections: Subsection 2.2.1 specifies terms and their understanding in the area of services and service management. The subsequent subsection 2.2.2 enlightens meaning and terminology around network effects. Lastly, subsection 2.2.3 explains the target groups addressed by the artifacts provided through this thesis and positions those groups in the context of corporations. Being positioned in the field of language engineering, the present work dedicates a specific section to this discipline (chapter 3.1) before applying it in Part B.

2.2.1 Definitions Related to Services

Software-as-a-Service groups all applications, running in the cloud and accessible to end-users as metered services on demand [1, 3]. These end-user applications comprise of application services. These building blocks could be basic applications like OpenId, or of composite nature, e.g., OpenSocial [3]. Important in this thesis is not size or software design (like being RESTful or SOAP-message based). It is their federation and supply-mechanisms over more or less distinct networks of service providers and consumers, which defines the respective service management design. In the remainder of the thesis, the term *service* stands for any of those services. In cases where a more fine granular consideration is required, like in the subsequent analysis of Web service providers or when exemplifying software-as-a-service platforms, terminology becomes technically specific. This thesis uses the following definition:

Definition 1: A service stands for any kind of deployed software, provided over the Internet by service platforms on demand.

Management of those services has the goal of providing value to consumers through satisfaction of their service requirements [51]. Service management is about managing the whole service life-cycle, including the service strategy, service design, service transition, service operation and continual service improvement. Correspondingly, the goal of service management is to make capabilities and resources available that bring value to the consumer.

Definition 2: Service management in platforms describes the activity of managing the whole service life-cycle, service design, service transition, service operation and continuous service improvement with the objective to make capabilities and resources available that are required by the consumer.

The ITIL framework of best practices in service management provides a suitable definition for this newly introduced term *service value* in the context of software services: Value combines *utility* and *warranty*. The term value circumscribes those service attributes with positive effect on performance of actions, objects, and tasks. Warranty describes how utility is guaranteed through appropriate service availability, capacity, continuity and security [51].

Definition 3: Service value describes those service attributes with positive effect on performance of actions, objects and tasks.

The perception of value depends on the addressee. A potential consumer who is in need of a specific service might attribute a higher notion of value to this service than somebody without this specific need. A more macroscopic perspective of value considers the value contribution of a whole platform in respect of defined subsets. In analogy to Definition 3, service platform value can be defined as follows:

Definition 4: Service platform value denotes those attributes, arising from the whole platform or from defined subsets, which have positive effect on the performance of actions, objects and tasks.

Section 3.2 complements this value consideration for whole service platforms, substantiated through dynamic market theory. When not specifically distinguishing between value of services and value of service platforms, the present work simply uses the short term *value*.

Section 2.2.2 of this work signals that this value contribution is not necessarily only originating from the platform operator but might be provided from external participants or co-created through interaction with external participants. The present work calls a value flow into the platform or within the platform a *transaction*. The word transaction captures well the flow and exchange process of rights of property and liberty to act based on contractual agreement [52]. This might include e.g., the transfer of services, developed by external participants. Transactions may also include the transfer of anything which is eventually turned into service value on the platform as result of a process of co-creation or interaction. For example an increase in subscription of users to a service is a trust building and hence values creating positive effect as other users perceive the increase in users as a sign of quality of service. The remainder of this thesis refers to any kind of value creating activity on the platform and in interaction with the platform as *activity*.

Definition 5: Transactions describe any kind of transfer into, from or within the platform, which eventually could create or reduce value in the platform.

Definition 6: Activities are value creating activities on and in interaction with the platform.

2.2.2 Definitions Related to Network Effects

Network effects play a central role in this thesis. Prior to part B, Section 3.2 provides detailed system-theoretical background, which serves as foundation for the subsequent language engineering. This subsection therefore focuses on definitions, which are required in the remainder of Chapter 2. Rohlfs [53] introduces a theory where he communicates the idea of interdependence of demand, substantiated with the example of the telecommunication industry. Also based on the analysis of the same industry Oren and Smith [54] speak of demand externality when a subscriber's benefit depends on the total quantity of subscribers. Katz and Shapiro [55] generalize this theory and use the term *network externalities* for positive external consumption benefits to the adoption of a specific technology. Continuing on this line of thinking, Shapiro and Varian [56] use the term *network effects*, when describing the phenomenon that products only become valuable when large numbers of people are using them. They discover that many digital goods are subject to network effects. Sterman [8] shows that network effects can be strengthened or weakened by complementary network effects. Rochet and Tirole 2003 [57] reveal that in many cases where network effects occur, two or more distinct participant groups benefit from each other. They term the business model behind *two or multisided markets or platforms*. Eisenmann, Parker and van Alsthyne speak of demand-sided network effects for those effects discovered by Rohlfs respectively Katz and Shapiro. They find out that platforms, which open for external supply, can encompass *cross-sided network effects* [19, 22, 58]. In these effects, *supply side* and *demand side* are interdependent.

A key notion of network effects is the self-organization of the concerned participants. The term *self-organization* originates from system theory and non-equilibrium physics, i.e. from the research of Nicolis, Prigogine et al. [59, 60]. They revealed that in self-organizing systems small changes can incite amplified network effects. They see scope of application of that theory on physical as well as on social systems. De Wolf and Holvoet [9] describe self-organization from a theoretical perspective as an adaptive process, where a system organizes and acquires structure without external control. In their view, structure can be spatial, temporal or functional. Self-organizing systems have a series of features which favor network effects [9, 59, 61].

- Autonomy – the parts of the system act without central control;
- Adaptability – the parts of the system are able to cope with changes autonomously and rapidly and align to a new temporal equilibrium. The system as a whole becomes robust to perturbations;
- Sensitivity to change – the system is sensitive to a changing requirement and can react rapidly. This requires the prerequisite to keep the parts of the system in an instable condition. Nicolis and Prigogine [59] call this a far-from-equilibrium state.

The self-organizing autonomous participants represent the platform ecosystem. The described service platforms benefit from their autonomy, adaptability and sensitivity to change. The sensitivity to change however requires the provision of suitable information to the participants in the platform ecosystem as a prerequisite to reactivity.

The provision of information about activities on a platform to participants that are active on a platform is called *feedback*. The participants' reaction on this information causes reciprocity, meaning modified activity on the platform leading to renewed feedback. Section 3.2 goes deeper into this with more theoretical background and pinpoints the far-from-equilibrium state in a phase plot of a platform's market share as a function of its participants' involvement (Figure 9).

Definition 7: Feedback is the targeted provision of information on platform-based activities to a participant, active in and in reciprocal relationship with the platform.

Definition 8: Platform ecosystems are sets of autonomous participants around service platforms, which are in reciprocal relationship with the latter.

Definition 9: Self-organization in the context of service platforms describes the line-up of platform ecosystem participants over time to a temporary situation of equilibrium, attained through feedback.

Definition 10: Network effects describe the reciprocal relation between the value of a service platform and the quantity of involved service consumers and service providers. Network effects are driven by self-organization of the platform ecosystem.

An important technical requirement to the parts of the platform which are affected by network effects is *scalability*. Neuman [62] describes scalability in distributed systems as a system's ability to "handle the addition of users and resources without suffering a noticeable loss of performance or increase in administrative complexity." According to Neuman, scale has three dimensions: the number of users and objects that are part of the system, the distance between the farthest node and the number of organizations that exert control over the system. Scale affects a system in various ways, i.e., reliability of the distributed system, load that needs to be managed, administration of an increasing amount of nodes and heterogeneity of architecture, which is likely to grow with an increased amount of nodes [62]. Binnig, Kossmann et al. [63] state the requirements that cloud services should scale in a linear way and infinitely with constant costs per web interaction. As a means to measure scalability the authors suggest increasing the web interactions per second and counting the produced responses in a specific time interval.

In the context of service platforms, the present work suggests the following definition for scalability:

Definition 11: Scalability describes to ability to meet increased workload, through a planned incremental increase of capacity

Capacity to scale depends on various factors, in particular on suitable compute and storage capacity, but also on the way that services and networks can handle increased numbers of users. *Elasticity* refers to the system's capability to handle sudden increases and decreases in load [1]. Such fluctuations are not related to network effects as those are characterized through inert movements in specific directions (section 3.2 deepens this aspect). Requirements of elasticity therefore rather depend on the specific business models. As an example, if a service handles salary management and payments, it can be expected to have high loads between the 25th and the end of a month and less during the remainder of the time. In such cases, the platform architect needs to conceive suitable mechanisms. Those activities, however, go beyond the scope of this work.

2.2.3 Definitions Related to the Addressed Target Groups

The present work states as target group for the graphical modeling language those persons in charge of design and evolution of service platforms. As corporate structures are diverse, the target profile needs to be clearly defined. The following paragraphs look at organizational structures to help locate and define the targeted job profiles. It requires a short look at theory and best practices in organizational management.

Most organizational structures are represented by a company board defining corporate strategy, vision and mission. Operationalization and execution of these goals happens under a director in charge, generally referred to as Chief Executive Officer (CEO). Responsible for service platform implementation and management in functionally structured companies are Chief Technical Officer (CTO) and Chief Marketing Officer (CMO), both reporting to the Chief Executive Officer (CEO). The CTO is in charge of strategic corporate decisions at the intersecting roles of technical expert and of business strategist. His task is to choose technologies based on their likelihood to generate the highest rate of return and growth in the context of a business strategy [64].

The CMO is in charge of strategy implementation with focus on solution, price, distribution and communication. In platform businesses, solutions, technical infrastructure and communication are all part of the technical platforms, leading to overlaps of these positions. Solution managers or product managers operationally implement those tasks. They have to take into account expectations coming from diverse corporate stakeholders, including research and innovation, development, sales and marketing and support. External stakeholders are partners and consumers, as well as potential consumers, competitors and analysts [65]. Depending on the company structure, product managers report to CTO, CMO, or in a matrix structure to both.

Companies with strong relevance to information technology introduced the position of a Chief Information officer (CIO), either alongside the CTO or instead. Trenner [66] identifies three types of CIO: the role of a *function head* whose profile is mainly the operational management of corporate IT, the transformational leader focusing on change management within corporate processes and their enablement through IT, and the business strategist. The latter has the highest level of responsibility. His focus is to steer corporate strategies in pursuit of accomplishing corporate goals. Given the importance of information technology for their corporate success, IT-focused companies like Google choose another approach and put the CEO in person in the lead of product development and technical strategy[15].

The described options reflect established solutions and trends. They cannot be exhaustive. However, they elicit that various job profiles may be in involved in modeling or designing service platform. Degrees of responsibility and involvement depend on the respective corporate structure. But all constellations occupy two roles albeit different denominations: the role of the analyzer who identifies economic parameters to be considered in the platform design and the platform architect who is in charge of designing the platform technically.

Definition 12: The solution manager brackets those job profiles, which identify and communicate business requirements, opportunities and goals for a service platform.

Definition 13: The platform architect brackets those job profiles in charge of the overall technical platform design.

[15] https://www.google.com/intl/en/about/company/facts/management/

2.3 Service Intermediaries

In pursuit of defining Cloud computing ‘to enhance and inform the public debate on cloud computing’[1], the National Institute of Standards and Technology (NIST) provides a categorization of cloud service models into SaaS, PaaS and IaaS. This serves well its purpose of providing general overview on Cloud computing. In an early classification on Internet-based distribution models, Tapscott [67] classifies networks of legal entitites in the web in accordance to their level of integration and control into agora, aggregator, alliance and value chain. Agora stands for self-organized electronic marketplaces with negotiable pricings in contrast to the more hierarchically organized aggregator. For business models of higher integration, Tapscott [67] defines the alliance as a self-organized value creating community, while the value chain is the most hierarchical and highly integrated form of a distributed network. Meier and Ullrich [68] build on this taxonomy, dividing the value chain into integrator and distributor. In contrast to distributors, integrators host services within their proper infrastructure and have detailed insight into and direct influence on them. Integrators can host 3rd party services as well as proprietary services. With the example of web-service intermediaries, Legner [47] provides a rather functional, tripartite taxonomy consisting of electronic Market (*e-market*), electronic hub (*e-hub*) and *infomediaries*. Legner suggests as differentiating parameters: information provision, customization, matching, transaction, assurance, logistics, collaboration, integration and standardization. Legner’s [47] group of electronic markets unifies agora and aggregator into one single category. E-hubs are intermediaries that facilitate supply chain collaboration. Complementing this categorization with Meier and Ullrich’s [68] category of *integrators* provides the best fit. This extended categorization allows for technical differentiation in view of the intermediaries’ capacities to exert control, or to allow for self-organization. Table 2 summarizes the categorization.

Intermediary Groups	Infomediary	E-hubs	E-market	Integrator
Goal	actively collecting, pre-processing and providing information	enabling supply chain colla-boration, no sales	marketplace for web services	optimized, highly integrated supply chain
Characteristics	off-site Web services and traffic	off-site web services and traffic	web services off-site, traffic routed through	web service residence and integration on-site
Stakeholding power	none	exclusion mandate for web services	influence on portfolio	influence of web service design and portfolio, owns supply infrastructure
Examples	Seekda	xmethods, Remotemethods, Web Service List	e.sigma	StrikeIron

Table 2: Technically oriented categorization of service intermediaries

This thesis suggests the term *stakeholding power* when talking about the platform operator's authority over quality of offered services.

Definition 14: Stakeholding power describes the degree of authority of a platform operator over an ecosystem participant or activity.

Looking at stakeholding power from an architectural perspective Figure 3 demarks three areas of staged authority for intermediaries related to supply infrastructure and in the context of the present longitudinal analysis.

Definition 15: Control area is the section, where the platform operator can exert full stakeholding power through enforcement. He can observe and steer all events and structures within the control area

Definition 16: Influence area is the subsection of a platform ecosystem, where the platform operator can only exert limited stakeholding power through incentives. He cannot observe but only inquire information on events within the influence area.

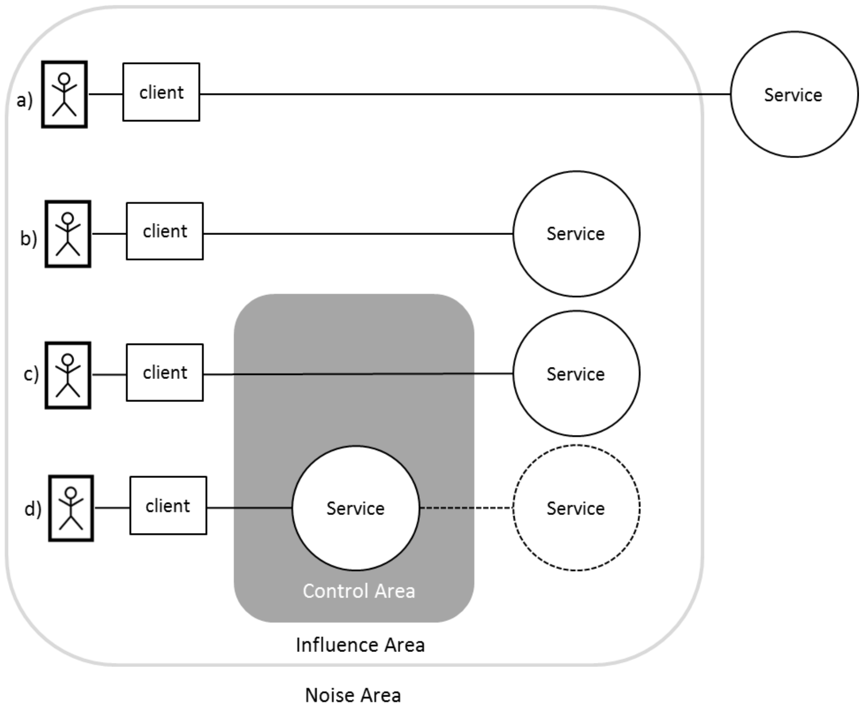


Figure 3: Areas of staged stakeholding power

Definition 17: Noise area is the subsection outside the platform ecosystem, where the platform operator has no stakeholding power.

The term *control*, used in this context, originates in control theory (Foellinger 1984) and exceeds the meaning of *Kontrolle* in German or *contrôle* in French, which are closer to *verification*. Definition 18 terms *control* in the context of service management on platforms.

Definition 18: Control describes service management actions by the platform operator in order to change a set of parameters from a current status (actual value) to a target status (setpoint). Control in a platform context operates as a closed-loop, meaning with monitoring feedback in the context of a regulatory process. The mechanisms which are used to control such a process are called control mechanisms.

The following examples illustrate this understanding. Infomediaries correspond to type a) in Figure 3. The consumer choses cooperation with the intermediary; he is therefore placed in the influence area. The services, which are simply crawled by the infomediary lie outside the influence area. E-hubs do not have access to any data traffic, while federating a service. However, both consumer and service provider actively choose cooperation with the e-hub. Therefore both are located in the influence area. This corresponds to type b) in Figure 3. E-markets (type c) represent the first supply concept with limited enforcing authority. E-markets can control and tab all traffic between the client and the service provider(s), as it is routed through the control area. The Integrator (type d) is omniscient to all traffic coming from and going to the client. He has enforcing power over the service as it is deployed within the control area. However, this omniscience and stakeholding power shrinks, once the service is of a composite nature with services outside the platform’s control area (dotted line and service).

Services deployed within the control area might request services outside (tier-(n+1)-success). Although tier-(n+1) services are of certain risk to quality, none of the analyzed platform operators ruled-out loose service coupling with tier-(n+1) services. To ensure proper quality of service of third party services within their own platform in spite of their composite nature, the platform operators either prescribed the application of a programming specification or even of a programming environment. When choosing a programming specification, the platform operator provides technical programming guidelines, mostly supported by tools and templates, allowing for a smooth integration of services into the platform. In such cases, the platform operator needs to enforce its directives in a second step through a compliance test, after the provider has uploaded the ready-made service. In the case of a programming environment, the service development takes place within the platform environment and can be continuously tracked. Technical non-compliance at deployment time can be ruled out by filtering (details explained in the subsections on control mechanisms). But further to such pre-filtering, the platforms need to manage non-compliance to quality requirements throughout operation time by permanent monitoring and relevant control mechanisms. The programing models and environments have also a role in facilitating QoS-monitoring.

2.4 Central Control and Quality

The above Sections showed that different designs of intermediation lead to different levels of authority (stakeholding power) on the activities of service providers. This section analyses, the extent to which the different types of intermediation were able to ensure quality of service.

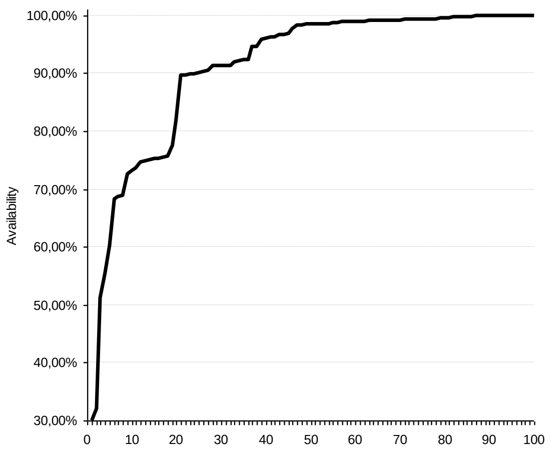
Lewis and Booms [69] define quality of service as a measure to describe the match of service delivery and consumer expectations. Service delivery involves a contractual relationship, comprising a set of service levels to express a consumer's demanded service quality [70]. The dimensions of service quality depend on the domains of application. Parasuraman, Zeithaml et al. [71] derive parameters for service quality for business-oriented services. They include parameters such as reliability, responsiveness, competence, courtesy or credibility. Keller and Ludwig [72] as well as O'Sullivan [73] researched quality of online provided services, in particular of Web services. In their analysis, the monitoring of technical parameters during service delivery (e.g., response time or throughput) has been focused to shed light on the gap between expected and delivered service quality. Maximilien and Singh [74] as well as Vu, Hauswirth et al. [75] define QoS-attributes such as service reputation and endorsement as parameters for service discovery. Kalepu, Krishnaswamy et al.[76] described service quality as degree of compliance of a service during service delivery with the previously advertised Service Level Agreement (SLA). Cardoso and Sheth [77] analyze quality of composite services which are generated within a network. They include attributes such as services to be delivered, deadlines, quality of products, and cost of service.

The present work builds on the Web-service Quality Model WSQM [78], for several reasons. First it is already focused on services, supplied via the Internet (however originally written for a very special type of service). Second, it integrates functional and non-functional aspects, which give a necessary holistic view, and which integrates technical as well as economic viewpoints. The present work regroups the WSQM's quality factors into more coarse granular categories of quality parameters. The reason for this restructuring is the intent to cover all subject areas of quality, which are relevant in the context of services delivered via service platforms.

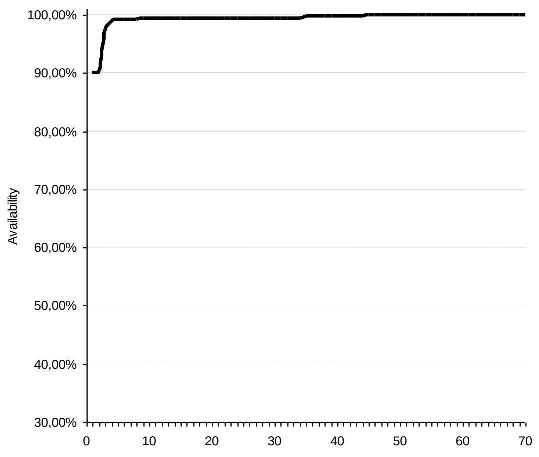
The thesis simplifies them to provide better clarity i.e. through reduction of the quality parameters from 86 down to 21 (Table 2).

- *Business Value Quality* for economic and functional consideration (e.g., service sustainability);
- *Service Level Measurement Quality* describing user perception on how fast a service is provided and how stable it is. Those parameters are related to design of service and deployment environment (e.g., availability);
- *Business Process Quality* describing the ease of service integration into other processes (e.g., collaborability);
- *Suitability for standards* describing compliance to platform service specifications (e.g., conformability);
- *Security Quality* describing the level of built-in security mechanisms within a service (e.g., trace management);
- *Manageability* describing the level of observability (introspection), the ability to write to a system and to a system's internal information (controllability) as well as of the provisioning of management information to platform operator and consumer (notification) [78].

Definition 19: Quality of service describes the match of service delivery with functional and non-functional consumer expectations. Quality of service includes the quality groups Business Value Quality, Service Level Measurement Quality, Business Process Quality, Suitability Quality, Security Quality and Manageability Quality.



a) Web service samples 1-100, different intermediaries



b) Web service samples 1-70, only Integrator (StrikeIron)

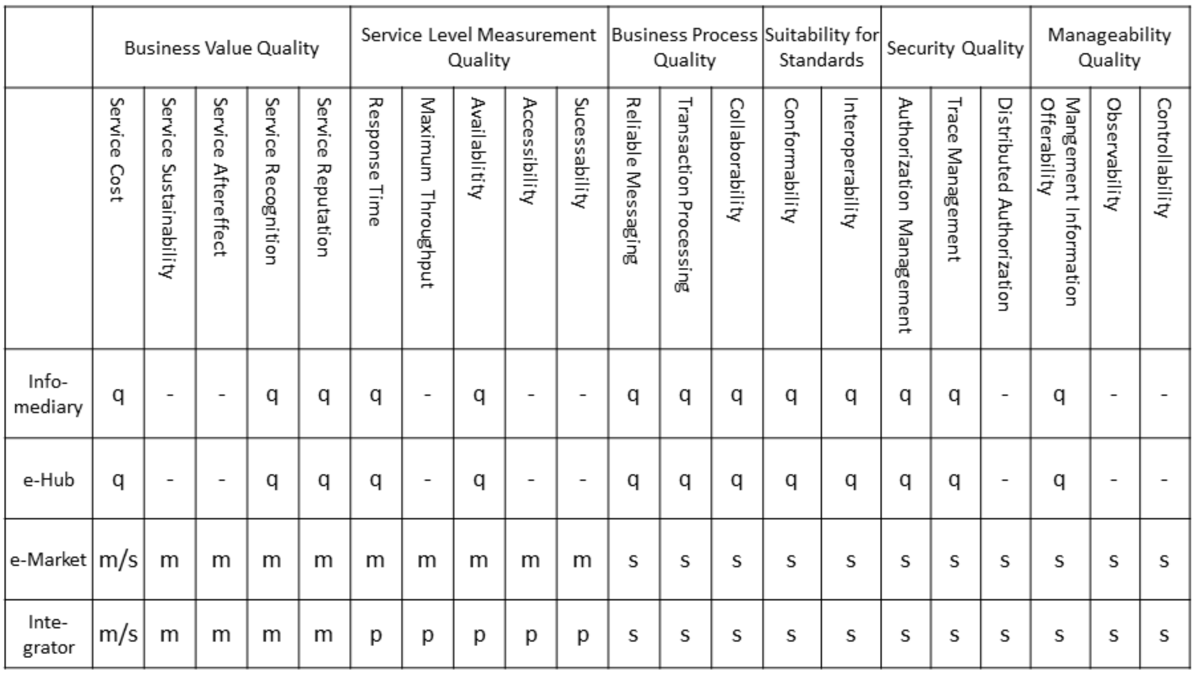
Figure 4: Comparison of Long-Time Availability of Web Services as ordered sets, structured by increasing availability

The conducted longitudinal study analyses a sample set of Web service intermediaries. The sample set includes one infomediary (SeekDa) and one integrator (StrikeIron Market-Place). The analyzed companies are comparable with respect to size and maturity. The analysis' objective is to draw conclusions on dependency between service quality and the intermediary's stakeholding power over service providers. The study uses sets of logs on service availability, collected over a minimum of 6 months per provider. It produces a correlation of 100 Web services for SeekDa (Figure 4a) and 70 Web services for StrikeIron (Figure 4b) to their availability over a period longer 6 months. The data logs originate from raw data of Seekda's daily-performed availability test (www.seekda.com). Figure 4a represents a set union of Web service logs, originating from all different intermediary types. The set union of Figure 4a brings to light that 23 out of the 100 analyzed Web services performed below 90% availability. A total of 22 Web services showed availability between 90% and 97.99%. While 25 Web services were available between 99% and 99.99% of the time, only 14 of 100 Web services achieved 100% availability. The analyzed sample of 100 Web services accomplished an average availability of 91.22%. The logs of the integrator-hosted services showed a much better performance than the non-hosting service intermediaries. Therefore, the survey looked in a second analysis at the specific intermediary type integrator, analyzing a new sample set of 70 integrator-hosted services (Figure 3b) with the example of StrikeIron. The services from the integrator showed an average availability of 99.37%[16]. The remainder of this subsection investigates on the reasons for the differences in performance.

Table 3 relates stakeholding power over service quality with the each intermediary type. Further elaborating on an initial categorization by Scholten, Fischer et al. [48] the present work categorizes into *query*, *monitor*, *specify* and *prescribe* to circumscribe the level of stakeholding power. The higher levels of stakeholding power include the lower levels. *Query* describes the weakest level of authority. Quality information is not available through the system, but needs to be collected from publicly available sources, through inquiries or through automated tests (pinging of services). *Monitor* is a level, achievable for those intermediaries, which route service consumption traffic through their domain and which can collect data like business quality data through suitable systems e.g., reputation systems (Figure 3c). Monitoring does not include enforcing power over a quality parameter. Intermediaries, which control the traffic, the service or the deployment environment can actually prescribe non-functional quality parameters. Prescription in that context means they can define and adapt those quality parameters which depend on platform resources (memory, storage, compute), on distributed architecture (e.g., time until eventual consistency is reached) and on specific aspects of service design e.g., through code optimization of binary executables in cases when the service is deployed on different hardware and

[16] Services on platforms can be identified due to the service's address. As e-Hubs and e-Markets do not allow for a clear correlation of services and platform they are not part of this quantitative comparison. Exemplary verification of their service's availability showed a quality level around the average of the infomediary.

requires different native machine instructions. Whereas infomediaries and e-hubs are limited on querying, e-markets and integrators can exert active service portfolio management. E-markets and integrators can specify service design. As they can observe and theoretically sanction non-fulfillment, this specification can have binding or non-binding annotation. The integrator however can go one step further. It can prescribe all non-functional quality parameters, e.g., through service design which mandatorily takes place in a programming environment and which only gives selective design options.



	Business Value Quality					Service Level Measurement Quality					Business Process Quality			Suitability for Standards		Security Quality			Manageability Quality		
	Service Cost	Service Sustainability	Service Aftereffect	Service Recognition	Service Reputation	Response Time	Maximum Throughput	Availablitity	Accessibility	Sucessability	Reliable Messaging	Transaction Processing	Collaborability	Conformability	Interoperability	Authorization Management	Trace Management	Distributed Authorization	Mangement Information Offerability	Observability	Controllability
Info-mediary	q	-	-	q	q	q	-	q	-	-	q	q	q	q	q	q	q	-	q	-	-
e-Hub	q	-	-	q	q	q	-	q	-	-	q	q	q	q	q	q	q	-	q	-	-
e-Market	m/s	m	m	m	m	m	m	m	m	m	s	s	s	s	s	s	s	s	s	s	s
Inte-grator	m/s	m	m	m	m	p	p	p	p	p	s	s	s	s	s	s	s	s	s	s	s

Table 3: Mapping table of quality factors and intermediaries; q: query, m: monitor, s: specify, p: prescribe

The study now maps the above described levels of stakeholding power per intermediary type with the actual measured performance. It used information on service management, given in the respective company websites (terms and conditions, programmers' guidelines). Table 4 summarizes the results. Being powerful in terms of stakeholding power towards Web service providers, the integrator clearly has the potential to outperform the e-market eSigma in terms of quality monitoring and the possibility to prescribe. With the services deployed in the integrator's domain, the integrator has better potential to monitor and adapt the quality parameters; especially those depending on the infrastructure.

The analyzed e-market operates with a defined quality management process for the release of Web services. It is based on a specification of expected service features to insure appropriate quality of service. The e-market's terms and conditions state that the deployment of services is restricted to compliance with these specifications. However, the e-market does not feature successive quality management procedures for released services. Availability of services federated by this market is only marginally differentiable from the average market performance. Because they route all traffic through the control area (Figure 3, constellation c), e-markets dispose of basic portfolio management options. They could for instance improve their service portfolio through eliminating low-performers from their portfolio of federated services. Further, they have the power to apply motivational influence onto their ecosystem.

	Quality Factor	**SeekDa!**	**WebService List**	**Xmethods**	**Remote Methods**	**eSigma**	**StrikeIron Market Place**
	Business model	Infomediary	e-Hub	e-Hub	e-Hub	e-Market	Integrator
	Residence	off-site	off-site	off-site	off-site	off-site	on-site
Business Value Quality	Service Cost	yes	no	no	yes	yes	yes
	Service Sustainability	no	no	no	no	no	no
	Service Recognition	yes	no	no	no	no	no
	Reputation (score)	yes	yes	no	yes	no	no
	Reputation (Descriptive)	yes	no	no	yes	no	no
Service Level Measurement Quality	Response Time	yes	no	no	no	yes	yes
	Availability	daily availability check	no	no	no	hourly availability check	minute-by-minute availability check
	General information through trial	no	yes	yes	yes	yes	yes

Table 4: Comparative analysis on exerted Stakeholding Power to ensure Web service quality

E-hubs and infomediaries have the lowest levels of stakeholding power. Perspectives for service intermediaries depend on their ability to respond to the potential to meet and consequently to have influence on the provided quality of service. In conclusion, they depend on the level of stakeholding power and hence their type of intermediation.

Due to the low performance of infomediaries and e-hubs due to their limited stakeholding power, the present work focuses in the remainder on those supply concepts, which have stakeholding power over service and / or traffic (e-markets and integrators). No commonly accepted terminology has yet emerged for those platforms offering Software-as-a-Service in the concept of e-markets and integrators which offer own and / or federate third party SaaS. This thesis applies the terms *service platforms*.

Definition 20: A service platform offers own or third party software as metered on demand services. Software can be partially or completely deployed outside the service platform. The traffic, when a service is consumed passes through the service platform.

2.5 Autonomy and Self-Organization

Self-organization as introduced in Definition 9 describes self-paced alignment of the platform participants, which is not centrally enforced [61]. Ecosystem participants autonomously self-organize towards a more beneficial structure, motivated by their individual pursuits of benefit-maximization. Similarly to the supply-side of a platform, the consumer-side may self-organize. Prerequisite to self-organization are feedback structures for suppliers and consumers (*Definition 9*). Several service platforms analyzed in the explorative analysis included structures that enable self-organization around the platform.

One approach on the supply side is to open up service platforms to services provided by autonomous third parties and thus to benefit from external value creation. Several among the analyzed platform operators chose that approach, e.g., Facebook, Netsuite or Salesforce. Many of the analyzed third party services were of composite nature. As an example, Figure 5 depicts a fraction of a service network from the Salesforce platform Force.com. The figure shows two tiers of an ecosystem for payment processing and storage. It depicts the fraction of tier 2 services that are published on the Salesforce platform. The complete network goes beyond the 2nd layer of service providers. PayPal for instance, a second tier service provider in Figure 5, is interconnected on subsequent tiers e.g., VISA, MasterCard or American Express [79]. Services depicted without links into a second tier – like O2B –did not provide network information.

Centralized handling and management of networked service portfolios like that of Salesforce would be difficult, in view of complexity through loosely coupled service networks, the multitude of service providers, transaction types and short life cycles. The self-organized approach reduces this complexity by giving the ecosystem participants the autonomy to act independently with the potential to cause better adaptability of the service portfolio and a better sensitivity to change (see Section 2.2.2). The endorsement of external services in a self-organized way can hence lead to the creation of networks around service platforms, which could eventually cause network effects. But opening to self-organization implies renouncing degrees of control.

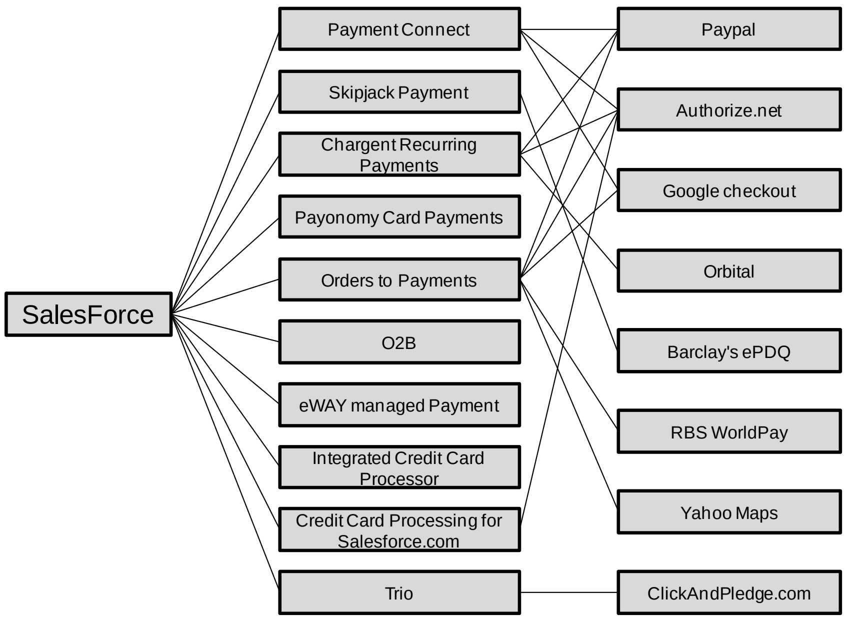


Figure 5: A Network of services around Salesforce

The network of services, described in Figure 5 would look different when considered from the viewpoint of e.g., Paypal. In consequence, a modeling language will need to model a platform from a specific view point, defined prior to starting the modeling process. The present work refers to the entity, whose view point is reflected in a model, as *protagonist*. It could be any kind of company or consortium, operating a service intermediary.

Definition 21: A protagonist is the entity, whose view point is reflected in a model.

The analysis brought to light structures for self-organization on the demand-side e.g., implemented by Dropbox, Salesforce or Trello. These platforms either motivate consumers to contribute (e.g., in Salesforce's AppExchange Platform), or enable consumers to invite additional (non-subscribed) users to mutual collaborative activities on the platform.

Pure self-organization without the platform's ability to manage service quality would be disadvantageous as seen in Section 2.4. All of the above stated platform operators with self-organizing structures therefore also implement service management mechanisms. The following section looks more closely at this mixed approach.

2.6 Managed Self-Organization

The two previous subsections substantiated that self-organization of service providers is a way to pursue fast growth in service provision and to ensure the reactiveness of large service portfolios on evolving consumer requirements. However, they also showed that stakeholding power is required to maintain a defined, appropriate level of quality of service. In result, the graphical modeling language, building the main artifact of this work, needs to include both. This Section builds a conclusion from Sections 2.4 and 2.5 and consolidates the findings into one integrated concept. The present work denominates this integrated concept of self-organization and control *managed self-organization*. Such a concept is a trade-off. The limitation of self-organization implies an increase in control and vice versa. The weighting of both parameters is subject to the decision of the modeler in the design process.

Definition 22: Managed self-organization describes the trade-off between the level of control exerted over service quality and the degree of self-organization.

Service management of high volumes of services and consumers, even though with a strong degree of self-organization is beyond manual manageability. The explorative and longitudinal studies showed that these service management activities need to impact on several stages of service management, i.e. during design time, during deployment time and during operations time. Such a system requires suitable automated or partially automated control mechanisms to be operable at all. In a graphical modeling language, these control mechanisms need to be conceived in a way that they can be allocated at the correct procedural point (e.g., before deployment) and in the right structural position (e.g., within the deployment environment).

The remainder of this subsection looks first at managed self-organization concepts through enforcement (Subsection 2.6.1) and then continues with incentivizing concepts (Subsection 2.6.2).

2.6.1 Managed Self-Organization through Enforcement

This subsection looks at the management of third party services or service providers through enforcement in the context of the service management life-cycle. It then complements this with additional aspects on managed self-organization of consumers through enforcement. Janiesch, Niemann et al. [80] structure the service management cycle in the Internet of services into 5 cyclically repeated steps: service design (embracing conceptual design and development), service deployment, the concurrent steps service delivery and service monitoring as well as service change. The present work groups the concurrent service delivery and monitoring steps into one phase of service operations. To the platform, the phase of service change driven by the service provider is like deployment of a new service together with the potential undeployments of the obsolete version. The present work therefore integrates deployment and change into one phase of deployment and updates. If the platform operator needs to update third party services (e.g., adapt services on new deployment environment), it does it during the operation phase, without interaction with the service provider (Figure 6).

The service platforms, analyzed in the explorative study and during the experiments on the three operative platforms, exert service management during all phases of service life cycle (Figure 6).

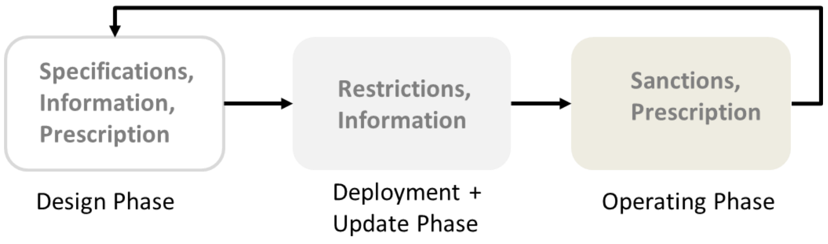

Specifications, Information, Prescription
Restrictions, Information
Sanctions, Prescription
Design Phase
Deployment + Update Phase
Operating Phase

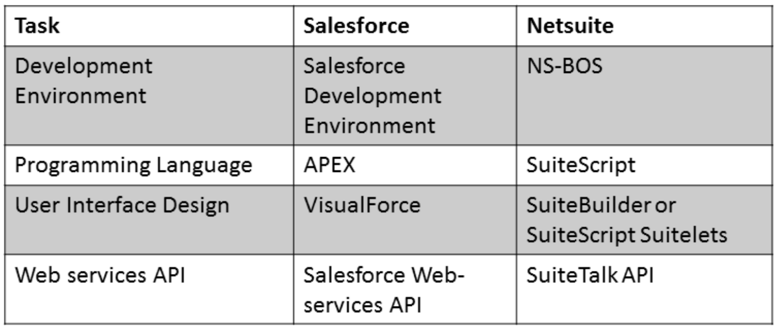
Figure 6: Service management actions during the service life cycle

Design phase:

The conducted studies revealed several approaches of exerting influence on service development already in the design phase. One elicitated approach of steering of service development in the design phase is through communicating a programming specification as service design specification. The platforms following this approach attain authority of enforcement over service design, when they prescribe external developers to develop their service design within a development environment on the platform. The development environment can impose a diversity of features on services including structure, the utilization of predefined building blocks or a specific programming language, as exemplified in Table 5. Limiting the scope of freedom for the external developers allows the platform operator to attain external services of defined minimum quality

(Table 3), fully responding to his design requirement. One important objective of this approach is service manageability (*controllability, observability, management-information offerability*).

As an example, platforms operators like Salesforce.com or Netsuite prescribed service development within their proprietary programming environment, allowing the platform operators to enforce defined service architecture, interface design and interoperability (Table 5). Programming environments make specifications binding and enforceable (*prescription*). As design takes place within the platform's control area, the platform operator can continuously monitor the process and interfere if required.



Task	Salesforce	Netsuite
Development Environment	Salesforce Development Environment	NS-BOS
Programming Language	APEX	SuiteScript
User Interface Design	VisualForce	SuiteBuilder or SuiteScript Suitelets
Web services API	Salesforce Web-services API	SuiteTalk API

Table 5: Development environments of Salesforce.com and Netsuite

Uniform interface design ensures compliance to the platform operators' standards and produces the same look-and-feel for all services in the platform portfolio. Through the prescription of a Web services API, the platform achieves suitable business process quality for internal and external service coupling.

The graphical modeling language, which is the central artifact in this thesis, needs to be able to express the design stage. All involved players as well as their transaction and activities need visualization. Representation needs to be able to differentiate between an offsite approach of a programming specification and an onsite approach of a programming environment, including a prescriptive dimension of service management. Information and motivation as a means to exert influence in the context of service management requires visualization.

Deployment phase:

The studies revealed that service management continues during the deployment phase. Deployment in the analyzed platforms is conditional. Filtering mechanisms restrict the deployment of external services for the accomplishment of the previously described service qualities (Figure 6). The Facebook platform had a fully automatic release process, requiring 10 Facebook friends to test the service as prerequisite for releasing it. At the time of Scholten's experiment on

Force.com, the operator Salesforce estimated the release process as approximately 2 months. This leads to the estimation that the company's release process has also a manual stage.

Similarly to the service management features within the design phase, the graphical modeling language needs to express the restrictive filtering prior to deploying a service. Also the deployment itself requires visualization.

Operating phase:

During operating time, the analyzed platform operators were in a position to continuously monitor service quality, allowing them to adapt, in the worst case undeploy specific services (as done in example by Salesforce or Netsuite). The attained Service Level Measurement Quality enables them to monitor features, which may depend on platform or on the service. Availability or maximum throughputs are typical platform-related features. Depending on consumption of a specific service, the platform needs to be able to replicate it, or to increase respective platform resources. The specified service manageability quality allows services to be adapted, e.g., code optimization on specific hardware or modification of a specific API based on changed platform infrastructure. Platform areas which are subject to network effects may exhibit strong growth with respect to the activities of service deployment or consumption. Those areas consequently need to offer suitable scalability of the deployment and consumption environment.

The graphical modeling language needs to be able to express the platform operator's scope for service management related to this stage, i.e. the prescriptive mechanisms that enable enforcing adaptation of the platform service parameters, if required. It also needs to be able to articulate the sanctional power of the platform operator, being able to undeploy services if underperforming or in violation with given specifications. The areas that exhibit network effects require explicit visualization in the language, as they might require a scalable environment. Also, the consumer groups which are part of the network effects require representation.

In managed self-organizing platforms, platform operators are limited to controlling tier-one services. In a scenario of composite services (Figure 3d), tier-(n+1) services are situated outside the control area and beyond the operator's prescriptive authority. Several of the analyzed service providers took advantage of this limited influence and focused on the development of universal tier-2 service. On the service platforms they provided a simple requestor service, compliant with the platform specific programming specification or environment, which communicated with the more complex tier-2 service. The simplest solutions found were designed with an inline-frame which requests on the universal tier-2 service (e.g., by the provider of file-share services Box Inc.[17]). Thus even the tier-2 UI was reusable.

[17] https://www.box.com/, retrieved 15.02.2013

The graphical modeling language needs to be able to express external service provisioning and a suitable mechanism to control their quality.

Consumer side:

The cycle of service consumption is less extensive than the cycle of service supply. Consumers are enabled to subscribe self-paced and autonomously. They have to follow a subscription process, where the consumer has to accept the terms and conditions prescribed by the platform operator. The platform operator limits access to the platform to those consumers who accept the terms and conditions and who supply the requested deliverables, e.g., name and address or credit card details for payment processing. On the platform, the consumer has access to certain environments, dedicated to consumption of the services, agreed upon in the terms and conditions. If a service consumer does not act in compliance to the agreed terms and conditions (e.g., payments are not released), the platform operator can initiate an escalation routine, with possible steps ranging from an accepted period for amendment to immediate exclusion from the platform. All analyzed platforms followed these steps with varying nuances. The graphical modeling language needs to be able to model these restrictive and sanctional mechanisms.

2.6.2 Managed Self-Organization through Incentives

Apart from management through enforcement, the surveys revealed several approaches of managing self-organization through incentives. One approach is the support of self-organization through consumer-based collaborative feedback mechanisms. The second is the provisioning of information to support self-organization among service providers or service consumers. The last encountered approach is the motivation of service providers or service consumers through rewards. Frey and Oberholzer-Gee [81] differentiate between intrinsic and extrinsic incentives. Provisioning consumer-based collaborative feedback or processed information incentivizes intrinsically. Motivation through rewards impacts extrinsically. Extrinsic motivation is triggered through monetary or non-monetary rewards. The collaborative feedback mechanisms discovered through the survey were only applied for service consumers. The two other mechanisms found application for service suppliers and service consumers.

The encountered collaborative feedback mechanisms can be divided into collaborative sanctioning systems (reputation systems) and collaborative filtering systems (recommender systems). Collaborative sanctioning penalizes services of poor quality and highlights perceived quality of service [82, 83]. The feedback is incentivizing to users as it creates a trust-basis for a service decision. Examples for elements of reputation systems in service platforms are service rating as done on Force.com, or social plug-ins such as like buttons, comment boxes or recommendation boxes on the Facebook platform[18]. Recommender systems are more focused on specific communities and are based on the assumption that tastes and preferences are specific per community [82]. Examples for recommending mechanisms are e.g., send buttons, follow buttons, activity feeds or facepiles [84]. Recommender systems leverage on the inherent trust relationship of social networks [85]. A graphical modeling language needs to be able to model such collaborative feedback mechanisms.

Platform operators also provisioned preprocessed information to their service providers or consumers. The platform operator can generate this information explicitly at the request of consumer feedback, or implicitly.

Implicit feedback is based on observing the consumer's reaction in the service delivery process. The platform operator can gain feedback in the process of offering, searching, and delivering a service, as well as after delivering a service. He can aggregate behavioral reactions to implicit feedback on consumer dissatisfaction [86] e.g., consumers migrating to other services or consumer complaints. Claypool, Le et al. [87] validate implicit interest indicators experimentally. They detect e.g., reading time and scrolling behavior as predictors for overall explicit ratings of Web page content. Fox, Karnawat et al. [88] reveal that certain implicit feedback information, i.e. click-through time and exit type of a Web search session. The platform operator Netsuite provides consumption information to key service providers. Several platforms express suggestions to service consumers based on consumption behavior by consumers with similar consumption behavior. In pursuit of engineering a graphical modeling language able to express the management of services and its providers and consumers this provisioning of preprocessed information must also be depicted when needed.

The last encountered incentivizing approach is the motivation of consumers of service providers towards a specific action. Frey and Oberholzer-Gee [81] differentiate between extrinsic and intrinsic motivation. Whereas information motivates intrinsically, motivational control works extrinsically. Extrinsic motivation is triggered through monetary or non-monetary rewards. Several among the analyzed platforms apply motivational control through monetary incentives. Platform operator use seed funding to support service development in line with the platform operators' goals, e.g., Facebook's fb fund[19]. Other platform operators create financial incentives through initial cost-free or cost-reduced subscription periods. On the supplier side, the platform operator can exert extrinsic motivational control e.g., through the provision of free-of-charge SDKs and thus reduce entrance barriers to potential developers. Further, extrinsic motivation can be accomplished though subsidized access to the service ecosystem, open license models for

[18] http://developers.facebook.com/docs/plugins/, retrieved 08.02.2013
[19] https://www.facebook.com/fbFund

specific code to facilitate service contribution. The backup service platform Dropbox [89] gives free storage space for additionally acquired customers. In a general analysis of software industry, Shapiro and Varian [56] suggest free versions of digital goods as suitable way to attain critical mass with respect to network effects. The disadvantage of extrinsic motivation in contrast to intrinsic incentives is that it is not cost-neutral. The modeling language also needs to represent this motivational approach to managed self-organization.

2.7 Base Value and Critical Mass

The conducted explorative study shows that successful platforms benefit from a suitably high quantity of managed services supplied by self-organized service and from high quantities of subscribed users. The unsuccessful ones as analyzed in the longitudinal analysis did not accomplish self-enforcing network effects. The present work refers to the value propositions that are expected to incite network effects as *base value*. Base values need to exceed a minimal threshold called *critical mass*. The concept of a critical mass receives theoretical substantiation in Section 3.2.

Definition 23: The base value of a service platform is the value proposition, offered by a platform operator to ecosystem participants to incite network effects.

Definition 24: The critical mass describes the threshold required for a base value to incite a network effect.

The base value could be a static contribution from the platform operator, e.g., Salesforce.com's CRM software. In this case, the value would be a *close to static* stock due to the limited team of contributors. Close to static means that the content grows organically. That is through the own development of the internal team in charge. Other base values depend on activities from ecosystem partners, i.e. Salesforce.com's service providers, which are motivated to contribute to the Force.com platform due to the large number of consumers that actively subscribed to the platform. If successfully implemented, these base values grow through the activated network effect.

Attractiveness to attain a sufficient amount of consumers depends on the platform's potential to offer a sufficient number of services of appropriate quality[20]. This capability is limited to

[20] The interdependence of attractiveness, quantity of services deployed and quantity of customers subscribed is formulated and explained as quantitative differential equation in the subsequent foundations-chapter (Section 3.2).

e-markets and integrators. None of the intermediaries analyzed in the longitudinal study succeeded. All of them were discontinued (or fundamentally changed in concept) due to the lack of critical mass of consumers.

The most successful SaaS-platforms such as Salesforce.com, Intuit, Appirio, Long Jump, Integrify or Netsuite [4] have integrator structures. Most of them started acquiring consumers with a developed critical mass of proprietary services. Only in a second step, a subset of them opened up to third party services and / or created social networks among their existing consumer base. They used the subscribed consumer base as critical mass to incite network effects.

Although the successful examples described above are all integrators. However, the e-market approach will also remain in the scope of the planned modeling language. First, this is because some analyzed platform operators followed a mixed approach including e-market designs (e.g., SAP's App Store). Second, this is in view of W3C's linked data vision to improve manageability of distributed application integration [90-92]. Thirdly, because of ongoing research in the fields of Social Cloud building on virtualized resources contributed by users which are made available to friends in the context of a social network [85, 93]. Aware of the shortcomings in manageability of external resources the researchers suggest (a) compensating by using prescribed and parsed service designs and sandboxing, (b) lessening the risk of malicious action by encryption of data exposed to misuse and (c) reducing the risk of loss or corruption of files through redundancy, achieved by the utilization of several providers for the same service provisioning in parallel

One additional particularity, revealed in the surveys is that the duality between service providers and service consumers becomes indistinct. Both can provide value or be attracted by value. Both may also provide services. As an example, the platform operator Salesforce offers a market place, where service consumers can offer their add-on services to other consumers. Experimental platforms like SOAlive support the development, deployment and management of services for those social and business communities, desiring to exchange services [94, 95]. Social Cloud projects like the research e-markets BOINC[21] assemble for research projects like Milkey@Home more than 300.000 computers from approximately 150.000 participating users as virtual resources[22]. The graphical modeling language needs to be able to handle the increasingly dual roles of the ecosystem participants.

In conclusion, the graphical modeling language needs to be able to correctly abstract e-markets and integrators. It also needs to express the base value that is intended to incite a network effect. Ideally, the model should allow for differentiation between a static or growing base value. The model should also be able to handle duality or changing roles of value provider and consumer.

[21] Berkley Open Infrastructure For Network Computing (BOINC), website: boinc.berkley.edu
[22] Source: http://boincstats.com/en/stats/61/project/detail, retrieved 23.03.2013.

2.8 Research Requirements

The previous sub-chapters derive three fundamental factors that need to be addressed in the graphical modeling language: Structural elements, process elements and the allocation of service management mechanisms.

Requirements related to structural elements (RQ1)

Section 2.3 revealed that areas of staged stakeholding power (Figure 3) play an important role to the platform operator. These areas determine where and to what extend the platform operator can intervene in value creating processes and where he can interfere directly with technology. The language needs to be able to represent these areas as elements, structuring platforms and their environment. Its grammar needs to be able to express the resulting implications to service management. Further the language needs to capture areas on the platform, which can be subject to network effects, as those areas need to be scalable. It also requires depicting finite quantities of areas of similar full stakeholding power (Section 2.6).

Requirements related to process elements (RQ2)

The present problem identification exposes process elements of relevance, which the language needs to formalize. Section 2.5 showed that the sources and targets of any relationship can be external players, internal entities, or interactivity between diverse participants. The thesis gives the example of service providers or consumers as external players. Examples for elements within platforms are the departments which contribute value e.g., the Salesforce's Relationship Management product, discussed in Section 2.6. The language requires an element to represent participants, which is robust to potential role changes over time between those of value consumer or provider. The language further needs to depict base value contribution, as done e.g., by Salesforce.com's CRM software in the start phase of the platform (Section 2.7). The language needs to find a concept on how to handle players outside the platform's influence, e.g., competitors in Figure 3. Further, the present work repetitively emphasizes that network effects are originating from more implicit relationships, where cooperating participants are not distinct but are rather part of an unspecific group of players. Those groups require representation. Another element type, which finds repeated discussion in chapter 2 and which requires representation are the activities, e.g., service deployment, subscription, social networking or the exchange of applications as in the example of Salesforce's AppExchange.

The consideration on managed self-organization in Section 2.6 highlights the requirement to model transactions from ecosystem players and the exertion of influence on these transactions by the platform operator or any other player. The platform needs to be able to model the interplay of these relationships. This includes the modeling of centrally controlled process elements as well as self-organized processes in the ecosystem, leading to network effects.

Given that self-organization is driven by groups of ecosystem players, the language needs to be able to express this in representation and grammar

Requirements related to service management (RQ3)

Chapter 2 revealed a set of mechanisms for service management, used in the context of *managed self-organization.* In particular, Section 2.6 identifies enforcing mechanisms like prescription, restriction and sanctioning, as well as incentivizing mechanisms for self-organization in the ecosystem and on the platform, through consumer-driven feedback, information and motivation. The language needs to depict these mechanisms. Its grammar needs to be aware, which mechanism can be applied in which context.

The research requirements lead to the formulation of the research hypotheses in Section 2.9. Table 6 summarizes the research requirements.

Perspectives	**Requirements**
Structure	RQ1: Representation of • All areas of staged stakeholding power • Areas that need to scale, • Finite areas of similar behavior
Process	RQ2: Representation of: • (unspecific) players in/outside the ecosystem and elements within the platform • Activities (e.g. deployment, subscription, social networking, exchange of applications) • Transactions • Influence Representation of • the interplay of relationships, leading to network effects
Service Management	RQ3: Abstraction of managed self-organization: • Enforcing mechanisms (prescription, restriction, sanctioning) • Incentivizing mechanisms (consumer-based feedback, information, motivation) leading to external and internal self-organization and network effects

Table 6: Overview of research requirements

2.9 Research Hypotheses

The overall question, defining this work is:

How can a modeling language support the design of service platforms, targeted at harnessing network effects?

In response to this question, the present work formulates the following main hypothesis and three related subhypotheses.

Main Hypothesis MH: *A modeling language targeted at representing network effects around service platforms supports platform design through guided modeling, in particular by providing structural elements, process elements and control mechanisms.*

The main hypothesis addresses all aspects of platform design and service management claimed in the research requirements. The language and its underlying grammar are able to provide a seamless answer on the different research requirements, enabling modelers to produce well-formed platform models of sufficient level of detail. These models allow for subsequent analysis.

The subsequent three sub-hypotheses break down the main hypothesis, shedding light on the major directions, set through the research requirements RQ1 – RQ3.

Sub-hypothesis H1: *A language providing the structural perspective of areas of staged authority and structural divisions enables better structuring and improved exploitation of stakeholding power.*

The present work provides concepts on where to position ecosystem participants and interaction with the platform operator, in view of the platform operator's desired stakeholding power. Building on that, the thesis defines structural modeling elements for the control area, the influence area and the noise area. It also provides subdividing elements for the control area in order to enhance clarity and to express scalability of technical environments and areas of similar structure. It embeds the resulting structuring elements in a seamless grammar.

Sub-hypothesis H2: *A language providing the procedural perspective of interrelated specific and unspecific participants as well as of their interaction in and with the platform allows for modeling improved causal loops and related processes.*

This hypothesis requires general consideration of network effects in the context of service platforms. Considering network effects also necessitates an understanding of the originators of value contributions in and around service platform in the context of the processes that drive these network effects. Related research outcomes embrace a conceptualization of these processes as well as their grammatical representation.

Sub-hypothesis H3: *A language that adds control mechanisms onto its elements, allocating managed self-organization, can model options to turn causal loops into network effects.*

The present work provides concepts on how to exploit or strengthen latent network effects and on how to create new ones. For this purpose, the thesis introduces control mechanisms and explains when and in which context to place them onto the procedural elements. The thesis further integrates these control mechanisms into the language's overall grammar.

The present work validates the hypothesis as follows: It begins with a conceptualization of structures, processes and control mechanisms as claimed in the subhypotheses H1-H3. The suggested terminology and abstractions are able to represent multiple relationships in and around platforms, leading to network effects as well as options to manipulate them. Building on these abstractions, the thesis develops the Dynamic Network Notation, a graphical modeling language, which seamlessly integrates the above described conceptualizations according to subhypotheses H1-H3 into one grammar and related semantics. Both represent platforms, platform ecosystems and their competitive environment as well as immanent network effects, while allocating categories of control mechanisms to manipulate them. The thesis endows the Dynamic Network Notation with building blocks of structured experience, formalized as pattern language and embedded in a framework to develop and evolve a pattern repository. The thesis instantiates the grammar within and editor, complemented with an analysis environment, subsequently guiding the modeler towards more efficient solutions. The validation closes with a sequence of field studies confirming well-formedness and expressiveness of the Dynamic Network Notation and its editor as well as its ability to guide to quality models.

3 Related Theory

The present work's major deliverable is a graphical modeling language for service management around platforms and their ecosystems in the context of network effects. This chapter creates foundations on language engineering and network dynamics in the context of service platforms.

Section 3.1 introduces language engineering. It first builds the necessary foundation on language and grammar (3.1.1 and 3.1.2), then introduces the groundwork for the following graphical language engineering (3.1.3). The latter subsection suggests a model stack for graphical language engineering, motivated by model-driven development. This stack allows creating and maintaining clarity throughout the present work, as it provides a clear demarcation and allocation of the various abstraction levels needed in this thesis. The second part of this chapter gives a detailed introduction into network effects around service platforms. The theory presented originates from the disciplines of system theory and dynamic markets (Section 3.2). The chapter closes with an introduction into control theory (Section 3.3).

3.1 Language Engineering

The major artifact produced within this thesis is a graphical modeling language. The following subsection lays the foundations with *language* and *grammar.* These are the basic pillars of language engineering. It proceeds with an introduction into more specific graphical language engineering. Both Sections are used as groundwork for the deduction and description of the engineering concepts of the dynamic network notation, as they provide necessary theory and terminology.

Information technology has been a focal sphere of activity for language engineering. This is even intensifying since the emergence of the Internet. There is on the one hand a broad basis of *generic* languages such as. XML or Java, which allow for nonspecific applications, detached from any subject matter or area. On the other hand there is an increasing requirement for domain specific languages (DSL) such as the Web Service Description Language (WSDL), designed for an efficient description of web services. The term *domain* stands for specific technical or application purpose. Business Process Model and Notation (BPMN) and the Business Process Execution Language (BPEL) are languages, engineered to effectively design and execute business processes. The Markup language used for interchange of data between advertising systems (AdsML) is specifically addressing the data exchange needs of newspapers or advertising agencies. Subject of this thesis is a domain-specific language for service management.

The art of language engineering lies in the choice of the right abstraction level that suitably simplifies reality to create the right expressiveness all while preserving the necessary minimum attention to detail. Its theory is subject of the following two subchapters. A subsection on language and grammar opens the introduction into language engineering (3.1.1), followed by an explanation of the concept of generative grammars (3.1.2). The subsequent subsection 3.1.3 immerges into the area of graphical language engineering

Much of the theory in software language engineering originates from the academic discipline of natural language research (linguistic) further adapted to the purposes of software language engineering. This section uses theory from both disciplines. It builds on key concepts and accuracy of classical language engineering theory (e.g., context-free grammars) and benefits from recent research in the fields of software language engineering. General terminology of language engineering sometimes causes inadequacies, when applied to the context of Information Technology. Linguistic speaks of *sentences* as elements of a language. The term sentence however describes well a sequence of words in a natural language but appears inappropriate when talking about mathematical languages or software languages. In mathematical contexts, the term *statement* might be suitable. In IT language engineering *model*, *program* or *artifact* might be suitable – always depending on the context.

Kleppe [96] suggests the all-embracing coinage *linguistic utterance* or alternatively the fusion word *mogram* (being made-up of model and program). For the sake of clarity and general applicability, this thesis uses the word *utterance* in definitions originating from linguistics. In the context of specific graphical language engineering, the present work utilizes the word *model*. For an utterance of the dynamic network notation it uses the term *Dyno model*.

3.1.1 Language and Grammar

The basics in language engineering go back to the US-American Scientist Noam Chomsky. In the 1950s, he laid down the foundations for language engineering including software langauge engineering in his search for a simple grammar that enabled the generation of all sentences of the English language. The following definitions are based on Chomsky [97] and complemented by the attribute of *well-formedness* as measure of grammatical correctness [98]:

Definition 25: A language L is constituted of a (finite or infinite) set of utterances of finite length, constructed from a finite alphabet of symbols through a grammar.

Definition 26: A grammar G is constituted through a finite set of production rules, describing how the elements of the languages alphabet are concatenated. An utterance that conforms to such production rules is called well-formed.

These production rules are the core of language engineering. They define how utterances (referred to as variables or non-terminals) are produced as a string of terminals from the language's alphabet [96, 97]. Kleppe's [96] example of a simple grammar helps to visualize the concept of production rules. The grammar specifies a part of the English language. Note it is just a fraction of the grammar which specifies the English language. It defines the structure of sentences S, noun phrases NP and verb phrases VP. The angular brackets […] state optional elements, the arrow → can be read as *is produced by*.

The grammar can be formulated as:

S → NP VP (3.1)

NP → [article] [adjective] noun (3.2)

VP → [verb] verb (3.3)

The utterance "The wonderful Dynamic Network Notation is convincing" for example has the structure *article adjective noun verb verb* in compliance to rules (3.1), (3.2) and (3.3); it is grammatically correct. The term *Dynamic Network Notation* only complies, as it is one proper pre-defined noun. The opposite case is worth a consideration as it shows the limits of this simple grammar: considering *Dynamic Network Notation* as independent words would turn it into one adjective and two nouns, leading to the following incompliant phrase structure:

article adjective adjective noun noun verb verb.

In IT-literature, the word *grammar* is often used synonymously with *syntax*. These words however are not synonyms. To cater for a precise application of terminology, this Section provides disambiguation. Redefining grammar as a function of syntax leads to:

Definition 27: Grammar is a set of rules for languages, consisting of three segments: morphology, syntax and phonology. Morphology prescribes the representation of linguistic units in a language (e.g., of words). Syntax defines the assembly rules of utterances through linguistic units. Further, Syntax determines adaptation of specific representation of linguistic units. Phonology defines the organization of sounds within utterances.

The following utterance serves to illustrate the disambiguation:

Dyno produces graphical models.

The utterance is produced with the English grammar. The morphology provides the words *Dyno, produce*, *graphical* and *model*. The syntax takes care of the production of utterance and case specific adaptation of the words *produce* and *model*. Phonology, the organization of sounds, plays a subordinate role in IT language engineering although it allows for correct pronunciation of language during oral discussions. In this thesis, phonology will be added wherever required. In the above case, it would explain how to pronounce the phase properly. In the standardized phonetic language, the word Dyno would be depicted as *daɪ.nə*. Phonology also includes the description of the language rhythm and accentuation.

The above linguistic rules are able to create correct utterances. However to assign a meaning to those utterances, semantics is required. The present work expresses semantics in natural language and in a non-formalized way.

Definition 28: Semantics assigns a meaning to elements of an utterance (symbols, words) and to the utterance the as a whole.

The formalisms to describe how production rules formulate a language are called meta-language.

Definition 29: The term meta-language defines languages conceived to describe other natural or artificial languages with the help of specific terminology and symbols.

The following subsection applies a simple meta-language that was used by Chomsky [97]. In the course of the thesis, more meta-languages i.e. the Extented Backus Naur Form (EBNF) or the Unified Modeling Language (UML) find exemplification.

3.1.2 Generative Grammars

Chomsky's research in the 1950s [97, 99] deserves consideration as it provided important foundations for language engineering. His process of research further illustrates the mightiness of transformational grammar as compared to other concepts of grammar. The concept of transformational grammar will be introduced / derived hereafter. Chomsky investigated on 3 conceptions for grammars for language engineering: (a) Finite State Markov Process (b) Phrase Structure and (c) Generative Grammar.

(a) Finite State Markov Process:

The Finite State Grammar G is a system of

- a finite number of states $S_o, \dots, S_q$, (3.4)
- a set of transition symbols $A = \{a_{ijk} | 0 \leq i,\ j \leq q;\ l \leq k \leq N_{ij} for\ each\ i, j\}$ (3.5)
- a set of pairs of states C, said to be connected = $\{(S_i, S_j)\}$ (3.6)

Moving from state Si to state Sj, the system produces the symbol $a_{ijk} \epsilon\ A$.

The initial sequence of states is

$$S_{\lambda_1}, \ldots, S_{\lambda_m} \mid \lambda_1 = \lambda_m = 0; (S_{\lambda_i}, S_{\lambda_{i+1}}) \in C \; for \; each \; i < m \,. \tag{3.7}$$

When moving from state $S_{\lambda_i}$to $S_{\lambda_{i+1}}$ the grammar G produces concatenations of a_{ijk} from appropriate choices of $k_i \leq N_{\lambda_i \lambda_{i+1}}$, which represent correct utterances of the form:

$$a_{\lambda_1 \lambda_2 k_1} \| a_{\lambda_2 \lambda_3 k_2} \| \ldots \| a_{\lambda_{m-l} \lambda_m k_{m-l}} \,. \tag{3.8}$$

The totality of utterances, produced by this rule-set is called language L_G generated by G.

The language is very limited and failed due to this in experiments to produce natural languages i.e. English (Chomsky 1956). Better results were generated with a phrase structure.

(b) Phrase Structure Grammar:

Still being a finite state process, phrase structure grammars are able to produce realistic results, when limited to a small subset of simple languages. A Phrase Structure Grammar, also called (∑, F)-grammar, consists of

- a finite vocabulary V_p, (3.9)
- a finite set of initial strings in V_p depicted as ∑, (3.10)
- a finite set of rules that rewrite x through y, depicted as $F = \{ x \rightarrow y \mid (x,y) \in V_p \}$. (3.11)

The resulting language L is a set of strings derived from the (∑, F)-grammar. It is consequently called a derivable language. However, the (∑, F)-grammar was still producing inadequate derivations and required additional refinement.

(c) Generative Grammar:

Chomsky [97] generated the best results when going one step further through transformational grammars, in later publications referred to as generative grammars. Those build on a limited kernel of simple utterances, complemented with a set of grammatical transformation rules, transferring one correct utterance or fraction of an utterance to a new correct one. In successive publications, Chomsky derives a hierarchy of generative grammars, consisting of a series of 4 groups of types with increasing expressive power, summarized in the following definitions. In the illustrating examples the symbols x, y, z, describe terminals, A and B stand for non-terminals and w groups (possibly empty) strings of terminals [96, 99]:

Definition 30: Recursively enumerable grammars (Type 0) have no restriction at all in the production rules; e.g., xxAByz → zzz.

Definition 31: In Context-sensitive grammars (Type 1), the number of elements in the string on the left-hand side must be smaller or equal to the number of elements on the right side; e.g., xAzB → AAxyzBBB .

Definition 32: In Context-free grammars (Type 2) all production rules produce a single non-terminal on the left hand side; e.g., A → AzyB. Context free grammars can be applied when describing recursive language structures (nesting).

Definition 33: Regular grammars (Type 3): All production rules produce the form A → wB or A → Bw. These grammars can be applied on natural languages which do not require nesting.

The definitions show that all grammars include relevant higher type grammars

$$\{\text{Grammar of Type}_{i+1}\} \in \text{Grammar Type}_i \,. \tag{3.12}$$

Prominent examples of meta-languages used to describe software grammars are the *Backus-Naur Form (BNF)* and the ISO-standardized *Extented Backus Naur Form* (EBNF). One of their advantages (apart from a clear and comprehensive design) is their ability to describe context-free grammars, meaning recursive structures. The following Backus-Naur grammar describes a recursive language, expressing multiplication and addition.

*<expr> ::= <number> | (<expr> * <expr>) | (<expr> + <expr>)*

In the example, the non-terminal <expr> is recursively produced by terminals and the non-terminal <expr>. The above used grammars are all textual grammars.

Definition 34: Textual grammars build languages through linear catenation of text.

An important limitation to the expressiveness of such textual grammars is their linear structure. Such linear conception makes it difficult to describe complex interdependencies. Overcoming the limitation of linearity requires the step from textual grammars to graphical modeling. The languages modeled graphically do not necessarily need to be graphical. On the other hand, textual languages such as *Pascal* or *XML* could well be modeled graphically as well.

Definition 35: Graphical grammars are a special form of generative grammars which describe the elements of a language and their relationships graphically.

The remainder of the thesis focuses languages, dealing with the application of information technology. Although the term *software language engineering* is established in academia, this thesis uses the word *IT language engineering* to include aspects such as hardware or user ecosystems.

3.1.3 Graphical Language Engineering

The title term of this sub-chapter *graphical language engineering* requires clarification. It could mean *the engineering of graphical languages* as well as the *graphical engineering of language*. In this thesis both connotations apply as it will produce a graphically engineered graphical language. Whenever necessary to focus on one of these specific meanings, the text provides clarification.

The following three definitions introduce the necessary terminology:

Definition 36: A language is graphically engineered with one or more graphical grammars, optionally complemented by textual grammars. One or more graphical meta-languages complemented with optional textual meta-languages document the grammar.

Definition 37: Graphical languages are a special form of artificial languages which describe their elements and relationships graphically.

Definition 38: Graphical meta-languages are special versions of meta-languages expressed as graphical language.

This thesis uses – similarly to publications in the disciplines of computer science - the terms *graphical* versus *textual representation* of data. Publications of more psychological orientation use the terms diagrammatic versus sentential [100, 101].

Examples of graphical languages are the Business Process Modeling Notation [102] or Ladder Logic [103]. A prominent example of such graphical meta-languages is the Unified Modeling Language (UML) [104]. Graphical languages benefit when compared to textual languages from the advantage of non-linear expressiveness. Through visual representation, complex interdependencies can be expressed more easily [96]. In general, graphical representations support the human perception process when positioning elements relative to each other with respect to structural affiliation or graphical order of importance (expressed through retinal techniques, i.e. color, shape, size, saturation, texture). From such explicit representation, the user’s brain is able to draw conclusions through simple and direct perceptual operations. Publications in the fields of psychology denominate the human capability to infer on something based on a visual stimulus *perceptual inference*. Graphical encoding, which focuses on the message to be conveyed and which is adapted on the target group can amplify the overall cognition process [14, 15, 100, 101, 105, 106].

Part B realizes information visualization based on theory of graphical encoding.

An utterance of a graphical language is a collection of tuples:

$$u \subset \{(o, l) | o\epsilon O \wedge l\epsilon L\} \tag{3.13}$$

with O being a set of graphical objects and L a set of locations [15].

The goal of graphical languages is to correctly express a set of relations and their structural properties, the latter including the domain sets (the various entities involved) and their functional properties. A fact is expressible in a graphical language, if it contains at least one utterance that encodes all facts in the set and not more. This means, when modeling a specific set of information, the language must not provide additional data which is not defined in the set. The effectiveness of languages describes how well the expressed facts can be read. This however also depends on the perceiver [15]. In a logical consequence, languages should be designed to meet a specific target group's way of thinking. Established graphical languages targeting the same group might give guidance.

To improve *effectiveness* of a grammar, Mackinlay [15] suggests the principle of *importance ordering*, requiring more important information to be presented more predominantly. Lastly, *efficiency* depends on the composition of a graphic utterance. Composition efficiency however depends strongly on the modeler. Procedural models or tool-based guidance of the modeler can help improving efficiency.

The following definitions summarize the above:

Definition 39: The expressiveness of a graphical language describes its ability to express desired information in a semantically and syntactically correct way.

Definition 40: The effectiveness of a language describes how well a language can express information with respect to a specific target group.

Definition 41: The efficiency of an utterance describes how well it is modeled.

The remainder of this chapter deepens syntax, morphology and semantics in the particular case of graphical language engineering. In definitions given by Clark, Evans et al. [107] and André [108] a software language 'consists of models for concrete syntax, abstract syntax and for the semantic domain'. This definition shows 3 deviations from the natural language oriented definition 27. The terms morphology and phonology do not find further consideration. On the other hand, the definition introduces a differentiation between abstract and concrete syntax. All compared sources synonymously define abstract syntax as set of production rules of an utterance without details on concrete representation [96, 107-109].

Originating from compiler theory, abstract syntax gives a high-level description of programs. The abstract syntax in that context describes a tree structure with respect to statement, expression and identifier [109].

Definition 42: Abstract syntax gives a high-level description of syntax, leaving out the specific technical implementation.

However, standard specifications for languages such as BPMN also include graphical representation of nodes and edges. The language engineered in this thesis shall have a distinct representation (morphology) regardless of a technical implementation. The present work therefore applies a more inclusive definition of a grammar on graphical language engineering, leading to:

Definition 43: Abstract morphology prescribes the representation of graphical elements, leaving out the specific technical implementation.

Definition 44: Abstract grammar gives a high-level description of a grammar, using abstract syntax and abstract morphology, but leaving out the specific technical implementation.

Following this line of thinking and again looking at the original definition in compiler theory, the source syntax (called concrete syntax in language engineering) gives the actual representation and the machine oriented encoding [109]. Likewise, Kleppe [96] defines concrete syntax of a graphical language as a syntax, specific to the modeling environment. The present work adopts this interpretation, which is more restrictive than definitions by Clark, Evans et al.[107] or André [108] which do not specify the restriction of specificity to the modeling environment. The enhanced precision will avoid ambiguity in the following chapters, i.e. when speaking about grammar definition, grammar instantiation and platform modeling.

Definition 45: Concrete syntax in graphical languages is a set of production rules specific to the technical implementation.

Definition 46: Concrete morphology in graphical languages prescribes the representation of graphical elements specific to the technical implementation.

Definition 47: Concrete grammar in graphical languages is a set of rules consisting of concrete syntax and concrete morphology.

Similar to other software language engineering efforts e.g., BMPN, this thesis does not give specific attention to phonology. The word *model* appears in the remainder of this thesis in various contexts. For the sake of differentiation, but also for the sake of illustrating the interdependence of the different levels of models, the chapter suggests a stack motivated by model driven development [110]. Table 7 describes the model stack applied on Graphical Language Engineering.

Level	Model	Graphical Language Engineering
M3	Meta-Meta-Model. Abstract level to define M2	UML model for the meta-modeling language applied in M2.
M2	Meta-Model. Defines the structure of a model, i.e. classes, attributes and associations.	Abstract Grammar. Distinct production rules with respect to Syntax and Morphology of a language but agnostic to the technical implementation
M1	Model. Specific instantiation of the meta-model from level M2	Concrete Grammar. Distinct production rules with respect to Syntax and Morphology of a language and specific to the technical implementation
	Transformation M1 ↔ M0	Transformation M1 ↔ M0
M0	A concrete object	A concrete model, expressed through the graphical language

Table 7: Stack for meta-modeling applied on graphical language engineering

Level M0:

On the lowest abstraction level, this chapter starts with the concrete object and moves step by step into higher levels of abstraction. In graphical language engineering, the *graphical language models* are the concrete artifact; they are the real-life object to be produced during the modeling process. The finite set of well-formed utterances of a grammar (loaded with the specific semantics) makes up a language. This object layer is called M0. In the context of Dyno, the concrete artifacts are service networks.

Level M1:

In IT languages, an editor generates the concrete object from the concrete grammar, placed on the subsequent layer M1. Part B and C of this work exemplify a set of such models. M1 describes the concrete grammar of a language, specific to the technical implementation. In the context of this thesis, the technical implementation is specific to the chosen ORYX framework. ORYX serves as modeling environment and repository for the Dynamic Network Notation. Chapter 6 further elaborates on an instantiation of Dyno within ORYX.

Level M2:

This level accommodates a meta-model, defining the abstract syntax of a language and the abstract morphology giving the visual representation. Both are necessary and sufficient to complete a graphical language as they describe the production rules and the representation. Therefore, specifications of languages like BPMN also include both. This thesis applies the graphical meta-language Unified Modeling Language UML [104] to produce the abstract syntax model for the graphical language, complemented through the Object constraint language OCL. OCL is a supporting textual meta-language. It specifies also how to adapt graphical representation in design-time and in function of context-specific syntactical requirements (chapter 5). Graphics and describing text specify the generic representation of all objects, given by the abstract morphology. Chapter 5 provides more detail on this.

Level M3:

The meta-languages used to define the meta-model sit on level M3. In the specific context of this thesis, they define UML and OCL. Layer M3 is not further considered in this thesis. Details are provided by the respective specification documents of the standardization bodies [104, 111].

In some languages, transformation between M2 and M1 can be automated. This thesis choses a manual approach for creating a concrete grammar from an abstract syntax. Specific for language engineering, a modeling environment accomplishes the translation from a concrete grammar (level M1) into a concrete object (see chapter 6 for details).

3.2 Network Effects

Central to the modeling language developed within this thesis is the theory of network effects. Therefore, the present section focuses at an explanation of network effects. The present work adapts and applies the related theory on service platforms. The section ends with a discussion on why currently existing theory and the modeling language originating from business dynamics and system theory are not sufficient to produce models that support platform design oriented on the exploitation of network effects. The term *network effect* originates from system dynamics. System dynamics is macroscopic system behavior over time as described in system theory. Business dynamics are economic systems which exhibit this behavior.

Definition 48: System dynamics describe macroscopic system behavior over time, built through interaction of sources and sinks, stocks, flows and feedback loops

A traditional way to depict system behavior in the discipline of business dynamics is the *Stock and Flow Diagramming Notation* [112], as depicted in Figure 7. It is shaped through four elements: *Sources* or *sinks*, *stocks*, *flows* and *auxiliary variables*. Sources or Sinks represent the origin and final destination of any supply and are depicted as a cloud. In system dynamics, they are assumed to have infinite capacity, meaning they can never be fully depleted or filled. When being a source, it causes an inflow into a system, when being a sink, it is the destination of an outflow. Stocks describe the storage capability of a system. They are depicted through a rectangle. A stock can accumulate or deplete over time. It gives memory and inertia to a system. Flows change the stock over a certain period of time. They can be inflows, filling the stock, or outflows, emptying it. Auxiliary variables can influence the flow. They act like a regulator to the flow. Auxiliary variables may be constant or variable exogenous inputs (meaning they originate from a source outside the modeling focus). They may be also functions of stocks. Last, they may be a result of cascaded influence from exogenous inputs or stocks within the system. The present work speaks of *second level variables*, when those influence auxiliary variables and not flows. Their toeholds are depicted through the symbol of a valve, implying the regulation of the amount of flow, entering the stock.

Definition 49: Stocks describe a vessel which can accumulate or deplete.

Definition 50: Flows circumscribe the inflow into or outflow from a stock.

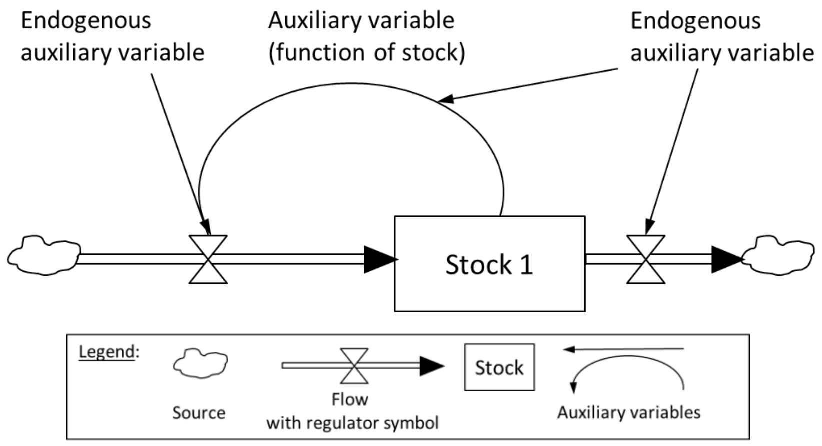

Endogenous auxiliary variable
Auxiliary variable (function of stock)
Endogenous auxiliary variable
Stock 1
Legend:
Source
Flow with regulator symbol
Stock
Auxiliary variables

Figure 7: Stocks, flows, auxiliary variables and causal loops,
depicted in the Stock and Flow Diagramming Notation (Forrester 1961)

Flows can be positive or negative. In example, the outflow from stock 1 (Figure 7), could be modeled as inflow from the right cloud into stock 1. In that case, the flow would have a negative sign. The role of a cloud as source or sink may vary over time. The present work brackets both as *source*.

Definition 51: Sources are stocks outside the boundary of the model with assumed infinite capacity.

Definition 52: Auxiliary variables regulate the flow. They can be cascaded. Their origins may be exogenous or functions of stocks.

Figure 7 simulates a basic interplay of stocks, flows and auxiliary variables in the *stock and flow diagramming notation* [112]. *Feedback loops* describe the reciprocity that a stock has on its own filling or depletion, when impacting on the flow through an auxiliary variable. The loop in Figure 7 consists of the stock as origin, the auxiliary variable which is a function of the stock, and the flow filling or depleting the stock, whose magnitude is regulated by the auxiliary variable (Sterman 2000).The present work refers to the feedback loops of auxiliary variables that are functions of stocks and which impact flows as *causal loops*.

Definition 53: Causal loops are feedback loops, where the magnitude of a stock amplifies the flow by a certain factor, which in reciprocity increases the stock again.

Service platforms with consumers on the one side and service providers on the other side comply with the definition of 2-sided markets. 2-sided markets may include various alternatives for causal loops. Those loops can be *same-sided* e.g., *demand-sided* or *supply-sided.* They can also be *cross-sided*, involving both sides of the platform [20, 22, 113].

The present subsection uses the example of a two sided platform to derive and illustrate network effects. In explicit, it considers network effects around the subscription base of consumers in a platform, as well as the base of third party services (complementary services). The subscription base of a platform is a stock. It can have a causal loop, which might be of minor impact if consumers are not really sensitive. The causal loop describes that the bigger a subscription base is, the more it attracts potential consumers to subscribe. This subscription rate is the flow. As schematized in Figure 7, the resulting increased subscription base (stock) further amplifies the flow. Mathematically, this effect can be expressed through the exponential function

$$B = \exp(gt) * B_0 \tag{3.14}$$

where B describes the subscription base, B_0 denominates the subscription base at the time $t_{0,}$ *g* the fractional growth and *t* the time. In reality, exponential behavior is limited to s-shaped behavior due to saturation effects. This effect is also called the logistic curve. The system behaves differently when accumulating and when dispersing due to its stock. This behavior is called non-linear.

Definition 54: Non-linearity is the effect within causal loops, where manipulations are not linearly reversible, due to the accumulative behavior of stocks.

Academic literature offers several options to mathematically describe and model system dynamics and their respective ecosystems. A common approximation to estimate size and growth in networks around service platforms is the Bass diffusion model, describing the share of the potential new adopters subscribing at time t as equal to a linear function of previous adopters [114, 115].

$$\frac{b(t)}{1-B(t)} = p + \frac{q}{M} * [A(t)] \tag{3.15}$$

With

M - the potential market (meaning the ultimate number of subscriptions),

b(t) - the portion of M that subscribes at time t,

B(t) - the subscribed base at time t,
A(t) – the cumulative subscribers at time t.
p - the coefficient of innovation,
q - coefficient of imitation.

Although being regularly applied in the context of two-sided markets, the Bass diffusion model has a range of disadvantages, deterring the adoption of this option for service management on platforms:

- The Bass model assumes the potential market to be constant. However the market of platform users is open, with continuous inflows from mature industries. Holt, Weiss et al [4] speak of industry takeaways. Those takeaways can only be roughly approximated but not be quantitatively modeled;
- Similarly, the coefficients of innovation and imitation could only be roughly estimated, particularly in the context of new markets and disruptive solutions, where comparable coefficients are lacking.
- The calculated results risk being erratic with no applicability.

An alternative approach to capture these network effects is to quantitatively model platforms. The Section therefore continues with modeling service platforms in their competitive environment based on causal loop diagrams. Figure 8 shows a causal loop diagram around a service platform, modeled with the simulation environment VenSim [116]. The generation of such a model with reliable data is challenging as holistic data on competitors and consumers, required to quantitatively model such a complex environment is highly volatile and difficult to attain.

It integrates patterns for models on network and complementarity effects by Sterman [8], applied and adapted on a service platform use case and its business ecosystem. Platform 1 represents the analyzed platform. All competition is encapsulated into platform 2 and its respective business ecosystem. The model shows a same-sided causal loops around the feedback loops R1. These loops originate in scale effects on the platform's attractiveness, limited through competitive attractiveness and total demand. Scale-based attractiveness depends on subscription base, the consumers' sensitivity to the installed based and a certain threshold for scale effects. Total attractiveness of a platform is also influenced by attractiveness of complementary third party services. The attractiveness through complementary effects is part of a cascading loop R2, including the stock of third party services. This stock of third party services is again function of scale effects of the platform, market share and additional service offer. The model aggregates other effects in loop R1 through a random function.

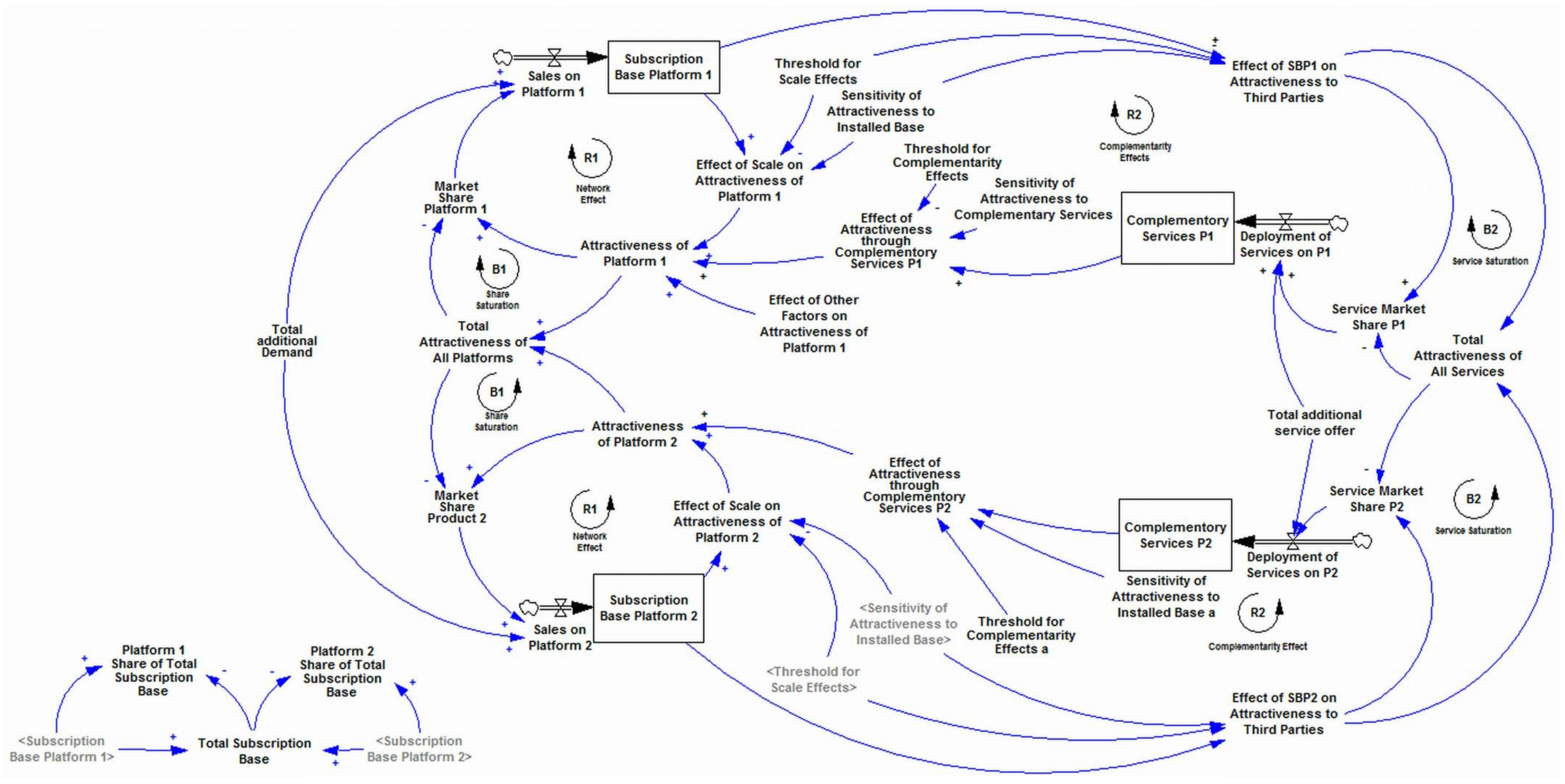
Following Sterman's [8] theory on dynamic markets, attractiveness A_{P1} of a Service platform P_1 is the product of various types of attractiveness, impacting on total attractiveness Aj, e.g., attractiveness of price, attractiveness through a trust relation, but also attractiveness resulting from causal loops e.g., through the number of subscribed users inciting other users to subscribe.

$$A_{P1} = \prod_{j=1}^{n} A_j \tag{3.16}$$

The market share of a platform is

$$M_{S1} = \frac{A_{P1}}{A_{P1} + A_C} \tag{3.17}$$

where A_{P1} stands for platform P1's attractiveness and A_C for the aggregation of all other platforms' attractiveness.



Sales on Platform 1
Subscription Base Platform 1
Threshold for Scale Effects
Sensitivity of Attractiveness to Installed Base
Effect of SBP1 on Attractiveness to Third Parties
R2
Complementarity Effects
R1
Network Effect
Effect of Scale on Attractiveness of Platform 1
Threshold for Complementarity Effects
Sensitivity of Attractiveness to Complementary Services
Market Share Platform 1
Effect of Attractiveness through Complementory Services P1
Complementory Services P1
Deployment of Services on P1
B2
Service Saturation
Attractiveness of Platform 1
B1
Share Saturation
Effect of Other Factors on Attractiveness of Platform 1
Service Market Share P1
Total additional Demand
Total Attractiveness of All Platforms
Total Attractiveness of All Services
B1
Share Saturation
Total additional service offer
Attractiveness of Platform 2
Effect of Attractiveness through Complementory Services P2
Service Market Share P2
B2
Service Saturation
Market Share Product 2
R1
Network Effect
Effect of Scale on Attractiveness of Platform 2
Complementory Services P2
Deployment of Services on P2
Sensitivity of Attractiveness to Installed Base a
Subscription Base Platform 2
<Sensitivity of Attractiveness to Installed Base>
Threshold for Complementarity Effects a
R2
Complementarity Effect
Sales on Platform 2
Platform 1 Share of Total Subscription Base
Platform 2 Share of Total Subscription Base
<Threshold for Scale Effects>
Effect of SBP2 on Attractiveness to Third Parties
<Subscription Base Platform 1>
Total Subscription Base
<Subscription Base Platform 2>

Figure 8: Simulation of a service platform including ecosystem and competition, simulated with the tool VenSim [116], based on two models on n network and complementarity effects by Sterman [8]

The attractiveness resulting from the network effect (network attractiveness) in loop R is

$$A_i = \exp(\mathscr{s}_{S0} * \beta_N) \tag{3.18}$$

where $\mathscr{s}_{S0}$ is the sensitivity to the subscription base and the normed subscription base β_N (which is the subscription base relative to the threshold). The critical mass is the threshold of required subscribed subscribers, above which the subscription base has impact on attractiveness. Formula (3.18) shows that network attractiveness needs, to incite visible exponential behavior, a basic amount of subscriptions (or other relevant activities) above a threshold.

Definition 55: Network attractiveness is defined through an exponential function of the product of a network participant's sensitivity to a stock and the magnitude of that stock relative to a threshold. The threshold delimits the magnitude where the impact starts.

Definition 56: A network effect takes place within a causal loop, when network attractiveness grows exponentially.

Definition 55 highlights a difficulty in this quanitative model. Sensitivity and threshold are theoretical parameters which help understanding the mode of action. However they would be difficult to determine a real life model. Figure 7 shows many valves which could be the toeholds for causal loops. To generate strong network effects, the causal loop has to address an inflow, coming from a source of large size. If the source is small, the network effect will abate, once the source is used up. However, a causal loop, starting at a source of (theoretically) infinite size could cause a strongly growing network effect. The present work uses the term *strong network effect* referring to this growth behavior. For those causal loops where either network attractiveness is below the threshold or the capacity of the source remains small, the remainder of the text remains unspecific and uses the term *causal loop*.

Definition 57: A strong network effect takes place when at least one causal loop fulfills two necessary conditions: accumulation in a stock exceeding the delimiting threshold and a sufficiently high source allowing for exponential growth of network attractiveness.

Definition 58: The critical mass of a stock is the magnitude that is equal to the threshold required to exhibit exponential behavior.

The dependency on a threshold of participation entails that a platform starting from zero needs a base value contribution outside a causal loop, which initially starts off this process and attracts the required subscriptions to surpass the normed subscription base β_N, unless the threshold is very small. The longitudinal study showed that all Web-service intermediaries apart from SeekDa and StrikIron lacked such a base value outside their causal loops. The services offered services based on a cross sided network effect (as shown in Figure 8). In such a constellation, successful platforms need to surpass a critical mass of services on offer to attract users. They also need a successful number of users to attract service providers. None of the intermediaries achieved this. The explorative study also brought to light causal loops with require only small thresholds for network attractiveness. The platform for online file hosting Dropbox achieved network attractiveness in offering cooperation between several users on the same file. This lead to a small critical mass of the magnitude m = 2. Also the project platform Trello uses collaborative scenarios to create network attractiveness with small quantities of users. The company IMVU provides instant contact possibilities with other members of a community of shared value and motivates users to join through the intrinsic motivation of being awarded with social contacts [117].

The network dynamics described above result from the network effect R1 (Figure 8) in conjunction with the counter-acting loop of share saturation (B1), caused by the activity of competition. The App-Exchange marketplace by Salesforce in its initial design is an illustrative example for such a same-sided causal loop as it is restricted to Salesforce consumers. The more consumers offer their application for application exchange, the more consumers are inclined to look for the offers. However, there are more loops in this model. Important are the cross-sided network effects, added through complementarity effects with service providers (indicated through a complementarity loop R2 displayed Figure 8.

A platform's attractiveness to a service provider is also exponentially dependent on the subscription base. This leads to a cross-sided effect, when platform attractiveness to consumers is further increased through an increased provision of services. In return, the increased consumer-base will again increase platform attractiveness to providers. It is plausible that the more cross-supporting loops are created, the stronger is the overall network effect. Salesforce's Force.com platform illustrates this cross-sided effect. With more consumers looking for offers, the attractiveness rises for the third party service providers to offer services. To cream off cross-sided network effects, Salesforce.com complemented their same-sided marketplace with the development and deployment environment for external service *Force.com*. The size of Salesforce's consumer base has direct effect on the suppliers' motivation to offer own services. The increasing amount of services offered increases the platform's attractiveness to the consumers.

Figure 8 illustrates an important characteristic of attractiveness related to network effects: Through the subscription base, it scales up with the causal loop of the network effect. The effects of *Other Factors on Attractiveness on Platform 1* in contrast remain constant (abbreviated as c).

c in the Salesforce example has a share of impact through the company's CRM-as-a-service. Before any network effect occurs, it helps filling the initial subscription base until it surpasses the threshold. The relative subscription base (meaning: relative to the total subscriptions in the market) of Platform 1 is

$$\beta_{rP1} = \frac{\beta_{NP1}}{\beta_{NP1}+\beta_{NP2}} \tag{3.19}$$

Substituting attractiveness in (3.17) with (3.18) and (3.19) leads to

$$M_{S1} = \frac{1}{1+c*\exp(\delta_{S0}*\beta_{NP2}*\left(1-\frac{1}{(\beta_{rP1})^{-1}-1}\right))} \tag{3.20}$$

which displays market share of platform 1 as a phase-plot of relative subscription, with the aggregated non-network related attractiveness *c* as amplifier.

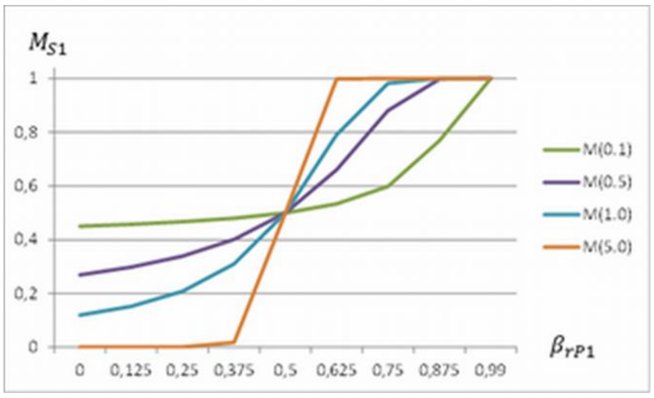


Figure 9: Market share of platform 1 (for sensitivities 0.1, 0.5, 1, 5) as a phase-plot of relative subscription base, based on [8]

Figure 9 depicts this phase plot. In a first step, focus is the network impact under the condition c=1. On the 45° line, the system is in equilibrium, as the current market share is equal to the relative subscription base. In cases where the slope is > 1, small changes in the relative subscription might cause significant changes in market share (*instable equilibrium*). On the other side, areas where the slope is strongly <1 do not provide much scope for influence (*stable equilibrium*).

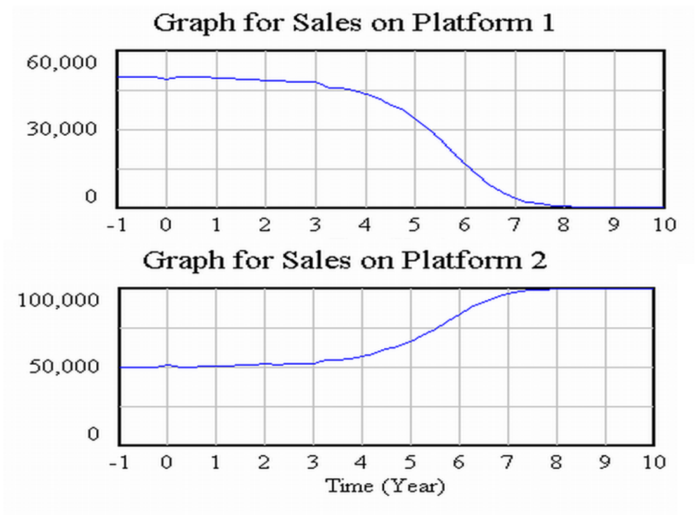


Figure 10: Simulation based on the model in Figure 8, with 2 competing platforms of a same-size consumer base of 50.000 users

Once, a dynamic process sets off, loops may self-fertilize and potentially grow towards a market dominating position (*lock-in situations*), making it particularly difficult for challengers to successfully introduce competitive solutions [8]. Figure 10 shows a simulation based on the model in Figure 8, where the market of platform 1 collapses in year 5, although both platforms started with the same subscription base of 50.000 consumers. However, the phase plot in Figure 9 illustrates that similar inertia applies, when a new entrant tries to initiate system dynamics from the beginning. Prerequisite to success is the initial value proposition (*base value*), which needs to be important enough attract a critical mass of first movers among service providers. With respect to the phase plot Figure 9, the best start off condition for an initial value is to have a low sensitiveness to scale. The value proposition with the sensitivity $s_{S0} = 0.1$ has much easier grounds to start off than a value proposition with high sensitivity (e.g., $s_{S0} = 0.1$). Compliant to this requirement, Salesforce.com started off offering only own services, which are only subject to minimal scale effects. In the longitudinal study from Section A, all sample intermediaries (apart from Seekda) depended on quantities of third party Web services and of consumers. In the phase plot this corresponds to an elevated sensitivity to scale. This means, network effects only appear, when either a high subscribed base of consumers or of deployed 3rd party services is already on board. StrikeIron's Marketplace started with a range of own services. But figures show that this base value was not sufficient to attract a consumer population, strong enough to incite a cross sided network effect with 3rd party service providers. Seekda's unsuccessfulness cannot be explained through considerations of one-sided network effects, as – through crawling – they mediated more than 70.000 Web services. Data shows that Seekda did not incite a cross-sided network effect in spite of this high quantity of mediated services. The available data does not provide sufficient information to explain this. Theoretically, the identified low average quality of service and the lack of business model could be probable reasons.

The above calculations and simulations provide a holistic and integrated explanatory model of network effects around service platforms and their ecosystems. However, this quantitative explanatory model is unsuitable to provide answers to the modeling requirements of platform architects and solution managers. The reasons lie in the quantitative aspects of the model and in the lack of architectural consideration.

Conclusion

The quantitative model relies on estimated data. Formula (3.16) formulates market share as a function of the relative subscription base. The function has an exponential constituent. Small inaccuracies in sensitivity or threshold can lead to significantly different results. As accurate data on competition and on behavioral parameters such as sensitivity and threshold are not available, small variations in estimates might even produce diametric results. In consequence, a quantitative model as output of the graphical modeling notation promises to be more reliable.

3.3 Theory on Control

Katz and Shapiro [55], Schilling [118], Boudreau [119], Parker and Alstyne's [20], as well as Hagiu and Lee [120] research control in the context of opening technical platforms, enabling cooperation of distinct supplier and user groups. The researchers consider control from a perspective of power through technology ownership, decision of technical evolution and distribution rights. Being in control includes rights to appropriate value from a technology. Parker and Alstyne's [20] research explicitly comprises service platforms. They discuss implications of platform openness on platform controllability. All the researchers of this group include network effects into their reasoning. In contrast to the differential equations in the system theoretical approach, their mathematical formulations take a static comparative or game theoretical perspective. They are equation based, describing points of equilibrium.

Closer to the system theoretical consideration of causal loops of Section 3.2 is the theory of feedback loop control [121]. This theory dates back to the beginning of the 19th century. It describes the concept of a technical system being regulated by a control device aligning a reference value with the fed back system output (Figure 11).

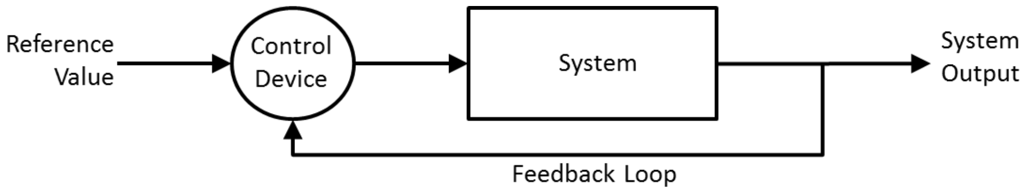


Figure 11: Feedback controlled system

The concept of feedback controlled systems was later extended under the name *cybernetics* on social or organizational systems [122]. In that context, control theory became a sub-discipline of system theory.

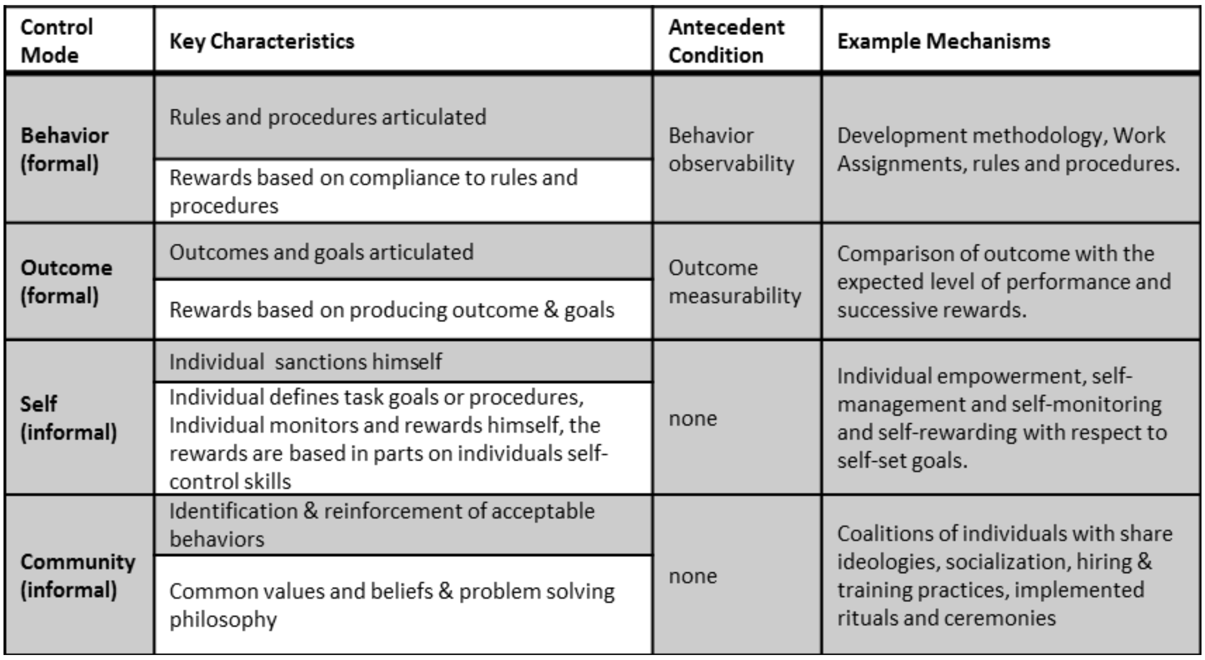
Definition 59: Feedback controlled systems are systems, being regulated by control devices, *aligning the reference value with the fed back system output.*

Other researchers [43, 123-125] provided foundations on control modes for the management in software engineering projects and provide knowledge as well as terminology and concepts. Kirsch [43] groups control mechanisms within a taxonomy of four control modes. Further, she provides guidance on the operationalization of control mechanisms in the context of software development projects. She differentiates between *formal* and *informal modes* (Table 8).

Ouchi [124] and Eisenhardt [123], refer to those formal modes as *performance evaluation strategy* and to control mechanisms a *bureaucratic mechanisms*. They refer to *informal modes* as *informal social structure* [124] or *social* or *people-based strategy* [123].

In Kirsch's [43] taxonomy, the set of formal modes contains (a), *behavior* characterized through articulated rules and procedures and (b) *outcome*, defined by expressed project outcomes and goals. Formal control modes can be designed to be observable and are hence suitable in enforcement and reward-oriented approaches [43, 123, 124]. Informal modes include (c) *self* and (d) *clans*. The present work replaces the term clan by *community* as this term established in the platform context. The control mode *self* relies fully on an individual's ability and competence to self-control. Community-modes are suitable, where coalitions of individuals group around common values and beliefs [43]. Informal modes lack observability and hence their successful implementation is difficult to observe. However, an organization can benefit from interpersonal feedback-seeking dynamics in social structures self-regulating social processes [125]. Table 8 summarizes the four control modes.

Beyond Kirsch's [43] software-engineering focused taxonomy, Ouchi [124] contributes a market view through market-based mechanisms of control. In creating a situation of choice and the availability of sufficient comparative information from the market, the market-based mechanisms enable the service consumer to choose the option with the highest value to him.



Control Mode	Key Characteristics	Antecedent Condition	Example Mechanisms
Behavior (formal)	Rules and procedures articulated	Behavior observability	Development methodology, Work Assignments, rules and procedures.
	Rewards based on compliance to rules and procedures		
Outcome (formal)	Outcomes and goals articulated	Outcome measurability	Comparison of outcome with the expected level of performance and successive rewards.
	Rewards based on producing outcome & goals		
Self (informal)	Individual sanctions himself	none	Individual empowerment, self-management and self-monitoring and self-rewarding with respect to self-set goals.
	Individual defines task goals or procedures, Individual monitors and rewards himself, the rewards are based in parts on individuals self-control skills		
Community (informal)	Identification & reinforcement of acceptable behaviors	none	Coalitions of individuals with share ideologies, socialization, hiring & training practices, implemented rituals and ceremonies
	Common values and beliefs & problem solving philosophy		

Table 8: Modes and Mechanisms of Control, based on Kirsch [43]

B Solution Design

4 Conceptual Model

Part A has led to the formulation of 3 sets of research requirements, RQ1 formulating requirements on structural representation, RQ2 describing requirements on process representation and RQ3 stating the requirements on representation related to service management. For the purpose of formalizing these requirements, this chapter provides related terminology and abstractions. These form the basis for subsequent language engineering in Chapter 5. Table 1 outlines the conceptualization of requirements into elements and groups of control mechanisms accordingly.

Perspectives	Requirements	Conceptualization
Structure	RQ1: Representation of • All areas of staged stakeholding power • Areas that need to scale, • Finite areas of similar behavior	Conceptualization of structural elements: • Areas (Control Area, Influence Area, Noise Area) • Divisions • Division Groups
Process	RQ2: Representation of: • (Unspecific) players in/outside the ecosystem and elements within the platform • Activities (e.g. deployment, subscription, social networking, exchange of applications) • Transactions • Influence Representation of • The interplay of relationships, leading to network effects	Conceptualization of process elements • Participants and participant groups • Activities • Transactions • Influences Conceptualization of relationships between elements • i.e. those representing causal loops and network effects
Service Management	RQ3: Abstraction of managed self-organization: • Enforcing mechanisms (prescription, restriction, sanctioning) • Incentivizing mechanisms (consumer-based feedback, information, motivation) leading to external and internal self-organization and network effects	Conceptualization of control mechanisms: • Service policies, prescriptive control, restrictive control, sanctioning control • Market-regulative control, informative control, motivational control

Table 9: Conceptualization of knowledge

The subsequent sections derive and conceptualize structural and procedural elements as well as groups of control mechanisms: First, structural elements respond to RQ1, embracing different areas of staged stakeholding power, areas with scale capability according to needs and areas of similar behavior. Second, process elements relate to RQ 2, represented through nodes and edges, allowing the modeling and analysis of causal loops and network effects. Finally, control mechanisms relate to RQ3, supporting the implementation of service management. Each of these subsections derives functional design requirements, which are kept non-specific to any graphical language. The functional design specifications can thus also serve as a basis for the engineering of the graphical modeling language in Chapter 5. The way they are formulated, they can also serve to derive other kinds of models, e.g. of models without representation or models of mathematical nature. When phrasing functional design requirements, subsections reference the IEFT request for comment document RFC2119 on key words *must, must not, required, shall, shall not, should, should not, recommended, may* and *optional* [126]. Consequently, the application of these key words for language specification supports the usage of prescriptive voice. In the remainder, the subsections do not further explicitly reference the RFC2119 documents.

4.1 Structural Elements

Research requirement set RQ1 asks for structural elements, representing the areas of staged stakeholding power and their implications to service management. It also asks for elements, representing areas with need to scale and for those which find infinite repetition. Subsection 4.1.1 derives the structural elements. Subsections 4.1.2 to 4.1.5 conceptualize and explain the elements in detail.

4.1.1 Derivation of Structural Elements

The sections revert to the findings of Section 2.2 on areas of staged stakeholding power, as represented in Figure 3. The solution design adopts the terminology from that section. It reformulates the definitions, given in Section 2.2 in the context of the constructs, formulated in this section. Table 10 provides an overview of the solutions, elaborated in this section and substantiated in the subsequent subsections.

Research Requirement RQ1	**Solution Design**	**Attribute**	**Degree of exertable stakeholding power**
Represent areas with full stakeholding power to the platform operator over all service provider and consumer activities	Control Area	none	Control
Represent areas with limited stakeholding power	Influence Area	none	Influence
Represent areas without stakeholding power	Noise Area	none	None
Divider allowing to substructure the control area, able to depict technical environments which need to scale.	Division	scalable (Boolean)	Control
Divider group allowing to substructure the area of full stakeholding power.	Division Group	scalable (Boolean)	Control

Table 10: Mapping of requirements with solution design

Subsections 4.1.2 to 4.1.4 conceptualize the areas of staged stakeholding power. The solution design first proposes the structural element *control area*, representing an area of full stakeholding power over all activities of service providers and consumers (4.1.2). The platform operator can control all activities in this area. It is therefore the area where control mechanisms need to be placed. Subsection 4.3 derives the control mechanisms when representing service management. To avoid an overload of information, the present work specifies one single control area. As a means to group elements within the control area, the section introduces the elements *division* and *division group* subsequently. The influence area represents the area of limited stakeholding power. The modeler cannot place any control mechanism in this area. However, he can represent the exertion of influence through the allocation of the process element *influence*. Subsection 4.2.6 elaborates on this influence element. The present work only formulates one influence area, embracing the whole surrounding platform ecosystem. Iterations with user groups did not reveal any requirement for sub-structuring elements in this area. The last introduced area is the noise area, where the platform operator cannot exert any influence or control.

A second set of requirements in RQ1 are structuring elements within the area of full stakeholding power. In particular they shall allow representation of those technical environments which need to be able to scale. Also the requirement elicitation asks for representation of areas of full stakeholding power, which find finite repetition. Subsection 4.1.5 suggests *divisions* and *division groups*. The attribute *scalable* depicts whether a specific technical area needs to scale to cope with the impact of network effects.

The present work refers to all these elements as *structural elements*. In many cases, the areas also have a spacial implication (e.g. the location of a server, of a customer or of a supplier). None of the graphical modeling languages, refered to in the related research Section 1.1 conceptualized staged areas of authority.

4.1.2 Control Area

The control area is the structural area where the platform operator has full stakeholding power. From a technical point of view, that means having the capability of enforcing his technical infrastructure on his servers or on his virtual machines in an infrastructure-as-a-service environment and on all technically enabled activities which take place in the platform. Examples for the latter are service consumption or service provisioning through third party providers. From an organizational point of view, this means that the platform operator can exert full stakeholding power over workforce, e.g., internal teams, or external entities working on assignment. In the remainder of this work, those players are referred to as *internal participants*, as opposed to *external participants* who are outside the control area.

The allocation of control mechanisms as points to exert service management is limited to the control area. Also, from the control area the platform operator exerts influence over the ecosystem participants. Rephrasing Definition 14 in the context of this chapter's constructs leads to the following definition:

Definition 60: Control area is the area where the platform operator can exert control over activities and internal participants. It is also the area from where he influences ecosystem participants that are placed outside the control area.

The following functional design requirements result from the above: First of all, the control area must be a structural element. All process elements within this area must be equipped with control mechanisms. Those mechanisms shall be set false by default and can be activated by the modeler. Process elements within the control area shall be allowed to be a source of influences, pointing at ecosystem participants in the influence area. Table 11 maps the concepts on the control area with respective functional design requirements.

Control Area	
Real-life condition	**Functional Design Requirements**
Control area is the area, where the platform operator can exert control.	The control area must be a structural element. Within the control area, all procedural elements must be equipped with control mechanisms. Those mechanisms shall be set false be default. They can be activated by the modeler.
From the control area, the platform operator influences the ecosystem participants outside the control area.	Procedural elements within the control area shall be allowed to be source of influences, pointing at ecosystem participants in the influence area.

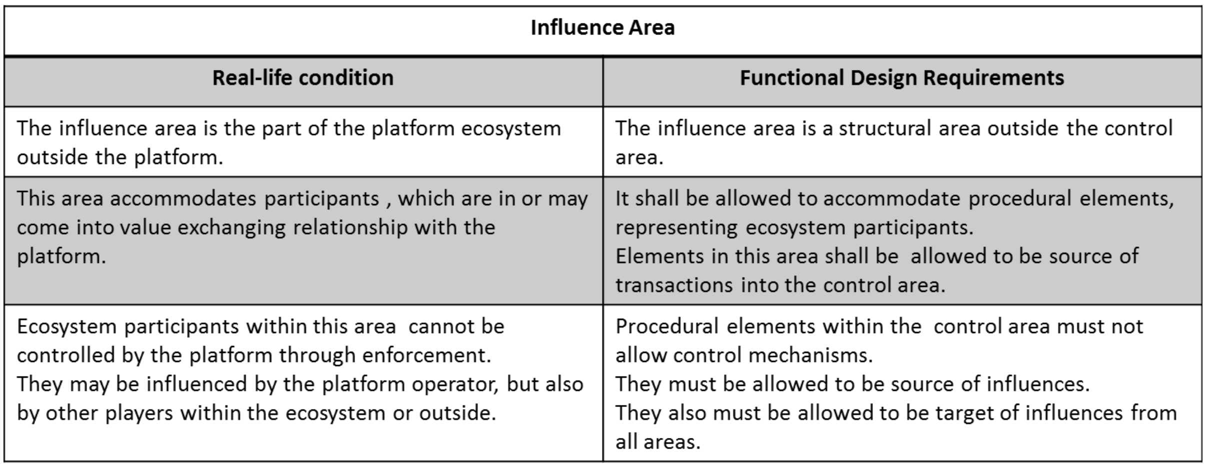
Table 11: Summary table of functional design requirements for the control area

4.1.3 Influence Area

The influence area is the structural area of the ecosystem around the control area. Ecosystem players, which are in or may come into a value exchanging relationship with the platform, are located in this area. The section on process elements introduces elements to represent ecosystem participants as well as to express relationships (4.2). This area does not allow for control through enforcement, as it is outside the space where the platform operator can exert full stakeholding power. The influences area therefore requires indirectly operating mechanisms of control, which the platform operator exerts from the control area. The remainder of this work refers to those mechanisms as *incentivizing mechanisms*. Details on how to exert indirect control follows in the sections on control mechanisms (i.e. 4.3.3 - 4.3.8). In the influence area, the ecosystem participants can also influence each other. Lastly, they are subject to influences of entities external to the ecosystem, e.g., competitors.

Definition 61: The influence area is the area where participants are located, which are in or may come into value exchanging relationship with the platform. Ecosystem participants within this area may be influenced by the platform operator, but also by other players within the ecosystem or outside.

The concepts within this section allow the formulation of a series of functional design requirements. The structural element *influence area* shall be allowed to accommodate process elements, representing ecosystem participants. Section 4.2 defines the possible elements. Elements in this area shall be allowed to be a source of transactions into the control area. Being outside the reach of the platforms full stakeholding power, the process elements within the control area must not carry control mechanisms. To be influenced from the control area, all elements shall be allowed to be the target of influences with source in the control area. The process elements within this area shall also be allowed to be target of influences from all areas. Subsection 4.2.6 provides a restrictive condition, defining when process elements can point directly at process elements within the influence area and when they should pass a merging gateway. Table 12 maps the concepts from this subsection with respective functional design requirements.



Influence Area	
Real-life condition	**Functional Design Requirements**
The influence area is the part of the platform ecosystem outside the platform.	The influence area is a structural area outside the control area.
This area accommodates participants , which are in or may come into value exchanging relationship with the platform.	It shall be allowed to accommodate procedural elements, representing ecosystem participants. Elements in this area shall be allowed to be source of transactions into the control area.
Ecosystem participants within this area cannot be controlled by the platform through enforcement. They may be influenced by the platform operator, but also by other players within the ecosystem or outside.	Procedural elements within the control area must not allow control mechanisms. They must be allowed to be source of influences. They also must be allowed to be target of influences from all areas.

Table 12: Summary table of functional design requirements for the influence area

4.1.4 Noise Area

The noise area describes the area around the ecosystem. Figure 3 in Section 2.2 shows that in this area, the platform operator cannot exert any influence on the players. It accommodates competitors and players uninterested in becoming customers. Players in the noise area are neither in a relationship of value exchange with the platform operator, nor can they exert influence on the platform operator or the included players. However they may influence the ecosystem participants around the platform and may thus cause a backflow of value (e.g., the unsubscription of players from the platform).

Definition 62: The noise area embraces all areas outside the platform ecosystem. Whereas the platform operator cannot exert any stakeholding power, the players in this area can influence the ecosystem participants in the influence area. No value flow happens between noise area and control area.

The following set of functional design requirements result from this construct: The noise area is a structural area around the influence area. The process elements within the control area must not allow control mechanisms in this area. Neither shall elements in this area be allowed to be target of influences from any area. On the other hand process elements in the noise area must be allowed to be the source of influences, pointing into the influence area. Lastly, and as the elements are not in a relation of value exchange with the platform, elements in this area shall not be allowed to be source of transactions into the control area. Table 13 maps knowledge and the resulting functional design requirements.

Noise Area	
Real-life condition	**Functional Design Requirements**
The noise area embraces all area outside the platform ecosystem.	The noise area is a structural area around the influence area.
The platform operator or any other ecosystem participant cannot exert any stakeholding power on the players in this area.	Procedural elements within the control area must not allow control mechanisms. Elements in this area must not be allowed to be target of influences from any area.
Players in the noise are can influence ecosystem participants in the influence area, but not the platform environment directly.	Procedural elements in the noise area must be allowed to be the source of influences, pointing into the influence area.
No value flow happens between noise area and control area.	Elements in this area shall not be allowed to be source of transactions into the control area.

Table 13: Summary table of functional design requirements for the noise area

4.1.5 Divisions and Division Groups

To better structure the control area, the present section specifies two sub-structuring elements, which include only minor logic but which help structuring the control area. The *division* is a frame to group elements, which belong together, e.g. because they build a set of solutions or because they are in the same physical location. The modeler shall decide which aspect is worth highlighting.

Division groups describe finite sets of similar technical environments, which could be servers at customer locations, or native applications on client PCs or mobile phones, controlled by the platform operator. In areas where network effects apply, the modeler has to be aware that the environment may require to scale. This concerns involved servers, services, databases or networks. Divisions can also group internal participants. In that case, there is no reason to talk about technical scalability.

Definition 63: A division is a structural entity of homogeneous conditions within the control area.

Definition 64: A division group is a finite set of divisions within the control area.

As resulting functional design requirements, divisions shall be available but not mandatory to group elements, limited to the control area. If those elements are available in a multiplicity, division groups shall be available to group them. To be able to indicate those areas in need of scalable infrastructure in cases, where network effects apply, both divisions and division groups shall carry a Boolean scalability attribute. This attribute shall be set false by default but can be activated by the modeler. Divisions and division groups must not have any logic interdependence with elements within the control area.

Divisions and Division Groups	
Real-life condition	**Functional Design Requirements**
A division represents a structural entity of homogeneous conditions within the control area.	Divisions must only be allowed in the control area.
Divisions , which are part of network effects, need to be modeled in a way that they can cope with rapid increase and decrease of load.	Divisions and division groups shall carry a scalability attribute. This attribute shall be set false be default. They can be activated by the modeler.
	Divisions must not have any logic , expressing interdependence with elements within the control area.
A division Group represents a finite set of divisions within the control area.	Division Groups represent a group of divisions.

Table 14: Summary table of functional design requirements for the influence area

4.2 Process Elements

From the process perspective, the requirements elicitation calls for a representation of (a) unspecific and specific players in and outside the platform and the platform ecosystem; (b) influences and transactions as well as (c) network effects. To conceptualize the process elements, Table 15 bases on system theory and suggests the process elements *participant groups, participants, activities, influences* and *transactions.* Causal loops that incite network effects consist of a concatenation of elements and therefore do not require an element of their own.

Requirements of RQ2	**System Dynamics**	**Solution Design**	**Location**
Unspecific groups players in and outside the ecosystem	Sources (with infinite capacity)	Participant group	Noise area, influence area
Specific players on the platform as well as in and outside the ecosystem	Sources with small capacity or origins of endogenous variable	Participant	Noise area, influence area, control area
Activities	Stocks	Activities	Control area
Influences	Auxiliary Variables	Influences	From noise to influence area, from influence to influence area, from control to influence area
Transactions	Flows	Transactions	From influence to control area, From control to control area
Interplay of relationships, leading to network effects	Causal loops of sources, stocks, auxiliary variables and flows	Concatenation of participant groups, activities, influences and transactions.	In influence and control area

Table 15: Mapping of requirements with solutions originating in System Theory

Whereas Subsections 4.2.2 to 4.2.6 conceptualize and explain the process elements in detail, Subsection 4.2.1 starts with the derivation of process elements. Importantly, Subsection 4.2.1 does not contain the final notation. It is rather a working notation that helps formalizing process elements, while making the subject matter less abstract. The Dynamic Network Notation (Chapter 5) will be more expressive than what can be depicted through the Stock and Flow Diagramming Notation used to derive the process elements.

4.2.1 Derivation of Process Elements

Figure 12 explains the derivation of process elements, which are described in detail throughout the following subsections.

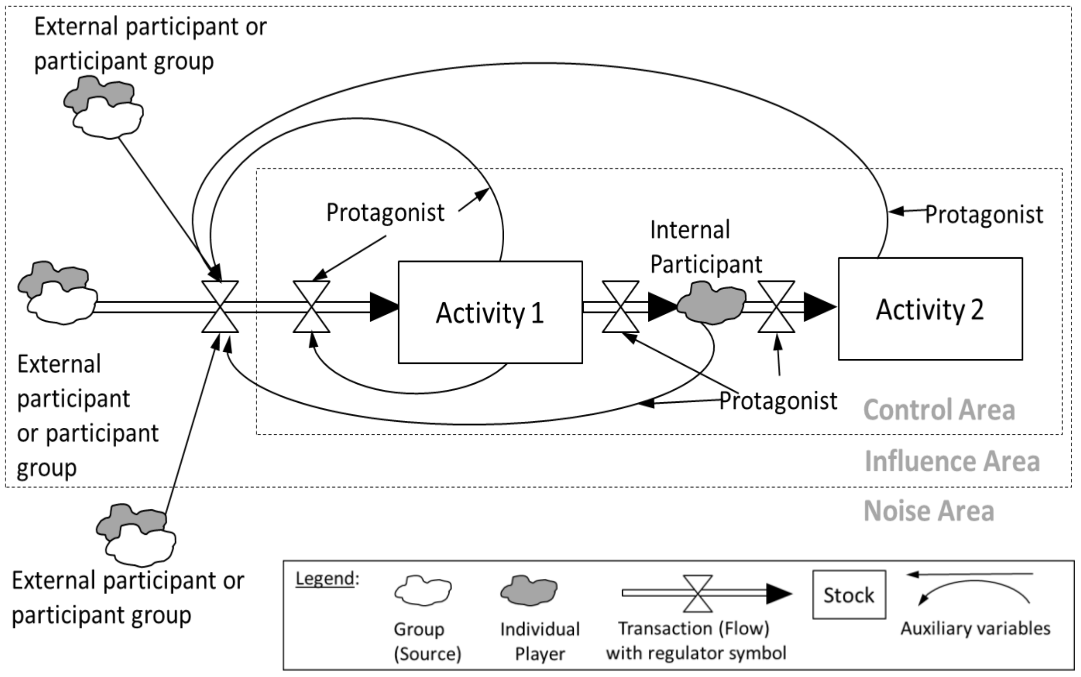


Figure 12: Derivation of process elements based on System Theory

With reference to Figure 7 in Section 3.2, Figure 12 adapts the Stock and Flow Diagramming Notation to the purposes of this thesis and applies terminology as well as elements and groups of control mechanisms as introduced earlier in this chapter. It sets the model in the context of the structural elements. Correspondingly, external *participant groups* are modeled as source in the original sense of system dynamics. Those external *participant groups* consist of external players of unspecific size and constellation (Subsection 4.2.2). *Activities* of cooperation, in turn, depict stocks in the original sense of system dynamics. They depict points of action and interaction with the platform and mostly technically enabled through the platform (Subsection 4.2.4), e.g., service consumption or collaborative scenarios[23]. Next, the system dynamic symbol for flows expresses *transactions* as flows of value into the activities in the platform (Subsection 4.2.5). Those activities accumulate, or, if the flow is inverted, deplete. Further, the Stock and Flow Diagramming Notation is able to depict *auxiliary variables,* impacting transactions, coming either from

[23] Following requirements of test users in field studies, the present work later loosened the restriction "technically enabled through the platform" and allow also less technically interpreted forms of interaction.

activities or from endogenous origins. In extension of the original stocks and flow diagramming notation, Figure 12 depicts the source of those auxiliary variables as well.

Quantitative system dynamics looks at the interplay of sources and stocks, and, hence, does not focus on individual participants. When intending to model service platforms in a well-formed way, meaning with the inclusion of all relevant aspects specified in RQ2, the language needs to be able to express those individual players. Figure 12 therefore complements the notation with a symbol of an individual player, (Subsection 4.2.3), representing internal or external *participants*. Being limited to one individual player, a participant can only have minor, close to static stock capacity e.g. his employees or one customer can create limited value through subscription to an activity or through the supply of limited amounts of services into an activity within a period of time. They therefore could not serve as source or stock within a causal loop. The figure also represents the platform operator's intervention as another input to endogenous variables. It refers to the platform operator as *protagonist*.

Summarizing these insights on system dynamic theory leads in a second step to a simplified view on process elements used to develop the abstractions as depicted in Figure 13.

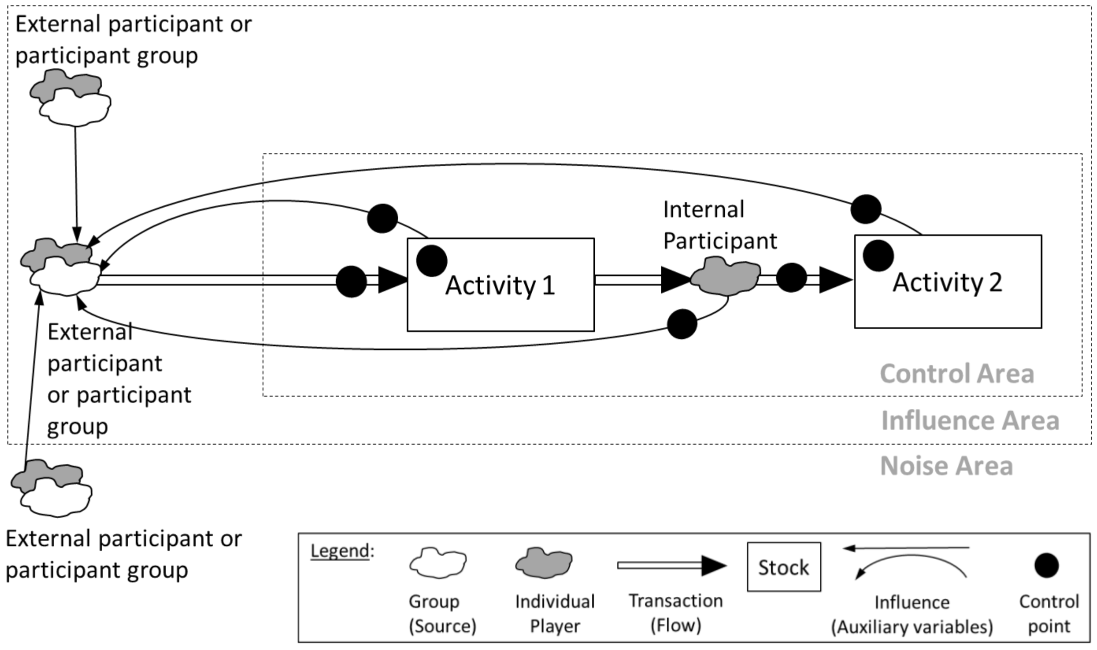

External participant or participant group
Internal Participant
Activity 1
Activity 2
External participant or participant group
Control Area
Influence Area
Noise Area
External participant or participant group
Legend:
Group (Source)
Individual Player
Transaction (Flow)
Stock
Influence (Auxiliary variables)
Control point

Figure 13: Simplified Stock and Flow Diagramming Notation with adapted terminology in the context of structural allocation

Within this context, modifications embrace the following changes:

- Auxiliary variables impacting on the outflow of a participant or a participant group in the influence area point directly at their symbol. This pinpoints the element under influence. Those auxiliary variables are now termed *influence*. As an example, the influence originating from activity 1 now addresses the symbol of the external participant or participant group. Section 4.2.6 analyzes this role in depth.
- Auxiliary variables within the control area vanish completely. 1st layer auxiliary variables from the protagonist, pointing at inflows, become a point of control on that transaction. Those pointing at outflows become a point of control within the element, causing the outflow. The figure also replaces 2nd layer auxiliary variables, originating from the protagonist through points of control on the influences.
- Loops around elements in the control area are now control points the element. As an example, the looped auxiliary variable in Figure 12 originating from activity 1 and pointing on its inflow is now represented through a point of control at the top left angle of activity 1. Those points of control can bundle several internal influences and are only depicted once.

The related graphical modeling languages, referred to in the related research section (Section 1.1) model specific participants, activities and value flows. Influence, exerted by the platform operator on ecosystem-based groups of unspecific size and constellation do not find consideration. Also none of the three languages considers or represents causal loops.

4.2.2 Participant Groups

Sources in system theory are accumulated stocks, perceived to be of infinite size. Their accumulation process is outside the modeler's focus. The source *participant group* describes groups of players of unspecific size in or outside the ecosystem. Participant groups are sources to transactions into the platform. They may describe e.g., the source of service provisioning, of development activities, of subscription or of consumption. From an organizational point of view, they are groups of individual persons or legal entities. There are several reasons for considering them as sources. First, their formation process in not in the interest of the modeler. Second, their formation is outside the scope of influence. Modeling the whole formation process would unnecessarily overload the model.

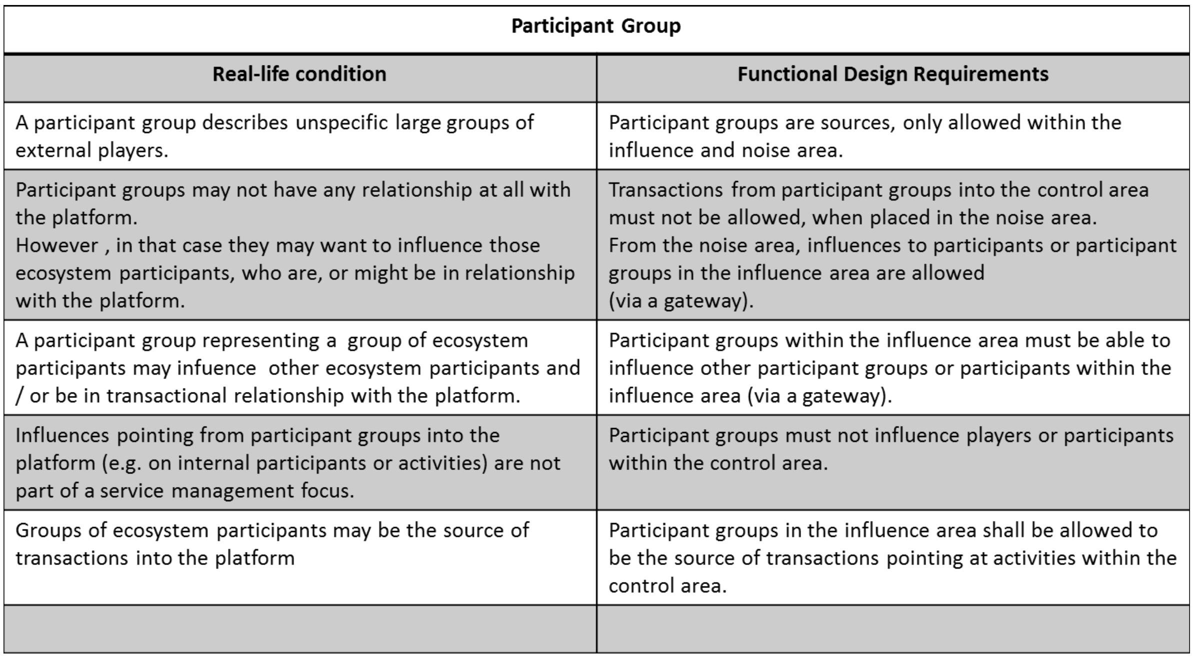
The present work does not adopt the system dynamic term *of infinite size,* as such abstraction would be unfamiliar in the context of service management. System dynamics only uses the term to describe the fact of being too big to deplete. The thesis instead speaks of *finite large size* and terms the element *participant group* (Figure 13). A modeling language shall only allow the use of the element in the influence and noise area, as it describes groups of players external to the platform. The element therefore must not be applied in the control area.

Definition 65: Participant groups are groups of individuals or of entities of finite large size within influence or noise area.

In some cases, participant groups may not have or not desire any relationship with the platform or vice versa. This stance rules out any exchange of value with the platform. As resulting functional design requirement, participant groups in the noise area must not be sources of transactions. However, as origin of an endogenous variable, those participant groups may influence ecosystem participants. For example, if a participant group represented the competition, it would want to disturb the value flow into the platform. Figure 13 depicts this as influence on the participant group, being source of the transaction to activity 1 (which is the inflow into the system). The resulting design specification is that participant groups, which are placed in the noise area, must be able to influence participants of participant groups in the influence area. Subsection 4.2.6 substantiates that these influences should only act via a merging gateway.

If a participant group represents a group of ecosystem participants, it may either influence other ecosystem participants as source of an endogenous variable, or may have a defined relationship with the platform. An example for influencing other players are third party service suppliers, which motivate targeted consumer groups to subscribe to the platform. The targeted groups of potential consumers are an example for participant groups, which are in a relationship with the platform. They are the source of a transaction into the platform. The resulting functional design requirement is that participant groups in the influence area shall be able to be the source of influences, pointing into the influences area and of transactions, pointing at activities within the control area.

In the context of service management, the present work rules out platform operators being influenced by external players other than through their value contribution. This means, only transactions may enter the control area. A platform's reactivity to the competitive environment would imply strategic adaption. Such adaption is part of corporate governance and concerns corporate boards. It is not part of the operational frameset of service management. As a functional requirement, participant groups must not be the source of influences, pointing at elements within the control area. Figure 13 depicts the possible roles of a participant group. The construct of a participant group does not differentiate between a consumer and a supplier. This accounts for the potential dual role of both and for the vanishing distinguishability between both roles.



Participant Group	
Real-life condition	**Functional Design Requirements**
A participant group describes unspecific large groups of external players.	Participant groups are sources, only allowed within the influence and noise area.
Participant groups may not have any relationship at all with the platform. However , in that case they may want to influence those ecosystem participants, who are, or might be in relationship with the platform.	Transactions from participant groups into the control area must not be allowed, when placed in the noise area. From the noise area, influences to participants or participant groups in the influence area are allowed (via a gateway).
A participant group representing a group of ecosystem participants may infuence other ecosystem participants and / or be in transactional relationship with the platform.	Participant groups within the influence area must be able to influence other participant groups or participants within the influence area (via a gateway).
Influences pointing from participant groups into the platform (e.g. on internal participants or activities) are not part of a service management focus.	Participant groups must not influence players or participants within the control area.
Groups of ecosystem participants may be the source of transactions into the platform	Participant groups in the influence area shall be allowed to be the source of transactions pointing at activities within the control area.

Table 16: Summary table of functional design requirements for participant groups

4.2.3 Participants

The element *participant* describes specific players in the platform, as well as in and outside its ecosystem.
From a system dynamic point of view, participants are small stock, perceived to be static in capacity and accumulation, where the formation process is not in the modeler’s focus, e.g. internal teams providing own existing software solutions. A participant may occur in any area. Therefore, the allocation of the element *participant* shall not be restricted. A participant may be the origin of an endogenous variable. Figure 13 depicts the possible roles of a participant.

Definition 66: Participants are individuals or entities with small capacity within the control, influence or noise area.

Similar to participant groups, participants with no relation to the platform lead to the functional design requirement that participants in the noise area must not be sources of transactions. As the origin of an endogenous variable, those participants must be able to influence participants of participant groups in the influence area via a merging gateway.

If a participant is an ecosystem participant, it may either have a defined relationship with the platform, or may influence other ecosystem participants as source of an endogenous variable. The resulting functional design requirement is that participants in the influence area must be able to be the source of transactions pointing at activities within the control area. They also must be able to be the source of an influence, pointing into the influence area.

In the context of service management, participants must not be the source of influences, pointing at elements within the control area. Figure 13 illustrates possible roles of a participant. Just like participant group, the construct of a participant can be equally applied for consumers and suppliers.

A participant is blocking within a causal loop, because of its close to static stock behavior. Therefore the language should to the greatest possible extent exclude the possibility that modelers apply it as sole source of influence on a transaction into the control area. A possible option is to bundle influences that are heading into the influence area, and start at a participant through a gateway, to allow for influences coming from activities.

As the objective is to fill the activities (stocks on the platform), meaning to increase platform value, there is no reason to focus modeling on their depletion. Therefore the language does not provide symbols for outflow into the influence area. A transaction with a negative flow could represent this, if required. However, case study experiments revealed that some modelers would like to model work flows, consisting of activity-participant-activity concatenations, connected through transactions[24]. Thus, they can express situations, where a flow comes from an activity, goes through an internal department of the platform operator for a specific treatment, e.g. quality verification, and then goes into the next activity. Depending on the case, activity 1 might deplete through that flow (e.g., in case of a development process on the platform), or remain unchanged (e.g., when a service from activity 1 is deployed on activity 2 in an enhanced version such as communication language translated to from English to Russian). Such a work flow is depicted through the Stock and Flow Diagramming Notation in Figure 13. It is built through two concatenated flow diagrams, passing an individual player. The specific concatenation is not in conflict with the fact that an individual player does not accumulate, as it simply passes the flow through, while enhancing it in the same time. Even though the language shall allow this option, it requires the modeler's attention. An internal participant between two activities may cause bottlenecks, when expressing human interaction within an automated process.

Figure 13 shows that the internal participant can be the origin of an auxiliary variable which stimulates a value flow into the platform. For example, an internal department provides an existing (considered static) stock of services for deployment into activity 2. This provisioning may serve as base value. The influence pointing from the participant to the external participant may stimulate a value flow to fill activity 1. As a resulting functional design requirement, participants within the control area shall carry the base value attribute. The default position shall be false.

[24] Modelers familiar with business process modeling languages opted to this approach.

Participant	
Real-life condition	**Functional Design Requirements**
A participant describes specific external players.	Participants are small sources, with limited capacity, allowed within the control, influence and noise area.
Participants may not have any relationship at all with the platform. However , in that case they may want to influence those ecosystem participants, who are, or might be in relationship with the platform.	Transactions from participants into the control area must not be allowed, when placed in the noise area. From the noise area, influences to participants or participant groups in the influence area are allowed (via a gateway).
A participant representing an ecosystem participant or internal players may infuence ecosystem participants .	Participants must be able to influence participant groups or participants within the influence area (via a gateway).
Influences pointing from participants into the platform (e.g. on internal participants or activities) are not part of a service management focus.	Participants must not be source of influences on activities or participants within the control area.
Ecosystem participants may be the source of transactions into the platform	Participants in the influence area shall be able to be the source of transactions pointing at activities within the control area.
A specific ecosystem participant may be in transactional relationship with the platform.	Participants shall be able to be source of transactions pointing at activities within the control area.
Some workflows may flow from points of interaction to specific internal players and then to an activity	The sequence activity-transition-participant-transition-activity must be allowed within the control area.
An internal participant can be the origin of an endogenous variable, starting off a network effect.	Participants within the control area shall carry the base value attribute, default position = false.

Table 17: Summary table of functional design requirements for participants

Similarly to participant groups, participants may model both consumers and suppliers.

4.2.4 Activities

Activities, as depicted in Figure 13, correspond to stocks in system theory. This means they can accumulate and deplete. Activities represent the location for interaction of participants or participant groups in and with the platform. They are the target of participants or participant groups, addressed through transactions. As shown in Figure 13, this leads to the functional design requirement that incoming transactions from participants and participant groups within the influence area as well as from activities and participants within the control area shall be allowed. Being technically enabled locations in the platform, a functional design requirement is that activities shall only be allowed within the control area. Also non-technically enabled interaction shall be positioned in control area. Outgoing transactions shall be allowed to participants and other activities, describing workflows.

Interpreting the behavior of an activity, any accumulation shall be considered an increase in value. This is justified by *Definition 3* and *Definition 4*, stating that value denotes those attributes with positive effects on performance of actions, objects and tasks. For example, the quantity of services deployed might have value to service consumers. Equally, the quantity of consumers subscribed might create value for other consumers. Quantities of consumers subscribed also represent value to potential third party service providers. Activities have, in contrast to participants, a time dependent state. The more participants get active, the more the stock *activity* accumulates and vice versa. The stock shrinks when they unsubscribe.

Definition 67: Activities are variable stocks, describing the magnitude and kind of interaction of participants and participant groups within the control area.

Activities can represent a base value to incite network effects. Such a base value within an activity is of explicit relevance due to its exponentially growing network attractiveness, if fed by a participant group (see formula 3.18 in Section 3.2). The resulting functional design requirement is that activities shall include a base value attribute. By the default, this attribute shall be set false.

The original intention of an activity was a purely technically enabled space of collaboration in the service platform. Iterative field studies showed that modelers bend this restriction. For example, an activity may be considered a stock for budget, generated from subscription and reutilized to influence additional subscriptions. As a result of the iterative learning process, this construct now includes also non-technical activities with or by the platform operator. They are also placed within the control area.

Activity	
Real-life condition	**Functional Design Requirements**
Activities are points of interaction in the platform which accumulate value.	Activities are stocks. They shall only be allowed within the control area.
Value flows may exist from ecosystem participants and participant groups as well as from participants and activities in the platform.	Transactions from participants and participant groups in the influence area shall be allowed. Transactions from activities addressing participants or activities within the control area shall be allowed.
Positive modeling objectives means only accumulating, but not depleting activities (stocks) of value are in the focus of the modelers interest.	No symbols describing outflows (transactions pointing into the influence area) should be provided.
An activitycan be parto of a causal loop, starting off a network effect.	Activities shall carry the base value attribute, default position = false.

Table 18: Summary table of functional design requirements for activities

4.2.5 Transaction

Figure 13 depicts transactions from a system theoretical view as flows into a stock. They represent explicit value flows within, into, or from a platform. As resulting functional design requirement, a transaction shall be a flow. The subsections on participant groups (4.2.2), participants (4.2.3) and activities (4.2.4) already derived and formulated most design requirements in context of the interaction with the respective elements. Those are only summarized here. Participants and participant groups in the influence area shall be allowed to be the source of a transaction. In the control area, activities and participants shall be allowed to be a source of a transaction. Each transaction must have exactly one source and only one target. Elements in the noise area must not be source to a transaction. Element in the influence and noise are must not be target of a transaction.

Value flows coming from the ecosystem can only target activities, as those are the only elements inside the platform, which exhibit stock characteristics. Cooperation of external participants with a specific internal participant is also a form of cooperation enabled by the platform operator. Therefore it shall be modeled as collaboration via an activity. The resulting functional design requirement states that transactions originating in the influence area must address an activity. The previous sections described various forms of work flows within the control area. Those lead to the functional design requirement that activities and participants in the control area shall be allowed as targets for transactions originating within the control area.

To ensure robustness of a model over time, a transaction's arrowhead always points into the ecosystem. Potential depletion such as the un-subscription of consumers shall be considered as negative flow in the unchanged transaction symbol. Of particular interest with respect to network effects are transactions coming from participant groups, as they can accumulate the stocks within an activity.

Reformulating

Definition 5 in the context of structural elements leads to

Definition 68: Transactions describe value flows into, within or from the control area. Their source is either in the influence or in the control area.

Transaction	
Real-life condition	**Functional Design Requirements**
Transactions circumscribe flows of value into points of interaction in the platform as well as flows within the platform	Transactions are flows. In the influence area, participants and participant groups shall be allowed as source. In the control area, participants and activities shall be allowed as sources. No element in the influence or noise area shall be allowed as target.
No interaction with participants or participant groups that have no relation to the platform	Elements in the noise area must not be allowed as source.
Transactions from external participants or participant groups exclusively flow into places of interaction. Cooperation with a participant takes place in those places of interaction.	Transaction with the source in the influence area must have an activity as target.
Transactions may describe work flows within the control area.	Transactions originating in the control area shall be able to have activities or participants within the control area as targets.
Positive modeling objectives means only accumulating, but not depleting activities (stocks) of value are in the focus of the modelers interest.	No symbols describing outflows (transactions pointing into the influence area) should be provided.

Table 19: Summary table of functional design requirements for transactions

4.2.6 Influence

Influences steer ecosystem participants or ecosystem participant groups through stimulation towards a specific value flow into the platform. From a system theoretic perspective, influences are auxiliary variables, which control the rate of flow of transactions through the stimulus of their sources. As resulting functional design requirement, influences must exclusively address participants or participant groups, either directly or via merging gateways. Their targets shall be limited to the influence area.

Various stakeholders might want to exert influence on ecosystem participants. First of all, the platform operator or persons active in activities within the platform are interested in value flow into the platform. Second, other ecosystem players may want to stimulate participation, e.g. service providers, whose services are deployed in the platform. Lastly, unrelated participants may want to exert influence. For example, competitors might want to hamper value flow into the platform. In consequence, the functional design specification demands that the sources of influences shall not be restricted to any area. Participants, participant groups and activities shall be allowed as source to an influence.

Definition 69: Influences are means to stimulate the rate of value flows at their sources.

The present work considers influences as group of stimuli on ecosystem participants. In an uncontrolled way, it might be any action without intervention of the platform operator and potentially without effect (e.g., exchange of information between ecosystem participants). When equipped with a control mechanism, this means that it is amplified through the platform operator's incentives, i.e. market-regulative control, informative control and motivational control. Subsections 4.3.6, 4.3.7 and 4.3.8 substantiate this.

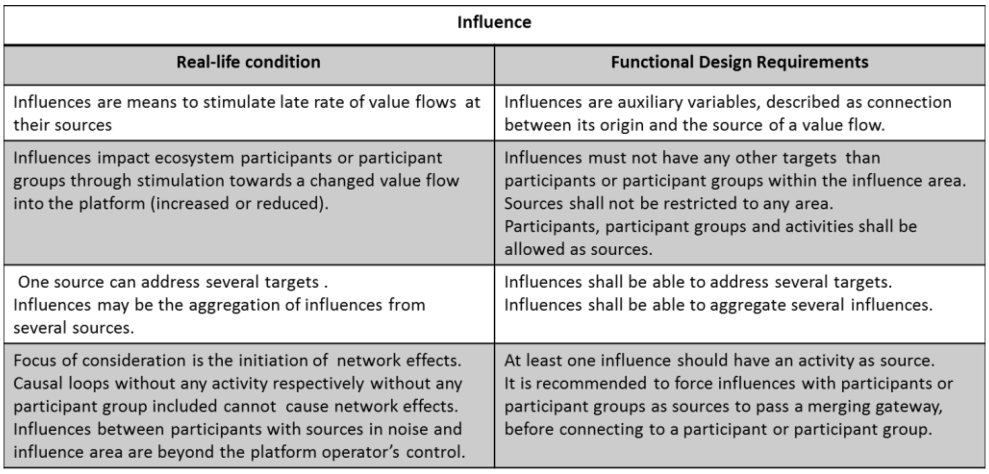
Transactions as introduced in Subsection 4.2.5 describe an explicit exchange of value between explicit entities (i.e. the platform operator, one ecosystem participant or an internal participant). Influences are less focused, i.e. when addressing participant groups e.g., with information on quantity of deployed services. Such information might influence members of a specific participant group to subscribe to a platform. The information on quantity of deployed services might even influence several participant groups. As a functional design consequence, influences shall be allowed to have several targets. Optionally, but not mandatorily a language might depict this through a gateway with one entrance and several outputs.

Also, influences of several sources may aggregate in one resulting source, stimulating a participant or participant group. As functional design specification this leads to the requirement that several influences shall have the possibility to address the same target. This design specification could optionally be accomplished through a gateway.

The influence of a participant or participant group in the influence or noise area on a participant or a participant group in the influence area is beyond control of the platform operator. In consequence, to ensure that the platform operator is not inactive in influences in the influence area, the modeling language should[25] support that at least one influence originating from the control area is among the influences. Also a language should[26] help to avoid the creation of loops which do not include stocks in the control area (activity) or large size sources (participant groups) in the influence area. A too restrictive specification may reduce a modeler's scope of freedom and hence the language's expressiveness. This functional design specification therefore recommends the following: Influences with activities as sources should be able to directly address participants or participant groups in the influence area. Influences from all sources other than from an activity should require a merging gateway, with one or more resulting influences pointing at one or more targets. The latter intends to focus the modeler's attention on the potential adding of an activity. As there may be specific cases where no activity should be in the loop, the gateway must not enforce one or more activities as source. To support the inclusion of participant groups into causal loops, the present work suggests a consecutive analysis with recommendation after the modeling process.

[25] The RFC2119 suggests applying the verb *should* when there may exist reasons to ignore the requirement. In consequence the implementation demands careful weighting.

[26] Again use of the verb *should* for the above reasons.



Influence	
Real-life condition	**Functional Design Requirements**
Influences are means to stimulate late rate of value flows at their sources	Influences are auxiliary variables, described as connection between its origin and the source of a value flow.
Influences impact ecosystem participants or participant groups through stimulation towards a changed value flow into the platform (increased or reduced).	Influences must not have any other targets than participants or participant groups within the influence area. Sources shall not be restricted to any area. Participants, participant groups and activities shall be allowed as sources.
One source can address several targets . Influences may be the aggregation of influences from several sources.	Influences shall be able to address several targets. Influences shall be able to aggregate several influences.
Focus of consideration is the initiation of network effects. Causal loops without any activity respectively without any participant group included cannot cause network effects. Influences between participants with sources in noise and influence area are beyond the platform operator's control.	At least one influence should have an activity as source. It is recommended to force influences with participants or participant groups as sources to pass a merging gateway, before connecting to a participant or participant group.

Table 20: Summary table of functional design requirements for influences

4.3 Control Mechanisms

The previous Sections of this chapter structured the knowledge on how to model platform design and how to bind in the ecosystem in favor of harnessing network effects. This section responds to research requirement RQ3 grouping service management mechanisms and allocating them in the context of structure and process elements to allow for managed self-organization.

Section 2.6 revealed mechanisms of intervention, which give successful platform operators ways to manage services and service consumption through *managed self-organization*. This section sets those into context with the above introduced structural elements and process elements. The section starts with an overall categorization of the control mechanisms, based on various control concepts as well as system theoretical concepts, introduced in Sections 3.2. and 3.3. Before categorizing the control mechanisms, the present work groups those prescriptions, which the platform operator communicates a priori to potential users with intent to steer their course of action and to set the rules of cooperation (Subsection 4.3.2). In addition, the section categorizes the enforcing mechanisms as prescriptive control (Subsection 4.3.3), restrictive control (Subsection 4.3.4) and sanctioning control (Subsection 4.3.5) leading to respective functional design requirements. Similarly, it categorizes incentivizing control mechanisms, in specific market-regulative control (Section 4.3.6), informative control (Section 4.3.7) and motivational control (Section 4.3.8).

4.3.1 Derivation of Control Mechanisms

The previous sections captured sources, flows, stocks and auxiliary variables set in the context of structural allocation around a service platform. Transactions represent the flows, activities describe stocks, participant groups describe the sources of flows and participants stand for additional sources with small variables. Participant groups, participants and activities can be the sources of influences, depicting the auxiliary variables. Figure 13 describes the interplay.

However, the model still lacks control mechanisms, which are able to steer the causal loops around the platform into the right direction. Adding control mechanisms into causal loops turns those loops into controlled feedback systems (compare Figure 11in Section 3.3). Figure 13 visualizes the allocation of these mechanisms through bullet points. Respecting the limitation of full stakeholding power to the control area leads to the exclusive positioning of control mechanisms within this area. Figure 13 shows three possible positions:

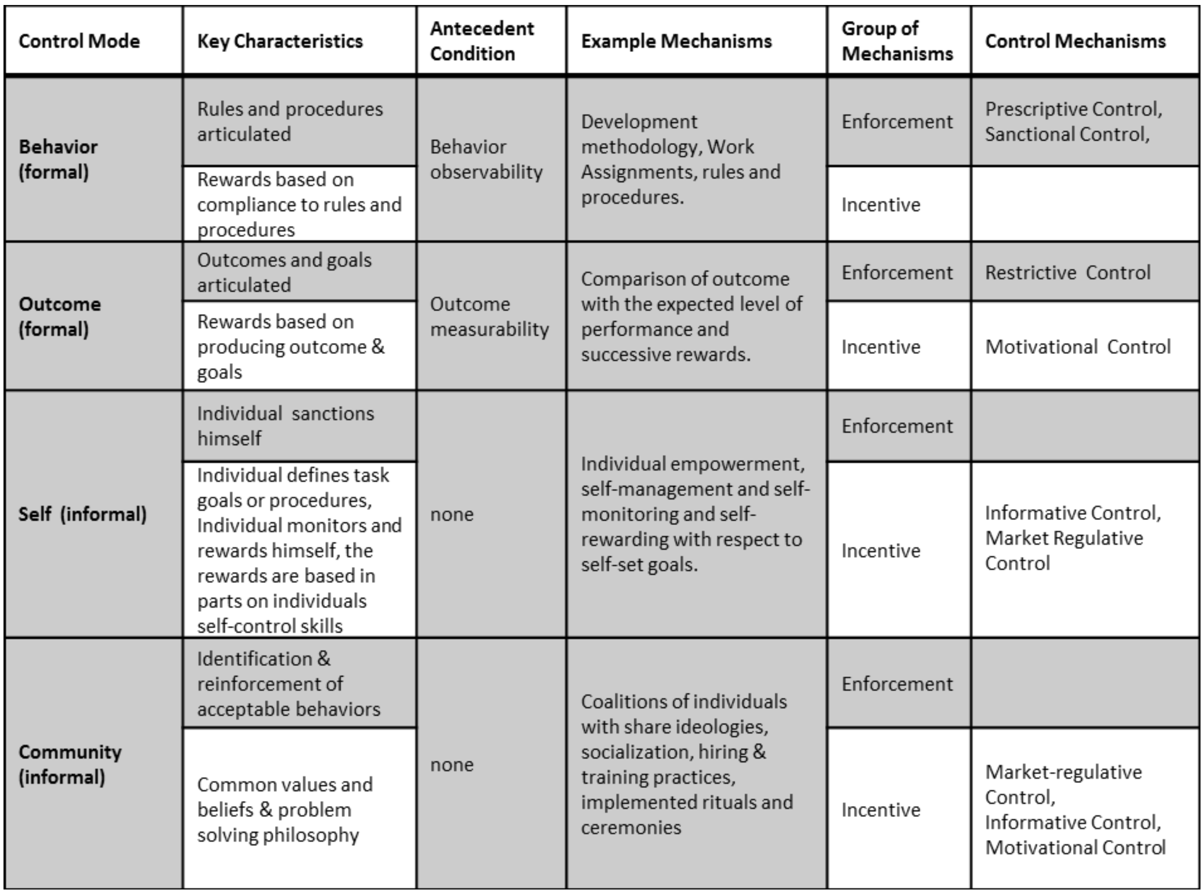
- on transactions, regulating the inflow (resp. outflow) into an activity;
- on activities, regulating the interaction;
- on influences, regulating the feedback into the ecosystem.

Applying the concept of feedback control on the present theory leads to the following definition:

Definition 70: Control loops are causal loops which carry control mechanisms on transactions, activities and influences to manipulate their progression.

The challenge is to find and allocate the right control mechanisms in pursuit of managed self-organization. Doing that needs incentivizing and enforcing control mechanisms to manage ecosystem participants and services.

The present work uses Kirsch’s [43] four control modes as starting point for the allocation of control mechanisms for service management in platforms and for the derivation of functional design requirements. Table 8 summarizes the four control modes. Beyond Kirsch’s [43] software-engineering focused taxonomy, Ouchi [124] contributes a market view through market-based mechanisms of control. In creating a situation of choice and the availability of sufficient comparative information from the market, the market-based mechanisms enable the service consumer to choose the option with the highest value to him.



Control Mode	Key Characteristics	Antecedent Condition	Example Mechanisms	Group of Mechanisms	Control Mechanisms
Behavior (formal)	Rules and procedures articulated	Behavior observability	Development methodology, Work Assignments, rules and procedures.	Enforcement	Prescriptive Control, Sanctional Control,
	Rewards based on compliance to rules and procedures			Incentive	
Outcome (formal)	Outcomes and goals articulated	Outcome measurability	Comparison of outcome with the expected level of performance and successive rewards.	Enforcement	Restrictive Control
	Rewards based on producing outcome & goals			Incentive	Motivational Control
Self (informal)	Individual sanctions himself	none	Individual empowerment, self-management and self-monitoring and self-rewarding with respect to self-set goals.	Enforcement	
	Individual defines task goals or procedures, Individual monitors and rewards himself, the rewards are based in parts on individuals self-control skills			Incentive	Informative Control, Market Regulative Control
Community (informal)	Identification & reinforcement of acceptable behaviors	none	Coalitions of individuals with share ideologies, socialization, hiring & training practices, implemented rituals and ceremonies	Enforcement	
	Common values and beliefs & problem solving philosophy			Incentive	Market-regulative Control, Informative Control, Motivational Control

Table 21: Modes and Mechanisms of Control, based on Kirsch [43], modified and extended

Table 8 extends Kirsch's [43] table, mapping the control mechanisms to the control modes. This enables the present section to build on existing knowledge on control modes, key characteristics and antecedent conditions, when defining the control mechanisms and when deriving respective functional design requirements. The mechanism *market-regulative control* complements the taxonomy in Ouchi's [124] market view. In addition, the table groups these mechanisms according to their enforcing or their incentivizing orientation.

Table 22 maps the revealed mechanisms with their accommodating elements. Activities carry the enforcing mechanisms prescriptive and sanctional control. They further accommodate market-regulative control, informative control and motivational control. Participants may just carry prescriptive control: transactions are limited to restrictive control. Influences can work with market-regulative control, informative control and motivational control. The following subsections provide substantiation on this.

	Enforcing Mechanisms			**Incentivizing Mechanisms**		
	Prescriptive Control	Sanctional Control	Restrictive Control	Market-regulative Control	Informative Control	Motivational Control
Activity	x	x		x	x	x
Participant	x					
Transaction			x			
Influence				x	x	x

Table 22: Activities, participants, transactions and influences with their suitable control mechanisms

Among the graphical modeling languages referred to in the related research section (Section 1.1), only e3control by Kartseva et al. [29] consider patterns of control. The goal of these control patterns is the exploration and mitigation of opportunistic behavior of the counter actor, leading to disrespect of contractual agreements. The authors limit their consideration on flowes of concrete value exchange, based on contractual agreement. They suggest six control patterns, i.e. enforcement through penalties, reward-based incentives, observation-based patterns (partner screening and execution monitoring) and provision of documented evidence (through proper contracting and through execution confirmation). The present work also conceptualizes enforcing mechanisms. Apart from a penalty-based sanctional control mechanism, it introduces two more enforcing categories, notably prescriptive and restrictive control. Also with respect to the incentivizing mechanisms, the present work is more exhaustive, suggesting one reward-based and two intrinsically motivating incentivizing mechanisms. Oberservation is no individual mechanism in this thesis, but an immanent element of each mechanism. Lastly, documented evidence does not find consideration in the present work, as processes in service platforms are considered tracable.

4.3.2 Service Platform Provisions

Through the provision of service platform provisions, platform operators define basic conditions to cooperation with ecosystem participants.

Definition 71: Service platform provisions are documents, defining restrictions to and rules of participation and cooperation for participants in activities in service platforms.

As condition for participation, platform operators may provide development frameworks in form of programming specifications or software development kits. The objective is to ensure development according to guidelines, in order to achieve a defined level of service level measurement quality, business process quality, suitability for standards, security quality and manageability quality. On the service consumption side, users e.g., have to conform to requirements such as the provision of a payment method or the explicit acceptance of the platforms terms and regulations.

Service platform provisions are documents that provide the basis for ecosystem participants to start and maintain a relationship with the platform operator. Those provisions are the basis for all restrictive and sanctional control. As resulting from functional design requirement, service platform provisions should be provided as textual documentation or a link to a document. It shall always possible to map any service platform policy with any restrictive or sanctional control mechanisms (Table 23).

Service Platform Provisions	
Real-life condition	**Functional Design Requirements**
Service platform policies are documents. Defining restrictions to and rules of participation and cooperation .	Service platform provisions are textual documentation or as links to documents.
Those policies are orientation for all restrictive and sanctional control mechanisms.	It shall be allowed to map any service platform provisions with any prescriptive, restrictive or sanctional control mechanism.

Table 23: Summary table of functional design requirements for service platform provisions

4.3.3 Prescriptive Control

Behavior of ecosystem participants, e.g., of developers in a software engineering process, can be guided through articulation of rules and procedures as defined in service platform provisions. Table 8 specifies as an antecedent condition the observability of behavior [43]. Platform operators can accomplish observability of behavior through the mandatory allocation of third party activities into the control area. Having the activities allocated within the area of full stakeholding power, the platform operator can a) observe and b) steer the activities of external participants and modify their outcome. The present work terms this interplay of observation, steering and modification *prescriptive control*. As internal participants are in hierarchical subordination to the platform operator's authority, the present work also equips them with prescriptive control. There may be cases where prescriptive control references service platform provisions.

Definition 72: Prescriptive Control is the sequence of observing and steering a participant's set of actions within activities as well as of internal participants. For actions in activities, it may further include subsequent corrective measures on their results through the platform operator.

As a result of functional design requirement, prescriptive control shall be located exclusively in all activities and participants, within the control area. The mechanism shall be set false by default. It may be activated by the modeler. It shall be able to represent service platform provisions as textual documentation or as link (Table 24).

Prescriptive Control	
Real-life condition	**Functional Design Requirements**
Prescriptive Control is the sequence of observing and steering a participant's set of actions within activities as well as of internal participants.	Prescriptive control is a control mechanism. It shall be located exclusively in all activities and participants, within the control area. It shall be set false by default. It may be activated by the modeler. It shall be able to represent service provisions as textual documentation or as link.

Table 24: Summary table of functional design requirements for prescriptive control

The following paragraphs illustrate the mechanism of prescriptive control with the example of the activities *service development*, *service deployment* and *customer subscription.*

Service development

The platform operator can observe the service provider's behavior during the service development phase through prescription of a development environment, located on the service platform. Enforcing development in such an environment gives an additional advantage, beyond observability of development behavior. It allows the developer's scope of freedom to be limited in such a way that only solutions compliant to the platform operator's expectation are possible. Identified ways to limit freedom are to prescribe development languages and tools (Chapter 2, Table 5). Thus the platform operator ensures:

- Manageability quality: The software design can be steered in a way that – once deployed - the platform operator can observe the final code and can manage it (e.g., adaptation of code sequences due to hardware changes). A prescription of a limited range of programming languages simplifies this task.
- Security Quality: The tools can enforce specific security solutions in function of specific requirements. The above stated manageability quality even allows for subsequent modification of the security quality in function of specific requirements.
- Suitability for Standards: The programming environment can enforce through limited choice of options and tools specific standards and interoperability, e.g., with respect to the platform's API.
- Service Level Measurement Quality: The platform operator can enforce a software design that facilitates enhanced observability with respect to service quality features i.e. response time, accessibility and successability.

Service deployment

When following the integrator concept, service deployment mandatorily takes place within the control area. This gives the platform operator the power to observe and manage service quality. The degree of influence depends on the platform policy. If full freedom of service development is given, as in the case of Facebook[27], the platform operator's scope of control is limited. If an activity *development* proceeds within a programming environment on the platform, the platform operator can benefit from prescriptive control scope during the whole service life cycle, through manageability quality, security quality, suitability for standards and service level measurement quality, as described above.

[27] http://developers.facebook.com/docs/sdks/, retrieved 11.02.2013

4.3.4 Sanctional Control

Activities in the platform are subject to an additional mechanism of control, called *sanctional control*. The platform can sanction services deployed or users subscribed. Sanctions take force in the moment of a breach e.g., copyright infringement or service underperformance. The platform gathers information e.g., through automatic verification, through service support (call centers) or through complaint management systems. After the discovery of such an infringement, an escalation routine takes place. Depending on aspects of safety, security or the importance of the breach, the escalation routine can vary from defined time for correction or statement requested from the participant to immediate undeployments.

Definition 73: Sanctional control describes the enforcing action of the platform operator on policy breaches in activities through an escalation routine, including discovery processes, scope and time of reaction for the participant and range of enforcements through the platform operator.

As a result of functional design requirement, sanctional control shall be located in all activities and only there. The mechanism shall be set false by default. It may be activated by the modeler. It shall always be possible to map any service platform policy with any restrictive or sanctional control mechanisms (Table 25).

Sanctional Control	
Real-life condition	**Functional Design Requirements**
Sanctional control describes the enforcing action of the platform operator on policy breaches in activities .	Sanctional control is a control mechanism. It shall be located exclusively in all activities. It shall be set false by default. It may be activated by the modeler. It shall be able to represent service platform provisions as textual documentation or as link.

Table 25: Summary table of functional design requirements for sanctional control

The following examples illustrate sanctional control. Facebook [84] specifies in its platform provisions potential restriction of service providers' and service's access to platform functionality, termination of agreements or any other necessary action in case of infringement[28]. Salesforce.com has a 2-staged escalation routine. The company proactively removes allegedly infringed services or applications after notification from the market. At the same time, Salesforce.com [127] informs the provider of the service or application. On the service provider's

[28] https://developers.facebook.com/policy/

request, Salesforce will the redeploy service or application. In the last stage of the escalation routine, a court or service provider and notifying body unanimously need to request undeployments to lead to the final removal from the platform[29]. All analyzed platforms possess termination clauses and routines in case of policy breaches, which the platform can enforce automatically.

The surveys typically encountered sanctional control during the operating phase. However, applications in other phases (i.e. during the design or deployment and update phases) should not be ruled out.

4.3.5 Restrictive Control

In service management of platforms, restrictive control applies on transactions, filtering the value flow into an activity. The necessary condition for a value to be let through is compliance with the service platform provisions. Restrictive control belongs to the formal control modes. As the platform operator does not observe the generation of a value (e.g., a process of service development or of participant subscription), the performance of restrictive control relies on outcome measurability as an antecedent condition (Table 8).

Definition 74: Restrictive Control is a filter mechanism on transactions placed within the control area and verifying compliance with service platform provisions.

As resulting functional design requirement, restrictive control shall be located exclusively in all transactions, within the control area. The mechanism shall be set false by default. It may be activated by the modeler. It shall always possible to map any service platform policy with any restrictive or sanctional control mechanisms (Table 26).

Restrictive Control	
Real-life condition	**Functional Design Requirements**
Restrictive Control is a filter mechanism on transactions, placed within the control area and verifying compliance with service platform policies.	Restrictive control is a control mechanism. It shall be located exclusively in all transactions, within the control area. It shall be set false by default. It may be activated by the modeler. It shall be able to represent service platform provision as textual documentation or as link.

Table 26: Summary table of functional design requirements for prescriptive control

[29] http://www.salesforce.com/au/company/legal/intellectual.jsp

The following examples illustrate restrictive control. On the participant's attempt to enter the activity *consume*, all analyzed platforms impose standard requirements e.g., the availability of all contact information, payment information or any other credentials. If the verifications exhibit incomplete or erratic information, the platform requires the user to complete it. After successful completion, the user is granted access to the platform. Service providers pass through a similar procedure.

Additional restrictive control applies when the service providers supply a service to the platform. The platform compares the service with the articulated provisions. In case of non-compliance, the platform refuses access and potentially asks for amendment. Such restrictive control can generate the same outcomes as enforced development within a programming environment, with respect to a service's manageability quality, security quality, suitability for standards and service level measurement quality. However, in contrast to the fully transparent service development process in the case of a programming environment (prescriptive control), the platform operator in the case of restrictive control only sees the outcome of the development process and needs to verify malicious or erratic deviation from the communicated expected outcomes. This approach carries the risk of missed deviations from the defined outcome within the phase of restrictive control.

Restrictive control also applies on external services, which are requested by services deployed on the platform or on e-markets which federate external services e.g., Facebook through its Graph API[30]. Such filtering limits susceptibility to errors and underperformance when calling external services, as discussed in the context of prescriptive control. Also transactions within the platform can be filtered through restrictive control.

[30] http://developers.facebook.com/docs/reference/api/, retrieved 08.02.2013

4.3.6 Market Regulative Control

Market regulative control categorizes control mechanisms which are fully driven by the ecosystem and which are generated through explicit feedback. Market regulative control can address service consumers (e.g., through ranking) as well as service providers (e.g., through recommendation boxes). Its objective is to communicate information on service quality. The ecosystem self-organizes when, in reciprocity, consumers adapt their consumption behavior and service providers amend service quality. It is a self-regulatory process; the platform operator's role reduces to the provider of the necessary infrastructure.

This self-organizing process represents a causal loop within the control area as well as into the ecosystem. In the Stock and Flow Diagramming Notation in Figure 12, the causal loop within the control area points from activity 1 back to the valve situated on the inflowing transaction inside the control area. Figure 13 replaces this causal loop by a filled circle, sitting in the upper left corner of activity 1. The new visualization has two reasons: First it reduces the load of information in the representation. The bold circle can accommodate all control mechanisms which act within the activity. Second, the representation in the original Stock and Flow Diagramming notation is irritating, as now value inflow does not come from the external source but is created inside. The causal loop within the activity creates value in a self-regulatory way. It can add value incrementally (e.g. through recommendation of services to consumers, as this increases trust). It is not able to turn into a strong network effect. Although it may fulfill the necessary condition of exceeding the necessary threshold, the limited size of the stock activity does not satisfy the second condition. A potential network effect would require a source of finite but large size.

Figure 13 depicts a second causal loop with an influence, pointing from activity 1 to an external participant or participant group. Placed on this influence, the market regulative control mechanism can incite new participants to transfer value to the platform (e.g. to subscribe). In the case where the ecosystem participant is a participant group, both conditions of a strong network effect can be theoretically fulfilled.

Another possibility of applying market regulatory control is on influences, pointing from internal participants to external participants or participant groups. For example, collaborative feedback systems may enable participants to recommend or advise against a value or base value contribution, provided by the participant.

Definition 75: Market regulative control is driven by participants. It gives explicit feedback to consumers or service providers in the platform and / or in the ecosystem on value, offered in activities or through participants. This causal loop incites a self-regulatory process.

In market-regulative control, service platforms make use of collaborative feedback systems, which allow for the collection, distribution and aggregation of information about a participant's activities [128] or the performance of a service. They can address internal or external participants and participant groups.

To create the best match of collaborative feedback systems with control modes, the chapter reverts to Jøsang, Ismail et al.'s [82] categorization into collaborative sanctioning systems (reputation systems) and collaborative filtering systems (recommender systems). Collaborative sanctioning is undirected and incentivizing to individual users as they create a trust-basis for a service decision. This subgroup of market regulative control mechanisms belongs to the control mode 'self' (Table 8), as all action is based on individual empowerment. Recommender systems serve the tastes and preferences of specific communities [82]. These coalitions of individuals with shared values fulfill the characteristics of the community control mode in Table 8. Whereas this mode incentivizes the consumers, it can both sanction and incentivize service providers, depending on the feedback. Table 27 maps the two types of feedback systems with implementation examples, retrieved from the surveys.

Category	Feedback System	Example
Collaborative Feedback Systems	Reputation Systems (Collaborative Sanctionning)	Rating, Like Buttons, Comment Box, Recommendation Box (consumer suggestions on services),
	Recommender Systems (Collaborative Filtering)	Send Buttons (allows to send content or suggestions to community members), Follow Buttons, (allows to follow a participants activities and contributions), Activity Feeds (allows to follow activities of community members), Facepiles (providing photos of community members who like a service).

Table 27: Classification of collaborative feedback systems

Recommender systems support the creation and evolution of observable communities of common values and beliefs on the demand-side. Such systems allow the platform operator in subsequent steps to separate monolithically modeled participant groups into several community-specific target groups, allowing for improved consumer management.

Guiding on and self-organizing around service consumption, reputation systems need to be modeled on activity elements. Recommender systems may be modeled on influences to communicate the feedback to non-subscribed suitable target group. They may also be modeled on activities, when they are intended to address subscribed participants. The feedback can be personalized, i.e. addressing a specific participant, potentially within a participant group. It can also be designed as a general feedback to a participant group.

The platform operator also has the possibility to interconnect this self-regulatory process with active service management, e.g., through the definition of a minimum consumer satisfaction level per service within the platform service provisions. This allows concatenating sanctional control. The typical phase of application for market regulative control is the operating phase.

The following functional design requirements result from the above. Market regulative control shall be located exclusively in all influences and activities, within the control area. The mechanism shall be set false by default. It may be activated by the modeler. Optionally, market regulative control may provide a 2nd layer sub-categorization into recommender and reputation systems (Table 28).

Market Regulative Control	
Real-life condition	**Functional Design Requirements**
Market regulative control is driven by participants. It gives explicit feedback to consumers or service providers in the platform and / or in the ecosystem on value, offered in activities or through participants.	Market regulative control is a control mechanism. It shall be located exclusively in all activities and influences, within the control area. It shall be set false by default. It may be activated by the modeler. Optionally, it may provide a 2nd layer sub-categorization into recommender and reputation systems . It shall be set false by default. It may be activated by the modeler.

Table 28: Summary table of functional design requirements for market-regulative control

4.3.7 Informative Control

Informative control stimulates creativity in the ecosystem, targeting individuals or communities and providing them with preprocessed information, e.g., on service requirements, preferences or feedback on specific quality. It addresses the participants' intrinsic motivation. Intrinsic motivation relates to activities done because the acting participants expects personal satisfaction out of a specific activity [81]. The respective informative control mechanisms consequently need to highlight opportunities or invitations to participate in activities, which are of personal satisfaction to the consumer, e.g., selected social networks or user groups. In contrast to the contributions in market regulative control, which are community-based, the platform operator manages informative control.

The mechanism's goal is to incite a self-regulatory process of alignment in favor of the service platform. Like market regulative control, it is located on activities and influences pointing from the control area into the platform ecosystem. Information addressed to unspecific target groups could be e.g., the availability of new services, specific to the target groups' expected requirements. Information addressed to specific subscribed users could be e.g., information of insufficiently serviced consumer requirements, which are close to the service provider's subject area. In that case, it acts within an activity. Informative control operates in the control mode *self* and builds on the individual empowerment of a participant (Table 8). It can be applied in all phases of the service management cycle.

Definition 76: In informative control the platform operator preprocesses information and addresses it to existing or potential participants or participant groups. The analyses are customized on the addressed participants or participant groups and have the goal to incite a self-regulatory process among them.

Being in a position of monitoring, the platform operator has the means to aggregate information and to customize it to a specific service provider's or consumer's requirement. The service provider has two possibilities of collecting feedback on consumer requirements, explicit and implicit feedback. Placed on activities, informative control addresses and supports existing participants in the activity. For example, participants can gradually strengthen the impact of the activity, e.g., through more focused quality of service, which they improve through feedback. In the equation on network attractiveness (equation 3.18), this would enhance sensitivity. Placed on an influence, informative control addresses external participants. E.g., potential customers can be intrinsically motivated through the available information on existing services. Informative control improves network attractiveness through improved sensitivity (equation 3.18). Ways to communicate information to specific target groups are outside the focus of this research project.

For further reading in the discipline of marketing communications, please refer to e.g., Smith and Zook (2011) or (Ries 2011).

The following functional design requirements result from the above. Informative control shall be located exclusively in all influences and activities, within the control area. The mechanism shall be set false by default. It may be activated by the modeler (Table 29).

Informative Control	
Real-life condition	**Functional Design Requirements**
In Informative Control, the platform operator preprocesses information and addresses it to existing or potential participants or participant groups.	Informative control is a control mechanism. It shall be located exclusively in all influences and activities. It shall be set false by default. It may be activated by the modeler.

Table 29: Summary table of functional design requirements for informative control

4.3.8 Motivational Control

The platform operator has the possibility to incentivize desired activities. Motivational control means those mechanisms which set explicit incentives and potentially reward participants. Motivational control is placed on influence edges, pointing at participants or participant groups in the ecosystem.

Definition 77: Motivational control aims at steering ecosystem participants towards the accomplishment of specific outcomes through rewards.

Motivational control can be triggered through monetary or non-monetary rewards. An example for monetary rewards would be seed funding for specific participants. Examples for non-monetary motivation would be free-of charge or cost reduced subscription periods, free storage space. Placed on Activities, motivational control addresses existing participants in the activity. For example, the platform operator can financially motivate existing participants to produce services which are of strategic relevance to the platform in a certain segment, e.g., finance. In the equation on network attractiveness (equation 3.18), this would enhance sensitivity to target groups with specific interest in financial services. Placed on an influence, informative control addresses external participants. E.g., the backup service Dropbox motivates subscribed consumers to invite new participants through offering additional storage space. In that case, new participants are not attracted through network attractiveness but through increased trust, created through a player from his community. This increase of trust improves attractiveness (equation 3.16) and enforces network effect R1 in Figure 8. The disadvantage of motivational control as compared to informative control is that it is not resource neutral.

The functional design requirements resulting from the above are as follows. Prescriptive control shall be located exclusively in all influences and activities, within the control area. The mechanism shall be set false by default. It may be activated by the modeler (Table 30).

Motivational Control	
Real-life condition	**Functional Design Requirements**
Motivational control aims at steering ecosystem participants towards the accomplishment of specific outcomes through rewards.	Motivational control is a control mechanism. It shall be located exclusively in all influences and activities within the control area. It shall be set false by default. It may be activated by the modeler.

Table 30: Summary table of functional design requirements for motivational control

5 Dyno - Model and Notation

This Chapter describes the engineering of the graphical modeling language Dyno. Its grammar embraces all conceptualizations presented in Chapter 4. In explicit, it integrates all structural and process elements, as well as the introduced control mechanisms. The artifacts defined in this Chapter are located on level M2 of the model stack, with concrete models from level M0 (Table 31). For the sake of clarity, the Chapter describes Dyno's abstract morphology and semantics together.

Level	Model	Graphical Language Engineering	In specific: Dyno
M3	Meta-Meta-Model. Abstract level to define M2	Self-referring UML model for the meta-modeling languages applied in M2	
M2	Meta-Model. Defines the structure of a model, i.e. classes, attributes and associations.	Abstract Syntax Defined through an UML model (potentially with OCL complements), platform independent	Meta-Model Defined through UML and OCL

Table 31: Model stack including Dyno Meta-Model

After defining aspects of norms, scope and conformance to the language (Section 5.1), the Chapter continues with Dyno's abstract morphology and semantics (Section 5.2) and syntax (5.3), followed by a modeling scenario (Section 5.4). Extending the overall scope of the graphical modeling language, Section 5.5 introduces a pattern language and repository, which enables modeling, based on reusable building blocks. The Chapter closes with a guide on how to apply the Dynamic Network Notation (Section 5.6).

5.1 Precepts

To allow for a language implementation in compliance with the subsequent specifications, the following 3 sub-sections stipulate the norms, applied in the specification, the scope of language design and the limits of conformance, when implementing the language.

5.1.1 Norms

The language specification in the following Chapters references the following normative, dated documents. Those documents as at the specified date are therefore provisions to the Dyno language and editor specifications. In the remainder, the text does not further explicitly reference these documents:

- OMG Unified Modeling language, v.2.0 for meta modeling and meta meta modeling [104];
- OMG Object Constraint language, v.2.3.1 [129] for meta modeling;

IEFT request for comment document RFC2119 on key words *must, must not, required, shall, shall not, should, should not, recommended, may and optional* [126]. Using these key words for language specification, the remainder of the Chapter deliberately uses a prescriptive voice;

5.1.2 Scope

The Dynamic Network Notation focuses on essential aspects of the platform design process. It provides a language and process support to platform architects and solution managers in order to model a platform's surrounding business ecosystem (e.g., service providers, consumers, competitors) and to model suitable structures and control mechanisms in order to harness the platform's network effects. It further allows the evaluation of design alternatives. In this context, the present work understands service management as managing the whole service life-cycle on an operational level, including the service strategy, service design, service transition, service operation and continual service improvement [130]. It explicitly includes the management of service provisioning and consumption. The scope is purely at the executive level and does not include board related governance task such as corporate strategy formulation.

The scope further includes a service platform pattern langauge providing a common vocabulary for platform architects and solution managers to communicate and document concepts as well as to explore management alternatives and which serves as a reusable base of expertise through solutions retrieved from experience.

5.1.3 Conformance

To reach compliance, implementations of the Dynamic Network Notation shall follow abstract morphology and syntax exactly as defined in the Sections 5.2 and 5.3. As explicitly permitted by the meta-model, a compliant implementation of Dyno may include additional properties that are specific to certain modelers' needs or intentions. The semantics of Dyno must be fully followed; however it could be extended through additional properties.

The present work does not prescribe any concrete morphology or syntax and leaves the choice of the modeling environment open to the implementer. Also, analytics features on top of the Dyno logic are purely optional. The pattern language and repository, introduced in Section 5.5 is also not part of the core Dyno specification but may be implemented as a supportive mechanism for enhanced modeling effectiveness.

5.2 Abstract Morphology and Semantics

This Section introduces representation of all Dyno elements together with their semantic annotation. The Section starts off with an explanation of the visualization concepts used in Dyno (Subsection 5.2.1). It continues with a specification of the areas of staged stakeholding power (Subsections 5.2.2 - 5.2.5). Thereafter, the Section specifies nodes (Subsections 5.2.6 and 5.2.7) as well as edges (Subsesctions 5.2.8 - 5.2.10). The Section closes with the specification of control mechanisms (5.2.11).

5.2.1 Design Concepts

The language aims to serve its target users, solution managers and platform architects, following the *Definition 12* and *Definition 13* in Subsection 19. It shall provide support when conceiving platform-arrangements and ecosystems. At the same time it shall create a point of departure for implementation or modification of a technical platform design. For the sake of effectiveness and given the different background of the addressed users, graphical presentation needs to be equally intuitive for users with a business background, who are traditionally familiar with flow-chart-oriented formats and for users with an IT background, acquainted to modeling languages like UML or Graphs. The BPMN symbolism provides an intersection, being used by both target groups. Dyno's symbolism is therefore close to the BPMN 2.0 conversation elements, i.e. the symbols for Participants, Activities and Gateways (see next Section). Alternatives of language design would have been a stereotype-based UML extension or a language definition based on activity diagrams. Several aspects advocate the chosen solution. First decision factor is the favored proximity to the BPMN representation. Second, the required inclusion of both structural aspects (areas, control mechanisms) as well as behavioral aspects (influences, transactions, loops) excludes the choice of UML; UML exclusively allows either the one or the other approach.

Dyno shall allow insight with respect to visualization, communication and decision making. It also has a dimension of explanation. However, as opposed to quantitative modeling in explanatory models, explanation is restricted to the *who* and *what*, without investigating on *how many*. Quantitative modeling would neither be possible nor useful as seen in Chapter 6: first, much of the information required for modeling is not available; second, due to the exponential behavior of network attractiveness, small mistakes in the basic assumption may lead to erratic results.

Dyno is conceived in a focused way on domain sets and functional properties related to service platforms, their ecosystem and the respective network effects. This orientation and limitation promotes expressiveness.

Complex requirements on modeling network and control features as pointed out in the previous Chapter advocate the choice of a graphical language. The proper use of graphical visualization allows for amplified cognition of the above described relations and structural properties through visual data representation and hence improves effectiveness of the language.

Dyno's visualization is based on Card, Mackinlay et al.'s [131] six axes of cognition amplification:

Increasing the user's memory and processing

From a psychological perspective, visualization can help exceed the user's intake capacity through the brain's capacity of parallel perceptual processing of visual attributes as well as the offloading from the cognitive to the perceptual system. This feature however is immanent to graphical representation in general and not specific to the Dynamic Network Notation.

Reduction of search effort for information

The Dyno-grammar gives guidance in producing the right information in the right moment through its context specific representation of symbols, reinforced by the blinding out or deactivation of unsuitable properties. In consequence, the user receives customized information for each specific modeling step. Grouping and visually relating information in graphical blocks reduces search.

Enhanced detectability of patterns

Simplified visualization and selective omission of data allows the creation of an enhanced overview perspective about the situation on network effects. In particular the domain-specific nature of Dyno allows for all data, which does not contribute to the target of depicting network effects around service platforms to be omitted from the domain set. To allow for additional user support, the underlying grammar allows for graph-theoretical analysis, when implemented in an editor.

Enablement of perceptual inference operations

Perceptual inference is the human's capability to infer something based on a visual stimulus. Due to its focus on network effects, Dyno choses network representation for making loops or potential for loops obvious. However, networks are weaker in expressing hierarchies than tree structures

[14], e.g., feature modeling. The requirement elicitation highlights that hierarchal relations play an important role, i.e. the gradation of stakeholding power depending on location of a vertex or an edge. As a work-around, the Dynamic Network Notation needs to enhance perceptual inference graphically:

- Centering the strongest area of staged authority (control area) and positioning the areas of staged stakeholding power around.
- Circles symbolize buttons, representing control mechanisms and highlighting points of manipulation. An activated control mechanism is filled black, inactivated mechanisms remain white.
- The infinity symbol depicts the property of scalability.
- Line type (continuous or dotted line), line width and degree of color (black, levels of grey) reflect an order of importance.

Using perceptual monitoring

The Dyno syntax enforces specific changes in representation based on spatial movement of elements. E.g., graphically displayed control mechanisms disappear once a participant is dragged from the control area to the influence area. The operator perceives the change while moving objects and draws conclusions.

Manipulatable Medium

The models created with Dyno allow exploration and analysis and modification of user operations.

To better express the morphology of a language, Bertin [106] suggests a vocabulary describing the techniques to graphically encode information:

(a) *Marks*: points, lines, areas

(b) *Positional techniques:* 1-D, 2-D, 3-D. Dyno provides a 2-D spacial visualization, which allow to schematize structural allocation of elements within the areas of staged stakeholding power. The Dyno specification requests a third dimension to display properties per one element, activated through mouse-click. The specification leaves the designer of the modeling environment at liberty as to whether to place this third dimension into a drop-down menu next to the active element or into a configuration panel. The specification limits the display of attributes per element to avoid information overflow and to enforce focused analysis.

(c) *Temporal technique:* animation. The specification requests mandatory dynamic adaptation of elements, when being moved around by the modeler. For example the *controllable* - circle disappears, when the modeler moves the participant from the control area to the influence area. The modeler receives a visual stimulus. The participant group bounces back, when the modeler tries to move it from the influence area into the control area. This dynamic effect communicates the exclusion of this operation.

(d) *Retinal techniques:* Color, shape, size, saturation, texture. Dyno choses a shape set which is familiar to those modelers, which are experienced with BPMN. It further gives the modeler scope of freedom to color elements. For example, he could model more important base values in a different color than the less important ones.

During the design of morphology in the remainder of the Chapter, the text uses these terms and comments on the applied graphical encoding.

5.2.2 Control Area

Semantics

The Control Area defines the space, where the platform operator can exert full control over all participants, over all their own infrastructure and services and over third party service in the frame of contractually agreed legal frame-set. The Control Area needs to be modeled within the Influence Area.

Symbol description

The basic symbol shall be a rectangle with

- Background default color white RGB-code 255-255-255, Hex-code ffffff, the background color may be modified in a model as retinal technique for highlighting.
- Standard width: 740 pt, standard height 540 pt, may be resizable during the modeling process as retinal technique for highlighting.
- Surrounding line: continuous, 2 pt, gray RGB-code 190-190-190, Hex-code #bebebe.
- Label: font orientation h: bottom, v: right, size 22 pt, color gray RGB-code 190-190-190, Hex-code #bebebe, font type not defined.
- Non-true to scale example given in Figure 14.

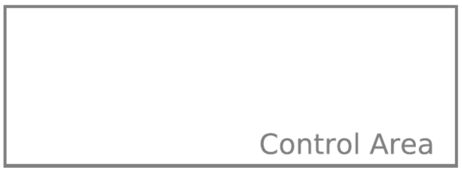


Figure 14: Control area

5.2.3 Influence Area

Semantics

The Influence Area delineates the space around the Control Area, where the platform operator can exert degrees of influence on participants. This area is out of scope for control activities. The Influence Area needs to be modeled within the Noise Area.

Symbol description

The basic symbol shall be a rectangle with

- Background default color white RGB-code 255-255-255, Hex-code ffffff, the background color may be modified in a model as retinal technique for highlighting.
- Standard width: 1000 pt, standard height 800 pt, may be resizable during the modeling process as retinal technique for highlighting.
- Surrounding line: dashed, 2 pt, gray RGB-code 190-190-190, Hex-code #bebebe.
- Label: font orientation h: bottom, v: right, size 22 pt, color gray RGB-code 190-190-190, Hex-code #bebebe, font type not defined.
- Non-true to scale example given in Figure 15.

Comments: none

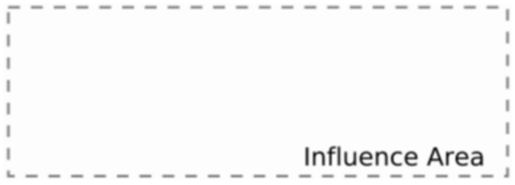


Figure 15: Influence area

5.2.4 Noise Area

Semantics

The Noise Area describes the zone, where the platform operator has no influence. With the consecutive areas (Influence Area, Control Area) it forms an ordinal order of graded controllability, where Influence Area is positioned at the medium and Control Area at the highest rank.

Symbol description

The Noise Area does not have a representation. The basic canvas of the modeling environment should be considered Noise Area.

Comments: none

5.2.5 Divisions

Semantics

Divisions help to divide the control area into structural units. That might be organizational units or geographical units e.g., rented compute and storage resources on an Infrastructure-as-a-Service or an own server, physically placed in a customer or partner location. When modeled by a consortium, division may give additional means to differentiate between specific areas of responsibility or ownership by specific partners (e.g., division of consortium member 1, division of consortium member 2, …).

Divisions carry a scalability attribute. This property can be set to true for structural units (divisions) within the control area, which are specially conceived for rapid scale behavior. As a Boolean attribute, it can only serve as qualitative indicator, pointing the modeler's attention through retinal technique to specific areas, where he has to be explicitly considerate when designing the technical environment. These scalable environments may be required by activities within loops. Details may be textually formulated or referred to in form of document identification number or hyperlink.

The explorative analysis encountered cases where platform operators create finite numbers of control areas outside the platform's domain. Examples are own servers placed at customer sites (e.g., by S.Chand Edutech, the example is revisited in the evaluation), or controlled environments with prescriptive rights to the platform operator placed as native applications on client PCs e.g., Google Drive [132]. Division groups allow the bundling of these groups without obliging the modeler to quantify the number of applications placed.

Symbol description

The basic symbol shall be a rectangle with

- Background default color white RGB-code 255-255-255, Hex-code ffffff, the background color may be modified in a model as retinal technique for highlighting.
- Standard width: 200 pt, standard height 750 pt, may be resizable during the modeling process as retinal technique for highlighting.
- Surrounding line: continuous, 1 pt, gray RGB-code 190-190-190, Hex-code #bebebe.
- Label: font orientation h: bottom, v: right, size 22 pt, color gray RGB-code 190-190-190, Hex-code #bebebe, font type not defined.

Conditionally visible properties:

- Scalability,
- Symbol for property: '∞', font size16 pt, color gray RGB-code 190-190-190, Hex-code #bebebe, font type not defined, position: top left corner of element.

Comments: none

Special version of the symbol: Division Groups

- Special Semantics: Dyno suggests the symbol 'division group' for the ease of grouping sets of many areas of similar nature (e.g., remote virtual servers at customer location).
- Deviating representation: Division Groups shall be depicted through 3 overlaid Division symbols. In cases where scalability = true, the infinity symbol shall be placed on the overhead Division symbol.
- Non-true to scale example given in Figure 16.

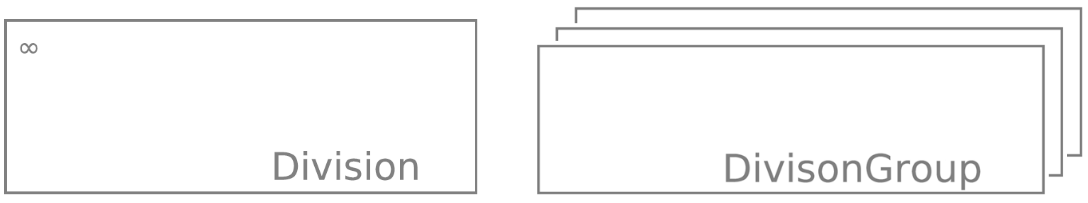


Figure 16: Division (displayed with *Scalability*-symbol) and Division Group

5.2.6 Participants

Semantics

Participants describe specific entities without stock behavior. They are considered static in short term view. This does not rule out linear evolution (e.g., the entity *service development* hires new personnel or/and develops new services). Platforms include the attribute *base value*. A base value is placed on and can be set true for participants within the control area. Base Values of a platform ecosystem are those values which the modeler considers to be valuable enough to incite a network effect. Base values may vary during different modeling stages and depend on the goal to be accomplished. A participant further carries the attribute *controllable*. If it is located in the control area it is symbolized by a circle, depicted in the element's representation. In that case it can carry one or more control mechanisms of prescriptive control (Table 22). The controllability of internal participants originates from their subordination to the platform operator. A participant within the control area can be internal entities like departments or workgroups, but also external suppliers which work on contractual assignment. Details may be textually formulated or referred to in form of document identification number or hyperlink. A base value on a participant may be required to start off a causal loop.

Symbol description

The basic symbol shall be a rounded rectangle with

- Background default color white RGB-code 255-255-255, Hex-code ffffff, the background color may be modified in a model as retinal technique for highlighting.
- Standard width: 150 pt, standard height 100 pt; may be resizable during the modeling process as retinal technique for highlighting.
- Surrounding line: continuous, 1pt, black RGB-code 0-0-0, Hex-code #000000.
- Label: Font orientation h: middle, v: center, size 12 pt, color black RGB-code 0-0-0, Hex-code #000000, and font type not defined.

Conditionally visible properties (depending on syntax):

- Base-value.
    - When not activated, no base value symbol is visible.
    - Symbol for property when activated: 'ß', font size16 pt, color RGB-code 0-0-0, Hex-code #000000, font type not defined, and position: top right corner of element.
- Controllability:
    - Symbol for property: circle, stroke: black, 1pt, position: top left corner of the element.

- background color when no control mechanism activated: white RGB-code 255-255-255, Hex-code #ffffff;
- background color when one or more control mechanisms are activated: black RGB-code 0-0-0, Hex-code #000000.

Comments

- Dyno adopts the participant symbol from BPMN Conversation, however with rounded angles to create sufficient differentiation from the divisions.

Special version of the symbol: Participant Groups

- Deviating semantics: The participant groups do not address explicit entities but groups of implicit nature with indistinct boundaries, e.g., group of users requiring a specific service.
- Deviating representation: Participants groups shall be depicted through 3 overlaid Participant symbols.
- Non-true to scale example given in Figure 17.

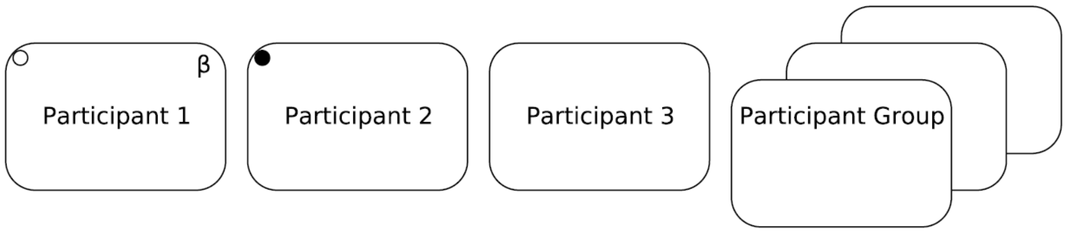


Figure 17: Participant representations.
Participant 1 (controllable, no control mechanism placed, base value activated);
Participant 2 (controllable, one or more control mechanisms placed, no base value);
Participant 3 (uncontrollable, therefore no option for base value);
Participant Group (uncontrollable, therefore no option for base value).

5.2.7 Activities

Semantics

Activities group sets of micro-activities e.g., service development, service consumption or service deployment. Activities shall represent a stock that has accumulated in the past and which might increase, stagnate or decrease in quantity in the future. As by definition, an activity can only take place in the control area, the symbol mandatorily carries the circle in the upper left corner, as it is always controllable. An activity can carry a finite number of control mechanisms of prescriptive control, sanctional control, informative control and market –regulative control (Table 22). If one or more of them are activated, the circle is filled. Activities also carry the attribute *provisions* to describe or refer to applicable terms and conditions. This attribute however is not visible and requires a modeling environment with the option to visualize it in the configuration panel or drop down menu. Configuration panel or drop down menu represent a third dimension according to Mackinlay [15].

Symbol description

The basic symbol is an equal-sided rectangle with

- Background default color white RGB-code 255-255-255, Hex-code ffffff.
- Background color may be variable during the modeling process as retinal technique for highlighting.
- Surrounding line: continuous, 1pt, black RGB-code 0-0-0, Hex-code #000000.
- Standard width: 100 pt, standard height 100 pt; may be resizable during the modeling process as retinal technique for highlighting importance.
- Label: Font orientation h: middle, v: center, size 12 pt, color black, font type not defined.

Visible property:

- controllable
    - symbol visible by default:
    - symbol for property: circle, stroke: black, position: top left corner of the element.
    - background color when no control mechanism activated: white RGB-code 255-255-255, Hex-code #ffffff;
    - background color when one or more control mechanisms are activated: black RGB-code 0-0-0, Hex-code #000000.

Conditionally visible properties (depending on syntax):

- Base-value.

  - When not activated, no base value symbol is visible.
  - Symbol for property when activated: ‘ß’, font size16 pt, color RGB-code 0-0-0, Hex-code #000000, font type not defined, and position: top right corner of element.
- Non-true to scale example given in Figure 18.

Comments

- To depict the ‘Activity’, Dyno adopts the communication symbol from BPMN conversation.

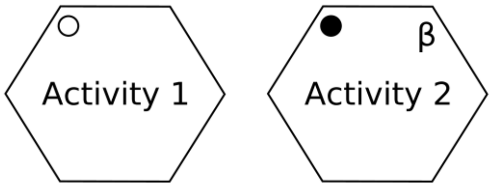


Figure 18: Activities;
Activity 1 (no control mechanism, no base value),
Activity 2 (one or more control mechanisms placed, on base value activated)

5.2.8 Transaction

Semantics

Transactions describe a flow of value, which is transferred

a) from a participant or participant-group to an activity or
b) from an activity to an activity or
c) from an activity to a participant inside the control area.

The options a) and b) describe typical flows in the sense of a migration, which accumulates a stock. A transaction can theoretically be negative. This describes a drain. Option c) allows for the modeling of directed process chains within the control area, e.g., when a participant retrieves information. It is an auxiliary construct to increase the modelers’ scope of expression but is not central to Dyno’s conception.

Transactions mandatorily carry the attribute ‘controllable’. It turns true if one or more restrictive control mechanisms are activated (Table 22). Transactions also carry the attribute *provisions* to describe or refer to applicable terms and conditions. This attribute however is not visible and requires a modeling environment with the option to visualize it in a third dimension the configuration panel or drop down menu.

Symbol description

Transactions shall be displayed as arrows with a continuous line. The direction shall be indicated at the target-side through a circle.

A transaction must be depicted with a label ‘Transaction’, first letter capital, all other letters small. Similarly to the controlled elements, a circle is placed at the target side on the symbol.

Graphic features:

- Line: continuous, 1pt, black RGB-code 0-0-0, Hex-code #000000.
- Label: Font orientation should be sitting middle-out on the arrow, size 12 pt, color black RGB-code 0-0-0, Hex-code #000000, font type not defined.

Visible property:

- Controllability:
    - symbol for property: circle, stroke: black, 1pt, position: should be near the target of the arrow.
    - background color when no control mechanism activated: white RGB-code 255-255-255, Hex-code #ffffff;
    - background color when one or more control mechanisms are activated: black RGB-code 0-0-0, Hex-code #000000.
- Non-true to scale example given in Figure 19.

Comments: none

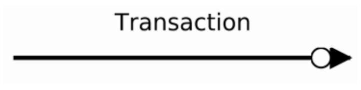


Figure 19: Transaction (no control mechanism)

5.2.9 Influence

Semantics

Influences describe efforts of indirect control over participants or participant groups, located in the influence area. From the business dynamics point of view, they correspond to rates, which influence value flows i.e. transactions. Sources to influences can be activities, participants and participant groups. To guide the modeler towards the design of causal loops, the language implements the construct for directed edges. It only allows activities to directly influence participant groups or participants. Participants or participant groups are allowed as part of a set of sources, aggregated through a merging gateway. The modeler might circumvent the guiding mechanism through concatenation of one or more influences pointing at a gateway, which addresses them to a participant group. Through the lack of activity, no causal loop can be created. Within a modeling environment, subsequent analysis could detect such inert constructions.

Influences carry a controllability attribute to display, whether or not they can be controlled. The relevant Boolean status is defined by the location of the influences' source. If the source of an influence is located in the control area, the controllability attribute is true and the element carries the circle depicting controllability. In such cases, the influence can carry a finite number of control mechanisms of motivational control, informative control and market–regulative control (Table 22). If one or more of them are activated, the circle is filled.

Symbol description

Relationships shall be displayed as arrows with a continuous line. The direction shall be indicated at the target-side through a filled arrow head.

An influence must be depicted by the label 'Influence', first letter capital, all other letters small. Similarly to the controlled elements, a screw head is placed at the source- or target side on the edge, whenever the attribute controllability is set to the value 'true'. Specific features:

- Line: continuous, 1pt, black RGB-code 0-0-0, Hex-code #000000.
- Label: Font orientation should be sitting middle-out on the arrow, size 12 pt, color black RGB-code 0-0-0, Hex-code #000000, font type not defined.

Conditionally visible properties (depending on syntax):

- Controllability:
    - symbol for property: circle, stroke: black, 1pt, position: should be near the source or target of the arrow depending on the syntax;
    - background color when no control mechanism activated: white RGB-code 255-255-255, Hex-code #ffffff;
    - background color when one or more control mechanisms are activated: black RGB-code 0-0-0, Hex-code #000000.
- Non-true to scale example given in Figure 20.

Comments: none

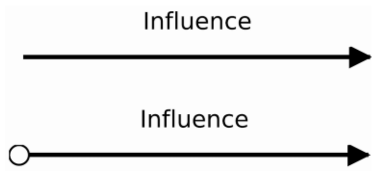


Figure 20: One uncontrollable Influence and a controllable Influence without control mechanism

5.2.10 Gateways

Semantics

A Gateway is a synchronizing merge gate, awaiting all incoming edges to complete before triggering one or more outgoing edges.

Symbol description

Gateways shall be represented by the diamond symbol with an included '+'sign. Details:

- Standard width: 50 pt, standard height 50 pt; may be resizable during the modeling process as retinal technique for highlighting.
- Surrounding line: continuous, 1pt, black RGB-code 0-0-0, Hex-code #000000.
- 'Plus' symbol centered within the diamond, height and width approx.36 pt, line: 3 pt.
- No labeling, no resizing.
- Non-true to scale example given in Figure 21.

Comments

Dyno uses the merging gateway symbol from BPMN 2.0.

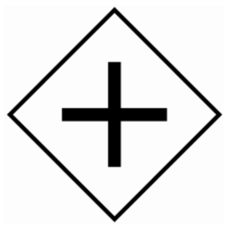


Figure 21: Gateway

5.2.11 Protagonist Control

Dyno models from a specific protagonist's view point. A complete Dyno model with Participants, Activities and Relationships shall be interpreted as control center for the protagonist. For the sake of clarity, Dyno represents control mechanisms as aggregated control points, displayed through one only circle. In an editor, configuration panels or drop down menus shall display the configurable attributes. The following list describes the included attributes, which are based on Chapter 4 (Table 3 and Table 5).

Control Mechanisms

- Prescriptive control is the mechanism to control participants and activities inside the control area. Prescriptive control means that the platform operator can fully prescribe the steps to take in activities and participants on the platform. Details and related provisions may be textually described or referred to in form of document identification number or hyperlink.
- Restrictive control is the mechanism to control inbound transactions from third parties. Third parties may be e.g., customers, suppliers. Restriction may be based e.g., on compliance level of an inbound transaction to the platform provisions. Details and related provisions may be textually described or referred to in form of document identification number or hyperlink.
- Sanctional control incites an escalation routine at any time an activity exhibits a certain level of incompliance to policy or regulations of the platform operator. Details and related provisions may be textually described or referred to in form of document identification number or hyperlink.
- Motivational control groups all incentivizing activities towards Participants within the Control Area. Details may be textually described or referred to in form of document identification number or hyperlink.
- Informative control gives the suppliers (customized) information on consumer preferences, requirements, etc. and aims at supporting the platform operator towards an optimization of his services and eventually of the whole service portfolio. Details may be textually described or referred to in form of document identification number or hyperlink.
- Market-regulative control uses feedback from consumers to exert control (i.e. through reputation and recommender systems). Details may be textually described or referred to in form of document identification number or hyperlink.

5.3 Abstract Syntax

Following Definition 42, abstract syntax gives a high level definition of syntax, leaving out particularities to technical implementation, but precise enough to describe representation of and production rules for actual utterances (i.e. of graphical models). It builds on the functional design specifications, derived from the conceptual model in Chapter 4. It complements those with specifications, originating from theory of language engineering (Section 3.1) and as well as from theoretical design concepts as described in Subsection 5.2.1.

The present work uses UML as a meta-language to display the abstract syntax' assembly rules conforming to the model stack of Chapter subsection 3.1.3. According to Definition 27, syntax needs to additionally prescribe the adaptation of graphic representation in function of specific conditions, e.g., the case-dependent representation of the control points on nodes and edges. Therefore, OCL complements UML to prescribe adaptation of graphical representation in design time and in function of the context specific syntactical requirements.

In the UML meta-model (Figure 22) classes give the meta-view on the elements *nodes* and *edges*, related and conjugated through production rules. The classes encapsulate characteristics, immanent to those elements as attributes. Some attributes define basic properties immanent to every element i.e. *identification number (id), name* and *documentation*. There are further attributes, which are only carried by specific elements, i.e., *base-value, controllable, controllableSource, controllableTarget, scalability, location* and *provisions*. An important property to a subset of nodes and edges in the context of service management are the control mechanisms, i.e. *prescriptive control, sanctional control, restrictive control, informative control, market regulative control and motivational control*. The meta-model depicts the control mechanisms not as properties but as classes. Subsection 8.3.6 (Control Mechanisms) details the chosen solution.

Considering the meta-model without the *ExtensionAttribute*-Class, it designs a regular grammar compliant to Definition 33. In a regular grammar, terminals on the right side produce a single terminal on the left side, e.g., A→wB. However, the Extension Attribute is nested in the root element (and may hence be called up recursively). This makes the grammar context-free, according to Definition 32. In a context-free grammar, terminals and non-terminals on the right side produce a single non-terminal on the left side, e.g., A→wAB.

The following subsections detail important parts of the meta-model. For better clarity, the subsections revisit only selected fractions of the overall meta-model (Figure 22). Subsection 5.3.1 starts with an introduction to root element and the core meta-model. Subsection 5.3.2 introduces edges and nodes and their relationship in detail. Subsection 5.3.3 explains how the meta-model handles control mechanisms.

Following the conventions applied by [133], the present work begins class names with a upper case and attributes with a smaller lower case letter. OCL conditions under a class name describe conditions for a class instantiation. E.g., *Division {{OCL} self.location = controlArea}* describes that necessary condition for a division is that its location is within the control area. OCL conditions stated behind an attribute describe conditions for an attribute to appear. E.g., *controllable: Boolean {{OCL} self.location = controlArea}* defines that an element only carries this attribute, if it is located within the control area.

When speaking of classes of the meta-model, the present work uses this differentiation of upper case and lower case. When speaking about the actual (instantiated) elements in the context of models, the text refers to those purely in lower case.

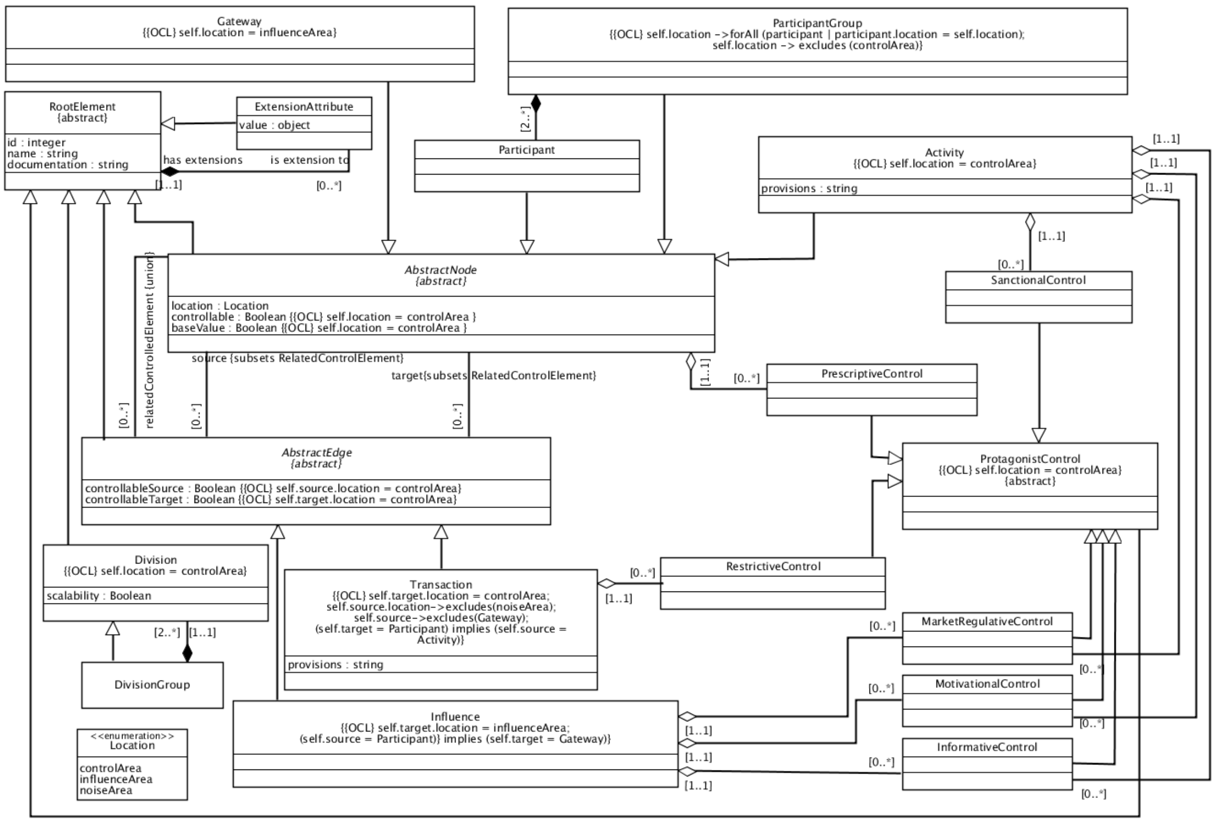

Gateway
{{OCL} self.location = influenceArea}
ParticipantGroup
{{OCL} self.location ->forAll (participant | participant.location = self.location); self.location -> excludes (controlArea)}
RootElement
{abstract}
id : integer
name : string
documentation : string
ExtensionAttribute
value : object
has extensions
is extension to
[1..1]
[0..*]
[2..*]
Participant
Activity
{{OCL} self.location = controlArea}
provisions : string
[1..1]
[1..1]
[1..1]
[1..1]
[0..*]
AbstractNode
{abstract}
location : Location
controllable : Boolean {{OCL} self.location = controlArea }
baseValue : Boolean {{OCL} self.location = controlArea }
SanctionalControl
relatedControlledElement {union}
source {subsets RelatedControlElement}
target{subsets RelatedControlElement}
[1..1]
[0..*]
PrescriptiveControl
[0..*]
[0..*]
[0..*]
AbstractEdge
{abstract}
controllableSource : Boolean {{OCL} self.source.location = controlArea}
controllableTarget : Boolean {{OCL} self.target.location = controlArea}
ProtagonistControl
{{OCL} self.location = controlArea}
{abstract}
Division
{{OCL} self.location = controlArea}
scalability : Boolean
Transaction
{{OCL} self.target.location = controlArea;
self.source.location->excludes(noiseArea);
self.source->excludes(Gateway);
(self.target = Participant) implies (self.source = Activity)}
provisions : string
[0..*]
[1..1]
RestrictiveControl
[2..*]
[1..1]
DivisionGroup
MarketRegulativeControl
[0..*]
[0..*]
MotivationalControl
[0..*]
[0..*]
<<enumeration>>
Location
controlArea
influenceArea
noiseArea
Influence
{{OCL} self.target.location = influenceArea;
(self.source = Participant)} implies (self.target = Gateway)}
[1..1]
[1..1]
[1..1]
InformativeControl
[0..*]
[0..*]

Figure 22: Complete Dyno meta-model

5.3.1 Root Element and Core Meta-Model

All elements in the Dyno meta-model apart from the enumeration *Location* inherit their basic features from the abstract class *RootElement* (Figure 23). Apart from the RootElement, the meta-model has 3 more important abstract classes, all inheriting from the *RootElement*: the *abstract-Node* as generalization of all nodes in the meta-model, the *abstractEdge*, generalizing all edges and the *ProtagonistControl* class, generalizing all control mechanisms and representing the control center.

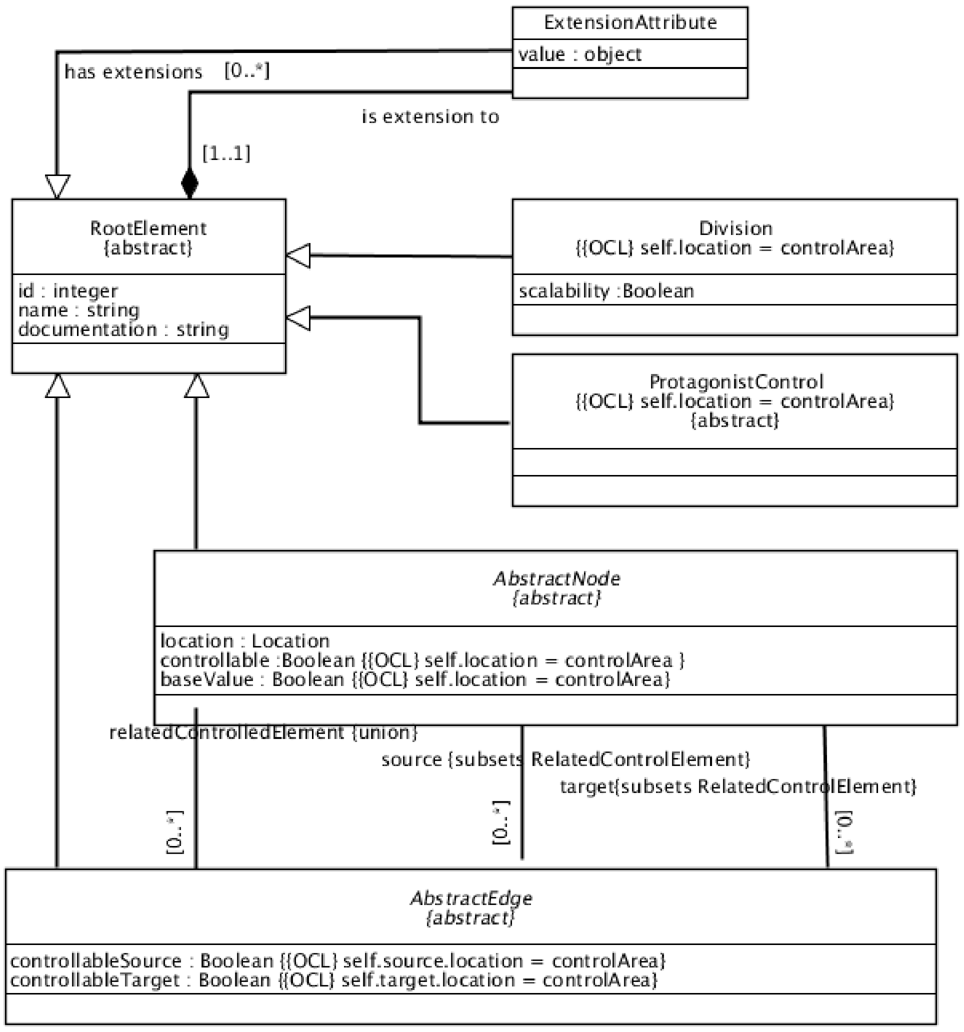

ExtensionAttribute
value : object
has extensions [0..*]
is extension to
[1..1]
RootElement {abstract}
id : integer
name : string
documentation : string
Division {{OCL} self.location = controlArea}
scalability :Boolean
ProtagonistControl {{OCL} self.location = controlArea} {abstract}
AbstractNode {abstract}
location : Location
controllable :Boolean {{OCL} self.location = controlArea }
baseValue : Boolean {{OCL} self.location = controlArea}
relatedControlledElement {union}
source {subsets RelatedControlElement}
target{subsets RelatedControlElement}
[0..*]
[0..*]
[0..*]
AbstractEdge {abstract}
controllableSource : Boolean {{OCL} self.source.location = controlArea}
controllableTarget : Boolean {{OCL} self.target.location = controlArea}

Figure 23: Root Element and Core meta-model

Only the element name should be represented in the element in the case of a node. The modeling environment shall allow for the documentation string to be edited in a third dimension. The element identification number does not need to be displayed, if the modeling environment assigns and handles them automatically. The edges by default depict their character *Transaction* or *Influence*. The modeling environment can give the modeler the freedom to change these default names to customized ones.

In analogy to the BPMN meta-model [102], the Dyno meta-model is equipped with an extension attribute. The Dyno language provides core grammar and semantics to model networks in an around service platforms with particular focus on control. However, in cases of different or broader problem statements, there might be other features, which merit consideration, e.g., monitoring mechanisms or security mechanisms. The root element *extension attribute* allows them to be added as properties without violation of the Dyno specification. At the same time it prescribes how to add additional properties without damaging Dyno's grammar. However, including additional properties through the ExtensionAttribute-class risks producing semantic incorrectness. Producing semantically correct extensions to Dyno lies solely in the accountability of the language engineer. The meta-model rules out the inclusion of additional classes, as those might destroy the correctness of the Dyno Models.

5.3.2 Nodes and Edges

Figure 24 describes the main nodes and edges in the Dyno meta-model. Main nodes are the classes *Participant* and *Edges*. They are specializations of the superclass *abstractNode*. Main edges are *Influence* and *Transaction*, inheriting from the parent class *abstractEdge*.

To be able to differentiate between source and target of an edge, the meta-model subsets *relatedControlElement* into *source* and *target*. *RelatedControlElement* forms the union of both as no other associations between *abstractNode* and *abstractEdge* shall be allowed.

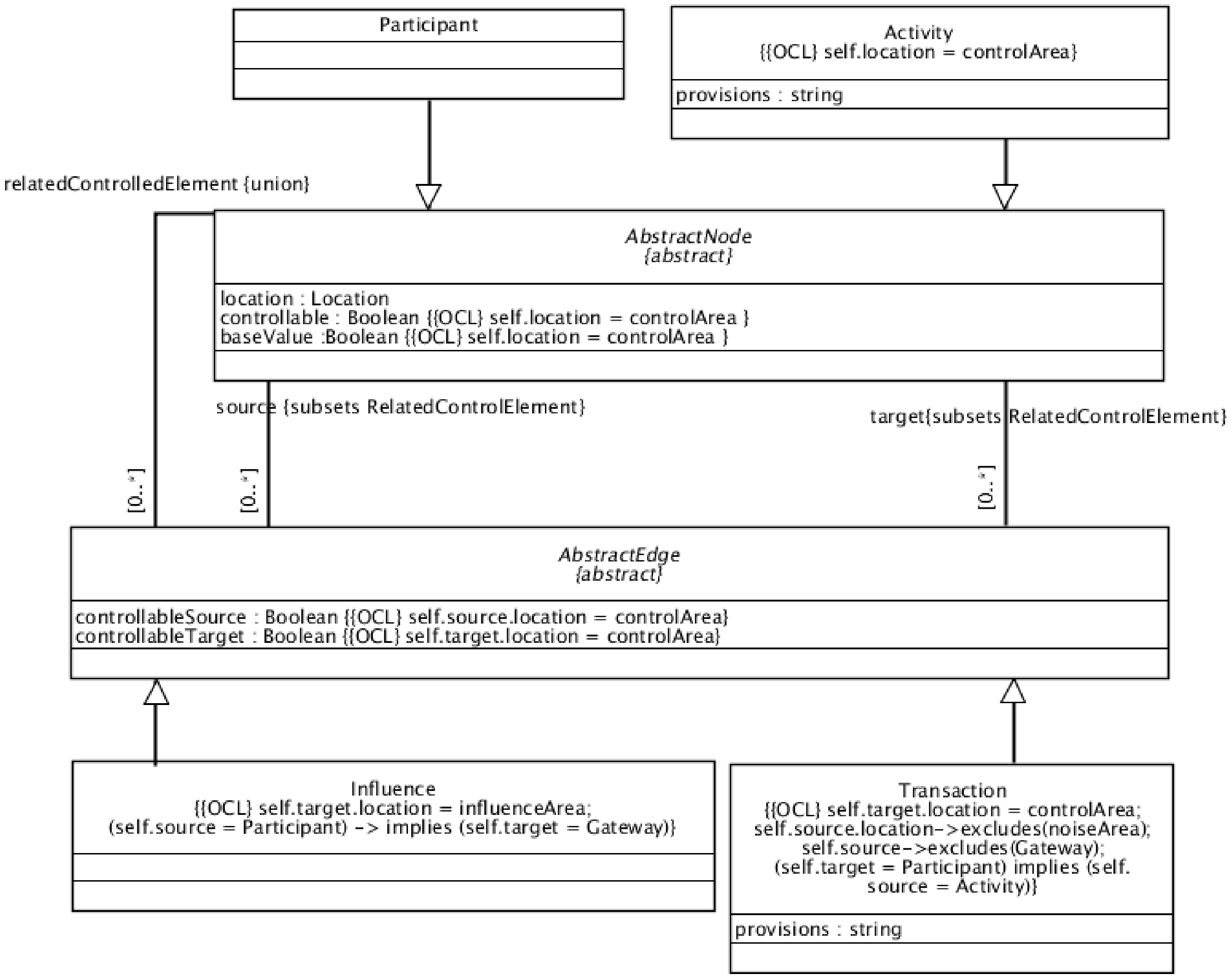

Participant
Activity
{{OCL} self.location = controlArea}
provisions : string
relatedControlledElement {union}
AbstractNode
{abstract}
location : Location
controllable : Boolean {{OCL} self.location = controlArea }
baseValue :Boolean {{OCL} self.location = controlArea }
source {subsets RelatedControlElement}
target{subsets RelatedControlElement}
[0..*]
[0..*]
[0..*]
AbstractEdge
{abstract}
controllableSource : Boolean {{OCL} self.source.location = controlArea}
controllableTarget : Boolean {{OCL} self.target.location = controlArea}
Influence
{{OCL} self.target.location = influenceArea;
(self.source = Participant) -> implies (self.target = Gateway)}
Transaction
{{OCL} self.target.location = controlArea;
self.source.location->excludes(noiseArea);
self.source->excludes(Gateway);
(self.target = Participant) implies (self.source = Activity)}
provisions : string

Figure 24: Core nodes and edges

Nodes

Figure 25 isolates the nodes from the meta-model. In addition to *Activity* and *Participant*, the superclass *AbstractNode* shall be specified into *ParticipantGroup* and Gateway. *Participant-Group* is an aggregation of [2..*] participants which shall all be based in the same area. An OCL constraint specifies this through

{{OCL}self.location → forAll {Participant.location = self.location}

{{OCL} self.location → forAll {Activity.location = self.location}

Through the *AbstractNode*, all children shall inherit the attribute location which refers to an enumeration of the location properties *controlArea*, *influenceArea* and *noiseArea (*Figure 25*)*. Through an OCL-based restriction of the property *controllable* to the *controlArea*, the syntax shall ensure that the *controllability* symbol only appears when the respective node is located within the control area:

controllable: Boolean {{OCL} self.location = controlArea}

The gateway shall be a synchronizing merging gate. As specified by the OCL constraint within the *AbstractNode*, a gateway must never be controllable. An additional OCL constraint restricts all activities to the control area. This follows the understanding that cooperation outside the control area can only be viewed as a black box activity within an external participant. For example, a platform operator cannot observe the loose coupling of services from the service provider. He can only observe the behavior of the resulting composite service, when it is called up by a requesting service that is deployed on the platform.

All activities can refer to defined service platform provisions. *Provisions* therefore are an attribute of the class *Activity*. Control mechanisms regulate the potential enforcement of these provisions.

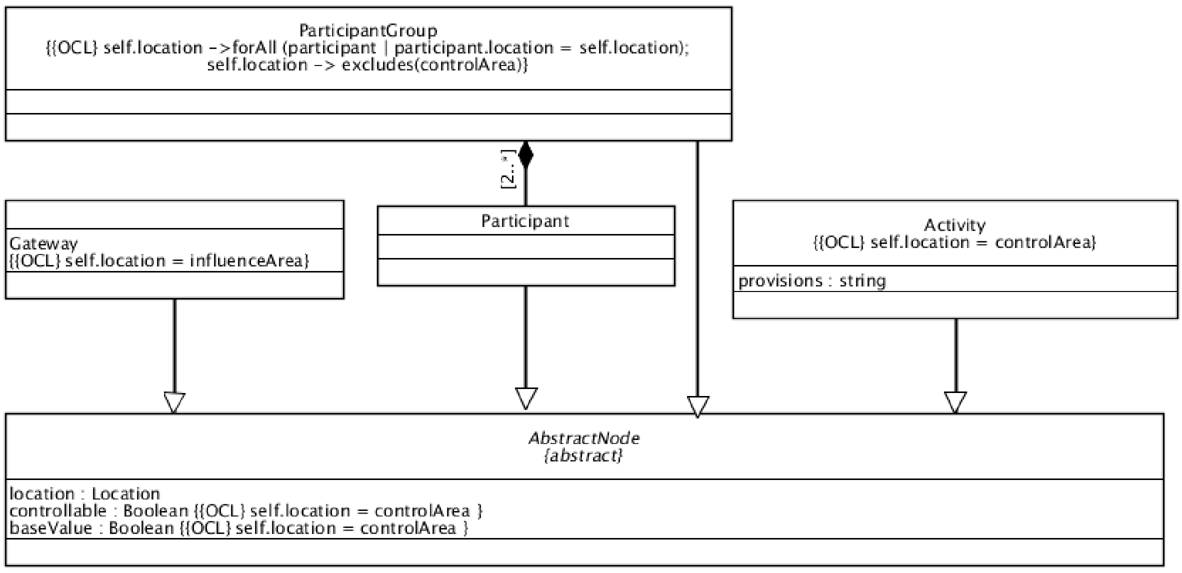

ParticipantGroup
{{OCL} self.location ->forAll (participant | participant.location = self.location);
self.location -> excludes(controlArea)}
[2..*]
Gateway
{{OCL} self.location = influenceArea}
Participant
Activity
{{OCL} self.location = controlArea}
provisions : string
AbstractNode
{abstract}
location : Location
controllable : Boolean {{OCL} self.location = controlArea }
baseValue : Boolean {{OCL} self.location = controlArea }

Figure 25: Nodes

Division and division group shall be modeled as structuring element within the boundaries of the control area (Figure 26). They shall not hold any logic. Division and division group possess the Boolean property ‘scalability’.

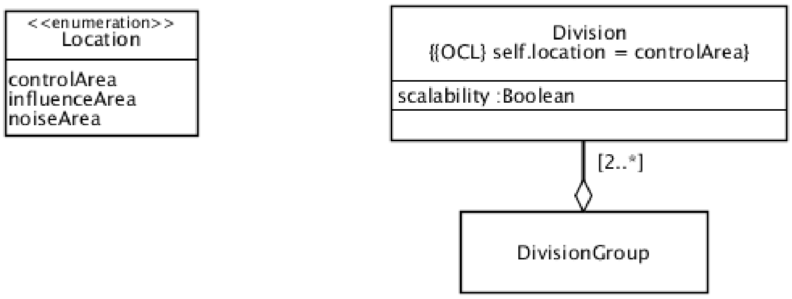


Figure 26: Location and divisions

Control area, influence area and noise area are modeled as an enumeration. In a modeling environment, they could allow for documentation in a third dimension. Also it is permitted to edit the area names.

Edges

As discussed on various occasions throughout the present work, the Dynamic Network Notation offers remains purely qualitative. Reasons for this are the unavailability of reliable market data and the danger of wrong conclusions based on slightly varying data, due to exponential behavior of equation 3.20, describing the causal loops. In consequence, the edges in Dyno do not carry any weighting.

Figure 27 shows the *AbstractEdge* with its two specializations, *Influence* and *Transaction*. Similar to controllability of nodes, controllability of edges shall be restricted to the control area for sources as well as for targets of edges:

controllableSource: Boolean {{OCL} self.source.location = controlArea}

controllableTarget: Boolean {{OCL} self.target.location = controlArea}

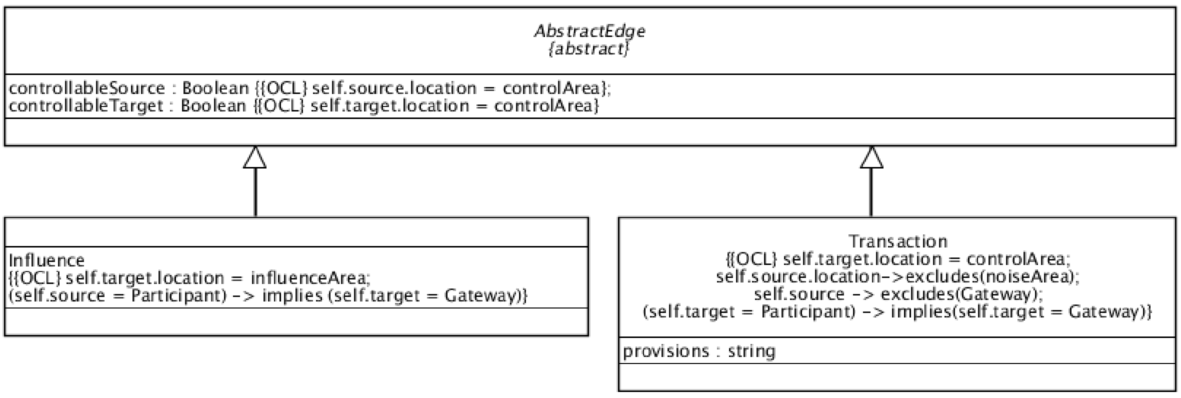

AbstractEdge
{abstract}
controllableSource : Boolean {{OCL} self.source.location = controlArea};
controllableTarget : Boolean {{OCL} self.target.location = controlArea}
Influence
{{OCL} self.target.location = influenceArea;
(self.source = Participant) -> implies (self.target = Gateway)}
Transaction
{{OCL} self.target.location = controlArea;
self.source.location->excludes(noiseArea);
self.source -> excludes(Gateway);
(self.target = Participant) -> implies(self.target = Gateway)}
provisions : string

Figure 27: Definition of edges

Influence and *Transaction* relate to each other through a logical exclusive-or disjunction. This facilitates subsequent realization in concrete syntax through an editor. The editor only needs to provision one single *super-edge*, embracing Influence and Transaction. In design time, the editor can then verify based on the concrete grammar, whether the representation shall be an Influence or a Transaction.

OCL constraints the Influence to targeting the influence area through the expression

{{OCL} self.target.location = influenceArea; …}.

Similarly it constraints the Transaction to point into the control area:

{{OCL} self.target.location = controlArea; ...}

The Transaction must in addition not originate within the noiseArea to rule out malicious value transfers:

{{OCL} ...; self.source.location → excludes(noiseArea); ...}

Also it makes semantically no sense to allow for aggregation of transactions outside the controlArea as the origin would be hidden to the platform operator. Therefore OCL constraints apply:

{{OCL} …; self.source → excludes(Gateway)}

Given the fact that participants do not accumulate (in contrast to activities), they could only be a side path in a causal loop. The grammar therefore rules out that participants alone influence participant groups in the influence area and direct them via a merging gateway.

{{OCL} …; (self.source = Participant) -> implies (self.target = Gateway)}

In a modeling environment, an analyzer should subsequently verify the involvement of an activity in loops. All transactions can refer to defined service platform provisions. Provisions therefore are an attribute of the class *Transaction*. Control mechanisms regulate the potential enforcement of these provisions.

5.3.3 Control Mechanisms

The UML meta-model depicts all control mechanisms as independent classes, although they could have been defined as *attribute of type string* to an element. The chosen approach is justified through several reasons. First, it expresses the control mechanisms belonging to the control center *ProtagonistControl* in the meta-model. *Protagonist control* is a generalization to all control mechanisms. Second, this approach emphasizes the fact that specific control mechanisms can be aggregated by several elements. Third, it gives the language the possibility to mature over the next releases. The control mechanisms are key to the Dynamic Network Notation. The isolated modeling allows language engineers to evolve them over time, e.g., through specific new properties, dependencies or association relationships (e.g., aggregations or compositions). For example, a subsequent release of the Dyno specification could equip market-regulative control with the options reputation system and recommender system, potentially regulated through constraints, specified in OCL. Another option would be to continuously add all emerging elements of recommender and reputation systems in the market to the modeling language.

AbstractNode (Figure 25) and *AbstractEdge* (Figure 27) limit controllability to those (parts of) elements which are within the control area. Table 22 prescribes the options which Dyno shall give to the modeller. The availability of such a control option must be limited to nodes and Sections of edges within the Control Area. Figure 28 describes the subset of the production rules related with control mechanisms.

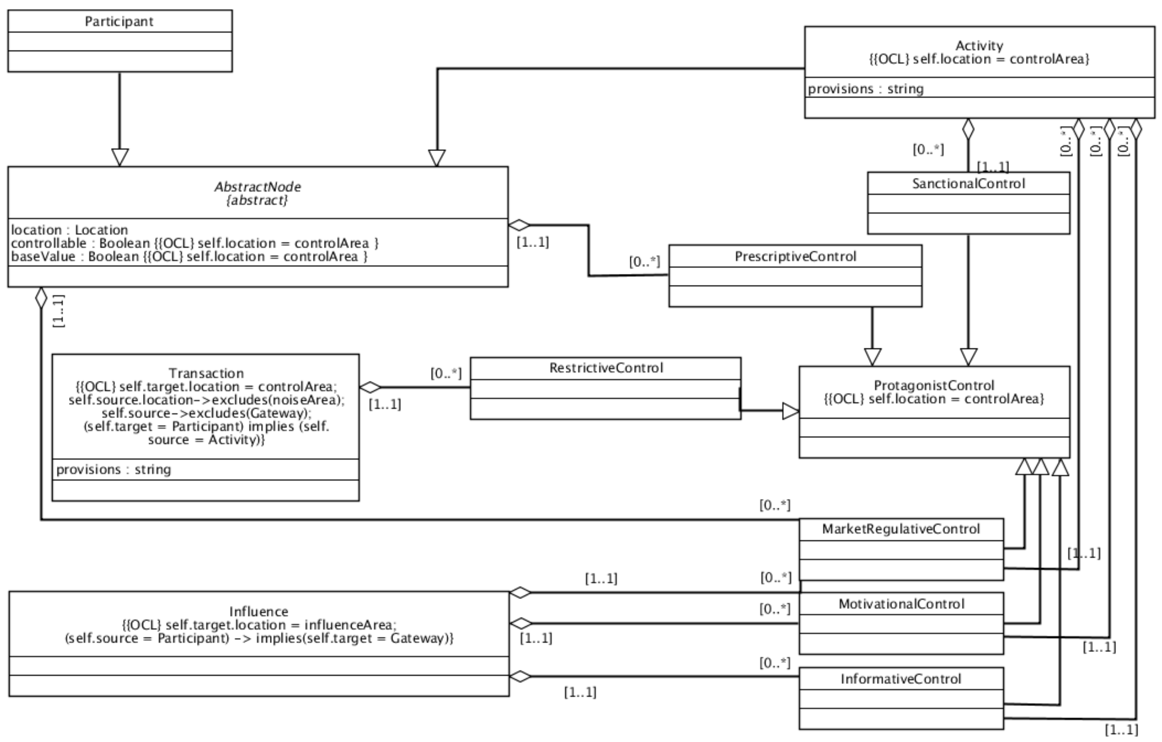

Participant
Activity
{{OCL} self.location = controlArea}
provisions : string
AbstractNode
{abstract}
location : Location
controllable : Boolean {{OCL} self.location = controlArea }
baseValue : Boolean {{OCL} self.location = controlArea }
SanctionalControl
PrescriptiveControl
Transaction
{{OCL} self.target.location = controlArea;
self.source.location->excludes(noiseArea);
self.source->excludes(Gateway);
(self.target = Participant) implies (self.
source = Activity)}
provisions : string
RestrictiveControl
ProtagonistControl
{{OCL} self.location = controlArea}
MarketRegulativeControl
MotivationalControl
InformativeControl
Influence
{{OCL} self.target.location = influenceArea;
(self.source = Participant) -> implies(self.target = Gateway)}
[0..*]
[1..1]

Figure 28: Nodes and Edges (those parts within the control area) and their suitable control mechanisms

Critical reflections on the control configuration

The case-sensitive allocation of control limits internal participants to prescriptive control involving a very narrow scope of freedom. For some corporate management concepts or even for some contexts this limitation of scope of freedom may be too strict.

However, the way it has been conceived gives the modeler choice. He may place those participants with limited scope of freedom (e.g., those in charge of core platform services) into the control area. On the other hand he may place position participants in charge of independent services outside the control area, giving them the management status of an independent unit with the implication of reduced stakeholding power, but with the potential of higher creativity. In such cases, control would occur on the corporate governance level through the setting of strategic targets for those independent entities. Such governance implications are outside the scope of Dyno. The next section gives an example for this differentiation.

5.4 Modeling Scenario in Dyno

This Section revisits and models several growth phases of Salesforce's service platform to illustrate Dyno. More models are to follow in the section on service platform patterns, giving additional examples of its reusable nature (Section 5.5.2). The pattern section reverts i.e. to demand-sided network effects as applied by Trello. It further models approaches of finite quantities of divisions at customer locations, as done e.g., by Dropbox, Google Plus or S.Chand Edutech.

Figure 12 describes four phases of growth of the service platform operator Salesforce. Figure 29, Figure 30, Figure 31 and Figure 32 model the different phases based on Dyno. The models do not claim to be exhaustive, but aim at providing familiarization with Dyno. The sequence starts with the launch phase and then depicts the different points of departure to growth phases II to III.

Launch Phase and Growth Phase I

Salesforce started off with proprietary software, offered as a service. Figure 29 depicts internal participants and activities on the platform within the borders of the control area. The reachable target group is located within the influence area, the competitors in the noise area. The model represents the internal service department, which is in charge of developing the CRM software, as participant.

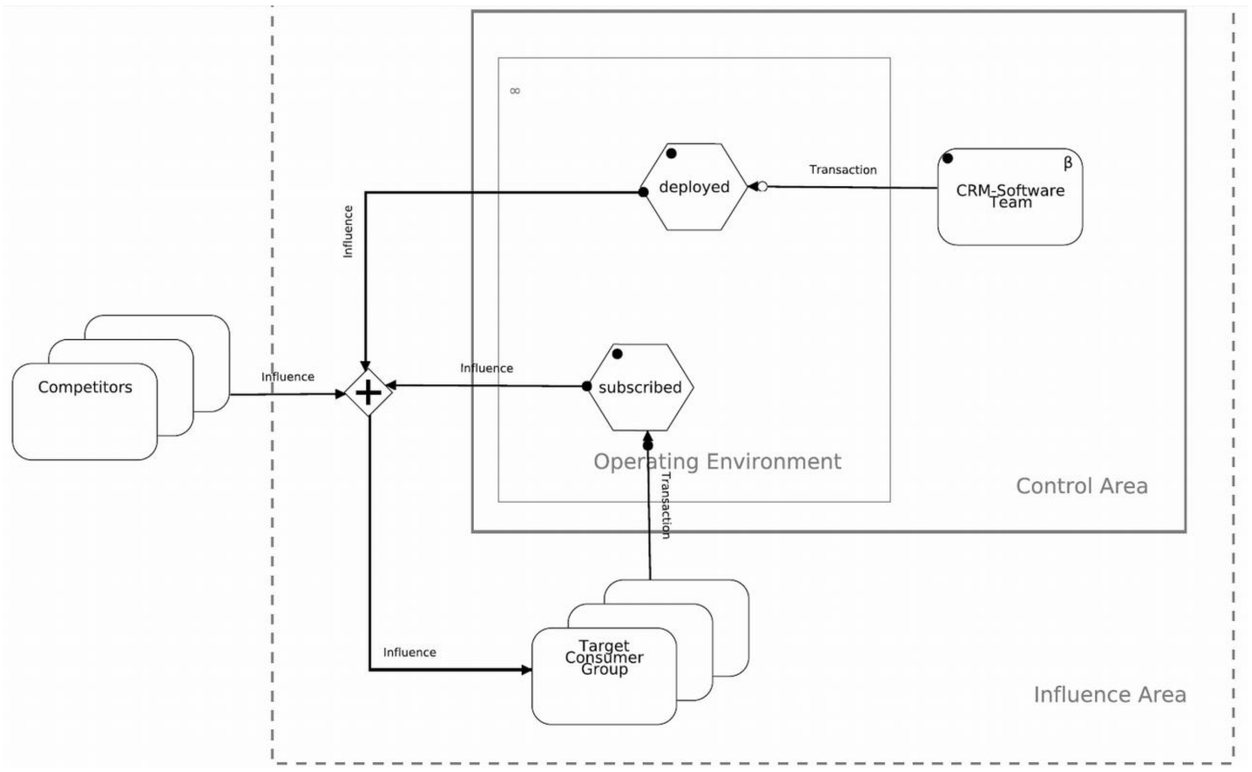


Figure 29: Salesforce, Launch phase, screenshot from Dyno-Editor

The internal participant is subordinate to the platform operator's authority. Therefore, the participant carries the symbol of a filled circle (*controlled*). The configuration panel of an editor would show that the platform exerts prescriptive control. As the CRM- team is the source of Salesforce's base value, it carries the β-symbol. Theoretically, the modeler could as well place the β-symbol into the activity *deployed*. Allocating the symbol into the participant emphasizes that the CRM-based base value does not grow through loops (in contrast to the activities in growth phases I to III). Putting the β-symbol into the activity shows, that deployment of new services is an ongoing activity of the participant, gradually filling the activity.

The activity *deployed* also indicates that it is controlled. Here, the platform exerts prescriptive control. It is able to modify software, e.g., to adapt it to new hardware. The deployed CRM software exerts attractiveness on the target consumer group, depicted by an influence. The platform operator enforces this attractiveness through informative control, which communicates information on available services to the addressed consumer group. The operator does not close any causal loop from the target group to the activity deployed. The model carries no network effect yet.

After a certain growth time, the activity *subscribed* accumulates an amount of subscriptions, which allows the model to be updated and to assign this activity with an additional base-value symbol (Figure 30). Figure 12 described this as growth phase I.

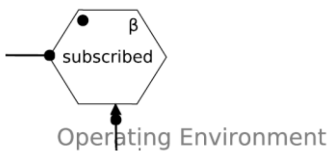


Figure 30: Salesforce, growth phase 1, fraction of screenshot from Dyno-Editor

However, a causal loop circles around the *subscribed* activity. The quantity of subscribed consumers has scale effects on other consumers which may subscribe as well. Not including any added functional value, the target group may have low sensitivity to the total number of users, which is the accumulation of activity *subscribed*. Given this low sensitivity, even though the number of subscribed users might exceed the threshold, the impact would remain small. The platform operator communicates information on the numbers of inscribed users to enforce the effect. Negative influences by competitors weaken the effect. In the model, the division operating *environment* carries an ∞-symbol, indicating that the technical infrastructure is scalable and thus prepared to network effects.

The base value *CRM software*, deployed as a service led to sufficient growth of the subscribed customer base. This allowed Salesforce in 2006 to shift to Growth phase II.

Growth phase II

In growth phase II, Salesforce opens an ‘on-demand application sharing service’ for subscribers, partners and developers’[31] (*AppExchange*). Figure 31 depicts the corresponding Dyno model. The activity *appex* represents the e-market. The model positions the activity into a dedicated scalable environment. By default, all subscribed participants get access to the e-market. Dyno visualizes this through a transaction from the activity *subscribed* to *subscribed to appex*. The model sets the base value attribute in *subscribed to appex* = *true*, as this activity is the basis to the targeted causal loops.

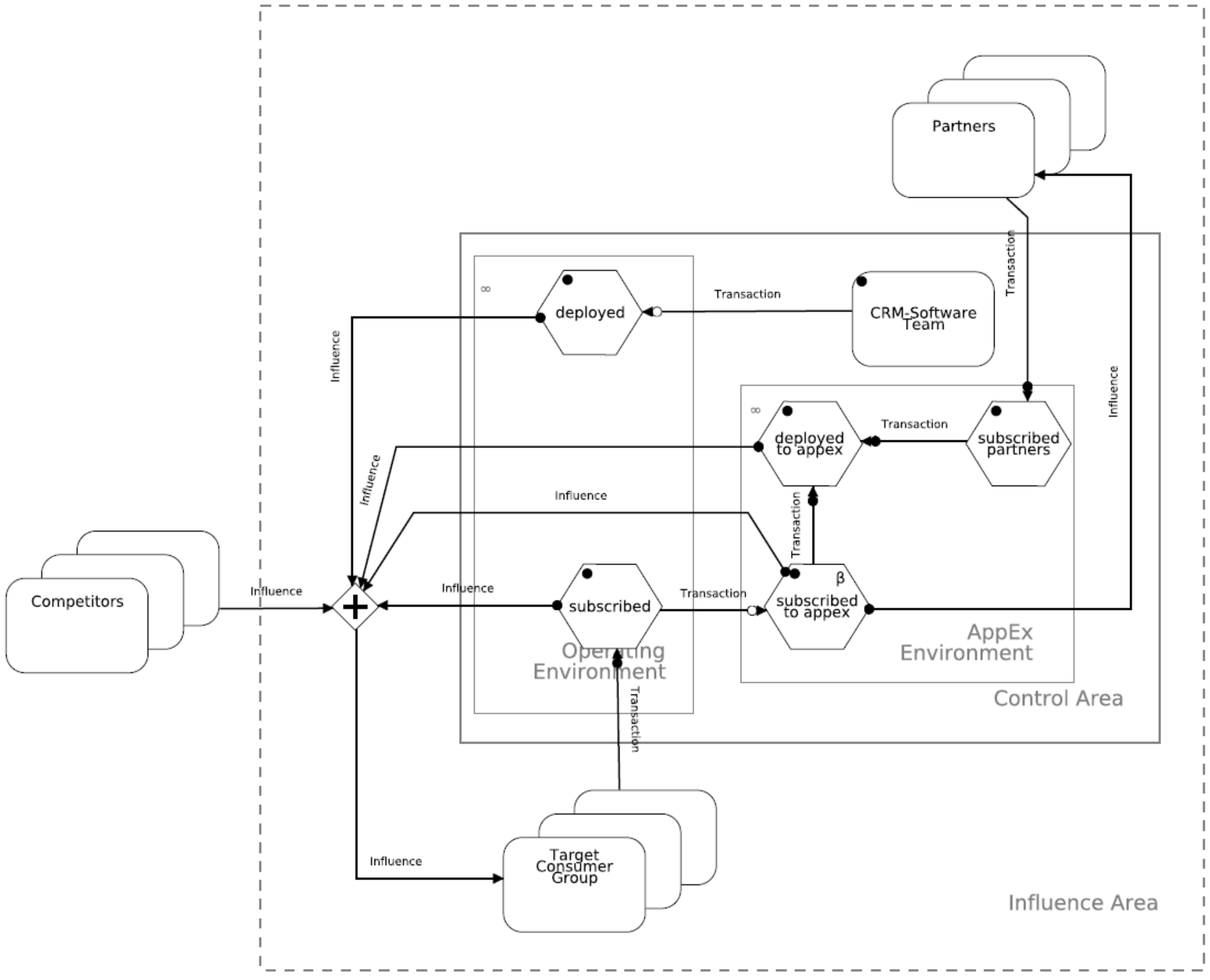


Figure 31: Salesforce, growth phase 2, screenshot from Dyno-editor

31 http://web.archive.org/web/20060213075840/http://www.salesforce.com/appexchange/whatis_appexchange.jsp, retrieved 19.02.2013

Salesforce provided a policy, specifying required information e.g., on application, functionality, external services or on manageability[32]. They are located on the transactions from participants (on activity subscribed to *appex*) as well as from partners (on activity *subscribed partners*) and pointing on *deployed to appex*. A restrictive control mechanism on both transactions represents the compliance check. Partners as well as consumers can be unsubscribed through a sanctional control mechanism on their respective subscribed-activities. On the *deployed to appex*-activity, the platform operator exerts sanctional control. Prescriptive control appears, due to the broad range of service design alternatives difficult

The platform design shows a demand-sided and supply-sided causal loop. On the demand side, consumers can provide solutions. The model depicts this through a transaction going from *subscribed to appex* to *deployed to appex*. The more solutions the consumers provide, the more attractive becomes the platform to potential consumers in the target consumer group.

The second loop is cross-sided. The more consumers subscribe, the more attractive becomes the platform for partners to provide services. The more services are provided, the higher is the attractiveness to new consumers to subscribe. The platform operator works with a series of mechanisms to support this. On the *subscribed to appex* activity, he applies market regulative control. The consumers rate services to each other. He does not place any recommender system. On the influence departing from the *subscribed to appex* activity, the platform operator uses informative control to inform the target consumer group on the growing quantity of consumers. On the influence departing from the *deployed to appex* activity, the platform operator sends motivational control (granting free testing of services on the platform). On the same influence, the platform operator places informative control (information of the services applied). Through market regulative control on the same influence, the platform operator communicates the service ratings aiming at creating a situation of trust in the target group.

The major disadvantage in this design is the limitation of prescriptive control on the *deployed to appex* activity and strong manual workload when testing the services during restrictive control on the ingoing transaction. Both shortcomings are caused by the insufficient service manageability quality due to broad scope of freedom in service design. Growth phase III compensates this shortcoming.

[32] Policy: http://web.archive.org/web/20060424014931/http://www.salesforce.com/appexchange/listingReqs_appexchange.jsp, retrieved 19.02.2013

Growth phase III

In growth phase III, starting in 2008 (Figure 32), Salesforce enhances its limited programming specification into a development and testing environment [134]. The whole division which handles external services turns into a Platform-as-a-Service solution (Force.com). A *developing* activity channels service design through prescriptive control into services of desired quality. Prescribed manageability quality allows for observability and controllability. Prescribed business process quality ensures collaborability. Suitability for standards allows for interoperability and conformability. The enforced quality allows loose coupling of those third party services with the CRM-as-a-service software. The consumer does the service configuration. To the platform operator, this implies an increased level of process automation with reduced requirement for human interference.

In this platform design, the platform operator can exert far reaching prescriptive control in the *deployed to appex* activity. This facility goes down to the level of technically modifying the service. This enhances the scope to influence service level measurement quality.

The Salesforce example would allow more detailed degrees of modeling: i.e. details like the testing environment *Sandbox* could have been added. Another fact worth modeling is Salesforce's contribution of services onto the two-sided market. The contributing team could be modeled as an individual participant. The participant's contribution of free of charge services could be modeled as motivational control on the influence from the *deployed to appex* activity into the ecosystem. In a more exhaustive model social networking applications such as chatter could also be modeled, showing additional network effects. The example describes how the structured approach of the modeling language enables visualizing complex multi-level dependencies. This evolutional example also shows that Dyno can help modeling platform evolution from various starting points.

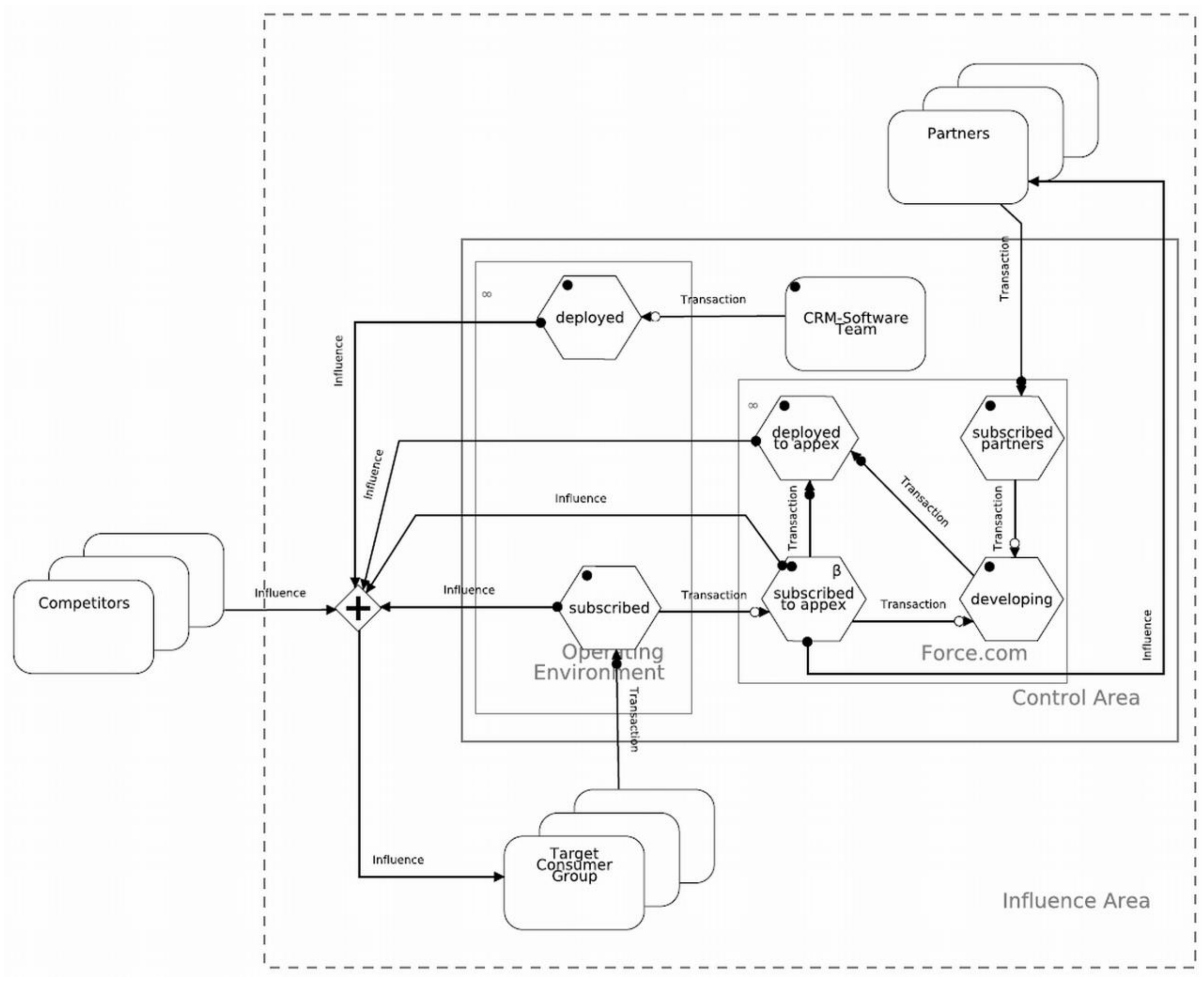


Figure 32: Salesforce, growth phase 3, screenshot from Dyno-Editor

5.5 Pattern Language for Dyno

The present section introduces a pattern language for service platforms and a coordinated community driven process to create a sharable quality-managed knowledge repository. Pattern language, process and repository are not part of the core Dyno language specification. They complement the language for increased comfort and modeling effectiveness. The present work uses the MoSaiC collaboration model [135] as underlying process model for a coordinated, community-based development of a pattern repository. An early version to this language was published by Scholten, Schuster et al. [44] and critically discussed at the 2012 5th IEEE International Conference on Service-Oriented Computing and Applications (SOCA).

Complementing the Dynamic Network Notation, platform service management patterns serve as mechanisms for documenting and communicating knowledge about successful and failed service management approaches around service platforms. Patterns in general help to “identify, name and abstract common themes” [17] in design. Those patterns help making knowledge reusable by capturing the information and intent behind a design. Patterns are condensed descriptions of the properties that succeeded in solving a specific problem. Platform service management patterns provide a common vocabulary for platform architects and solution managers to communicate and document concepts as well as to explore management alternatives and serve as reusable base of expertise through solutions retrieved from experience. For example, the explorative analysis of a set of platform samples revealed several patterns for service deployment on a platform: a free deployment approach, a programming-model-based deployment procedure and a programming-environment based deployment procedure. The remainder of this chapter revisits the two latter examples and embeds them in respective patterns and an anti-pattern.

The remainder of this subsection continues with a definition of service platform patterns (Section 5.5.1). Then it exemplifies a series of basic patterns (Section 5.5.2). Aggregating experience from many unconnected communities (e.g., researchers from different disciplines, professionals from various market segments) and staying in phase with their progression over time requires capturing patterns in an evolving manner. Therefore, the section closes with a collaborative management process suggesting a coordinated, community-based approach of creating and managing a repository of service platform patterns (Section 5.5.3).

5.5.1 Service Platform Patterns

The following definition of service platform patterns builds on terminology and concepts of pattern languages [16] and design patterns [17]. In general a pattern is an "abstract problem-solution pair, applicable for a specific environmental context" [136]. Plenty of pattern collections exist, addressing various contexts and levels of application. Gamma, Helm et al. [17] structure idiomatic class and object structures into design patterns, providing common vocabulary and constituting "a reusable base of experience for building reusable software". Workflow patterns as defined by van der Aalst, ter Hofstede et al. [137] operate on a higher level of abstraction, describing conditions, examples, problems and solutions in workflow style expressions. Platform service management patterns capture knowledge on best practices and experiences for controlling activities and participants as well as harnessing network effects in a service network platform ecosystem.

Definition 78: Service platform patterns are structured descriptions of best practices for harnessing network effects in service platform ecosystems.

The pattern language consists of building blocks of structured descriptive text and Dyno-utterances. The Dyno-utterances aggregate existing models from other patterns and elements in compliance with Dyno's abstract grammar, e.g., of the form $A \rightarrow ABw$, where A and B are building blocks and w are Dyno elements. Following Definition 32, the pattern language is context-free, as it allows for nesting. Explicitly, it allows the production of an upgraded pattern A based on the original pattern and extending patterns of Dyno elements. Figure 33 visualizes the corresponding meta-model to the pattern language.

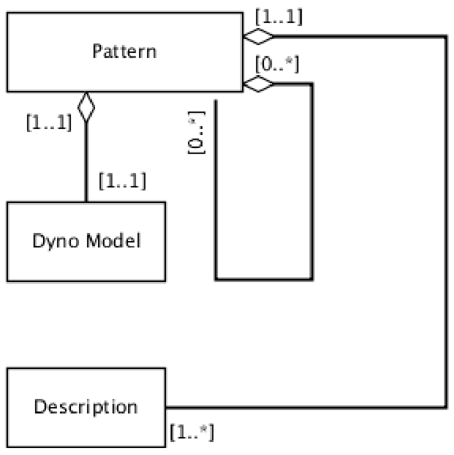


Figure 33: Meta-model to service platform patterns

A new service network pattern can include a multitude of identical, existing patterns. Existing patterns can be included into many new patterns. One specific instance of one Dyno model however can only be part of one specific pattern. Patterns can include only one Dyno model, although a Dyno model may be an utterance, embracing other Dyno models. The pattern language differentiates between patterns, supplying best practice in specific applications (e.g., one-sided service platform with network effect, platform with programming specification, platform with programming environment) and anti-patterns, practices where market studies proved that they lead to problems or underperformance (e.g., a Web service intermediary platform without base value contribution).

Building on Gamma, Helm et al. [17], platform service management patterns follow a structure as described in Figure 34. Departing from Alexander, Ishikawa et al. [16], research community around patterns [16, 136-138] calls the finite set of patterns constructible from a specific pattern type a pattern language. The grammar is textual and reverts to the English language, structured in a table of pairs of denominators and descriptions. The only exception is the diagram, which is a graphical element (Dyno-model), embedded into the table.

Pattern Id	Unique pattern identifier.
Pattern Name	Meaningful name of the pattern.
Version	Version number.
Date	Date of the current release
Authors	The authors that contributed to the pattern, followed by the release version in brackets.
Status	Under revision / released.
Pattern Type	Pattern or anti-pattern.
Intent	Description of the addressed service network management problem.
Applicability	Description of the contexts where the pattern can be applied including preconditions.
Solution	Detailed description of the pattern, its accomplishment, limitations, etc.
Diagram	Graphical representation of the pattern.
Frequent Features	Detailed description of functionalities that are often, but not always applied.
Consequences	Description of pros, cons, and limitations.
Sources	If code for parts or all of a pattern can be downloaded at a URL, this URL is included here, accompanied by additional information e.g., license information and deployment guides.
Examples	Real life examples described through company name, potentially through additional information,
Included Patters	Cross reference to included patterns described through pattern name and [pattern Id]. If the included pattern is a composite solutions, its sub-patterns do not need to be listed explicitly.
Related Patterns	Cross reference to closely related patterns described through [pattern name] ([pattern Id])

Figure 34: Structure of service platform patterns

The heading segments *pattern id, pattern name, version, date* (of last revision) and *authors* shall allow for distinct pattern identification. The cell *authors* denominates the author(s) of initial or revised version, complemented by the *version number* of respective involvement. *Status* shall show whether there is any limitation in usage due to revision or blocking. *Pattern type* should indicate, whether the depicted pattern is exemplary best practice or an illustration of solution that is likely to fail. *Intent* and *applicability* should describe the addressed problem and the context.

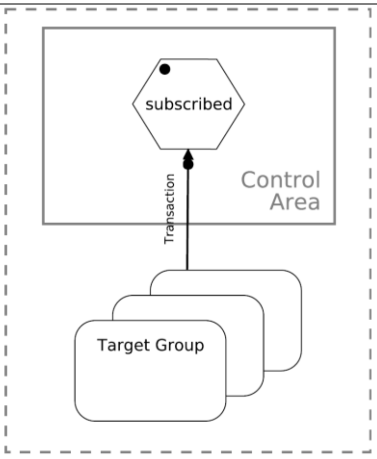
The detailed description of the pattern solution should be followed by a diagram providing the graphical representation. Frequently used but not always applied features may be listed in *frequent features*. Hereafter, *consequences* gives space for optional discussion of advantages and disadvantages of the pattern. *Sources* allows for the display of links to available code, *examples* describes real-life examples. The pattern structure shall close with a list of *included* and *related patterns*. The template may be extended through the community.

5.5.2 Pattern Drafts

This subsection gives examples of representative patterns. The first 3 patterns exemplify basic building blocks. Pattern 4 is a representative anti-pattern. Pattern 5 is composite patterns which embeds two basic patterns. Pattern 6 includes pattern 3 and pattern 5. The patterns are drafts as they are *under revision* and not yet released.

Platform subscription pattern draft

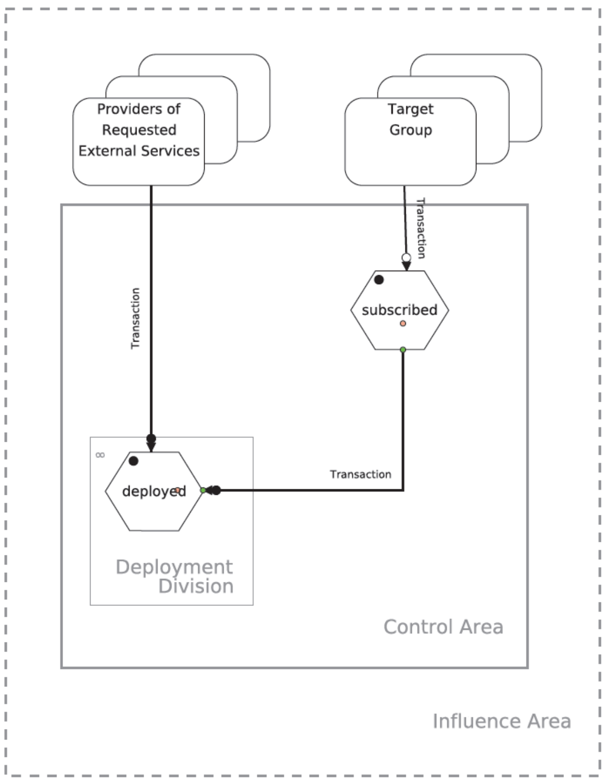
All analyzed platforms respond to the following subscription patterns.

Pattern Id	00001.0001
Pattern Name	Platform Subscription Pattern
Version	1
Date	08.12.2012
Authors	Ulrich Scholten (1)
Status	Under revision
Pattern Type	Pattern
Intent	This pattern provides a basic approach for managing subscription into a platform.
Applicabi-lity	The solution is suitable for service platforms. It helps onboarding and maintaining subscribers selectively. The subscribers can be service consumers, service providers or both.
Solution	Members of an addressed user group subscribe and *consume* services. The transaction edge pointing from the addressed target group to the *subscribe*-activity, which is endued with *restrictive control*. I.e., the user has to subscribe to the services and to accept the platforms' service provisions (or a subset). In the *subscribed*-activity, the platform exerts *prescriptive control*. The activity *subscribed* allocates limited amount of freedom to consumers. Within the activity *subscribed*, the platform exerts *sanctional control*, i.e., it retains the right and the power to exclude users.
Diagram	
Frequent Features	The provisions in transaction and activity often include the following: - Contractual agreement (acceptance of service platform terms), - Request for consumer details (address, payment details).
Conse-quences	The pattern is reduced to the core features of subscription. It blinds out any application specific feature, e.g., geographical or market segment-specific distinctions due to national law or mentality. Those need to be added, when applying the pattern.
Sources	not available
Examples	Facebook, Netsuite.com, Salesforce.com,
Included Patterns	-
Related Patterns	-

Basic External Service Deployment Pattern Draft based on Programming Specification

Deployment of external services exists in many variations. The following pattern describes a basic pattern, which ensures a rudimentary level of service manageability. It corresponds to the approach, which was chosen by Salesforce in their initial stage of expansion of the AppExchange / Force.com platform.

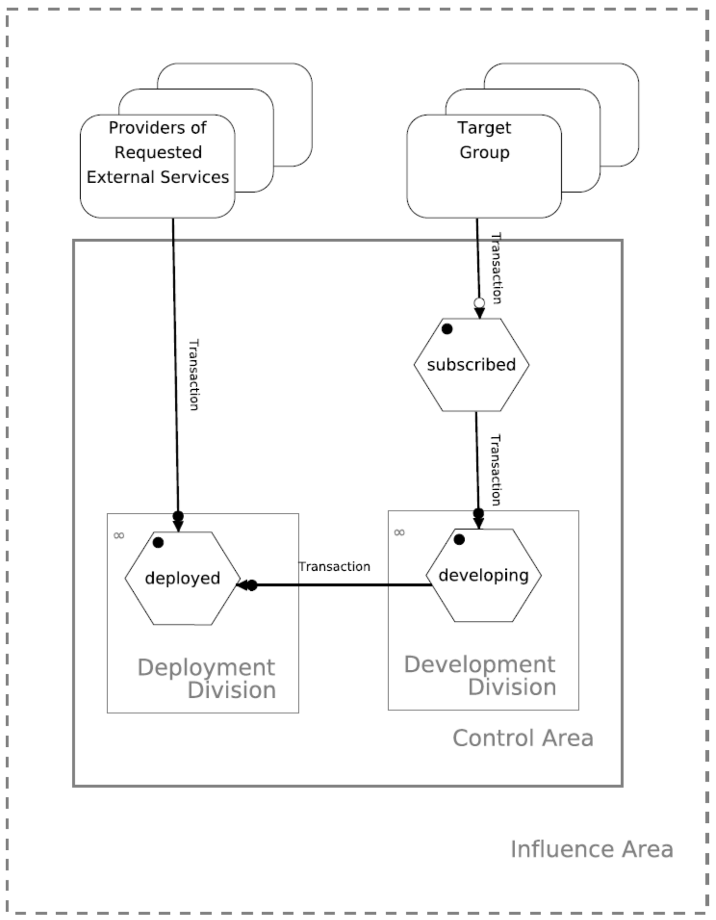
Pattern Id	00002.0002
Pattern Name	Basic External Service Deployment Pattern based on Programming specification
Version	2
Date	21.02.2013
Authors	Ulrich Scholten (1, 2)
Status	Under revision
Pattern Type	Pattern
Intent	This pattern provides a basic approach for managing third party service deployment into a platform.
Applicability	The solution is suitable for all service platforms. It helps onboarding and managing services.
Solution	Subscribed participants of a service provider group provide services for deployment on a platform. Regarding it as a dynamic system, the activity *deploy* has the function of a stock. The more services are provided, the more this stock replenishes. *Restrictive control* on the *transaction* pointing from *subscribed* to *deployed* filters the infeed of services in function of compliance to the *platform service provisions*, i.e. service design in compliance to the pro-gramming specification, but also to legal requirements. The activity *deployed* includes *sanc-tional control*, which regularly verifies compliance with the *platform service provisions* and initiates escalation routines. In addition, it includes *prescriptive control* to manage the de-ployed services. Services, requested from outside the control area through deployed services are subject to *restrictive control*.

Diagram	 Providers of Requested External Services Target Group Transaction Transaction subscribed Transaction deployed Deployment Division Control Area Influence Area
Frequent Features	Some service platform operators include a sandbox into the ‘deployed’-activity to test a service before deploying. Specific modeling of such a sandbox makes sense, if it is exploited in causal loop, e.g., to attract early adopters in a testing phase.
Conse-quences	The programming specification limits the service providers’ scope of freedom, i.e. with respect to reutilization of services on other platforms. As a consequence, service providers could be tempted to produce simple requestors, programmed based on the programming specification, calling up external services. Many of the challenges with respect to service quality could migrate from an upfront filtering through *restrictive control* to an ongoing verification through *sanctional control.*
Sources	not available
Examples	S.Chand Edutech with a programming specification based on the standard SCROM.
Included Patterns	Platform Subscription Pattern (00001.00)
Related Patterns	-

Service Development Pattern Draft with Programming Environment

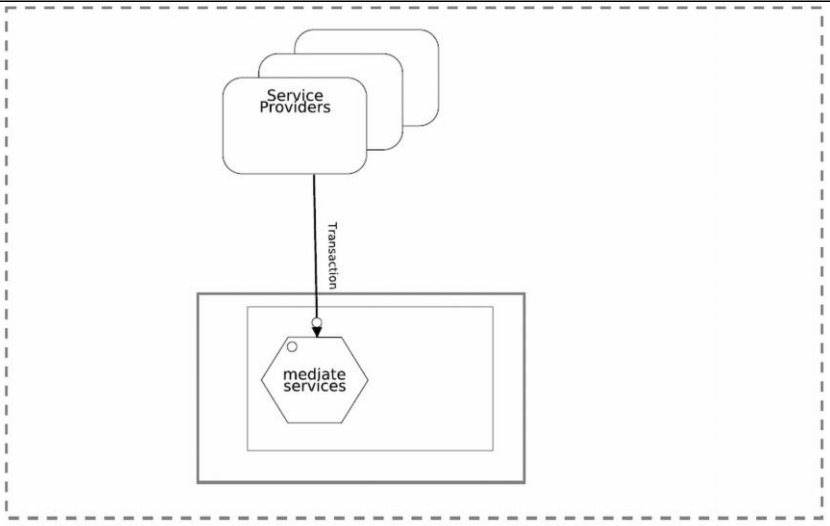
The following describes a service deployment pattern with a prepended programming environment, as used e.g., by Salesforce.com or Netsuite.

Pattern Id	00003.0001
PatternNa-me	Service Development Pattern with Programming Environment
Version	1
Date	21.02.2013
Authors	Ulrich Scholten (1)

Status	Under revision
Pattern Type	Pattern
Intent	This pattern provides an approach for managing third party service deployment into a platform.
Applicability	The solution is suitable for multisided service platforms. It helps onboarding, testing and maintaining services and service providers selectively.
Solution	Members of a target group provide services for deployment onto a platform. Regarding it as a dynamic system, the activity *deploy* has the function of a stock. The more services are provided, the more this stock is filled. A programming environment is prepended to the deployment environment, ensuring compliance to related service platform provisions. Apart from legal and administrative aspects, the programming environment on the *developing* activity ensures service manageability in the subsequent *deployed* activity. The activity *deployed* includes *sanctional control*, which regularly verifies compliance with the company policy and initiates escalation routines. In addition, it includes *prescriptive control*, managing the deployed services. Services, requested from outside the control area subject to *restrictive control*.
Diagram	 Providers of Requested External Services Target Group Transaction subscribed Transaction Transaction Transaction deployed developing Deployment Division Development Division Control Area Influence Area
Frequent Features	Some service platform operators include a sandbox into the *deployed*-activity to test a service before deploying. Specific modeling of such a sandbox makes sense, if it is exploited in causal loop, e.g., to attract early adopters in a testing phase.
Consequences	The level of manageability is higher than in the case of a pure programming specification (see 00003.00)
Sources	not available
Examples	Facebook, Salesforce, Netsuite
Included Patterns	00002.x - External Service Deployment Pattern based on Programming specification
Related Patterns	Platform Subscription Pattern

External Service Mediation Anti-Pattern Draft without mechanisms for quality and service control

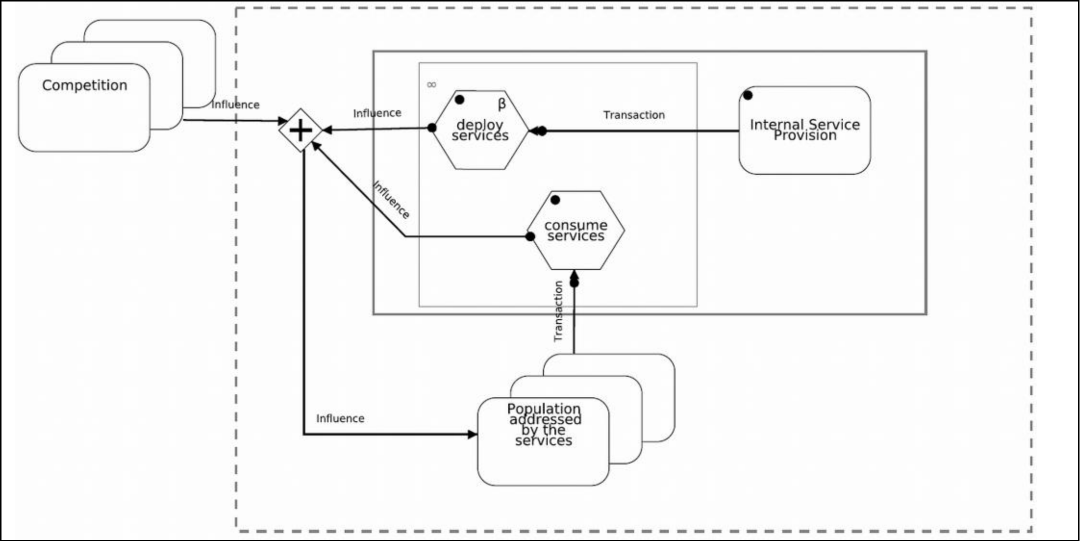
The following anti pattern describes a service mediation approach, which led to unsuccessful results.

Pattern Id	00004.01
Pattern Name	External Service Deployment Pattern without mechanisms for quality and service control
Date	09.12.2012
Version	1
Authors	Ulrich Scholten (1)
Status	Under revision
Pattern Type	Anti-Pattern
Intent	This pattern provides a negative example on third party service deployment into a platform
Applicabi-lity	The solution is unsuitable.
Solution	The platform offers third party services in an unmanaged way. The services are not deployed on the platform The platform's role is limited to a market place. Due to a lack of control stakeholding power, the platform operator lacks control over provided quality of service and service portfolio constellation. In addition, the *transaction* lacks *restrictive control*.
Diagram	
Frequent Features	Often, service mediation is done through the platform. Actual service evocation and the related traffic is done in a point-to-point way between service provider and service consumer. This limits the operator's role even further as he has no means to monitor the actually provided quality of service.
Conse-quences	Enforcing control mechanism on activity and transaction are vital to assure quality of service and a service portfolio congruent to corporate goals. All analyzed examples applying the above described anti-pattern are underperforming.
Sources	not available
Examples	Xmethods, RemoteMethods
Included Patterns	-
Related Patterns	Anti-pattern to 00002.x External Service Deployment Pattern based on Programming specification and to 00003.x External Service Deployment Pattern based on Programming Environment

Demand-sided Network Effect Pattern Draft

The following pattern describes a reusable structure for the creation and a management of demand-sided network effects.

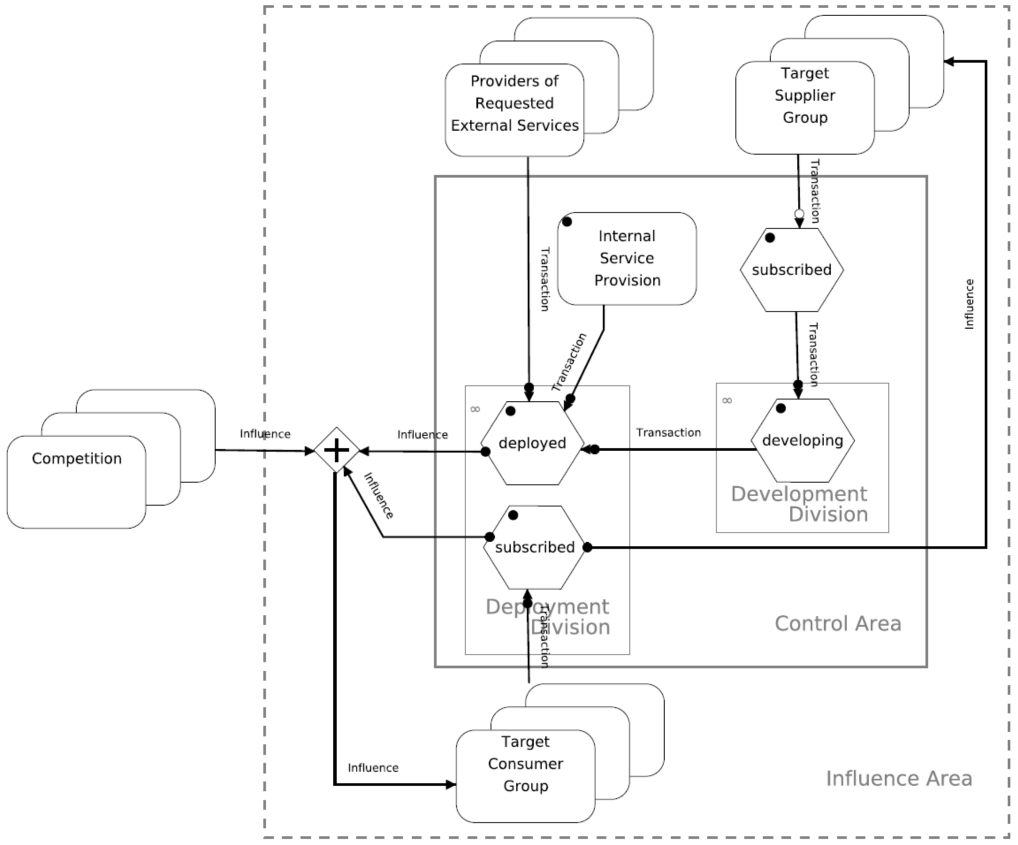
Pattern Id	00005.0003
Pattern Name	Demand-sided Network Effect Pattern
Version	3
Date	25.02.2013
Authors	Ulrich Scholten (1, 2, 3), Nelly Schuster (1)
Status	Under revision
Pattern Type	Pattern
Intent	This pattern provides a solution for network effects that shall be exploited to grow the consumer base.
Applicability	The solution can be applied in single-sided and multi-sided service platforms. The precondition is a deployment environment, scalable enough to cater for potentially accomplished dynamic growth of service consumption through a demand-sided network effect. Applying it through collaborative scenarios created cases, requiring small critical masses, starting at a magnitude of 2.
Solution	Members of a potential user group subscribe and *consume services* that were *deploy*ed by the platform operator. Regarding it as a dynamic system, the activity *consume services* has the function of a stock. The more subscribed users consume, the more this stock is filled. The activity *deploy services* also acts as stock. This activity represents the base value, which initially sets of the system (indicated by the β-symbol). The quantity of users can motivate – together with a quantity and quality of *deploy*ed services –new potential users to subscribe to the platform. This motivation is weakened by competitive offers. The pattern channels explicit effort on positive user influence on the addressed population through the application of several control mechanisms in the context of a control loop: On the *influence*-edge, linking the *consume* activity and the *Gateway*, platforms apply *informative* and *motivational control*. *Informative control* amplifies the impact of the size of the user group. The nature of communicated information may vary; however platforms in all analyzed cases communicated the subscribed number of users. Platform operators may utilize *motivational control* in various shapes. The transaction edge pointing from the user group to the *subscribe*-activity applies *restrictive control*. I.e., the user shall be required to accept the platforms' terms and conditions. Within the *consume*-activity, the platform shall exert *sanctional control*, i.e., it may make use of its right and the power to exclude users. The deployment of services in the described pattern is an internal activity, therefore limited to *prescriptive control* and *sanctional control*. Exerting prescriptive control means to manage the deployment environment. *Sanctional control* stands for the power to undeploy a service. The platform operator exert *prescriptive control* within the participant *Internal Service Provision* (e.g., in response to results from reputation systems) and may exert *restrictive control* on the *transaction* leading from *Internal Service Provision* to *deploy services*. The influence-edge, pointing from the *deploy* activity to the *Gateway* include the control mechanism of *informative control*. *Informative control* implies the clear and targeted information on the product. The activity *consume* needs to be placed in a scalable environment, to be able to respond to rapidly growing consumption. Strong growth behavior is realistic, as the activity is placed within a loop (= demand-sided network effect).

Diagram	 Competition Influence Influence β deploy services Transaction Internal Service Provision Influence consume services Transaction Influence Population addressed by the services
Frequent Features	• In many cases, the influence-edge, pointing from the *deploy* activity to the *Gateway* also includes *motivational control* and *market regulative control. Market regulative control* is applied through reputation mechanisms to reduce the entry barrier and to give decision support. • The activity *deploy services* can be replenished by specific participants, representing the service development departments within the platform. Linking edges are *transactions*.
Conse-quences	The pattern is reduced to the core features of a service deployment. It blinds out any complex feature, e.g., replication or synchronization. Those need to be added on a context based approach.
Sources	not available
Examples	Trello, Dropbox, Google Plus
Included Patterns	00001.x - Platform Subscription Pattern 00003.x - External Service Deployment Pattern based on Programming Environment

Cross-sided Network Effect Pattern Draft

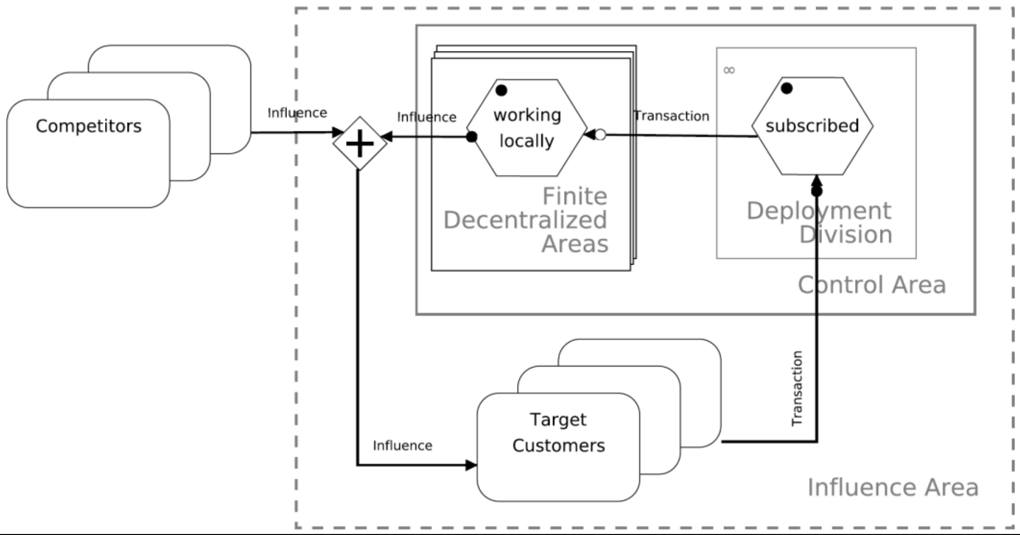
The following describes a reusable pattern for the creation and management of cross-sided network effects.

Pattern Id	00006.01
Pattern Name	Cross-sided network effect through service consumption and third party supply
Version	1
Date	08.12.2012
Authors	Ulrich Scholten
Status	Under revision
Pattern Type	Pattern
Intent	This pattern provides a solution for network effects that shall be exploited to grow the user base and service base.
Applicability	The solution can be applied in multi-sided service platforms. The precondition is a deployment environment, scalable enough to cater for potentially accomplished dynamic growth of service consumption through a demand-sided network effect and service deployment through suppliers-sided network effects.
Solution	Members of a potential user group subscribe and *consume* services that were originating from the platform operator and the service provider. Regarding it as a dynamic system, the activity *consume* has the function of a stock. The more subscribed users consume, the more this stock is replenished. The activity *deploy services* also acts as stock. This ‘filled stock of services’ represents the base value, which initially sets off the system (indicated by the β-symbol). The quantity of users can motivate – together with a quantity and quality of *deploy*ed services –new potential users to subscribe to the platform. This motivation is weakened by competitive offers. The pattern channels explicit effort on positive user influence through the application of several control mechanisms in the context of a control loop: On the *influence*-edge, linking the *consume* activity and the *Gateway*, platforms applies *informative* and *motivational control*. *Informative control* amplifies the impact of the size of the user group. The nature of communicated information may vary; however platforms in all analyzed cases communicated the subscribed number of users. Platform operators may utilize *motivational control* in various shapes. The transaction edge pointing from the user group to the *subscribe*-activity applies *restrictive control*. I.e., the user shall be required to agree to the platforms’ terms and conditions. Within the *consume*-activity, the platform shall exert *sanctional control*, i.e., it may make use of its right and the power to exclude users. The influence point from *consume services* to *Service Providers* embeds 2 control mechanisms: *informative control* and *market regulative control*. Deployed services are of internal and external origin. Services of internal origin are limited to *prescriptive control* and *sanctional control*. Exerting prescriptive control means to manage the services. Sanctional control stands for the power to undeploy a service. The platform operator exerts *prescriptive control* within the participant Internal Service Provision (e.g., in response to results from reputation systems) and may exert *restrictive control* on the transaction leading from *Internal Service Provision* to *deploy services*. Members of the *targeted consumer group* subscribe and *consume services* that were *deploy*ed by the platform operator. Regarding it as a dynamic system, the activity *subscribed* has the function of a stock. The more subscribed users consume, the more this stock is filled. The pattern channels explicit effort on positive user influence on the addressed population through the application of several control mechanisms in the context of a control loop: On the *influence*-edge, linking the *consume* activity and the *Gateway*, platforms shall apply *informative* and *motivational control*. *Informative control* amplifies the impact of the size of the user group. The nature of communicated information may vary; however platforms in all analyzed cases communicated the subscribed number of users. Platform operators may utilize *motivational control* in various shapes. The transaction edge pointing from the user group to the *subscribe*-activity shall apply *restrictive control*. I.e., the user is required to accept the platforms’ terms and conditions. Within the *sub-*

	scribed-activity, the platform shall exert *sanctional control*, i.e., it may make use of its right and the power to exclude users. The influence-edge, pointing from the *deployed* activity to the *Gateway* includes the control mechanism of *informative control. Informative control* implies the clear and targeted information on the services. The activity *subscribed* needs to be placed in a scalable environment, to be able to respond to rapidly growing consumption. Strong growth behavior is realistic, as the activity is placed within a loop (= demand-sided network effect). The influence-edge, pointing from the *subscribed* activity to the *Targeted Supplier Group* includes the control mechanism of *informative control. Informative control* implies the targeted information on consumption. The activities *subscribed* and *deployed* need to be placed in a scalable environment, to be able to respond to rapidly growing consumption. Strong growth behavior is realistic, as the activity is placed within a loop (= demand-sided network effect).
Diagram	 Providers of Requested External Services Target Supplier Group Internal Service Provision subscribed Transaction Influence Competition deployed developing Development Division subscribed Deployment Division Control Area Target Consumer Group Influence Area
Frequent Features	• In many cases the influence-edge, pointing from the *deploy services* activity to the *Gateway* also includes *motivational control* and *market regulative control*. Often suggestions are used, e.g., ‘other users, applying this service also applied XYZ’. *Market regulative control* in this case stands for the use of reputation mechanisms to reduce the entry barrier and to give decision support.
Conse-quences	This pattern couples two causal loops and enforces a cross-sided network effect. Quality control functions need to be properly in place to harness positive effects but also to avoid the negative scenario of negatively accelerating loops, e.g., initiated through the provision of low quality third party services.
Sources	not available
Examples	Salesforce.com, Netsuite
Included Patterns	00005.00 - Demand-sided Network Effect Pattern 00003.00 - External Service Deployment Pattern based on Programming Environment
Related Patterns	-

Pattern Draft for Finite Areas of Control

The following describes a reusable pattern for the creation and management of finite areas of control. The surveys brought to light several successful providers applying this pattern and thus mixing approaches of web-based applications and native apps.

Pattern Id	00007.0001
Pattern Name	Patterns for Finite Areas of Control
Version	1
Date	25.02.2013
Authors	Ulrich Scholten
Status	Under revision
Pattern Type	Pattern
Intent	This pattern provides a solution for network effects that should be exploited to steer and grow the user base.
Applicabi-lity	The solution can be applied in single-sided and multi-sided service platforms. The application has no precondition.
Solution	Subscribed consumers receive a native working environment on their local infrastructure (e.g., on a client computer or server). The platform operator uses the opportunity to be closer to user data to get access to this data in a legally permitted and from the consumer approved way. Local applications also help saving bandwidth and buffer potentially unstable Internet connection. *Prescriptive control* gives the platform operator control over this area. It also allows after release by the consumer to access user data (e.g., email addresses, settings etc.) or to change system settings on a customer infra-structure (e.g., default browser setting). *Sanctional control* allows the platform operator to terminate the application. *Motivational* and *informative* control in the activity *working locally* guides the sub-scribed users. An *influence* points via a *Gateway* to a targeted customer group. A *transaction* point-ing back to the activity *subscribed* and subsequently *working locally* creates a causal loop. Gained user data helps to address specific target customers from the subscribed users' communities.
Diagram	 Competitors Influence Influence working locally Transaction subscribed ∞ Finite Decentralized Areas Deployment Division Control Area Transaction Target Customers Influence Influence Area
Frequent Features	• Several platform operators deployed decentralized finite control areas on client computers. • Several platform operators requested consumers for permission to read out local email address-box to identify other subscribed users respectively to invite unsubscribed users. • One platform operator implemented synchronized environments on client servers to overcome instable Internet connection.
Conse-	Legal implications of and scope for decentralized finite control areas may vary, depending on the

quences	physical location of the customer infrastructure.
Sources	not available
Examples	Google Drive, Dropbox
Included Patterns	-

5.5.3 Coordinated Community-Based Management Process

To be attractive to potential users in a cross-industrial context, the repository needs to respond in sufficient variety to wide-spread options, contributed and verified by experts from the concerned communities. For this reason, the present work applies a community-driven approach to develop patterns for the repository. Following the line of thinking of network effects, this concept applies a demand-sided causal loop. This view is examined further in the remainder of the subsection. The quantity of initial patterns in a pattern repository needs to exceed a threshold to incite network effects. The coordinated model reverts to and further develops the service-oriented collaboration model (MoSaiC), introduced by Schuster, Zirpins et al. [135]. This collaboration model builds on two parts: the collaboration process and the coordination process.

Collaboration process:

The previous chapters on the Dynamic Network Notation gave an in-depth introduction into demand-sided network effects. A community-driven approach is a specific case of such a same-sided network effect. It makes use of a targeted group of potential (but unspecific) repository users that are in need of service patterns. It works, and is able to sustain itself, once a critical mass is reached and on condition that the network effect does not die off. In contrast to traditional community-driven approaches to content production, the present process includes quality control through explicit approval processes and defined collaboration processes. Following the collaboration model in MoSaiC, patterns and their elements are considered as services provided by the collaborating community members. Users make contributions in a self-paced way; a multitude of coordinators is in charge of quality control.

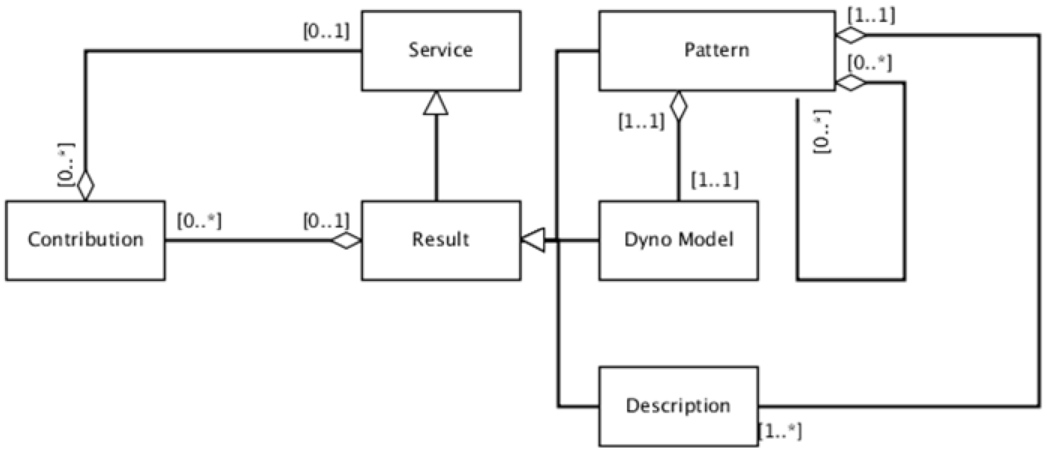


Figure 35: Meta-model for the collaborative composition process for Service Management Pattern repository based on Schuster, Zirpins et al. [135]

The person initially suggesting a pattern becomes coordinator, also taking this role during subsequent updates. He structures patterns P as an ordered tree of expected results R with $P = \{R, C_R\}$ with the set of results R and the relationship C_R. The relationship C_R is a subset of all possible relations within the set R, defined through the languages production rules (syntax). The possible set of relations can be described as $C_R \subset R \times R$. Figure 35 illustrates that a result can either be a Dyno model, a description or a pattern. The latter is possible due to the language's context-free nature. The meta-model for the collaborative pattern composition process discussed (Figure 35) is a refinement of an initial version, introduced by Scholten, Schuster et al. (2012). Looking at patterns as linguistic utterances generated through a collaborative composition process, they can be redefined as follows:

Definition 79: Patterns are ordered trees of results of a collaborative composition process, related through its production rules.

Definition 80: A result of a collaborative composition process can be a pattern, a Dyno model or a description.

The owner of a pattern is entitled to add and remove results throughout the collaboration process. Experts provide contributions as services into a pattern template (Figure 35). Those experts can be humans or software services (e.g., an Id generator or an automatically generated improved model subset, produced by an analyzer). The composition model includes several standard services, e.g., pattern draft service or pattern approval services. The number of contributions (on any part of the pattern) leading to a result is arbitrary and depends on the specific composition process and the eventual number of required iterations. The service-based approach is of particular importance as it allows for flexible assembly and decomposition of patterns, regardless of human or software-based origin.

b) Coordination process:

Coordination rules manage dependencies between pattern components and control the execution of services. The coordination rules are preset but can be theoretically modified. The pattern repository reverts to MoSaiC's event-condition-action (ECA) rule mechanism. For details on the rule mechanism see Scholten, Schuster et al. [44]. This mechanism comprises of

- Automation protocols for automating of service bindings as well as request / response protocols.
- Application-specific dependency management for the management of dependencies between individual patterns or subsets.

A coordination rule engine automates coordination and executes the rules. Service binding protocols bind a service to a specific contribution. In other words, a specific community member accepts responsibility for a result based on agreement with the coordinator of the result. Figure 36 describes the service binding process. A process starts once a community member accepts ownership for a pattern. He then initiates a pattern or updates an existing pattern. In case of an attempt for modification of an existing result, the community member requests or addresses the respective pattern owner. This owner can accept or decline the binding. In the case of an acceptance, the state changes from 'binding open' to 'bound'. The same protocol applies, when the coordinator invites a reviser, only with exchanged roles.

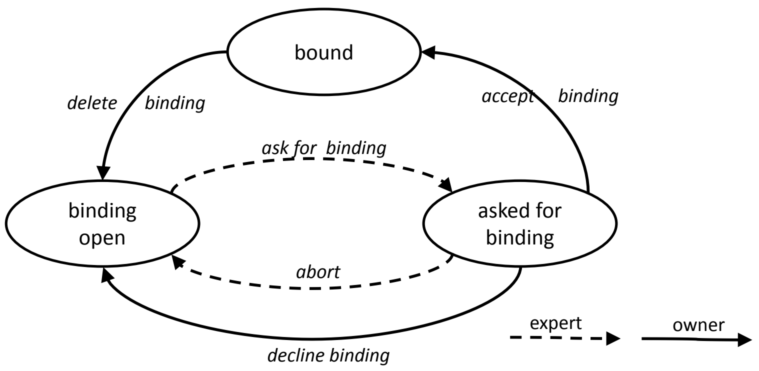


Figure 36: Service binding protocol [44]. Ovals depict a contribution's state; dashed arrows describe the experts' and solid arrows the owners' decisions. Binding open is the starting state

After successful binding of a service, service request/response protocols enable provisioning, update and approval of results (Figure 37). The following example serves to illustrate this process: The owner of a pattern first creates a result placeholder. Then he identifies a required contribution, e.g., for *frequent features*. As next step, he binds a service to the contribution which invoked the service. The expert then executes the service which writes the *frequent features* description into the pattern template. The state changes from *identified* to *created*. In the initial configuration of the rule engine, three approvers need to approve a process before a pattern can be updated. Until approval, the pattern rests in the *under revision* state. The initial definition of three approvers is assumption based and nothing more than a starting point for optimization in a subsequent optimization period. Those approvers are also community based.

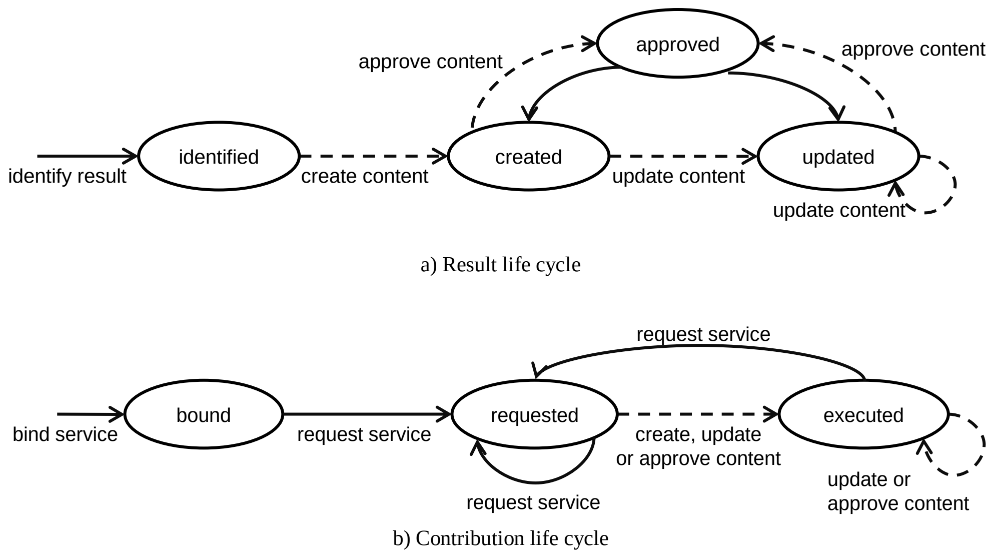


Figure 37: Service request and response protocol [44]

Figure 38 describes the Service Pattern repository in a Dyno Model. The repository will not attract an initial external user community until it has attained a sufficient amount of patterns. When this goal is accomplished, the activity *publish and update patterns* represents the base value with respect to the repository's network attractiveness on the target group. The model also reflects the approval process. Community users can provide updates. Other community members can release those updates. The released pattern goes to the activity publish and update patterns.

The Dyno model illustrates that the repository has by initial design already three same-sided causal loops. The users of the pattern repository might stimulate others to use the repository. The next causal loop is due to the MoSaiC coordination process (through binding). A coordinator invites specific experts or a group of potential experts for contribution and revision. Those may be acquired from outside the group of subscribed users. Finally, the continuously growing stock within the *publish and update patterns* activity will further increase the repository's value to potential users. To further stimulate the network effects, the modeler can further develop the indirect control mechanisms 'informative control' and 'motivational control, in each case on the influence edges. Scholten, Schuster et al [44] provide further deepening to the repository's coordination rules.

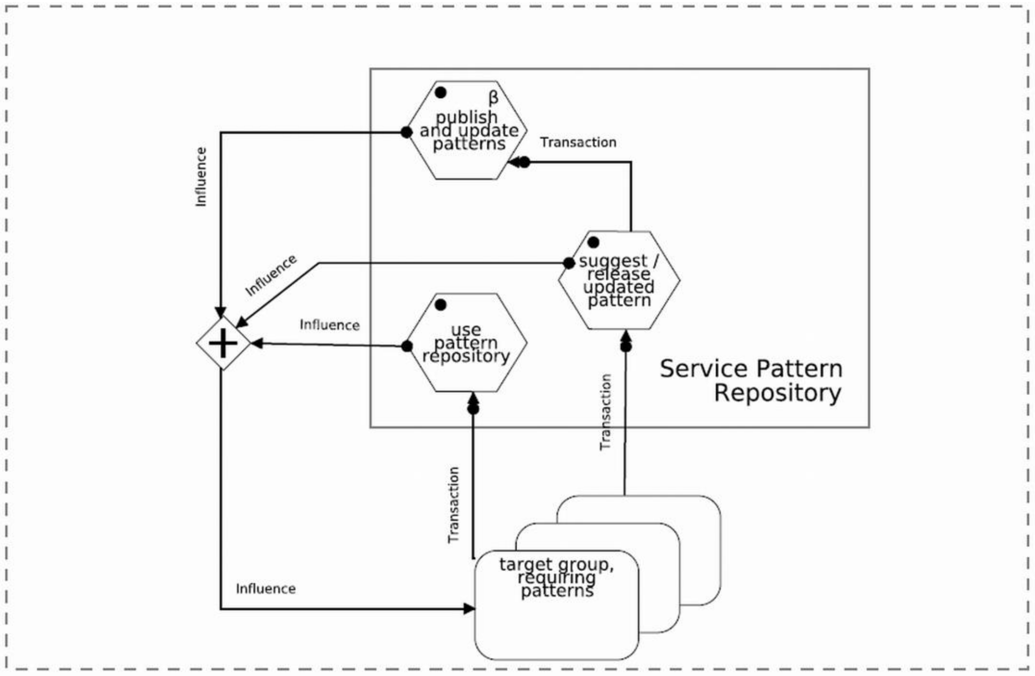

β
publish and update patterns
Transaction
Influence
Influence
suggest / release updated pattern
Influence
use pattern repository
Service Pattern Repository
Transaction
Transaction
target group, requiring patterns
Influence

Figure 38: Pattern repository, modeled with Dyno

5.6 Modeling Recommendations

One characteristic of Dyno’s grammar is that it allows for modeling free of any mandatory sequence. When implemented in an editor, the modeling environment only allows those options which are syntactically correct. This section therefore constrains itself to a list of good practices, which result from experiments with various test users.

Starting With a Basic Layout

It makes sense to start with a general layout. Placing the influence area and within it the control area (Figure 39)

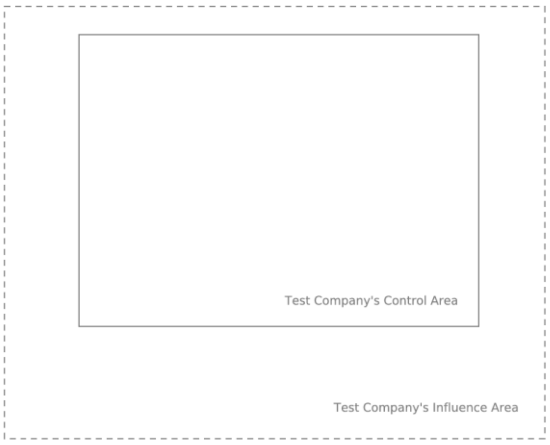


Figure 39: Basic Layout

Allocating and Connecting Important Participants and Activities

In a second step, the modeler could place important participants and activities. To assure clarity from the beginning, it is useful to place or group nodes into divisions.

The modeler should evaluate various levels of granularity when modeling activities and participants or participant groups. It may be helpful to split participant groups into several groups of distinct communities to be able to better focus the subsequent control mechanisms. Or he may split an activity (e.g. *deployed*) into a sequence (e.g. *deployed* and *in sandbox*) or into parallelized branches. This gives the option of additional loops, but reduces clarity of the model. In the illustrating models, the present work consistently differentiates between activities by consumers and activities by suppliers. This dyadic might dissolve in cases where the same group creates and consumes.

Depending on the modeler's line of thinking, two approaches could make sense:

- *Focal approach* - The modeler starts with an initial node, which he assumes carries the base value in the current model (activity or participant). Then he models one or several causal loops, around or near this node. Afterwards he places control mechanisms. Independent of the modeler's mindset, this approach could make sense for a startup, which can start off without many legacy structures requiring consideration.
- *Decentralized approach* – The modeler places all activities and assets into individual divisions. Then he decides on one or several base values to focus on in the current design. Then he inserts edges in pursuit of generating causal loops. In a next step, he places the control mechanisms to strengthen the desired network effects.

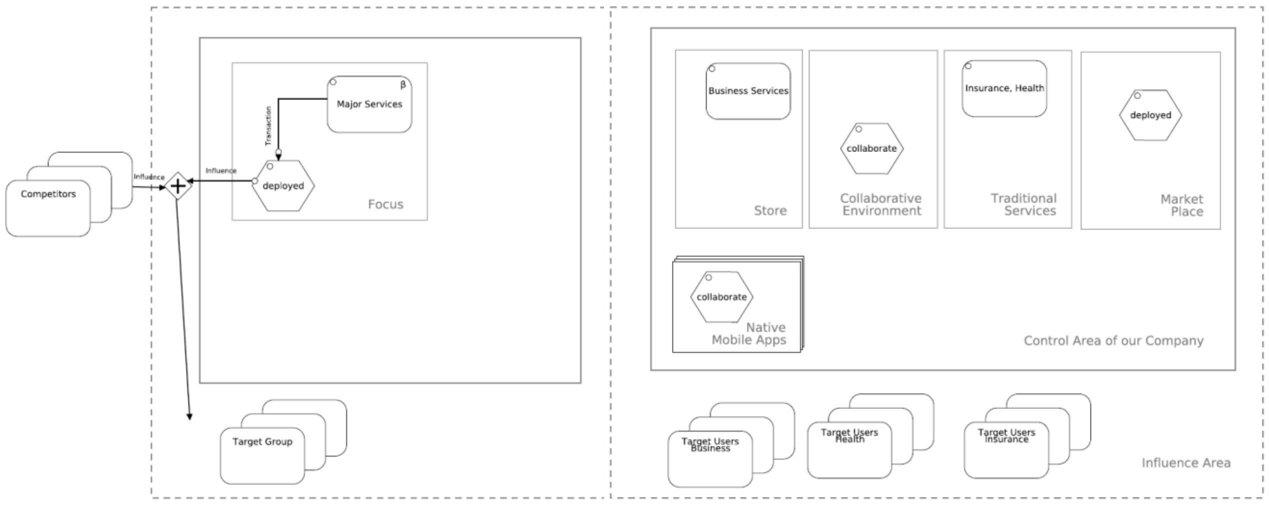


Figure 40: Focal approach (left) and decentralized approach (right) to modeling

Choice of Base Value

Within the above step of allocating and connecting nodes, the modeler needs to evaluate which potential base value or base values are suitable to incite network effects. The modeler could compare existing value contributions and rank them. Then he models the top ranked value contribution as the base value in the context of his model. A second approach is to do this selection based on comparison of alternative models. This approach has the advantage that not only the isolated base values are subject to comparison, but the whole loop, including targeted participant groups etc. When brainstorming on base values, the modeler should not restrict his listing to offered software services, but also look at other less tangible values, which might be of relevance to his target groups (e.g., subscribed consumers in other contexts).

For causal loops, the availability of a critical mass within an activity is important to create network attractiveness. If no sufficient critical mass is available, the modeler might need to rethink the setting of a loop with a network attractiveness of smaller thresholds (e.g., the loops designed with collaborative scenarios by Dropbox or Trello). Alternatively, he may build on an internal value contribution, modeled with a participant porting the base value. Such a base value is not exposed to network attractiveness, however it can incite the evolution of a new base value (e.g., consumer subscription base), which can subsequently serve as a starting point for a causal loop with network attractiveness.

Sequence of Models for Different Growth Phases

Platforms go through processes of evolution and maturity. Those different phases require different models. Design should always capture the actual setting and then build on it and modify it. Section 5.4 illustrates the different growth phases of the company Salesforce with different models reflecting different focus.

Parallelization of Causal Loops within One Design Period

In specific settings, i.e., cross sided loops, two or more loops are interdependent. In other cases, causal loops might exist in parallel to each other, without reinforcing or weakening effects. The present work cannot give exhaustive information whether one specific period of platform design should focus on one main one-sided or cross-sided causal loop. Supporting argument for such a design decision is the potential to interlink the detached loops and cause reciprocity on each other. Counter arguments that focus on one major loop may sharpen the value promise in the market. To the author’s best knowledge, no academically validated theory is available at the time of the write up of this thesis. Ries’ [117] standpoint to better focus on one *engine of growth* is not empirically substantiated. Companies like Google successfully operate several causal loops in parallel.

Platform Engineering versus Platform Strategies

The Dynamic Network Notation (Dyno) guides the modeler through the production rules designed for the specific case of harnessing network effects. It helps in the structural allocation of nodes, in the design of relationships and loops and in the allocation of control mechanisms for service management. The pattern language provides a complementing structured base on experience in favor of a good model. Subsequent automated analysis, if implemented, may even give further guidance in modeling a platform which has potential to be successful in the market.

With an engineering bias, the present work supports platform engineering through suitable artifacts. It supports, but does not replace platform strategies. Please refer to further going academic literature on platform strategies [6, 22, 139, 140].

C Instantiation and Evaluation

6 Instantiation

Part C instantiates and evaluates the concept of the Dynamic Network Notation in the editor DynoCloud.org, including two iterations of refinement after interventions with targeted companies, as suggested in the Action Design Research methodology [38]. The objective is to validate the hypotheses, stated in Section 2.9. This chapter presents the DynoCloud.org editor including a model designer and an analyzer. In particular, it describes the model designer's general conception (Section 6.1) and its concrete grammar being implementation-specific to the modeling environment (Section 6.2). The subsequent DynoAnalyzer shows that Dyno's grammar, the way it is conceived, can be the starting point for further analysis (Section 6.3). The analyzer features also serve as proof of concept and do not intend to claim to be exhaustive. In the contrary they aim to create grounds and to open perspectives for further going research.

In the context of the present work, the editor *DynoCloud.org* serves as instantiation to the Dynamic Network Notation to validate the hypotheses. It also supports additional reflections on grammatical effectiveness with respect to the possibility of subsequent analytics. The goal of this editor is to give solution managers and platform architects a tool for design, analysis, evaluation of design alternatives and the targeted evolution of platforms and platform ecosystems, optimized on network effects and quality of service.

Basic requirements of the editor are easy access, a comfortable graphical user interface, the potential to share designs and an un-expensive cost per user ratio. Specific requirements relate to model design and analysis. For design, the solution requires a modeling and storage environment as well as a storage repository that holds available the necessary structural and procedural elements as well as control mechanisms. It further necessitates a user interface for configuration of control mechanisms and element attributes. For analysis, it requires a unit able to prepare, analyze and display data.

Figure 41 provides a system architecture diagram of the DynoCloud.org architecture based on the semi-formal graphical notation Fundamental Modeling Concepts (FMC) [141]. The architecture accommodates the above specified features, grouped into two main components:

(a) Model designer, including a shape repository, a modeling environment, a property configuration panel, a Dyno model repository and the necessary backend.

(b) Analysis-environment, consisting of an analyzer and analysis plug-ins.

In the editor's current implementation, the data export from the modeling environment to the Analysis environment is a unidirectional read access, reading out the Dyno model as object file in

JSON format. At the time of the write-up, the Analyzer GUI does not yet use the browser, but works with a SWING-based graphical user interface.

The following sections describe the model designer (Section 6.1), its concrete grammar (Section 6.2) and the analysis environment (6.3).

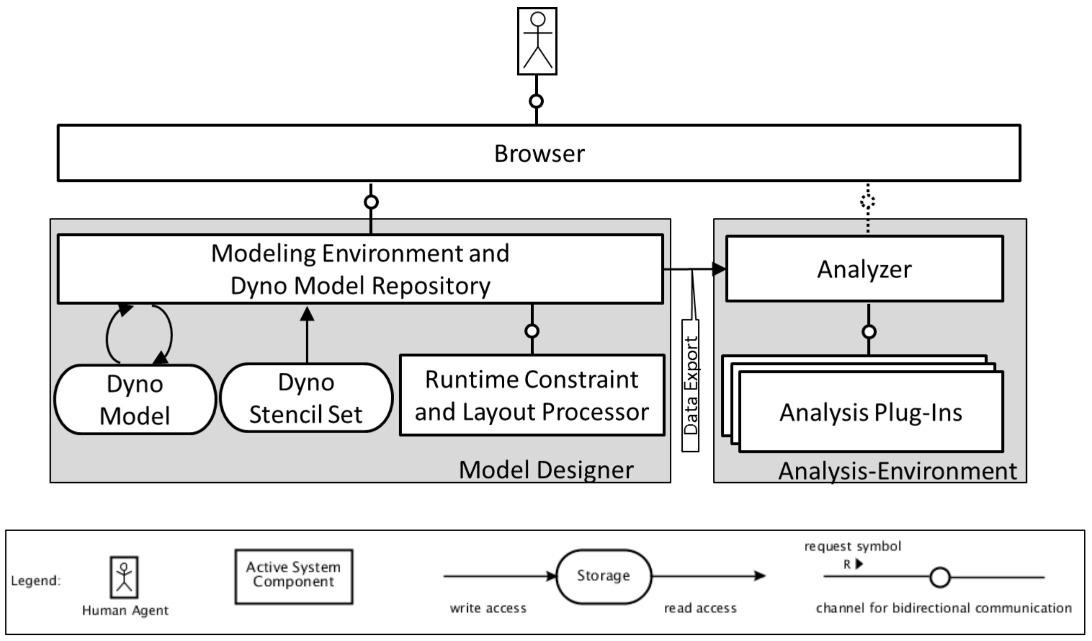


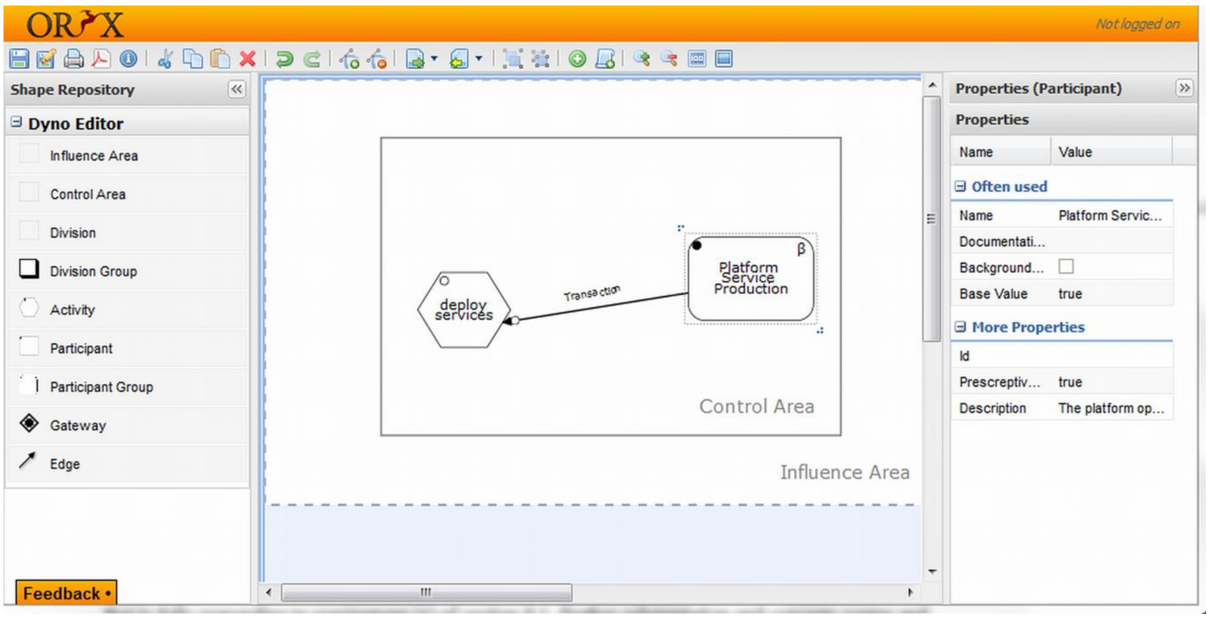
Figure 41: Editor architecture

6.1 Conception of the Model Designer

The configurable framework ORYX provides the infrastructure for an exemplary implementation of Dyno. ORYX is a web-based extensible modeling platform and repository supplied under MIT open source license [142]. Among others, ORYX has the advantage that it is conceived for business process modeling languages [143]. It is mature and proven in the BPM community through a variety of language implementations e.g. by Decker, Grosskopf et al. [144] or Cabanillas, Resinas et al [145]. A specific stencil set and a plug-in equip the ORYX framework with a concrete grammar for the Dynamic Network Notation and turn it into an operational modeling environment that is fully responsive to the above specified requirement (a). This section providers further substantiation on the general concept of the model designer.

Figure 41 describes the general editor architecture. The left bottom square describes the model designer. The architecture model depicts the modeling environment and process model repository ORYX as active component. This active component provides all the intelligence to operate, store and display business-process-oriented modeling and configuration for process models based on edges and nodes. The process model (in this context a *Dyno model*) is a passive element. The ORYX component has write access (save a model) and read access (reload a saved model). ORYX gains its Dyno-specific grammar through read access to the Dyno stencil set and through bidirectional communication with the active plug-in *Runtime Constraint and Layout Processor*. Bidirectional communication between ORYX and the browser allows for emulation of the modeling environment including its graphical user interface on client computers and the storage of models in the process model repository.

Figure 41 brackets the ORYX infrastructure in one active system component (*Modeling Environment and Dyno Model Repository*). For detailed description of ORYX, please refer to Decker et al. [143].



Name	Value
Often used	
Name	Platform Servic...
Documentati...	
Background...	☐
Base Value	true
More Properties	
Id	
Prescreptiv...	true
Description	The platform op...

Figure 42: Graphical user interface to the model designer including a Dyno-model, accomplished with ORYX

The graphical user interface (Figure 42) consists of 3 frames:

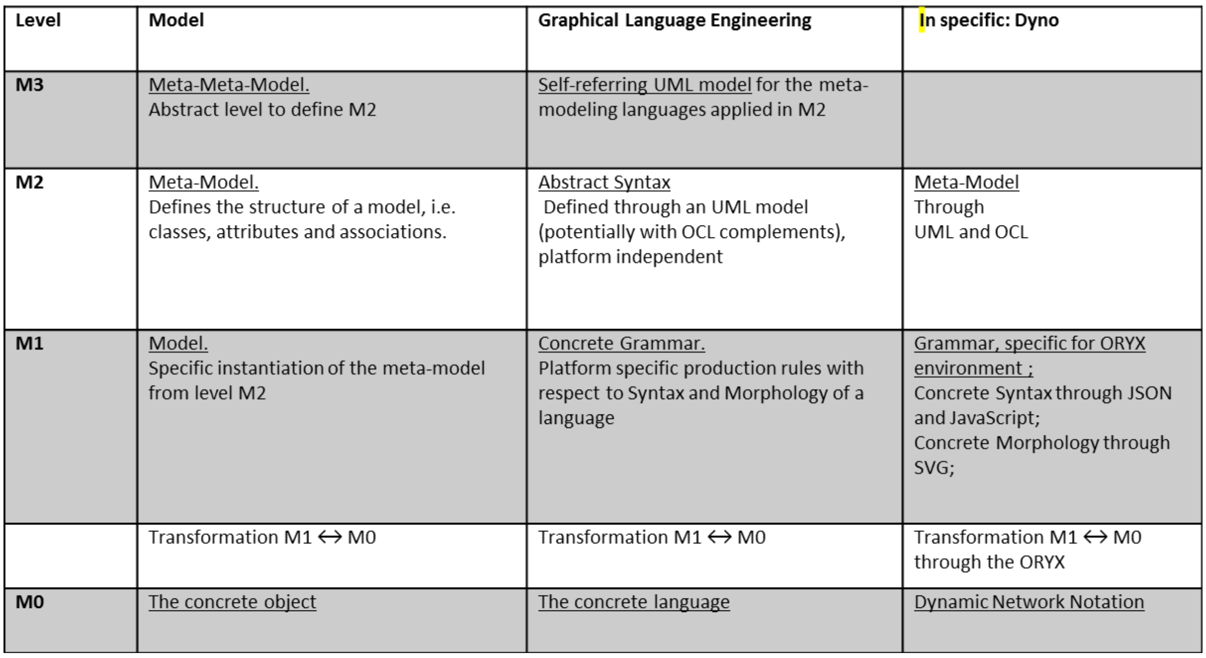
- a shape repository on the left side with all relevant Dyno-elements,
- the modeling canvas in the middle; in Figure 42 it shows a sample model of an activity and a participant connected through a transaction,
- a property configuration panel on the right side; Figure 42 shows the configuration of the participant *Platform Service Production* with the activated properties *Base Value* and *Prescriptive Control.*

In addition, the graphical user interface provides a horizontal tool bar with typical standard functionalities such as Save, Print, Copy, Past, Undo, Export, Resize. After login through OpenId the user can save his models and reload them in succeeding sessions.

ORYX's export feature allows for exporting Dyno-models in JSON format (JavaScript Object Notation) to the analysis environment. The model designer uses an export feature, which is specially modified for analysis purposes of Dyno models, i.e. it allows exporting full language information on a specific Dyno model, exceeding the normal data, exported by ORYX. Technically, the Model Designer runs on a virtual machine instance with 2CPU Cores, 2GB RAM and Secure Shell access as system platform, executing an Ubuntu Server LTS 10.04, 64-bit. An Apache Tomcat 5.5 serves as web server and servlet container. It uses a PostgreSQL server as database.

6.2 Concrete Grammar

The artifacts defined in this section are located on level M1 of the model stack (Table 32). In particular, they concretize morphology (Subsection 6.2.1) as well as the grammar (Subsection 6.2.2). Both concrete morphology and concrete grammar are specific to the ORYX environment.



Level	Model	Graphical Language Engineering	In specific: Dyno
M3	Meta-Meta-Model. Abstract level to define M2	Self-referring UML model for the meta-modeling languages applied in M2	
M2	Meta-Model. Defines the structure of a model, i.e. classes, attributes and associations.	Abstract Syntax Defined through an UML model (potentially with OCL complements), platform independent	Meta-Model Through UML and OCL
M1	Model. Specific instantiation of the meta-model from level M2	Concrete Grammar. Platform specific production rules with respect to Syntax and Morphology of a language	Grammar, specific for ORYX environment ; Concrete Syntax through JSON and JavaScript; Concrete Morphology through SVG;
	Transformation M1 ↔ M0	Transformation M1 ↔ M0	Transformation M1 ↔ M0 through the ORYX
M0	The concrete object	The concrete language	Dynamic Network Notation

Table 32: Concrete grammar in the context of the model-stack

The specifications in this section reference the following normative, dated documents. Those documents in the specified release date are therefore provisions to the Dyno language and editor specifications. In the remainder, the text does not further explicitly reference these documents:

- IEFT request for comment document RFC4329 on Scripting Media Types, i.e. JavaScript [146];
- IEFT request for comment document RFC4627 on application/json Media Type for JavaScript Object Notation (JSON) [146, 147];
- W3C recommendation on Scalable Vector Graphics and RGB color model [148].

6.2.1 Concrete Morphology

Concrete morphology in graphical languages prescribes the representation of graphical elements specific to the technical implementation. Concrete morphology in DynoCloud.org follows the requirements of the ORYX framework and becomes part of the ORYX stencil set (Figure 41). It is accomplished through Scalable Vector Graphics (SVG) [148]. SVG is an XML-based language for two dimensional vector graphics. Two of the strengths, which make SVG suitable for ORYX is that first, the SVG-format is browser readable. Second, the language's set of formatting rules for the portable and lightweight representation of graphics can be easily parsed by many other web-oriented languages, i.e. JavaScript.

Dyno's concrete morphology builds on this. Dyno elements contain different representations of an element in one SVG-file. E.g. the *participant* includes the standard symbol definition as a tag and a second tag for the context-sensitive controllability symbol.

The following line describes the tag for the basic participant definition

<rect id="participantView" fill="url(#background) #FFFFFF" stroke="#000000" width="150" height="100" rx="20" ry="20"/>

The element-name *rect* calls up a rectangle of the attribute value *participantView*, meaning it creates the participant shape. It has a white background and a black stroke, a width of 150 pts and a height of 100pts. The angles are rounded with a radius of 20pts.

The *Participant.svg* also includes a layout instruction for the graphical representation of the property 'controllable':

```
<g id="controllable" >
  <circle id="circlewhite" cx="10" cy="10" r="5" stroke="black" fill="white" stroke-width="1"/>
</g>
```

The concrete syntax (sub-section 6.2.2) adds the attribute *id=controllable* in design-time, whenever the context requires this.

The representation's adaptiveness allows the conception of the shape repository in a reduced and clearly arranged way. The shape directory includes only one edge-symbol for all possible variations of influences and transactions. The concrete syntax adapts them in design time. This ensures the Dyno model's syntactical correctness. In consequence, the editor guides the modeler, when designing an object by only allowing elements suitable to the context. Indirectly this also impacts the models' effectiveness. As the shapes appear with attributes mapped to the specific context, the modeler will unlikely forget considering them.

6.2.2 Concrete Syntax

Following Definition 32, syntax defines the assembly rules for modeling elements, as well as the adaptation of specific graphical representation of those modeling elements in function of the production rules. Concrete syntax in graphical languages is the set of production rules specific to the technical implementation. The present work implements the concrete syntax in the ORYX framework through two steps:

- Dyno-specific stencil, written in JSON-format. This file defines the assembly rules of modeling elements (a). The stencil set also includes the SVG-based concrete morphology.
- Dyno-specific plug-in, written in JavaScript. This routine handles the adaptation of specific graphical representation of modeling elements in function of the production rules in design time (b).

(a) Stencil Set

The Stencil set for Dyno uses the JSON format complemented by a set of jpeg images representing the icons of the shape repository. The JavaScript Object Notation (JSON) is a lightweight portable data exchange format to describe structured graphics, similar to XML. It is specified in the RFC4627 [147]. Its advantage in the ORYX framework is the proximity to JavaScript. The JavaScript statement executor function eval( ) turns JSON-definition into interpretable JavaScript Objects.

The JSON file provides definitions for all Dyno stencils including their properties and possible relations to other elements in compliance to the Dyno meta-model. It also includes implementation specific information such as the reference to its svg representation and in addition a reference to its respective icon (in jpeg-format) in the shape repository.

```
"type":"node",
"id":"participant",
"title":"Participant",
...
"description":"",
"view":"elements/participant.svg",
"icon":"elements/participant.png",
"propertyPackages":[
        "RootElement",
        "bgColor",
        "mayBeBase",
        "uncontrolled",
        "AbstractNode",
        "prescriptiveControlPackage"...]
```

The above example code segment defines the participant element. Specifically, it defines that it is of type *node*, including an empty string for the variable *description*. The latter defines the field description, which the user can fill in the configuration panel. The *view* refers to the related svg-element defining the morphology, the *icon* refers to the file supplying the symbol in the shape repository. The property packages refer to the inherited properties from the *root* element, to the standard color, to its capability of becoming a base value to and to the *AbstractNode*. The control packages refer define the control mechanisms of the element (the code fraction exemplarily states the *prescriptiveControlPackage*).

From the Abstract Node, the elements inherit (under specific constraints as defined in the meta-model) the property *controllable* with the default setting *false* as described in the following.

```
{
  "id":"controllable",
  "type":"Boolean",
  "title":"Controlled",
  ...
  "refToView":"controllable",
  "value":false
}
```

The runtime constraint and layout processor can set this property value *true* in compliance with the constraints in the meta-model in design time. If compliant, it also adds the circle symbol to the node by calling up the tag *refToView* in the svg file. A controllable node can be turned *controlled* by the modeler. This is again accomplished through the runtime constraint and layout processor in design time.

The JSON file defines allowed areas for each Dyno shape through containment rules as described in the following exemplary code-fragment for the control area.

```
"containmentRules":[
 {
 "role":"controlArea",
        "contains":[
            "activities",
            "divisions",
            "participants"
     ]
 },
]
```

The JSON-File further defines the allowed relationships through connection rules. The following example embraces all edges. The runtime constraint and layout processor (described below) defines in an *exclusive or* approach, whether the respective setting requires the morphology *Influence* or *Transaction*.

```
“connectionRules":
[
 {
 "role":"edge",
 "connects":[
            {
              "from":"participants",
              "to":[
                "activities",
                "gateway"
              ]
            },
            {
              "from":"activities",
              "to":[
                "activities",
                "gateway",
                "participants"
              ]
            },
          {
            "from":"gateway",
            "to":[
              "gateway",
              "participants"
            ]
    }
  ]
```

(b) Plug-In for a Runtime Constraint and Layout Processor

By definition, syntax also has the task to adapt specific graphical representations of modeling elements in function of the production rules. This task requires adaptation of the element representations in design time. ORYX provides a layout method. In ORYX, this subset of the concrete syntax needs to be programmed in JavaScript. The following code-fragment shows how a function is bound to the event that is in charge of laying out a participant. Laying out means the modeler touches or changes an element with the mouse.

```
construct : function (facade) {
  this.facade = facade;
  this.facade.registerOnEvent('layout.dyno.participant', this.handleNodeEvent.bind(this));
}
```

Whenever the modeler touches a participant, the function executes a series of methods. The following fragment shows how it adapts the property *controllability* on the constraints of a specific area.

```
_checkControllable : function (node) {
  var parentStencilID = node.getParentShape().getStencil().idWithoutNs();
  if (parentStencilID === "noiseArea") {
    this._setControllableFalse(node);
  } else if (parentStencilID === "influenceArea") {
    this._setControllableFalse(node);
  }
  else if (parentStencilID === "controlArea" || parentStencilID === "division" || parentSten-
cilID === "divisionGroup") {
    node.setProperty('oryx-controllable', true);
    parentStencilID = "controlArea";
  }
  node.setProperty('oryx-location', parentStencilID);
}
```

6.3 Conception of the Analyzer

The DynoAnalyzer is not part of the Dyno language specification. However, it is a helpful add-on for analysis of a model and its effectiveness as well as for the generation of suggestions to the modeler. Similarly to ORYX, Scholten and Reimchen [149] published the source code to the Dyno Analyzer under an MIT License.

The analyzer consists of an analysis environment and analyzer plug-ins (Figure 43). Figure 41 shows a dotted connection from the analyzer to the browser to indicate the objective of integrating model designer and analyzer into one browser-based GUI. The following sections describe the analysis environment (Subsection 6.3.1) and its plug-ins (Subsection 6.3.2).

The analyzer serves as proof of concept, substantiating that the Dyno grammar is suitably well-structured for consecutive analysis. This successive analysis is one of the claims of hypothesis MH. Subsection 6.3.2 provides exemplary examples, which confirm that the Dyno grammar includes sufficient information for the generation of subsequent analysis. The subsection exemplifies:

- Analysis of Completeness of control mechanism implementation,
- Detection of Loops,
- Analysis of availability of base values,
- Structured listing of included elements, their properties and descriptions.

6.3.1 Analysis Environment

The analyzer core is the central component within the analysis environment. It handles all major administrative tasks of the analyzer. The analysis functionalities however are treated by plug-in programs (see subsection 6.3.2).

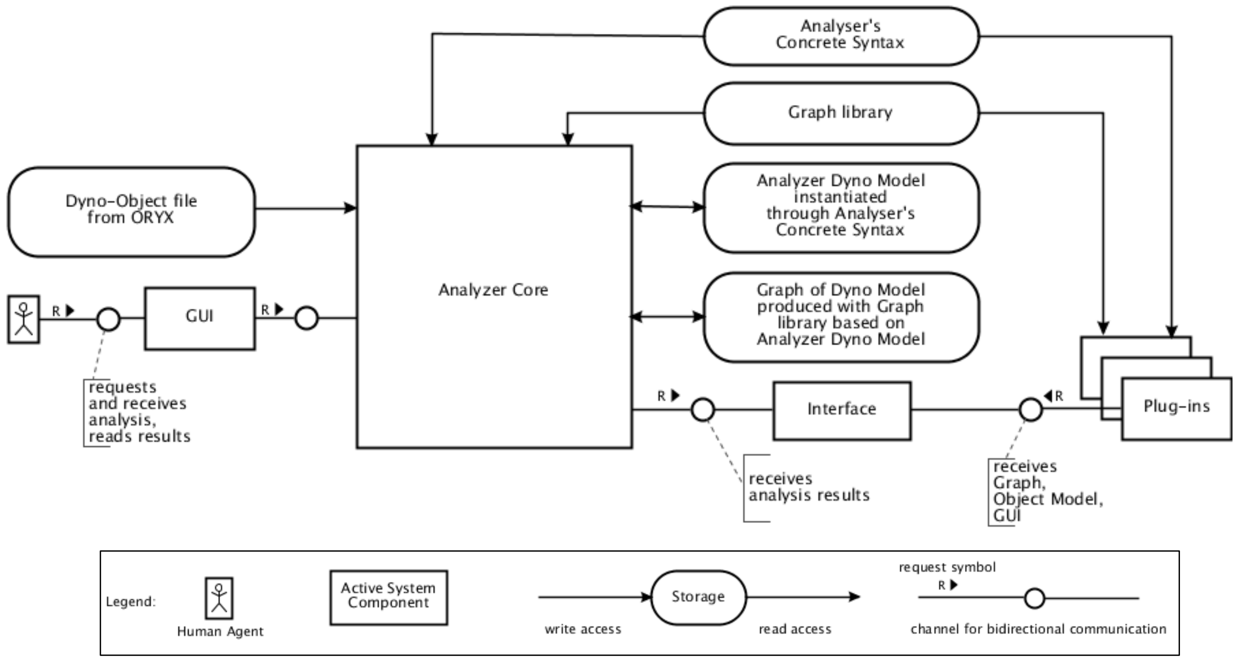

Analyser's Concrete Syntax
Graph library
Dyno-Object file from ORYX
Analyzer Dyno Model instantiated through Analyser's Concrete Syntax
Analyzer Core
Graph of Dyno Model produced with Graph library based on Analyzer Dyno Model
R
GUI
R
requests and receives analysis, reads results
R
Interface
R
Plug-ins
receives analysis results
receives Graph, Object Model, GUI
Legend:
Human Agent
Active System Component
Storage
write access
read access
request symbol
R
channel for bidirectional communication

Figure 43: Dyno analyzer

Figure 43 describes the analyzer. Its central analyzer core fulfills the following tasks:

- Read the Dyno object file, exported from the Model Designer in JSON format.
- Read the analyzer's concrete syntax, implemented in Java. This syntax is a concrete implementation of the Dyno meta-model and enables Java-based analysis through the plug-ins.
- Read the graph library. This prototypic implementation works with the library JGraphT [150].
- Create a new java-based Dyno object (analyzer Dyno model) as instantiation of the analyzer's concrete syntax. This new model does not have any graphic representation. It is serves subsequent analysis in the plug-ins.
- Alternatively or in complement to the analyzer Dyno model, create a graph of the analyzer Dyno model for potential subsequent graph-theoretical analysis. Technically, this is accomplished through a specific library for Graph analysis (JGraphT).
- Hand over to the plug-ins via bidirectional interface the following: graph, analyzer Dyno model and GUI-container.

Depending on their implementation and objective, the plug-ins read the analyzer's concrete syntax and / or the graph library. Through the received GUI container, they feedback analysis results to the analyzer core for visualization in the graphical user interface. The plug-ins define the result's graphical representation.

The conception of the analyzer shows that the object files, produced through the concrete implementation of Dyno allow for further going analysis. The subsequent analyses intend to:

- Substantiate the languages expressiveness beyond pure modeling, highlighting its ability to also convey information for analysis and recommendation in a semantically correct way.
- Provide evidence for the languages effectiveness, describing how well it can express information for its specific target group and purpose. I.e., the following section shows that the object file holds data for subsequent analysis with respect to network effects.
- Show that Dyno can improve the modeler's efficiency in guiding him through provision of structured information and potentially through subsequent recommendations.

6.3.2 Analysis Plug-ins

This subsection exemplifies a selection of prototypic analyses. Those analyses are not exhaustive. They analyze the model as given in Figure 44. This model deliberately has a series of shortcomings. Several control mechanisms are not applied at all. The supply sided causal loop cannot lead to any network effect as the modeler targets only a specific partner.

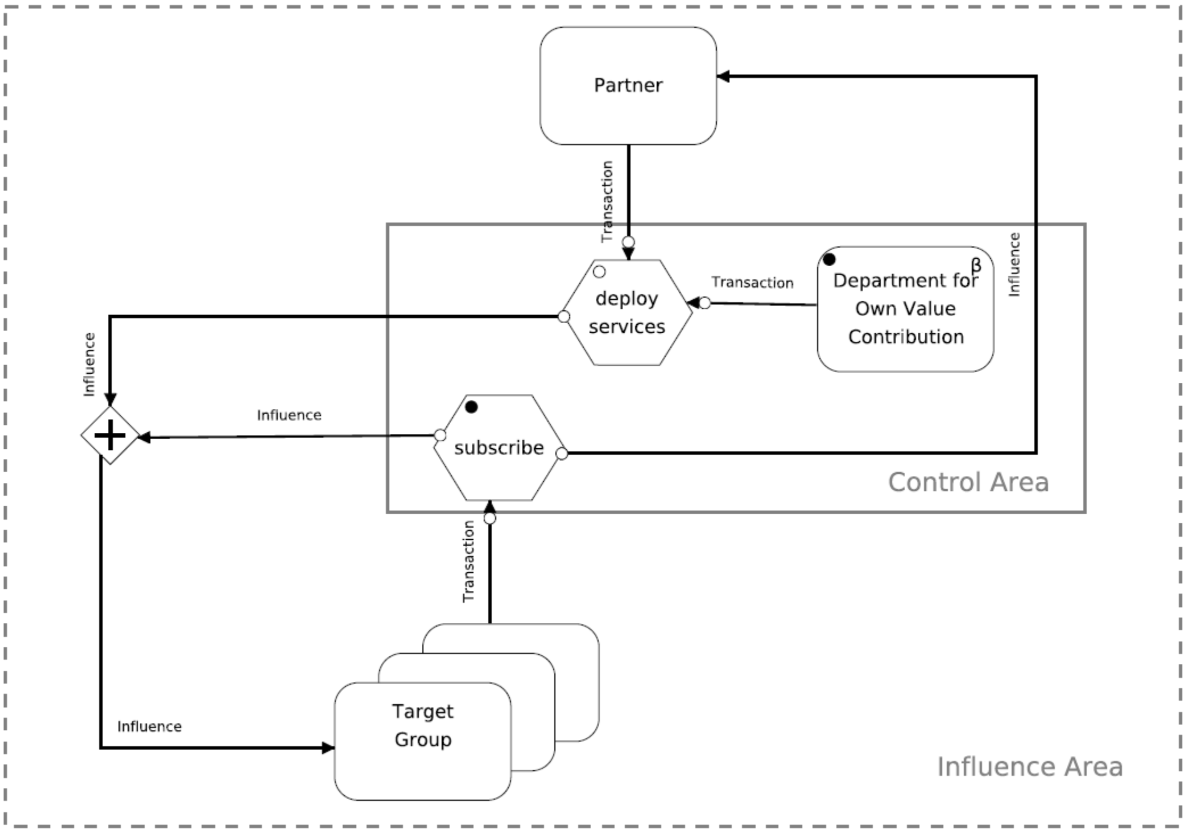
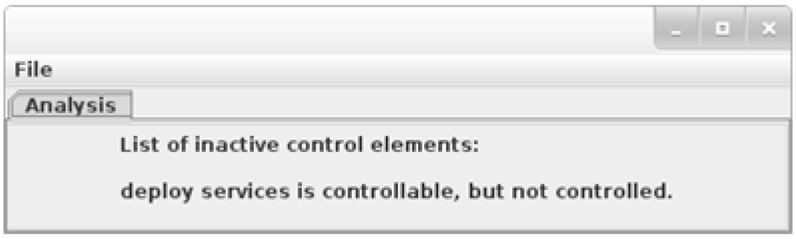


Figure 44: Dyno model

All subsequent analyses are based on the analyzer Dyno model. Experiments with the model's graph confirm that the model provides a basis for graphic theoretical analysis. Experiments provided e.g. a ranking of most prestigious and most central nodes [151]. Graph-theoretical analysis of Dyno models and the production of validated results remain subject to future research.

Analysis Plug-in for Completeness of control mechanism implementation (Nodes)

This plug-in parses the Analyzer Dyno Model and verifies whether or not there are nodes within the control area, where no control mechanisms are active. Based on this list, the modeler can reflect, whether it is of value to add additional control mechanisms. This solution is rudimentary and limited on nodes. In more comfortable realizations, the analyzer would include the edges. It could e.g. further check each possible control mechanism per element.



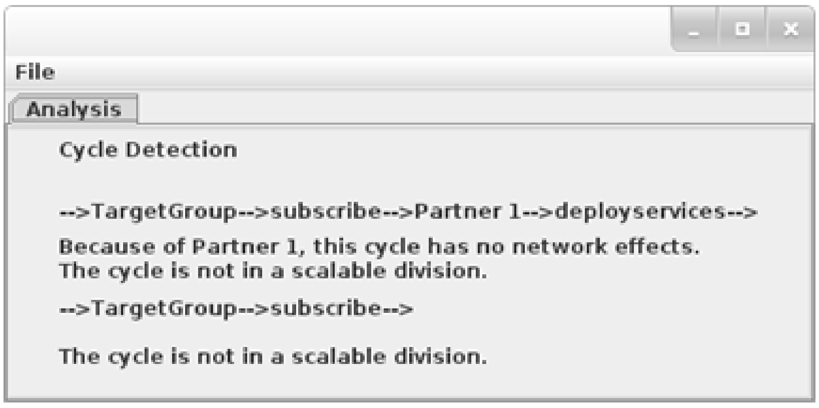
```
List of inactive control elements:

deploy services is controllable, but not controlled.
```

Figure 45: List of elements with missing control mechanisms

Figure 45 shows a list of elements with missing control mechanisms.

Analysis Plug-in for Detection of Loops

Figure 46 lists the loops in the Dyno model. It further highlights the shortcoming in the cross-sided loop. The modeler puts only one single partner into the loop. He also did not place the activity *deploy services* into a scalable environment. The plug-in also highlights that the demand-sided loop lacks a scalable environment. This analysis builds on the Analyzer Dyno Model.

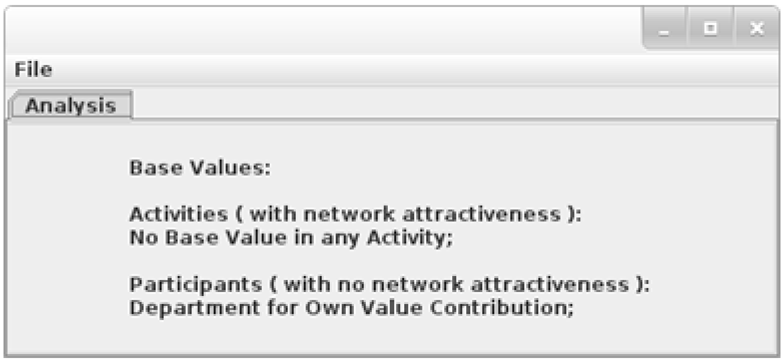


```
Cycle Detection

-->TargetGroup-->subscribe-->Partner 1-->deployservices-->
Because of Partner 1, this cycle has no network effects.
The cycle is not in a scalable division.
-->TargetGroup-->subscribe-->

The cycle is not in a scalable division.
```

Figure 46: List of loops and highlighting of the shortcoming

Analysis Plug-in for Verifying the Availability of Base Values

This plug-in verifies, whether the system includes base values. It further comments, whether the base values are within an activity or within a participant. This differentiation is important as the latter do not exhibit any network attractiveness.



Base Values:

Activities (with network attractiveness):
No Base Value in any Activity;

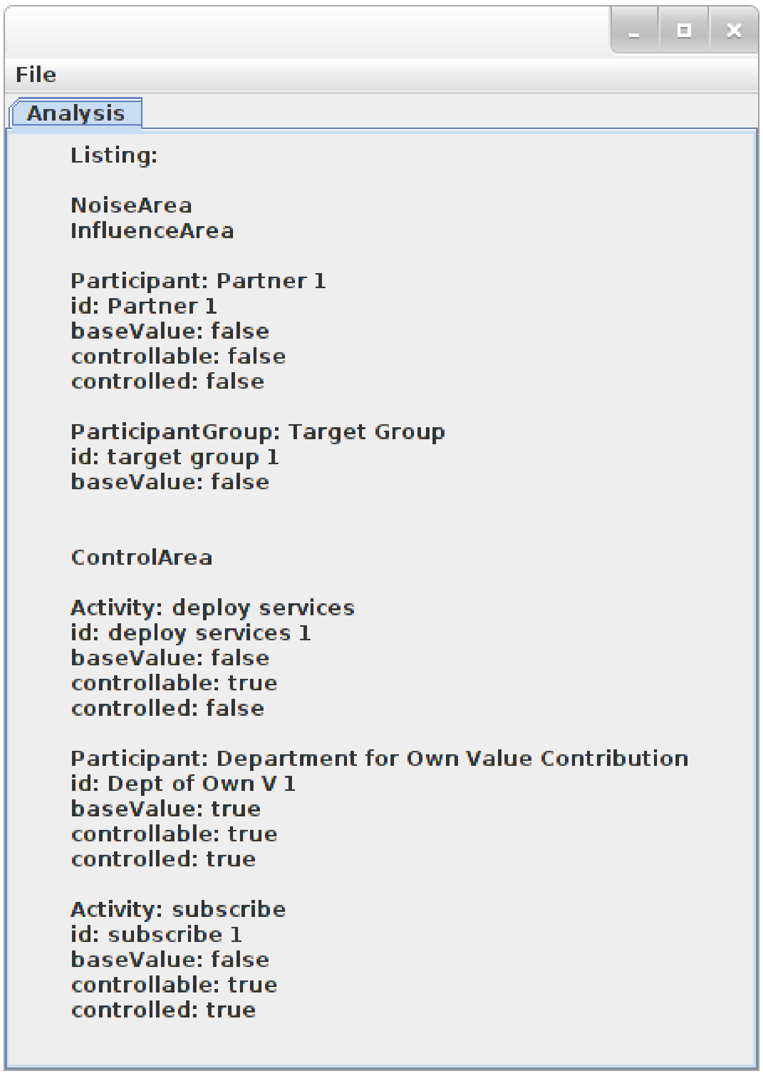
Participants (with no network attractiveness):
Department for Own Value Contribution;

Figure 47: Analysis indicating the participant carrying the base value and the fact that it lacks network attractiveness

The model in Figure 44 includes one base value represented by the *Department of Own Value Contribution*. As Figure 47 correctly states, this base value is carried by a participant and does not exhibit network attractiveness. Such a result should motivate the modeler to verify whether he could potentially model base values on an activity. One option for improvement would be e.g., to integrate causal loops through collaborative scenarios around the *subscribe* activity, designed on small thresholds. A future release of this plug-in could suggest alternative base value scenarios, e.g. through referencing suitable patterns in the pattern repository.

Analysis Plug-in for Structured listing of included elements, their attributes and descriptions

Figure 48 gives a structured listing of the model's nodes, as well as their attributes. The list of edges and divisions would look similarly.



```
Listing:

NoiseArea
InfluenceArea

Participant: Partner 1
id: Partner 1
baseValue: false
controllable: false
controlled: false

ParticipantGroup: Target Group
id: target group 1
baseValue: false


ControlArea

Activity: deploy services
id: deploy services 1
baseValue: false
controllable: true
controlled: false

Participant: Department for Own Value Contribution
id: Dept of Own V 1
baseValue: true
controllable: true
controlled: true

Activity: subscribe
id: subscribe 1
baseValue: false
controllable: true
controlled: true
```

Figure 48: Fragment of structured listing of elements including their attributes and description.

7 Evaluation

The present chapter presents the evaluation of the Dynamic Network Notation and its editor through a set of case studies. The field studies shall help scrutinizing DYNO and the editor DynoCloud.org and evaluate their ability to respond to the underlying hypotheses of the present work. They shall show that DYNO improves platform and ecosystem design in comparison to design without the tool.

Focusing on the Dynamic Network Notation, the analyzer is not part of the evaluating case studies. Its prototypic implementation substantiates the claim that Dyno is expressive enough to allow for further going analysis. Also the pattern language does not receive any consideration in the case studies. Its well-formedness is validated, once the Dynamic Network Notation is validated, as its patterns build on Dyno, complemented through descriptive text sections. Schuster [152] instantiated and discussed model and infrastructure to proof technical feasability.

This chapter first introduces the general evaluation approach (Section 7.1) and then sequentially implements two case study methods. Case study setting I conducts an expert workshop with one platform operator. The author and the experts jointly model and discuss the expressiveness and quality of the Dynamic Network Notation (Section 7.2). After integrating lessons learnt and implementing the editor DynoCloud.org, the author conducts questionnaire based case studues with three participating companies (Section 7.3).

7.1 Evaluation Approach

Following the design science research method [45], the Dynamic Network Notation and its instantiation require evaluation. On the basis of Pfleeger and Kitchenham [153] the present work accomplished the following steps to design and conduct the evaluating survey:

- Fixing specific and measurable objectives as well as defining the suitable population for the study;
- Composition of sample groups;
- Survey design and validation;
- Conducting the study;
- Analyzing and scoring the results.

Pfleeger and Kitchenham’s [153] suggestion slightly differs from the chosen sequence; i.e. they suggest selecting the participants (sample groups) after the validation of the questionnaire. The present work fixes the targeted population in step a), as the choice of a target group for notation and tool is directly related to the accomplishable objectives in our survey. It also defines the sample groups before designing the questionnaire, because the questionnaire requires a target-group specific language.

7.1.1 Composition of Sample Groups

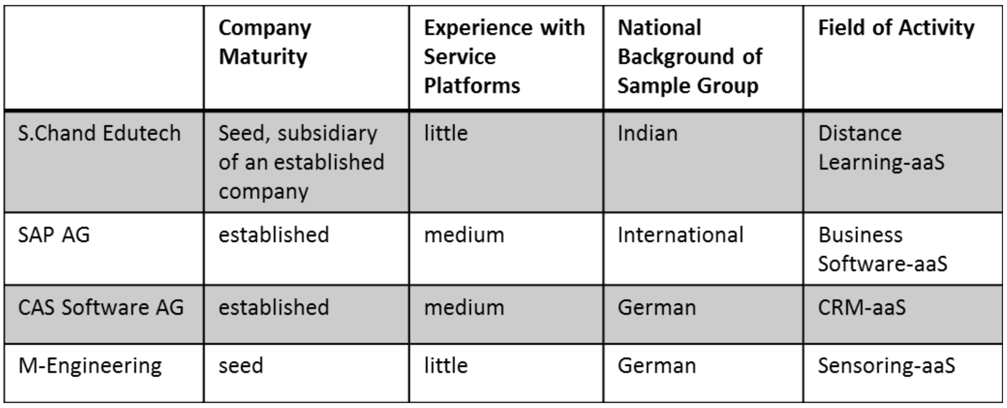
The target groups on which the notation was designed are platform architects and solution managers. The tool is hence positioned at an intersection: at the junction between engineering and management. Both target groups need to be met. They shall both be represented in the *population*. The term population denotes the complete target group of a survey. The *population*, targeted by DYNO, are the system architects and solution managers in charge of service platforms. In order to be able to produce results without analyzing the complete population, the survey requires suitable subsets called samples, or sample groups. To produce precise and reliable results, the sample groups need be well-defined and well represent the population. [154]. Challenges in the choice of sample groups are among others the heterogeneity of the service platform markets (from online games to business software platforms), different maturity of the companies, different cultural backgrounds. The present work chose a sample group of 4 companies with highly varied backgrounds to cater for heterogeneity:

(a) S.Chand Group[33], a provider of Distance Learning as a Service from own and external providers,
(b) SAP AG[34], a provider of business software,
(c) CAS Software AG[35], a provider of CRM software,
(d) M-Engineering[36] , a provider of SCADA[37] as a service.

The samples chosen in this case study are non-probabilistic, meaning they are chosen out of an accessible subset of the target population [155]. The reason for this choice is the novelty of that concept and the resulting limited availability of target groups. Also, the analyzed data is critical to corporate success. Therefore, companies without personal relation to the research group tend to be restrictive with the communication of data. To live up to the challenges stated before and to set grounds for a generalization of results, Scholten selected a subset with heterogeneous backgrounds (Figure 49). Two of the companies are older than 25 years and established in the market (SAP AG and CAS Software AG), one is in a seed stage but subsidiary of an established company (S.Chand Edutech) and one is an independent start up in seed stage (M-Engineering). The companies SAP AG and CAS Software AG have medium experience with service platforms

[33] www.SChandEdutech.com
[34] www.SAP.com
[35] www.CAS.de
[36] www.M-Engineering.de/
[37] SCADA abreviates: Surveillance, Control and Data Acquisition

(more than 2 years, but no market leader position). S.Chand Edutech is experienced with Internet-based distribution of books. M-Engineering is new to the service platform business. S.Chand Edutech is of Indian corporate culture. SAP has an international culture other members of the sample have different nationalities. CAS and M-Engineering are companies with a German culture. As the case studies are based on a limited quantity of samples, the present work uses the expression *qualitative case study*.



	Company Maturity	Experience with Service Platforms	National Background of Sample Group	Field of Activity
S.Chand Edutech	Seed, subsidiary of an established company	little	Indian	Distance Learning-aaS
SAP AG	established	medium	International	Business Software-aaS
CAS Software AG	established	medium	German	CRM-aaS
M-Engineering	seed	little	German	Sensoring-aaS

Figure 49: Sample group for evaluation of Dynamic Network Notation and DynoCloud.org editor.

Prior to the case studies, Scholten verified the questionnaire with a pilot group, consisting of 3 team colleagues. This preliminary study was designed to ‘help identify missing or unnecessary questions, ambiguous questions and instructions…., focus groups also contribute to the evaluation of instrument validity’ [156].

7.1.2 Survey Design and Validation

Theory offers a range of approaches to conduct surveys. Kitchenham and Pfleeger [154] categorize surveys into descriptive and experimental survey designs. Descriptive designs “describe a phenomenon of interest” [154], experimental design assess “the impact of some intervention” [154]. Given the present work’s design-scientific nature and the objective to evaluate the improvements brought about through the application of artifacts, experimental design is the most suitable. Four approaches of experimental survey exist:

(a) Concurrent control surveys: the scientists distribute samples randomly or non-randomly into parallel groups. The survey compares the outcomes of sample groups with and sample groups without treatment.

(b) Self-control surveys: the same sample group provides data on a situation before and after a treatment;

(c) Historical controlled surveys: the scientists repeat similar studies with respect to context and population over time to identify an evolution;

(d) Combined techniques, a mixture of several of the above approaches.

The present work chooses the option of a self-controlled study. This gives the possibility to compare results of the same sample group before and after support by the Dynamic Network Notation or the DynoCloud.org-editor. In particular, the study compares design experience without the tool with subsequent experience with tool support. Such a design provides information on changed user experience and benefit in the two different settings. In consistence with the present work’s overall research design, the survey follows an iterative approach of artifact building, intervention and learning [38].

To get an initial understanding on the notation’s usability and to be able to properly set expectations and to dimension design tasks in the survey, Scholten conducted the first evaluation in form of an expert workshop with the sample group S.Chand Edutech (Section 7.2). After documenting their initial planning on how to proceed with the Distance learning platform, Scholten accompanied 1 solution manager and one platform architect from S.Chand Edutech in modeling the platform. This procedure gave initial understanding on how the language was understood, what introduction the sample members would need, or what results could be expected from modeling. This initial experiment was still done without the editor, which was at that time still under testing. The results were used to modify the editor and to better conceive questionnaires and accompanying document for the second setting. In this second setting, done with CAS Software AG, SAP AG and M-Engineering, experts from the sample group had to model themselves, equipped with explaining material. During the experiments at SAP AG and CAS Software AG, Scholten was physically not present. At M-Engineering, Scholten was present to discuss ideas after the sample group had finished the modeling task.

Also in this second setting, Scholten used an iterative approach. The questionnaire-based surveys started with CAS Software GmbH. Including the lessons learnt, it continued with SAP and M-Engineering in a modified approach.

7.2 Case Study Setting I

Parts of the evaluation results from this case study setting were published and discussed at the International Conference on Service Oriented Computing and Architecture 2011 in Irvine [157]. The preliminary notation used dashed lines to display influences and continuous lines to represent transactions. Circles with an included plus sign represented inactive control mechanisms. Also, the base value did not have a representation yet. Another difference was that in the applied preliminary version, influence targets were signed (prefix + or -). As the editor was not yet implemented, all modeling was done with PowerPoint (Figure 50).

The subsection continues with a description of the necessary preparations (Subsection 7.2.1). It then describes the experimental procedure (Sebsection 7.2.2). Its main part (Subsection 7.2.3) describes the findings, summed up by the subsequent conclusion (Subsection 7.2.4).

7.2.1 Preparation

Pursuing an iterative research process, the research project required, apart from critical discussions in the research community, interventions with industry to gather feedback on the language expressiveness and effectiveness, as well as on the achievable quality of the produced models. The following case study took place in 2011 at the S.Chand headquarters in New Delhi, India. It was the first intervention in the market other than the feedback, gained with SAP Research. The case study took place in form of a modeling workshop with two experts from S.Chand Group. That was prior to the accomplishment of the editor and required the preparation of all major Dyno shapes in Microsoft PowerPoint. Scholten participated in the workshop and the modeling process to gain first hand understanding of Dyno’s strengthes and shortcomings.

7.2.2 Experimental Procedure

Goal was to evaluate the suitability of the Dynamic Network Notation in a first face to face session and to conduct potential improvements before starting the second round of experiments. The sample group consisted of 2 experts. One expert was in charge of the platform based-distribution concept and contributed the commercial view. The second expert was an IT specialist with experience in Internet-based distribution gained at the computer technology corporation Dell Inc[38].

In the beginning of the workshop, the experts presented their platform design views, documented with a structured list of required technical features. Afterwards, Scholten gave two hours of introduction into service platforms, based on structured presentation of successful service platforms, i.e. by Salesforce, Netsuite as well as on several smaller e-Learning platforms. Thereafter, he gave one hour of introduction into the Dynamic Network Notation and the editor, which he had prepared with MS PowerPoint. Then they modeled together for two hours, while discussing and evaluation the decisions and commenting on the notation.

Scholten interviewed the two experts after the experiment and produced a summarizing document. Given the open character of such a workshop, the setting would be less useful at the end of a research project when searching for final assessment. Interviews and discussions provided feedback for subsequent improvement, prior to conducting the questionnaire-based surveys.

S.Chand was deliberately chosen for this experiment. One reason is the experience with Internet-based distribution. The other is its strength in the market. S. Chand Group is the second largest academic publisher in India[39]. This market position endows S.Chand from the beginning with an existing large customer group, big enough to justify solutions with third party service providers. Through several initiatives the company tries to stay in phase with the rapid growth of Internet adoption. India accounts for more than 100 Million Internet users in 2010[40]. Internet access of students is still growing, supported by many government and private initiatives and fueled by many Internet Cafés [158]. In order to capitalize on this growing segment, S.Chand Group creates the subsidiary S.Chand Edutech (SCE)[41] assigned to design a service platform that

- scales with growing quantities of users and providers as well as seasonal effects through rented infrastructure, without the need for stockpiling of IT infrastructure (A);
- leverages on S.Chand’s existing position in the academic text book sector in terms of width and depth of portfolio and in terms of distribution channel (B);
- benefits from the existing multitude of e-Learning development companies (C) ;
- offers content that responds to a diversity of academic classes (D);

[38] Homepage of Dell Inc. India: http://www.dell.co.in/
[39] S.Chand Group Homepage, http://www.schandgroup.com/, retrieved 13.08.2011;
[40] Internet World Stats Homepage, http://www.internetworldstats.com /asia/in.htm, retrieved 13.08.2011;
[41] S.Chand Edutech Homepage, http://schandedutech.com/, retrieved 02.03.2013.

- works around shortcomings in national telecommunications infrastructure through placing own servers into university environments. The company manages the servers through TCP/IP based fileserver access and synchronizes overnight. On the university side, the servers are connected to the University LAN. The university only has webserver access (E);
- assures suitable quality levels (F).

7.2.3 Findings

Before starting to model, the experts had already made and documented design ideas verbally, without any tool support. They compare their initial ideas with those produced with Dyno throughout the modeling process. The experts started the development of the model from the beginning. The model as depicted in Figure 50 is the result of an iterative collaborative modeling process within the workshop. As with their core business, SCE targets students as primary consumers. The experts modeled the Dyno model as shown in Figure 50. The upper case letters followed by a number associate each element to the above letters A-F. When designing the model, cooperation with the partners (e.g. universities, or initial content providers) existed, but it was not embedded in the context of a platform-based concept. A first challenge was the base value arrangement. Before modeling, the experts thought about a Credit-Card based purchase process or a voucher sale within the universities. The Dyno-based modeling made them neglect this option, as it was not reverting to an established distribution channel. To build on an existing channel, the company chose the rented book shelves, which they modeled as base value 1. Secondly, the experts initially conceived a production process through outsourcing on assignment (dependent participants in the control area). However, the notation allowed them to model an alternative setting with independent providers in the influence areas and thus to create the complementarity loop. Experimenting on divisions, they decided to place the service platform on a scalable IaaS, provided by a third party, as DYNO showed that their own infrastructure would create a bottleneck within the important demand-sided causal loop. DYNO revealed a second bottleneck in the non-scalable server architectures within the universities. For the time being however, the experts accepted this bottleneck in a trade-off for better network performance to the students.

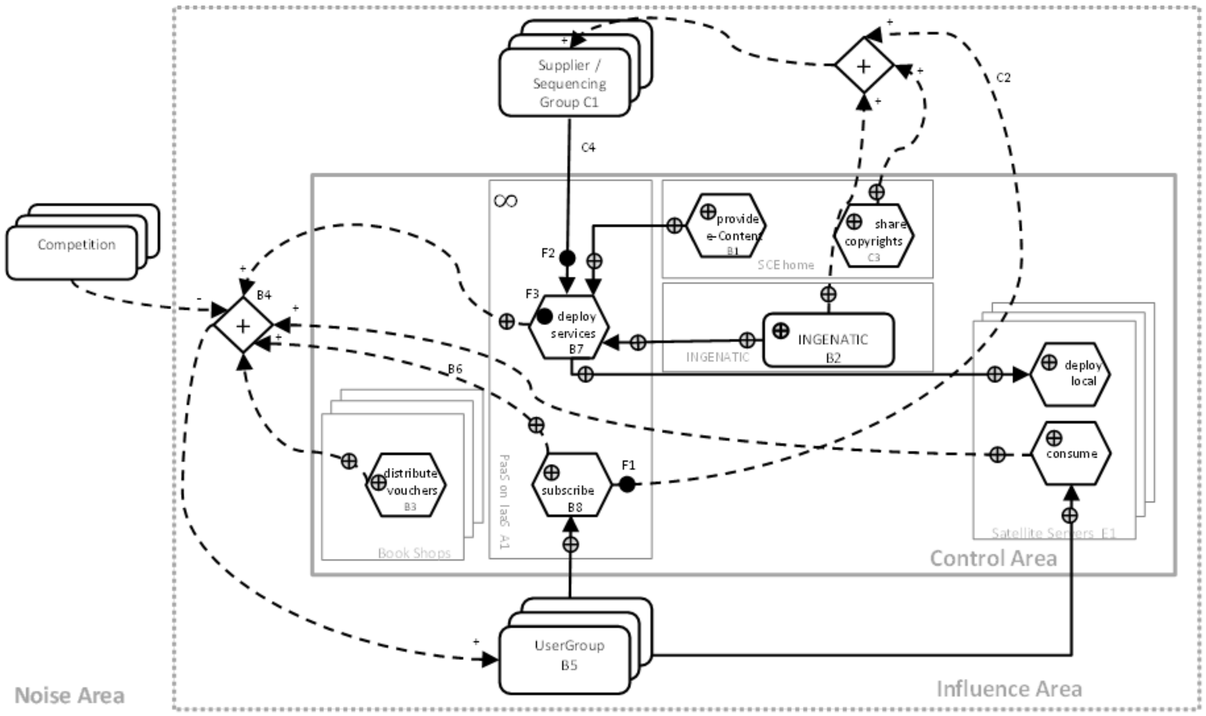


Figure 50: S.Chand Edutech platform design

Figure 50 reproduces the platform model as developed within the workshop. Some arrangements, e.g. activities without inflows from participants or participant groups are semantically questionable. However this reflects the modeler's way of thinking and is therefore reproduced unchanged in the present work.

Modeling & Analyzing Demand-side Base Values

Section 2.7 highlighted the necessity of an available base-value as prerequisite to incite a network effect. The experts identified and modeled three base values to set off this initial causal loop on the demand-side. Then, they aggregated base values through a merging gateway.

- *Electronic content provision* - SCE decides to offer digitalized versions of a significant amount of their academic books through the e-Learning platform. The experts modeled content provision as an activity (Figure 50, B1).
- *Take-over of an established e-learning developer* – In order to start off their business, SCE took over INGENATIC, a European Multimedia company with an existing portfolio of web-based academic distance learning content. The experts modeled this as

dependent participant within a specific division (Figure 50, B2) and supplying the *deploy services* activity (Figure 50, B7) to stimulate the consumer-sided network effect through a critical mass of content.

- *Distribution in book shops* – To reach the students, SCE decides to use S. Chand Group's existing book-shelves in the nation-wide network of book resellers. Non-transferrable time-limited subscription vouchers will be sold through these channels (Figure 50, B3). The division group symbol describes the availability of a non-quantified amount of book-shops. If the quantity was invariant and known, it could be added through a comment. This modeling fragment showed that modelers left the limits of IT-infrastructure located activities to integrate rented book-shelves into Dyno's line of thinking. Being controlled by SCE, allocating the bookshelves within their control area is correct.
- *Integrating through a Gateway* – The modelers aggregate the positive influences of all previously described values (1-3) within the Gateway (Figure 50, B4) to take effect on (potential) users. Competitors weaken this accumulated positive impact. Consequently, the competitor group's influence carries a negative prefix.

The activity *provide e-content* (Figure 50, B1) and the participant *INGENATIC* (Figure 50, B2) act indirectly through the activity *deploy services* (Figure 50, B7). The modelers positioned the target users into the influence area. They applied the symbol for participant groups to emphasize that they do not speak of explicit users, but of a non-quantified group (Figure 50, B5). Once the first users subscribe, the initial loop – a causal loop (B6) – is started off. Once the quantity of supplied content exceeds the threshold for network attractiveness through the activity *provide e-content* (Figure 50, B2) and through the participant *INGENATIC* (Figure 50, B2), the consumer-sided network effect comes into force.

Modeling & Analyzing Supply-side Network Effects and Base Values

During the time of the experiment, SCE was piloting with a handpicked selection of initial third-party suppliers to gather experience for a subsequent opening up to a multitude of providers. To remain valid during the two stages of ramp up, the experts modeled all providers as one group (Figure 50, C1). Becker, Rosemann et al. [159] call such robustness against changes over time *economic efficiency*. Having set off the dynamics on the demand-side, the suppliers are already attracted by the volume of users, promising potential for turnover (Figure 50, C2). SCE strengthens this in having the permission to using S.Chand's copyrighted material (Figure 50, C3) and a plentitude of design templates, provided by INGENATIC for ease and efficiency in development and coherence in look-and-feel (Figure 50, B2). As the suppliers deploy services (Figure 50, C4) in reaction to the stimulus (Figure 50, C2), it creates a complementarity loop [8]. The complementarity loop in conjunction with the demand sided loop form a cross sided loop, where the deployment of services (Figure 50, B7) and the subscription of users strengthen each other in reciprocity.

Modeling & Analysis of Complementary Portfolio Contributions

SCE decided to start off with technical contents. Still there is a multitude of academic classes to map, with differences in expectations and orientation, depending on the university. It starts off with ready-made courses to meet all content requirements. Also would it be impossible from an expertise, time and budget perspective to fill the missing spots. SCE's choice is to delegate value creation into the supply chain. The solution, again supported through SaaS providers is as follows:

> *Provision of sharable content objects (SCO)* and of *sequencing manifestos* – Instead of providing ready-made e-Learning courses, SCE demands the SaaS providers (Figure 50, C1) to provide fine-granulated SCOs. SaaS providers also provide IMS-manifestos, suggesting context related sequencings of reusable SCOs. XML-based IMS manifestos reference the included SCOs and their respective sequence of invocation[42]. E-Learning can be considered a network of SCOs, or in SOA terminology a composite service. Typically, the initial IMS-manifesto comes from the SCO-providers, giving a first suggestion for a course. Alternative manifestos can be created e.g., through lecturers, adapting the existing content base on their requirements, complemented through additional SCOs. To ensure the interoperability of manifesto and SCOs, SCE needs to demand compliance to the respective Sharable Content Reference Model[43] as programming model through their platform service provisions.

Modeling & Analyzing Scalability Requirements

With respect to the fast growing market, SCE decided not to invest into a proprietary IT infrastructure, but to build an e-learning service platform on top of an Infrastructure-as-a-Service (IaaS), provided by a third party provider. The experts modeled the IaaS as a technically scalable division (Figure 50, A1). Indian internet connectivity is of good quality in the metropolitan areas, but many university locations still suffer from a limited bandwidth. Working around this shortcoming and benefiting from the existing University-WLANs, SCE places proprietary 'satellite servers' into the Intranets of partner universities (Figure 50, E1). The servers are synchronized

42 IMS standard, http://www.imsglobal.org/content/packaging/, retrieved 13.08.2011

43 SCORM reference pages page by the US government's advanced distributed learning department, http://www.adlnet.gov/scorm/, retrieved 02.03.2013

through with the central server. The downside of this approach is the lack of scalability in the finite divisions on the university campuses. The experts modeled this through *scalability: boolean = false* in the subdivision group of satellite servers. Through using the symbol of a division group, they show that commercially, the project can grow with any additional campus connected. The lacking infinity symbol pinpoints a potential bottleneck, as the servers are a non-scalable potential bottleneck within a loop.

Modeling & Analyzing Control Points & Mechanisms for the PaaS Ecosystem

Working with external provisions implies a new paradigm of quality control and of enforcement for SCE. There are many possible control points and corresponding sets of control mechanisms in the model. The remainder of this subsection exemplifies one control mechanism on a transaction, one mechanism on an influence and one on an activity. This preliminary version of the language, concretized through MS PowerPoint, was not able to be more specific about the availability of control mechanisms as through a Boolean *true* of *false* information.

- *Influence-Control* - In this scenario, the experts place control mechanisms on an influence (Figure 50, C2). To amplify its impact on the influenced suppliers, they applied informative control and motivational control (Figure 50, F1). The experts discussed the specific mechanisms of informative control. First, to inform the (potential) supplier about the quantity of users subscribed in order to increase the attractiveness. Second, they suggest customized information per SaaS provider on user preferences to motivate the suppliers to optimize their offers on the user requirements.
- *Transaction-Control* – As the transaction C4 points into the control area (Figure 50, C4), the target-side is controlled). In the modeled solution, a restrictive control mechanism verifies through an automated routine the suitability of the provided service. I.e. it verifies compliance with the stipulated programming model (SCORM). Incompliant services are rejected.
- *Activity Control* – Consecutively, the modelers apply sanctional control in Figure 50, F3. In example, this give the platform operator away to undeploy services that are continuously rated poorly through the reputation systems (market-regulative control).

7.2.4 Conclusion

In the preliminary research stage at the time of the experiment, DYNO still showed scope for further refinement with respect to expressiveness and well-formedness. The signed targets of influences (plus, minus) were a first subject of discussion. Whether an influence or an aggregated influence has positive impact may change over time. To produce models that are robust over time, the release of Dyno, published in the present work abandoned the signs at the influence targets. Following suggestions by the experts, the version of Dyno published in this work visually represents the activation of the base-value attribute through a β–smbol. The modeling desire of voucher-based sales in SCE's book-shops led to a loosening of the requirement, that an activity necessarily is a technically enabled place for interaction.

The experts also discussed whether to change from a binary consideration of a base value to a qualitative. The present solution however did not integrate this proposal due to two reasons. First, a quantification of a base value is difficult to achieve, due to its dependency on various aspects of attractiveness. Second, being quantitative would weaken the models robustness over time.

Unresolved criticism lies in the binary consideration of scalability and base value. Being able to quantify those would allow for improved infrastructure dimensioning. Scalability is a subject to ongoing research. During the time of the publication of this thesis, no suitable metric on scalability was available. Those issues are subject to further research.

7.3 Case Study Setting II

This second case study setting is based on a questionnaire. It was conducted in summer 2012 (first company) and in winter 2012 / 2013 (2nd and 3rd company), after the lessons learnt from the first case study were implemented. Also the editor was already available.

The following subsection starts with a substantiation of the questionnaire design and the execution of the study (Subsection 7.3.1). Then it describes the experimental procedure (Subsection 7.3.2), followed by a description of the findings in the three consecutive interventions (Subsection 7.3.3). After another iteration following the investigation at the first company, the investigations at the 2nd and 3rd company allow a final evaluation at of language and editor. Final results in the context of the present work represent the actual research and implementation achievement submitted in the frame of this thesis. The Action Design Research method theoretically is an ongoing process of cycles , interweaving theory generation, design and intervention respectively evaluation, and thus contributing to an iterative refinement of the artifact [38]. The section closes with a conclusion (Subsection 7.3.4).

7.3.1 Preparation

For the design of a questionnaire Kitchenham and Pfleeger [160] suggest to prepend research of comparable studies to build on past experience. As language and tool are new artifacts, no previously acquired data is available. However, Scholten reverted to questionnaire design by Wittern and Zirpins [161], as well as Lenk[44], both evaluating modeling and engineering tools for service

[44] A.Lenk, unpublished work in progress, Karlsruhe Institute of Technology.

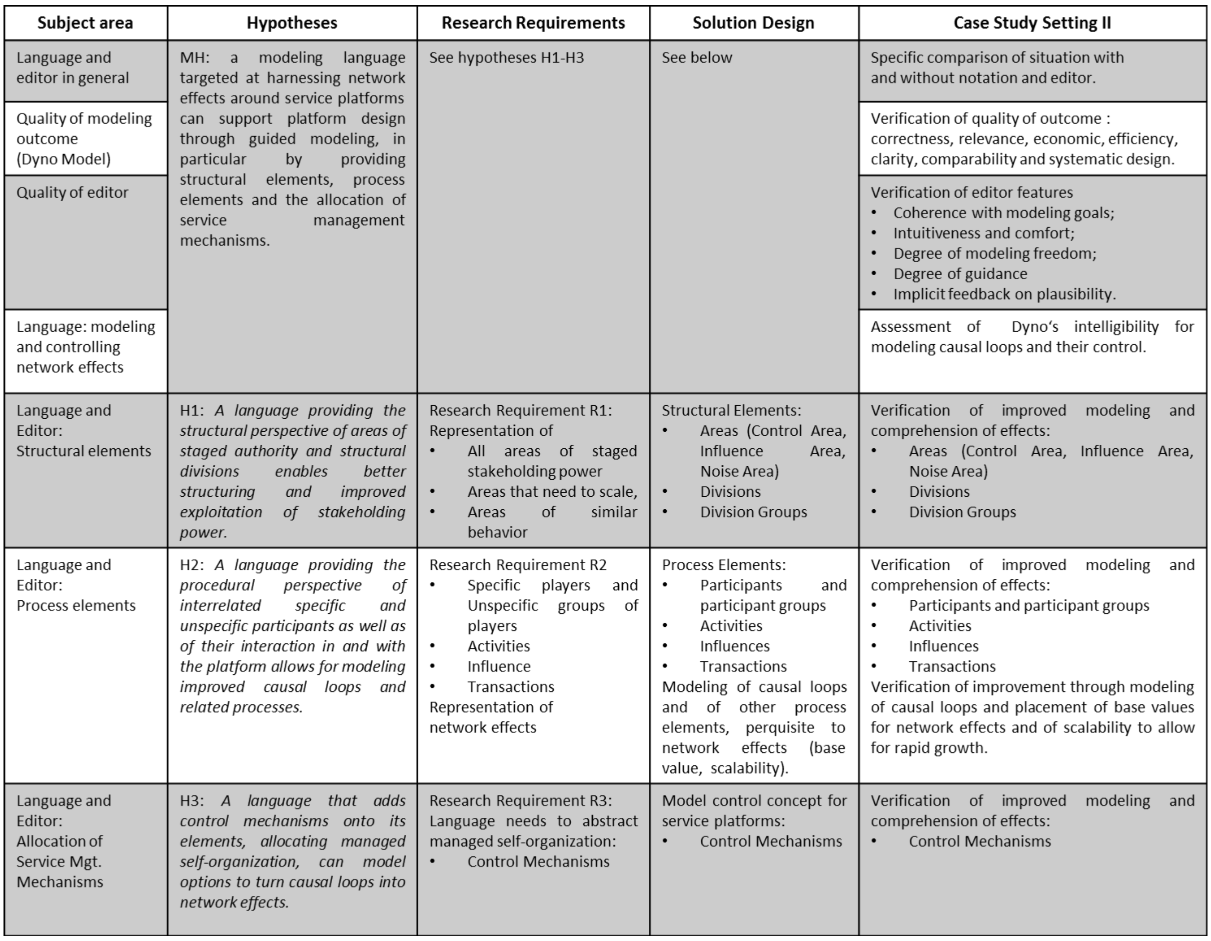
composition. Those studies gained experience on assessing tools and thus contributed to the questionnaire structure.

The questionnaire follows a series of design principles: It starts with an introduction of research group and purpose of the survey. The questionnaire also describes how the acquired data will be treated to create an environment of trust among the respondents [162]. The questionnaire mainly builds on closed questions mixed with few open questions. Closed questions allow quick response and thus promise a high response rate. They are easier coded for contexts of quantitative analysis or direct comparison [162]. The latter is decisive for the prioritization of closed questions in this thesis. Open questions elicit qualitative data but may also discourage, as they require more time to respond. It is also more difficult to analyze answers to open questions than those given on close questions [162]. The questionnaire allocates open questions, when explicitly addressing the respondents' creativity, e.g. when asking for suggestions of improvement. To understand the acquaintance of the respondents with the subject matter, a question of their familiarization with modeling software prepends the subject-oriented questions.

The questionnaire's objective is to verify whether language and editor respond to the research requirements and confirm the research hypotheses. To slowly introduce the respondent into the subject without deterring him from accomplishing this task to the end, the questionnaire follows a sequence of questions from general to particular and from factual to more abstract [162]. Given this frameset, it is not possible to sequentially step through the research requirements respectively through the research hypothesizes. The survey faces another challenge with respect of data elicitation. As the editor is a concrete implementation of the language, it is difficult to differentiate between pure editor and pure language features. As the main differentiating point is the quality of implementation of the editor's graphical user interface, the questionnaire specifically starts with this. After that, the questionnaire assesses the respondent's experience with and his assessment of the language. It starts with a request for general evaluation of the comprehensibility of the Dynamic Network Notation and its helpfulness to understand dynamic networks around platforms. Following the sequence of research requirements and hypotheses, the questionnaire walks through structural areas, process elements, causal loops and then control mechanisms. Given the importance of causal loops and network effects in the context of this thesis, the questionnaire dedicates a small visually separated block to causal loops. The questionnaire uses the term dynamic loops, a term used in preliminary stages of the model development, but replaced by the term causal loops. The term *causal loop* implies causality and ensures unambiguousness. In contrast, the term dynamic led to confusion during discussions, as addressees interpreted the word in diverse ways (e.g. synonymous to flexible or volatile). The subsequent block requests the respondent's feedback on control mechanisms from different angles, e.g., suitability of suggested mechanisms, limitation of freedom of decision making, usefulness. Base value and scalability receive individual sections.

To evaluate the language's expressiveness and effectiveness and the achievable efficiency of the models, the questionnaire verifies whether it can produce models which fulfill Becker, Rosemann et al.'s [159] quality guidelines of modeling. In particular, it verifies whether Dyno helps the respondents to produce models which respond to the following quality criteria [159]:

- *Correctness* – structure and behavior of the Dyno models are consistent with the real world;
- *Relevance* – Dyno represents all relevant elements and relationships from the real world;
- *Economic efficiency* – Dyno models are *robust*, meaning it remains relevant over time, even as quantities of subscribed users or service evolve;
- *Clarity* – Dyno models are clearly understandable and not overloaded;
- *Comparability* – Design alternatives are comparable as they follow the same grammar.



Subject area	Hypotheses	Research Requirements	Solution Design	Case Study Setting II
Language and editor in general	MH: a modeling language targeted at harnessing network effects around service platforms can support platform design through guided modeling, in particular by providing structural elements, process elements and the allocation of service management mechanisms.	See hypotheses H1-H3	See below	Specific comparison of situation with and without notation and editor.
Quality of modeling outcome (Dyno Model)				Verification of quality of outcome : correctness, relevance, economic, efficiency, clarity, comparability and systematic design.
Quality of editor				Verification of editor features • Coherence with modeling goals; • Intuitiveness and comfort; • Degree of modeling freedom; • Degree of guidance • Implicit feedback on plausibility.
Language: modeling and controlling network effects				Assessment of Dyno's intelligibility for modeling causal loops and their control.
Language and Editor: Structural elements	H1: *A language providing the structural perspective of areas of staged authority and structural divisions enables better structuring and improved exploitation of stakeholding power.*	Research Requirement R1: Representation of • All areas of staged stakeholding power • Areas that need to scale, • Areas of similar behavior	Structural Elements: • Areas (Control Area, Influence Area, Noise Area) • Divisions • Division Groups	Verification of improved modeling and comprehension of effects: • Areas (Control Area, Influence Area, Noise Area) • Divisions • Division Groups
Language and Editor: Process elements	H2: *A language providing the procedural perspective of interrelated specific and unspecific participants as well as of their interaction in and with the platform allows for modeling improved causal loops and related processes.*	Research Requirement R2 • Specific players and Unspecific groups of players • Activities • Influence • Transactions Representation of network effects	Process Elements: • Participants and participant groups • Activities • Influences • Transactions Modeling of causal loops and of other process elements, perquisite to network effects (base value, scalability).	Verification of improved modeling and comprehension of effects: • Participants and participant groups • Activities • Influences • Transactions Verification of improvement through modeling of causal loops and placement of base values for network effects and of scalability to allow for rapid growth.
Language and Editor: Allocation of Service Mgt. Mechanisms	H3: *A language that adds control mechanisms onto its elements, allocating managed self-organization, can model options to turn causal loops into network effects.*	Research Requirement R3: Language needs to abstract managed self-organization: • Control Mechanisms	Model control concept for service platforms: • Control Mechanisms	Verification of improved modeling and comprehension of effects: • Control Mechanisms

Figure 51: Comparison of research requirements, hypotheses, solution design and case study questions

The questionnaire applied a non-dichotomous ordinal scale to structure the respondent's opinion. As the modeling with language and editor was done by all groups in teams, each team represents one respondent and fills out one single questionnaire. An ordinal scale arises from rank ordering [163]. In contrast to a nominal scale (relation of equivalence), the ordinal scale incorporates a *more than* or *less than* relationship [164].

The current questionnaire applies a non-dichotomous approach, giving more than one choice. The ordinal scale documents the respondent's ranked opinion with

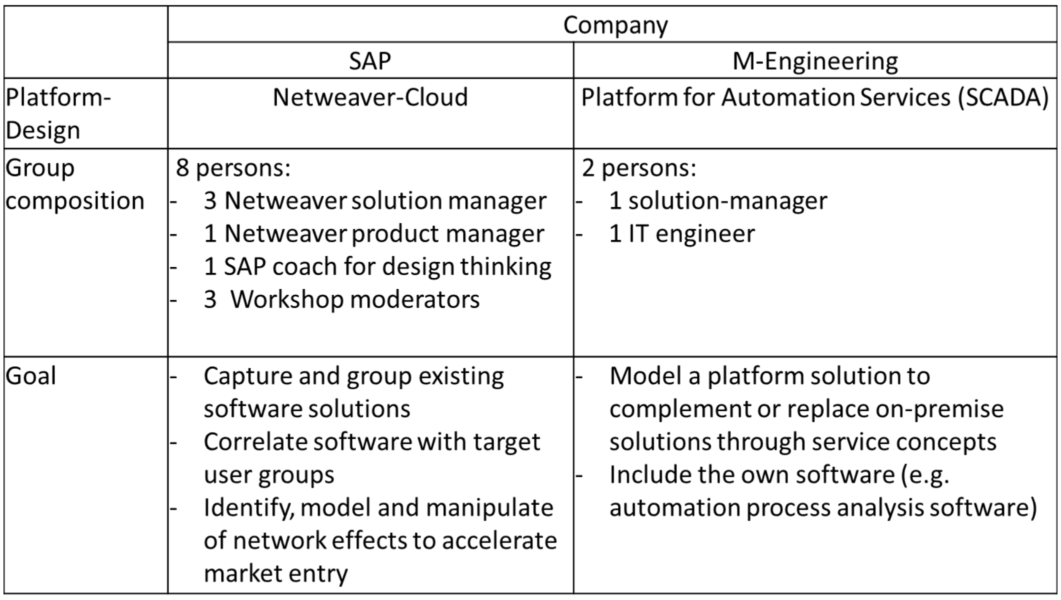
- Strongly agree (+2),
- Agree (+1),
- Disagree (-1),
- Strongly disagree (-2).

To check plausibility of answers, in two cases the questionnaire carries inverted questions, i.e. the respondent is asked to comment a negative instead of a positive effect, e.g., "… is unnecessarily limiting my scope of decision making." For these questions, the analysis inverts ranking and numeric weighting. To impose a respondent's decision towards agreement and disagreement, the questionnaire leaves out the undecided option (+/-0). The prepended assessment of the respondent's experience with modeling remains dichotomous. The goal of this pre-assessment is to make sure that the respondents have a basic experience with modeling.

7.3.2 Experimental Procedure

The case study at CAS Software AG was taken in July 2012. Based on the experience in that case study, the author complemented the questionnaire with additional and more detailed questions on structural areas and process elements. The case study at M-Engineering was conducted in November 2012, the study at SAP in February 2013.

CAS Software AG provided one solution manager to participate in the modeling experiment. No platform architect was present. The solution manager received detailed documentation on notation and editor, but no training. The solution manager modeled a scenario which he already conceived beforehand without notation and tool support. The results were moderate (see Subsection 7.2.3). In consequence, Scholten modified the case study setting slightly and prepended one hour of training for one participant of the remaining groups and extended the questionnaire.



	Company	
	SAP	M-Engineering
Platform-Design	Netweaver-Cloud	Platform for Automation Services (SCADA)
Group composition	8 persons: - 3 Netweaver solution manager - 1 Netweaver product manager - 1 SAP coach for design thinking - 3 Workshop moderators	2 persons: - 1 solution-manager - 1 IT engineer
Goal	- Capture and group existing software solutions - Correlate software with target user groups - Identify, model and manipulate of network effects to accelerate market entry	- Model a platform solution to complement or replace on-premise solutions through service concepts - Include the own software (e.g. automation process analysis software)

Figure 52: Overview of group composition of goals of SAP and M-Engineering

SAP modeled with the Dynamic Network Notation and editor in the frame of their design workshop on the service platform solution to SAP's Netweaver product[45]. The SAP workshop group consisted of three solution managers, one platform architect (called SAP product manager), one Design Science Coach and three workshop moderators. The company's goal was (a) to capture and group existing inhouse software solutions, suitable for such a concept, (b) to correlate the software solutions with suitable target user groups and (c) to identify, model network effects and suitable control mechanisms (Figure 52). Prior to the modeling with Dyno, the group already had several sessions, where they worked without the tool. The group jointed worked and discussed on one solution. One moderator served as modeler. The modeling progress was shared through an overhead projection.

M-Engineering plans to set up a service platform for SCADA services. Those SCADA services are in charge of surveillance, control and data acquisition and analysis of manufacturing processes. M-Engineering is a new entrant into the platform business, coming from the process analysis and control side. The company's objective is to complement their on-premise solutions. It intends to deploy existing solutions as a service, using Siemens' Web-enabled SCADA Toolsuite WinCC/Web Navigator[46]. Being too small to attract pools of third party service provid-

[45] Netweaver technology platform: http://scn.sap.com/community/netweaver; retrieved 24.04.2013.

[46] Siemens Win CC/Web Navigator: http://www.automation.siemens.com/mcms/human-machine-interface/en/visualization-software/scada/wincc-options/wincc-web-navigator/pages/default.aspx; retrieved 24.04.2013.

ers, the company plans to work with explicit partners on the supply side and to generate network effects on the consumer side.

7.3.3 Findings

This subsection summarized the responses, provided by CAS Software AG, SAP AG and M-Engineering UG. The questionnaires and the detailed answers are presented in the Appendix in the section Questionnaire.

Modelers' Familiarity Check

The prepended familiarity check elicited that all respondents had modeling experience.

First Iteration of Case study with CAS Software AG

The result of the first questionnaire-based case study at CAS Software AG produced an average score of 0.46, which can be read as a weak confirmation of improvement of the situation, as compared to modeling without tool. In the summarizing assessment, the group did not see any improvement in platform conception after getting using language and tool. However, the team acknowledged the improved time-effectiveness. In explicit, team did not see benefits from several of the core features of Dyno, e.g., from transactions and influences, from the areas of staged authority, or from the possibility to model two-sided markets. The team considered the guided suggestion of control mechanisms as unnecessary limiting of scope of decision and considered the whole concept difficult to grasp. The appendix in Part D includes the complete questionnaire and in answers by CAS Software AG.

Second Iteration after Modification of Study Design

The results led to the assumption that the respondents require enhanced introduction into the concept. Scholten prepared the successive field studies with SAP, and then with M-Engineering with a one hour training session for one representative per team. With a score of 1.58 (SAP) and 1.67 (M-Engineering), the results changed significantly, confirming that language and underlying theory require a basic amount of introduction. The feedback of SAP and M-Engineering showed an average deviation of 0.26 points. Although ordinal scales as used in the questionnaire only give a relative rank order, statistical methods find successful application on ordinal scales in various disciplines. Scores and deviation as given above and in subsequent analysis cannot be interpreted quantitatively, but they provide qualitative trends [163]. When speaking of the *respondents*, the remainder of this sub-section only considers SAP and M-Engineering, unless stated differently.

Evaluating the Main Hypothesis (MH)

The main hypothesis states that a modeling language for design of service platforms oriented on network effects can support improved modeling outcome. The case study scrutinizes this hypothesis from three angles. First it compares the situation with and without language and editor (a). Then it considers the language's orientation on causal loops and their control (b). It then assesses the quality of the Dyno models (c) and the concrete language implementation in the editor (d).

a) Change of situation through the provision of the language and its concrete instantiation (editor):

   Both respondents strongly agree that notation and tool help to produce better platform design than without the tool (both +2). The respondents strongly agree as well that the tool improves the efficiency of the platform design process as it reduces the required design time (both +2). Also, they agree that language and tool are quickly understandable. However the respondents deviate in their weighting, leading to an averaged score of 1.5. The improved score as compared to the results at CAS Software GmbH showed that language and notation are not purely intuitive and require introduction. Both respondents considered an hour of explanation for notation and editor as *quick*.

b) *Language orientation on modeling and controlling network effects:*

   The present work's main hypothesis states that a specific modeling language for design of service platforms oriented on network effects can support improved modeling outcome. Both groups strongly confirm the comprehensibility of Dyno during the modeling process. They also strongly confirm they get better understanding of network effects and respective control possibilities through Dyno. Similarly, they give strong confirmation that Dyno helps locating or creating causal loops. Also, they confirm that Dyno helps updating one-sided loops into two- or multi-sided loops. They thus confirm that Dyno goes beyond pure enablement of causal loop creation, given the complexity of multi-sided loops. SAP and M-Engineering evaluated the respective questions unanimously with full agreement (average score +2).

c) Quality of outcome:

   The questionnaire elicited whether the achievable quality of models, modeled with the Language and the editor satisfies the requirements of Becker, Rosemann et al.'s [159] guidelines of business process modeling.

Correctness: The respondents saw the requirement of correctness fulfilled (score +1), meaning they were able to produce one or more models, where structure and behavior of the models were consistent with the real world.

Relevance: The respondents strongly agree that all relevant elements from the real-world scenario find application in the model representation (both +2). The respondents agreed that all elements and relationships, applied in and required by Dyno are necessary to accomplish the design goals (both +1).

Economic Efficiency: The respondents consider their model as robust, meaning it remains relevant over time, even when quantities of users or services evolve (both +1).

Clarity: The two groups agreed that the produced model was clear, understandable and not overloaded (average +1.5).

Comparability: The two respondents agree that they can compare design alternatives on the same business case, as they follow the same grammar (average +1.5).

d) *Quality of the editor:*

The two groups (SAP, M-Engineering) were in cautious appreciation with the comfort of the editor. SAP's main point of criticism its lack of intuitiveness. Still, an overall ranking between agreement and full agreement on the editor features *comfort of GUI* and *level of modeling freedom* in the editor is an acceptable result for an editor, implemented for proof-of-concept. The two respondent groups strongly emphasize their benefit from the editor's guidance, based on Dyno's grammar and resulting context-related suggestions and restrictions. The questionnaire explicitly stated suggestions and restrictions with respect to transactions, influences, controllable and uncontrollable activities, case-specific mechanisms, areas and the implicit feedback on the modeler's modeling intentions. Although the questions were centered on the editor, the results support the main hypothesis (MH) on the general benefit to the modeling target group of a language which guides through context related suggestions and restrictions. The conclusion is viable as the editor is the concrete implementation of the language's grammar (complemented by GUI-features).

Respondents' Evaluation of the Dynamic Network Notation with Respect to the Structural Distribution (H1)

Hypothesis 1 (H1) states that a language providing the structural perspective of areas of staged authority and of structural divisions enables better structuring and improved exploitation of stakeholding power. The respondents agreed (average score +1.3) to the contribution of added value through areas of staged authority. The areas allowed them to better position their modeling elements (both ranked +1) and supported their understanding on the dependency of structural position of elements and their area-dependent controllability (both +2). Also they fully agreed to the contribution of division groups for repeated areas of similar behavior (both +2). The two respondents strongly deviated in the evaluation of individual divisions. Whereas SAP disagreed with their contribution (-1) agreed M-Engineering strongly (+2).

Respondents' Evaluation of the Dynamic Network Notation with Respect to the Procedural Perspective (H2)

Hypothesis 2 (H2) states that a language providing the procedural perspective of interrelated specific and unspecific participants as well as activities is able to model and improve causal loops and related relationships in and around platforms. Both respondents unanimously fully agree (both +2) that Dyno's elements (activity, participant, participant group, transaction, influence) enabled them to express the interplay of relationships within a platform. They also fully agreed that Dyno helped them to locate and create causal loops. They also confirmed that Dyno helps to connect 2 or more causal loops into interconnected loops (both +2). Given the complexity of cross-sided loops this emphasizes the level of ease felt by the respondents when working with Dyno.

Also the respondents were in full agreement that Dyno allowed them to select optimal nodes for placing base values (both +2). When asked whether Dyno allowed them to evaluate strong base value candidates or whether they still need to create suitable base value, the score was slightly lower (average + 1.5). When being asked whether Dyno could help them to judge whether – in view of the base value situation – it makes sense to enter the platform business, both unanimously agreed with a score of +1.

Respondents' Evaluation of the Dynamic Network Notation with Respect to the Allocation of Control Mechanisms (H3)

Hypothesis 3 (H3) states that a language that adds control mechanisms onto its elements, allocating managed self-organization can model options to turn causal loops into network effects. The two respondents produced a superimposable set of answers with an average score of 1.4.

In explicit, the respondents agreed that the language always provided the most suitable control mechanisms and that it helps with running services of improved quality. They strongly agreed that the context-specific suggestion of control mechanisms helps to design a platform, which influences growth of consumer and provider ecosystem and which helps to design a platform that focuses 3rd party services that fit into the corporate strategy. The respondents strongly confirmed that the suggested control mechanisms allowed to completely model the corporate platform.

Respondents' suggestions for improvement or additional features

M-Engineering commented that the notation and tool provide a result which is targeted and which goes further than what was achieved without the tool. The respondent suggests embedding the process of modeling into a consultant-moderated workshop "which is capable to lead through the first steps or better the use of the relevant information by specific question / answer session. Consequently better results through a common workout".

As improvement for the editor, CAS and M-Engineering suggested to include tooltips for the individual elements in the shape repository. All suggested the opening of the modeling canvas with pre-allocated noise, influence and control areas. Those two improvements need to be included in a future release of the editor, as they require modification in the ORYX-framework.

7.3.4 Conclusion

This subssection summarizes the finding revealed through the second case study. The main Hypothesis MH states that a modeling language targeted at harnessing network effects around service platforms can support platform design through guided modeling, in particular by providing structural elements, process elements and the allocation of service management mechanisms.

In result of the iterative research process, language and editor accomplish all aspects of platform design and service management claimed in the hypothesis; i.e. the case study confirmed that:

- The Dynamic Network Notation either improves the starting position for platform design or supports improvements in platform design as compared to the situation without language and tool.
- The Dynamic Network Notation helps solution managers and platform architects understanding and locating causal loops, which can lead to network effects.
- The quality of the designed Dyno models is high with respect to correctness, relevance, robustness, clarity and comparability.

Throughout the case study the editor also received positive acclaim, yet with suggestions for improvement.

The case study further substantiated the three subhypotheses: First, Sub-hypothesis H1 states that a language providing the structural perspective of areas of staged authority and structural divisions enables better structuring and improved exploitation of stakeholding power. The respondents confirmed this claim. Second, Sub-hypothesis H2 claims that a language providing the procedural perspective of interrelated specific and unspecific participants as well as of their interaction in and with the platform allows for modeling improved causal loops and related processes.

H2 received strong endorsement from the respondents. Third, Subhypothesis H3 states that a language, which adds control mechanisms onto its elements, allocating managed self-organization, can model options to turn causal loops into network effects. This subhypothesis finds the respondents' consent, as well. In summary, the case study fully validated all stated hypotheses.

The overall confirmation of the hypotheses also proved the iterative research design's suitability: The last iteration produced enhanced results as compared to the expert workshop with S.Chand Edutech (Section 7.4), where several inconsistencies and weak robustness of the models could be eliminated and as compared to the study at CAS which took place without prior training of the experts.

Yet, the omitted evaluation of conceptual model, analysis environment and pattern language require explanation: Both field studies focused on evaluating the Dynamic Network Notation as well as its editor, which base on the conceptual model elicited in Chapter 4. As Dyno represents reality in a semantically and syntactically correct way, the underlying conceptual model is consequently correct as well and does not require further evaluation. Moreover, the field studies did not explicitly evaluate Dyno's expressiveness with respect to subsequent analysis. In this regard, the prototypic implementation of the analysis environment (Section 6.3) already provided the exemplary proof of concept.

Finally, the thesis complements the Dynamic Network Notation with a pattern language based on building blocks of structured experience. These patterns themselves require continued and iterative optimization, which exceeds the timeframe of the present work. Consequently, the present work speaks of drafts rather than final patterns, indicating that these patterns shall be subject of further optimization throughout the coming years. As highlighted in Chapter 8 the elaboration of patterns remains a path for further research. As recommendation, the suggested pattern repository might be used as means to support this evolution. Its rules of coordinated collaboration themselves may evolve during the iterative optimization process.

D Conclusion

8 Summary and Discussion of Contributions

The emergence of service platforms and their opening to third party service contribution as well as to degrees of self-organization on the consumer side brought up new challenges to existing and future platform operators. A service as understood in the present work stands for any kind of deployed software provided on demand. The environments, federating such services are referred to as service platforms. The increased autonomy of suppliers and consumers and resulting network effects increase the complexity in the management of such platforms. The operators are well aware of the opportunities of such effects, particularly the desired strong growth of service consumption and provision. But they also see the risk of failure due to loosing influence on quality of service on the one hand and the danger of unachieved growth or rapid collapse of the consumer base due to negative network effects.

These opportunities and threats lead to the requirement of theory and design support, enabling platform architects and solution managers to design platforms that are oriented to harnessing network effects. In response to this requirement, the present work states the hypothesis that a modeling language, focused on harnessing network effects around service platforms can support platform design through guided modeling. Related sub-hypotheses emphasize three dimensions of solutions that are required: First, the dimension of structural elements: helping to allocate the different participants within and outside the platform’s ecosystem in function of the degree of authority which the platform operator can exert on them. Second, a procedural dimension: helping to represent the involved participants, as well as their various types of relationships and interactions. Lastly, the control dimension: helping to choose and allocate suitable control mechanisms to ensure quality of service without suffocating self-organization.

The present work introduces and derives a set of artifacts, the Dynamic Network Notation, validating this main hypothesis and its sub-hypotheses. These artifacts utilize, adapt and structure knowledge on causal loops and targeted network effects, originating from system, control and dynamic market theory and applied to service platforms. The main contributions of this thesis embrace the Dynamic Network Notation and its underlying model. They build on the knowledge, gained through a series of surveys and endowed with related theory. In addition, the present work proposes a language for service platform patterns and a coordinated, yet community-driven method to accumulate a pattern repository, both supporting solution managers and platform architects with building blocks of reference solutions retrieved from best practice. Finally, the thesis introduces the DYNO editor, an exemplary instantiation of the language, accessible at www.dynocloud.org.

8.1 Critical Acclaim

The present thesis emphasizes the importance of network effects for platform success and explores their implications on platform design and service management. Explicitly, the present work advances the state-of-the-art by providing the Dynamic Network Notation and the dedicated service platform pattern language. Notably, the representation of network effects around service platforms and the related control mechanisms do not find consideration in related graphical modeling languages Service Network Notation [26, 27], Service Network Modeling Notation (SNMN) and e3controls [29].

Conceptual Model **-** The conceptual model conveys the semantic backbone and the functional design requirements for the Dynamic Network Notation. It consists of three major groups: *structural elements, process elements* and *control mechanisms* for service management: (a) *Structural elements* consist of areas of staged authority, i.e. the *control area*, where the platform operator has full and prescriptive authority, the *influence area*, where he can influence the ecosystem players and the *noise area* for individuals, groups and legal entities, which are not influenced by and not part of the platform's ecosystem (but which can affect the ecosystem). (b) *Process elements* include nodes and edges. Nodes describe the activities on the platform or players active on and around the platform. The players can be positioned in the control area (e.g. departments of the platform operator) in the influence area (e.g. suppliers or customers), or if they are outside the platform's reach, in the noise area (e.g. competitors or neutral communities). They can be individual players, called *participants*, or unspecific groups call *participant groups*. Edges are either *transactions*, which transfer a value from one node to another, or *influences*, which influence participants or participant groups in the influence areas. Influences can be merged through merging *gateways*. Influences, participant groups, transactions and activities can form causal loops of reciprocity, which if successfully implemented in the context of its ecosystem, turn into network effects. (c) C*ontrol mechanisms* allow for service management on nodes and edges within the control area. Control mechanisms allow service management to exert control in diverse specificities. The present work differentiates between enforcing and incentivizing mechanisms. Enforcing mechanisms are *prescriptive control, restrictive control* and *sanctional control*. Incentivizing mechanisms are *market-regulative control, informative control* and *motivational control*. Control mechanisms turn causal loops into control loops, giving the platform operator possibilities to interfere.

Dynamic Network Notation **-** The Dynamic Network Notation (Dyno) adopts its semantics from the constructs of conceptual model. Likewise, the meta-model is based on functional design requirements, derived from the conceptual model. The meta-model formulates Dyno's *abstract* *syntax* through production rules, communicated in Unified Modeling Language (UML), in conjunction with constraints described in Object Constraint Language (UML). The notation explicitly comprises a *morphology* to represent structural and procedural elements, as well as control mechanisms. The Dynamic Network Notation represents, apart from controllability and applied control mechanisms, two important attributes: scalability of technical environments and base values, representing the platform's value proposition and expected to incite network effects. Dyno's meta-model explicitly allows the inclusion of additional attributes in concrete grammar implementations. As the notation addresses *solution managers* and *platform architects* in charge of designing service platforms, representation and grammar are conceived in a way that they address equally modelers with a business as well as a technical background. A Dyno model exhibits the modeler's specific view, called the *protagonist view*. Consequently, a Dyno model will look different when modeled from the perspective of a different company, even when it models the same situation.

Service platform pattern language **-** Platform service management patterns are reusable *building blocks of structured experience*, helping the modeler to revert to best practice solutions. They increase the effectiveness of modeling with regard to modeling the right solutions. Therefore, the present work formulates a *pattern language*. A pattern in this language consists of structured, descriptive text and a paradigmatic Dyno model answering the specific problem. Each pattern can be atomic, or include one or more patterns. The present work suggests several draft patterns, retrieved from various surveys. The thesis further provides a *coordinated community-driven process* to produce patterns for a pattern repository. That process considers and handles all elements of the pattern language as services allowing the handling of human and non-human contributors, similarly. The coordinated community driven process includes a *process model for collaborative pattern composition* as well as a *coordination process for automated service binding* and the *management of dependencies* between individual patterns and subsets. As a result, the integrative collaboration and coordination model allows for quality control through explicit approval processes and defined collaboration processes.

DYNO editor **-** The Dyno grammar and subsequent analysis have been instantiated with the DYNO editor. The DYNO editor embraces two functionalities; model design and model analysis. The model designer implements the notation's concrete syntax and concrete morphology through a stencil set and plugged-in runtime constraint and layout processor within the ORYX framework. The model designer includes a shape repository, accommodating the Dyno structural and procedural elements, a modeling canvas and a property configuration panel, allowing for configuring context specific attributes. Tested in field studies, the model designer verified the hypotheses: In confirmation of sub-hypothesis 1, the provisioning of the structural perspective of areas of staged authority and of structural divisions succeeded in enabling better structuring and improved exploitation of stakeholding power. Validating sub-hypothesis 2, the field studies confirmed that the procedural perspective of interrelated specific and unspecific participants as well as of participants' (inter-) activities on and with the platform allows improved modeling of causal loops and related processes in and around platforms. Lastly, it validates sub-hypothesis 3, confirming that the control mechanisms allocating managed self-organization on the Dyno elements, allows for modeling design options for turning causal loops into network effects. The model analyzer, in turn, allows for analysis of Dyno models, helping to improve their efficiency and effectiveness. The analyzer has a modular design and can accommodate plug-ins which perform specific analyzes. Providing a Dyno model and its graph, the plug-ins allows for a broad range of analysis including graph-theoretical exploration.

The present work develops artifacts for the specific organizational context of service platforms. Intentionally, the thesis does not follow the classic stage-gate approach as suggested by e.g., Hevner, March et al. [33] or Peffers, Tuunanen et al. [45]. It rather pursues an iterative approach of design, discussion or prototypical application and refinement as common in Design Thinking [37]. This approach brings elements of action research into the design scientific methodology and is referred to as Action Design Research [38]. The artifacts underwent a series of refinement cycles, through experimental modeling of real-life scenarios and discussion in research teams, critical discussion at academic conferences, as well as through field studies. Two final case studies freeze the status at the end of the research project and thus provide final validation to the hypotheses and substantiate the applicability of the developed Dynamic Network Notation to modeling service platforms. The case studies confirmed the *expressiveness* and *effectiveness* of the language, able to produce models that are not overloaded and still consistent with the real world as consequence of targeted abstraction. The models are also *robust to change* of parameters. The limits of such an iterative, design scientific approach relate to the fact that it does not produce per se generalizable results, but classes of solutions for specific contexts. The fact that the language is able to represent all major theoretical constructs found in literature back the goal of generality. Further substantiation comes from the fact that all settings of management of network effects by successful service platform, elucidated in an explorative analysis, can be modeled with Dyno.

The produced solutions however respond to problems around service platforms and have only been validated in this context. In what respect the findings and the Dynamic Network Notation can be applied other fields, e.g., product-based businesses, exceeds the scope of this work and requires further research.

Given the novelty of service platforms and related concepts and theory, future evolution of best practice and technology, as well as of related theory is predictable. The present work responds to this with a language meta-model, allowing complementation of attributes, while remaining compliant to the specification. It also answers with a concept for a coordinated but community-driven pattern repository, designed to rapidly embrace emerging best practices. The iterative Action Research Design method allows the provision of an academic frameset for such an evolution.

8.2 Outlook and Future Work

Based on the generated results, the present thesis opens four potential paths for future research:

A first path of further research relates to multi-layer categorization of control mechanisms. All developed mechanisms could be further refined through additional layers. For example, the mechanism of motivational control could be split into reputation systems and recommender systems. This new layer could again be filled with sets of mechanisms. Given that this layer is strongly evolutional and dependent on available technology; mechanisms could be placed together with a set of constraints in a web-based repository. This approach would take account of the short innovation cycles in technology. This complement would however exceed the frame of a well-formed grammar. It would be rather using the pattern repository as a set of building blocks. The current meta-model to the Dynamic Network Notation allows for such extensions through the class *ExtensionAttributes*. The repository of control mechanisms could be complemented through additional layers of categorization. Such categorization, if formulated independent of language engineering and technical implementation, could enrich the developed theory of management of network effects in service platforms.

Second, the Dynamic Network Notation could be further extended. The language in its present expansion state focuses on control mechanisms. Theoretically, attributes with other focuses may be added. As an example, monitoring and security aspects might be of added-value to specific solutions. Remaining on the attribute level, such extensions could be implemented without violating the meta-model. Even depiction of an activated attribute in the Dyno morphology is not ruled out. Logic could be formulated through OCL-based constraints. Such extensions however should avoid overloading the morphology as this would reduce clarity. In the current release, scalability and base value are depicted in a binary way. The attribute is either set true or false. Further research might allow a move from the simple binary perspective, providing an ordinal or interval scale. Having such an enhanced logic would allow the attributes to be more specific. However, such distinction makes the language more specific and reduces the models' robustness over time. Prerequisite to a scale of superior grade for scalability would be a progression in research. Currently, no generally accepted codes of practice exist for an ordinal or interval scale to scalability in the context of infrastructure of service platforms.

The third path of further research is the potential alignment with other notations. The Dynamic Network Notation provides a specific view point of service management of ecosystem interaction around platforms with the focus of harnessing network effects. The Business Process Model and Notation BPMN provide a rather focused level on explicit collaboration. A stack could help to set both notations into context and correspondences between both notations could be defined. An interesting path to follow would be to bridge the perspectives of network effects and service management with discrete processes as formulated in BPMN. A possible solution would be to enhance BPMN conversation through the integration of the Dyno logic. The advantage of such integration would be the combination of Dyno's advantages, such as allocated control mechanisms, and BPMN's advantages, such as defined translation to executable BPEL code.

Lastly, the concrete implementation can be further developed. Dyno has a set of implementations, conceived in a prototypic way, serving as proof of concept. Examples are the integration of Dyno's pattern language into MoSaiC. An enhanced integration of both solutions would allow for enhanced modeling of service platforms. Also, the current status of the analyzer invites further research. The currently provided analyses again just serve as proofs of concept. Implementing Graph-theoretical analytics would allow for extensive analyses. However, to do so, suitable theory needs to be expanded or adapted. Analyzer and pattern repository together could evolve towards an expert system, providing suggestions for improvement and design alternatives, integrating both, theory-based reflections and building blocks of structured experience.

E Appendix

Questionnaire

The following table lists the questions, asked in the questionnaire and the answers, given by the respondents CAS Software AG, SAP AG and M-Engineering U.G. The current questionnaire applies a non-dichotomous approach, giving more than one choice. Using an odd number of alternatives, the results support the revelation of tendencies. The ordinal scale documents the respondent's ranked opinion with

- Strongly agree (+2)
- Agree (+1)
- Disagree (-1)
- Strongly disagree (-2)

The first colomn refers either to the main hypothesis (MH) or to the subhypotheses H1-3. The letter behind the abbreviation MH refers to the enumeration a)-d) in Subsection 7.3.3 with respect to the topic *Evaluating the Main Hypothesis (MH).*

	No		**CAS**	**SAP**	**MEng**
Hypotheses		**Please position your experience with the editor**			
MHd	1	I found the features in the editor purposeful (zielführend)to model Platforms Ecosystems and the respective network effects with the editor	1	2	2
MHd	2	The Graphical User Interface was Intuitive, and easy to follow	1	-1	1
MHd	3	The Graphical User Interface was sufficiently comfortable to work with	1	1	1

	No		CAS	SAP	MEng
	4	The tool did not restrict me in my modeling efforts	1	1	2
		When designing, I benefitted from editor's guidance based on DYNO's grammar and its resulting context related suggestions & restrictions			
		with respect to			
MHd	5	… transactions and influences	-1	2	2
MHd	6	… controllable / uncontrollable activities and participants	1	2	2
MHd	7	… case-specific control mechanisms	1	2	2
MHd	8	… areas (control area, influence area, noise area)	-1	2	2
MHd	9	… getting a plausibility check of my modeling intentions	1	2	2
		Please position your experience with DYNO			
MHa	10	It was easy to understand the DYNO in modeling dynamic networks around platforms.	1	2	2
MHa	11	DYNO gives me a better understanding of network effects and control possibilities in platform ecosystems	1	2	2
		Dynamic Loops:			
MHb	12	DYNO helps to locate or create dynamic loops,	1	2	2
MHb	13	DYNO helps to upgrade one-sided loops to 2- or multi-sided loops, meaning to create interconnected loops to structures of 2 or more loops which directly impact on each other	-1	2	2
		Control Mechanisms:			
		The context-specific suggestion of control mechanisms			
H3	14	… always provides me with the the most suitable control mechanisms;	-1	1	1
H3	15	… is unnecessarily limiting my scope of decision making;	-1	1	1
H3	16	… is difficult to grasp;	-1	1	1
H3	17	… helps to design a platform running services of improved quality (your personal perception of the suggested solution);	1	1	1
H3	18	… helps to design a platform that influences growth of consumer and provider ecosystem;	1	2	2
H3	19	… helps to design a platform that focuses 3rd party services that fit into the corporate strategy;	1	2	2
H3	20	… allowed me to completely model my corporate platform (potentially describe in which contexts it failed to do).	1	2	2

	No		CAS	SAP	MEng
		Base Value:			
		DYNO allowed me			
H2	21	… to select the optimal nodes for placing base values;	0	2	2
H2	22	… to evaluate whether our company already possess strong base value candidates	0	2	1
H2	23	… to evaluate whether we still need to create suitable base values;	0	2	1
H2	24	… to see whether – in view of our base value situation – it makes sense to enter the platform business.	0	1	1
		Base Value:			
H2	25	I was satisfied with the ordinal metrics for 'Base Value' suggested by DYNO.	0		
		Scalability:			
		DYNO allowed me			
H1	26	to think about Scalability requirement per Division	0	0	2
		Please position your perception on the achievable quality of the models designed with DYNO and the editor			
MH c	27	**Correctness:** I was able to produce one or more models where structure and behavior of the models are consistent with the real world.	2	1	1
		Relevance:			
MH c	28	All relevant elements from the real-world scenario find application in the model-representation	1	2	2
MH c	29	All elements and relationships, applied in and required in DYNO are relevant to accomplish its design goal.	1	1	1
		Economic Efficiency:			
MH c	30	The model is *robust*, meaning it remains relevant over time, even when quantities of users, services etc. evolve.	1	1	1
MH c	31	The model is *adaptable*, meaning it can be adapted without big effort to a modified constellation (e.g. to implement strategic change or to respond on environmental change).	0	0	1
MH c		**Clarity:** The produced model was clear, understandable and not overloaded	1	1	2
MH c		**Comparability:** When modeling design alternatives on the same business case, I can compare them as they follow the same grammar.	0	2	1
MH c		**Systematic Design** The system-description from the Analyzer (Analysis View) the Model are consistent.	0	1	2

	No		**CAS**	**SAP**	**MEng**
		General perception			
MH a	35	The notation and the tool helped me to produce a better solution than without the tool	-1	2	2
Mh a	36	The design-time was shorter when using the editor as compared to a design without the editor	1	2	2
MH a	37	I was quickly able to understand both Notation and Tool	-1	1	2
		Questions on experience with DYNO, added in the second part of setting II (with SAP and M-Engineering)			
		Areas:			
		The concept of control area, influence area and noise area			
H1	37	allowed me to better position my modeling elements		1	1
H1	38	supported my understanding on the authority for service management I have, depending on my design decision		2	2
H1	39	The concept of divisions helped me to better structure my model		1	2
H1	40	The concept of division groups helped me to model repeated areas of similar behavior.		2	2
		Elements:			
		The following elements helped me to express the interplay of relationships within a platform			
H2	41	Activity		2	2
H2	42	Participant		2	2
H2	43	Participant Group		2	2
H2	44	Transaction		2	2
H2	45	Influence		2	2
		Total Score	**13**	**68**	**75**
		Number of answered questions	**28**	**43**	**45**
		Weighted average score per question	**0,46**	**1,58**	**1,67**
		average score SAP /M-Eng		**1,62**	

List of Figures

List of Tables

Glossary

Abstract grammar	Abstract grammar gives a high-level description of a →grammar, using →abstract syntax and →abstract morphology, but leaving out the specific technical implementation.
Abstract morphology	Abstract →morphology prescribes the representation of graphical elements in an →abstract grammar, leaving out the specific technical implementation.
Abstract syntax	Abstract syntax gives a high-level description of →syntax, leaving out the specific technical implementation.
Activities	Activities are value creating activities on and in interaction with the →service platform.
Areas of stages authority	See → control area; → influence area; → noise area.
Authority	See → stakeholding power.
Auxiliary variable	Auxiliary variables regulate →flows. They can be cascaded. Their origins may be →exogenous or functions of →stocks.
Base value	The base value of a service platform is the value proposition, offered by a →platform operator to →ecosystem participants to incite →network effects.
Solution manager	The solution manager brackets those job profiles, which identify and communicate business requirements, opportunities and goals for a →service platform.
Causal loops	Causal loops are → feedback loops, where the magnitude of a →stock amplifies the →flow by a certain factor, which in reciprocity increases the stock again.
Controlled system	A controlled system includes a →control loop.
Control loop	Control loops are →causal loops which carry →control mechanisms on →transactions, →activities and →influences to manipulate their progression.
Control theory	Control theory is the study on →controlled systems.
Control mechanism	Control mechanisms are regulators within a →controlled system which have the goal to minimize the offset between an actual system output and a desired reference value.

Control loop	Control loops are →causal loops which carry →control mechanism on →transactions, →activities and →influences to manipulate their progression.
Close to static	A →stock is considered close to static when it hardly accumulates or depletes within a period of time.
Concrete grammar	Concrete grammar in →graphical languages is a set of rules consisting of →concrete syntax and →concrete morphology.
Concrete morphology	Concrete morphology in →graphical languages prescribes the representation of graphical elements specific to the technical implementation.
Concrete syntax	Concrete syntax in →graphical languages is a set of production rules specific to the technical implementation.
Context-free grammar	In Context-free →grammars (Type 2) all production rules produce a single non-terminal on the left hand side. Context free grammars can be applied when describing recursive language structures (→nesting).
Context-sensitive grammar	In Context-sensitive →grammars (Type 1), the number of elements in the string on the left-hand side must be smaller or equal to the number of elements on the right side.
Control	Control describes →service management actions by the →platform operator in order to change a set of parameters from a current status (actual value) to a target status (setpoint). Control in a →service platform context operates as a →feedback loop, meaning with monitoring feedback in the context of a regulatory process. The mechanisms which are used to control such a process are called →control mechanisms.
Control area	Control area is the area where the →platform operator can exert control over →activities and internal →participants. It is also the area from where he influences →ecosystem participants that are placed outside the →control area.
Control theory	Control theory is the study of influencing →closed-loop systems through regulators.
Critical mass	The critical mass describes the threshold required for a →base value to incite a →network effect. The critical mass of a →stock is the magnitude that is equal to the threshold required to exhibit exponential behavior of → network attractiveness.
Division	A division is a structural entity of homogeneous conditions within the →control area.

Division group	A division group is a finite set of divisions within the →control area.
Dynamic system	See →system dynamics.
Dynamics	See →system dynamics.
Ecosystem participant	Ecosystem participants are →participants and →participant groups in the →platforms ecosystem.
Effectiveness	The effectiveness of a →language describes how well the language can express information with respect to a specific target group.
Efficiency	The efficiency of an →utterance describes how well it is modeled.
Expressiveness	The expressiveness of a →graphical language describes its ability to express desired information in a semantically (→semantics) and syntactically (→syntax) correct way.
Exogenous	See → exogenous variable.
Exogenous variable	A variable where the origin is not modeled within its utilizing model.
Feedback	Feedback is the targeted provision of information on platform-based →activities to a →participant, active in and in reciprocal relationship with the →service platform.
Feedback control	→ Feedback controlled system.
Feedback controlled system	Feedback controlled systems are systems, being regulated by control devices (see →control theory), aligning the reference value with the fed back system output. The special version of feedback controlled systems within this thesis is referred to as →controlled system.
Feedback loop	See → feedback.
Flow	Flows circumscribe the inflow into or outflow from a →stock.
Generative grammar	Generative →grammars build on a limited kernel of simple utterances, complemented with a set of grammatical transformation rules, transferring one correct utterance or fraction of an utterance to a new correct one. The hierarchy of generative grammars, consisting of a series of 4 groups of types with increasing expressive power: →recursively enumerable grammars, →context-sensitive grammars, →context-free grammars and →regular grammars.

Grammar	A grammar is constituted through a finite set of production rules, describing how the elements of the languages alphabet are concatenated. It consists of three segments: →morphology, →syntax and →phonology.
Graphical grammar	Graphical →grammars are a special form of →generative grammars which describe the elements of a language and their relationships graphically.
Graphical language	Graphical →languages are a special form of artificial languages which describe their elements and relationships graphically.
Graphical meta-language	Graphical meta-languages are special versions of →meta-languages expressed as →graphical language.
Graphically engineered grammar	A →language is graphically engineered with one or more →graphical grammars, optionally complemented by textual grammars. One or more →graphical meta-languages complemented with optional textual meta-languages document the →grammar.
Influence	Influences are means to stimulate the rate of value flows at their →sources.
Influence area	The influence area is the area where →participants are located, which are in or may come into value exchanging relationship with the →service platform. →Ecosystem participants within this area may be influenced by the →platform operator, but also by other players within the ecosystem or outside.
Informative control	In the category of →control mechanisms called informative control the →platform operator preprocesses information and addresses it to existing or potential →participants or →participant groups. The analyses are customized on the addressed participants or participant groups and have the goal to incite a self-regulatory process among them.
Internal participant	An internal participant is a →participant, located within the →control area.
Language	A language is constituted of a (finite or infinite) set of →utterances of finite length, constructed from a finite alphabet of symbols through a →grammar.
Linguistic units	Linguistic units are building blocks of an →utterance.
Linguistic utterance	See →utterance.

Managed self-organization	Managed self-organization describes the trade-off between the level of control exerted over →service quality and the degree of →self-organization.
Market regulative control	Market regulative control is a category of →control mechanisms, driven by →participants. It gives explicit →feedback to consumers or service providers in the →service platform and / or in the ecosystem on value, offered in →activities or through →participants. This causal loop incites a self-regulatory process.
Meta-language	The term meta-language defines languages conceived to describe other natural or artificial languages with the help of specific terminology and symbols.
Morphology	Morphology prescribes the representation of linguistic units in a language (e.g., of words).
Motivational control	Motivational control is a category of control mechanisms that aim at steering →ecosystem participants towards the accomplishment of specific outcomes through rewards.
Natural language	The term natural language describes →languages, which have not been artificially engineered but which emerged over periods of time within communities as means of communication.
Nesting	Nesting describes recursive →language structures.
Network attractiveness	Network attractiveness is defined through an exponential function of the product of a network participant’s sensitivity to a →stock and the magnitude of that stock relative to a threshold. The threshold delimits the magnitude where the impact starts.
Network effects	Network effects describe the reciprocal relation between the value of a →service platform and the quantity of involved service consumers and service providers. Network effects are driven by →self-organization of the →platform ecosystem. A network effect takes place within a causal loop, when →network attractiveness grows exponentially.
Noise area	Noise area is the subsection outside the platform ecosystem, where the →platform operator has no stakeholding power.
Non-linearity	Non-linearity is the effect within →causal loops, where manipulations are not linearly reversible, due to the accumulative behavior of →stocks.
Notation	A notation is a formalized →language, conceived to better describe complex relationships than a →natural language. They are used e.g., in music, mathematics or information technology.

Participant	Participants are individuals or entities with small close to static capacity within the →control, →influence or →noise area.
Participant group	Participant groups are groups of individuals or of entities of finite large size within →influence or →noise area.
Phonology	Phonology defines the organization of sounds within →utterances.
Platform architect	The platform architect brackets those job profiles in charge of the overall technical →service platform design.
Platform ecosystem	Platform ecosystems are sets of autonomous participants around service platforms, which are in reciprocal relationship with the latter.
Prescriptive control	Prescriptive Control is an enforcing category of control mechanisms, describing the sequence of observing and steering a →participant's set of actions within →activities as well as of →internal participants. For actions in activities, it may further include subsequent corrective measures on their results through the →platform operator.
Platform operator	The platform operator is the legal entity, managing a →service platform and legally responsible for all course of action.
Protagonist	A protagonist is the entity, whose view point is reflected in a model.
Quality of service	Quality of service describes the match of service delivery with functional and non-functional consumer expectations. Quality of service includes the quality groups Business Value Quality, Service Level Measurement Quality, Business Process Quality, Suitability Quality, Security Quality and Manageability Quality.
Recursively enumerable grammars	Recursively enumerable →grammars (Type 0) have no restriction at all in the production rules.
Regular grammar	Regular grammars (Type 3): Production which do not allow recursive structures (→nesting) These grammars can be applied on natural languages.
Restrictive Control	Restrictive Control is a category of →control mechanism on →transactions to filter value flows. The mechanisms are placed within the →control area and verify compliance with →service platform provisions.
Sanctional control	Sanctional control is a category of →control mechanisms. It describes the enforcing action of the →platform operator on policy breaches in →activities through an escalation routine, including discovery processes, scope and time of reaction for the →participant and range of enforcements through the →platform operator.

Scalability	Scalability describes to ability to meet increased workload, through a planned incremental increase of capacity.
Self-organization	Self-organization in the context of service platforms describes the line-up of →platform ecosystem participants over time to a temporary situation of equilibrium, attained through →feedback.
Semantics	Semantics assigns a meaning to elements of an →utterance (symbols, words) and to the utterance the as a whole.
Service	A service stands for any kind of deployed software, provided by →service platforms on demand.
Service management	Service management in →service platforms describes the activity of managing the whole service life-cycle, service design, service transition, service operation and continuous service improvement with the objective to make capabilities and resources available that are required by the consumer.
Service platform	A service platform offers own or third party software as metered on demand services. Software can be partially or completely deployed outside the service platform. The traffic, when a service is consumed passes through the service platform.
Service platform pattern	Platform service management patterns are structured descriptions of best practices for harnessing →network effects in →platform ecosystems.
Service platform policy	Service platform provisions are documents, defining restrictions to and rules of participation and cooperation for →participants in →activities in service platforms.
Service platform value	Service platform value denotes those attributes, arising from the whole platform or from defined subsets, which have positive effect on the performance of actions, objects and tasks.
Service value	Service value describes those service attributes with positive effect on performance of actions, objects and tasks.
Sink	Sinks are negative →sources.
Source	Sources are →stocks outside the boundary of the model with assumed infinite capacity.
Stakeholding power	Stakeholding power describes the degree of authority of a →platform operator over an ecosystem participant or activity.

Stocks	Stocks describe a vessel which can accumulate or deplete. The term vessel is used in the figurative sense.
Strong network effect	A strong →network effect takes place when at least one →causal loop fulfills two necessary conditions: accumulation in a →stock exceeding the delimiting threshold and a sufficiently high →source allowing for exponential growth of network attractiveness.
Syntax	Syntax defines the assembly rules of utterances through →linguistic units. Further, syntax determines adaptation of specific representation of linguistic units.
System dynamics	System dynamics describe macroscopic system behavior over time, built through interaction of →sources and sinks, →stocks, →flows and →feedback loops. System dynamic theory is part of →system theory.
System theory	System theory is the study of systems, self-regulating through →feedback.
Textual grammar	Textual →grammars build languages through linear catenation of text.
Threshold	See → network attractiveness.
Transformational grammar	See → generative grammar.
Transaction	Transactions describe any kind of transfer into, from or within the platform, which eventually could eventually create value to the platform.
Utterance	(Linguistic) utterances are elements of a →language (e.g. sentences, mathematical statements or graphical models).
Well-formed	See →well-formedness.
Well-formedness	An →utterance that conforms to the production rules of a →grammar is called well-formed.

List and Abstracts of Related Publications by the Author

Ulrich Scholten, Nelly Schuster, Stefan Tai (2012):

A Pattern Language and Repository for Service Network Management, Proceedings of the IEEE International Conference on Service Oriented Computing & Applications SOCA 2011, Taipeh.

Abstract:

Successful service platform operators foster their market performance by leveraging economic network effects, which implicitly control service ecosystems. Explicitly, third party services are used to complement the platform's intrinsic value to the users. Platform operators' key to success is the initiation of a snowballing interplay of consumers' preferences and a respective portfolio of service offerings. The Dynamic Network Notation DYNO supports modeling such service networks from a service management perspective, while defining system-interaction control and exploiting network effects. However, there is a need for an evolving and reusable base of experience that allows researchers and platform operators to learn from and to share knowledge on best practices. To this end, we introduce service network management patterns based on DYNO. In addition, to exploit the dispersed applications through various market segments, we present a community-driven pattern repository. The repository applies coordination and review means to ensure quality of patterns without restricting creativity during the pattern design process.

Ulrich Scholten, Robin Fischer, Christian Zirpins (2012):

The Dynamic Network Notation: Harnessing Network Effects in PaaS-Ecosystems, Proceedings of the Fourth Annual Workshop on Simplifying Complex Networks for Practitioners co-located to the www 2012, Lyon.

Abstract:

Web applications complement the Platform-as-a-Service (PaaS) value by satisfying widespread and rapidly changing consumer requirements within limited time and budget. Successful PaaS providers excel in governing their market performance by leveraging complex network effects, which implicitly control PaaS-ecosystems. There is currently no methodically sound and easy to use tool available to solution managers and software engineers of PaaS-offerings that address challenges and opportunities in launching and governing such highly dynamic networks. In this paper, we capture network behavior through elements of complex system and control theory. Our dynamic network notation (DYNO) builds upon these theories. In more detail, DYNO mod-

els PaaS offerings with a focus on identifying and shaping network effects towards a sufficient user-base and an optimized portfolio of Web applications, all while maintaining a high quality of service.

Nelly Schuster, Christian Zirpins, Ulrich Scholten (2011):

How to Balance Flexibility and Coordination? Service-oriented Model and Architecture for Document-based Collaboration on the Web, Proceedings of the IEEE International Conference on Service Oriented Computing & Applications SOCA 2011, Irvine.

Abstract:

Frequently, distributed work groups require the documenting of joint activities or collaborative authoring of documents comprising diverse resources, originating both from humans and the Web or enterprise systems. These creative collaborations involve ad hoc human interactions and unexpected changes which build an implicit situational process. Supporting such collaboration requires a balance between retaining flexibility of collaboration with space for individual creativity and the ability to coordinate the collaborative evolution of documents. Existing coordination approaches support (partial) automation of conventional business processes but are too inflexible for creative collaboration. Web-based collaboration tools do not explicitly drive coordination. In this paper we provide a collaboration model for the coordinated creation of documents as an evolving composition of services embedded into a RESTful scalable architecture. Through a prototype we substantiate how the approach leads to intended output documents.

Simone Scholten, Ulrich Scholten (2011):

Platform-based Innovation Management: Directing External Innovation Efforts in Platform Ecosystems, Journal of the Knowledge Economy, Springer, New York.

Abstract:

Modular platforms have become the centerpiece of collaborative value creation in platform ecosystems. Platform ecosystems co-create the platform's value proposition and support its market adoption as the more complementors join the ecosystem to supply complementarities, the more valuable the platform becomes to consumers due to a greater variety of choice. This poses new requirements on managing innovation in open platform environments. While academic research stresses the relevance of external complementary innovation for platform success, it lacks, however, a concrete understanding to guide platform owners in directing external innovational ef-

forts in coopetive platform ecosystems to co-create and deliver value, while ensuring the overall quality, reliability and consistency of the "whole" solution. Based on the challenges platform ecosystems place on innovation management, this paper explores and categorizes control mechanisms leading platform owners in the ICT industry have implemented to steer external complementary innovation efforts. From that an overall platform-based innovation management process is developed.

Ulrich Scholten, Robin Fischer, Christian Zirpins, Simone Scholten (2011):

DYNO: A Notation to Leverage Dynamic Network Effects in PaaS Ecosystems, Proceedings of the IEEE International Conference on Service Oriented Computing & Applications SOCA 2011, Irvine.

Abstract:

Platform-as-a-Services offerings continuously gain importance as two-sided markets, offering Software-as-aService (SaaS) to the respective customers. Market success is achieved by platforms which excel in shaping ecosystems of users and autonomous SaaS suppliers around their basic value proposition - and in controlling quality of service in function of customer requirements. In this paper, we suggest a notation for dynamic networks "DYNO", designed to help platform providers in creating PaaS, optimized on their specific requirements. In addition, DYNO gives support in properly allocating control mechanisms to guarantee high quality of service. In a use-case we describe how solution managers and service engineers may use DYNO models to conceive PaaS ecosystems.

Normann May, Ulrich Scholten, Robin Fischer (2011):

Towards an Automated Gap Analysis for e-Service Portfolios, Proceedings of the 8th International Conference on Services Computing SCC 2011, Washington.

Abstract:

Intermediaries for e-services continuously gain momentum, powered by a materializing Internet of Services. However, quality of service still exhibits considerable shortcomings, as no structured process to enhance consumer satisfaction is available yet. To improve the match of delivered e-service quality and expected service quality on the consumer side, we develop a portfolio optimization process that integrates both, the consumer’s as well as the intermediary’s perspective. First, we introduce a toolkit for an e-service-oriented gap analysis. Thereupon, we identify monitoring points to measure service quality gaps automatically. A subsequent aggregation of measured data into customized feedback information allows for applying the toolkit to continuously optimize e-service portfolios. Instantiated in the AGORA e-service market, we conclude with a report on our recent implementation results.

Ulrich Scholten, Robin Fischer, Dimitir Bojkov, Normann May (2011):

Supply Chain Control building on Emergent Self-Organizing Effects, Proceedings of the 4th Academic Symposium on Supply Management, Würzburg.

Abstract:

This paper sheds light on control mechanisms to improve and automate service quality respectively service portfolio management in platform ecosystems. Its focus is placed on e-service value networks as found in platforms such as the Apple App Store, Facebook, Salesforce or SAP ByD. The paper differentiates between direct and indirect control mechanisms and explains how they can be embedded within feedback controlled systems. Informative and Motivational Control mechanisms act on the macro level, indirectly influencing a whole system or subsystem towards a specific target. Sanctional and Restrictive Control in conjunction with Co-Regulative Control act on the micro level and directly influence specific services. Market Regulative Control indirectly influences specific services. The paper suggests and formalizes possible sequences to implement control mechanisms, allowing for optimizing and building on the platform ecosystem's emergent characteristics.

Ulrich Scholten (2010):

Service Level Management in Platform Ecosystems, Proceedings of the INFORMATIK 2010 conference on Service Science, Leipzig.

Abstract:

With growing importance of e-service platforms, enhanced Service Level Management (SLM) concepts are required, paying respect to the service providers' autonomy as important source for value creation within a platform ecosystem. This paper proposes a highly automated SLM concept, wherein traditional direct control mechanisms are complemented by indirect mechanisms, including reputation systems, selected motivational measures and information-based guidance of each individual service-provider. The concept makes use of the ecosystems' inherent emergent and self-organizing processes and is grounded on system and control theory.

Simone Scholten, Ulrich Scholten (2010):

Platform-based Innovation Management: Directing External Innovational Efforts in Self-Organizing Platform Ecosystems, Proceedings of the PICMET 2010 Conference on Technology Management for Global Economic Growth, Bangkok.

Abstract:

During recent years, modular platforms have become the centerpiece of collaborative value creation in customer-driven platform ecosystems. Platform ecosystems co-create the platform’s value proposition and support its market adoption as the more complementors join the ecosystem to supply complementarities, the more valuable the platform becomes to customers due to a greater variety of choice. This poses new requirements on managing innovation in open platform environments. While academic research stresses the relevance of complementary innovation for platform success, it lacks, however, a concrete understanding of how platform operators can direct external innovational efforts in complex self-organizing ecosystems to co-create and deliver value while ensuring the overall quality, reliability, and consistency of the ‘whole’ product. Based on case study results, this paper presents a categorization of control mechanisms currently applied in platform markets, enabling the platform operator to steer external complementary innovation within the context of a platform strategy. From that an overall innovation management process is developed.

Robin Fischer, Ulrich Scholten, Simone Scholten (2010):

A reference architecture for feedback-based control of service ecosystems,

Proceeding of the 4th IEEE International Conference on Digital Ecosystems and Technologies DEST 2010, DUBAI.

Abstract:

With the emergence of digital business ecosystems, new control mechanisms are required to sustainably ensure responsiveness on dynamically evolving consumer demand, as well as goal congruence with the platform providers' strategic goals. Based on system theoretical reflections, we propose a 3-layer reference architecture that collects data on service interactions, aggregates monitoring and feedback information and thus provides data basis for the said control mechanisms.

Simone Scholten, Ulrich Scholten, Robin Fischer (2010):

Composite Solutions for Consumer-Driven Supply Chains: How to Control the Service-enabling Ecosystem? Proceedings of the 3rd Academic Symposium on Supply Management, Würzburg.

Abstract:

In this paper, the shift from classical supply chains to more dynamic value net designs based upon modular product and service architectures is revised. The authors show that to enable consumer-driven supply chains, platform operators have to orchestrate distributed value creation efforts and ensure continuous supply, coherence and quality. However, if the platform performance deviates from the expected output, how should the platform operator react to get the system (back) on target? What are the strategic and operational means of acting on the control path to control outputs to-wards desired values, thus ensuring a desired level of performance? In response to this void, the authors develop a control process for service-enabling ecosystems, which allows to systematically assigning control mechanisms to different value creation phases prior to, during and after service supply, categorized into six categories. Feedback loops play a central role: the provision of extended consumer information stimulates overall platform performance by empowering autonomous service enablers to optimize their service portfolio according to the most recent consumer needs and, therefore, to increase the customer perceived value of the overall platform solution.

Robin Fischer, Ulrich Scholten, Simone Scholten, Stefan Tai (2009):

Information-based Control of Service-enabling Ecosystems, Proceedings of the Second International Workshop on Enabling Service Business Ecosystems, Athens.

Abstract:

Continuous optimization of value co-creation in networks of consumers and autonomous service enablers describes a major challenge to mediating platform operators. In analogy to systems theory, we propose to introduce customizable feedback loops from the service-enabling ecosystem to the service enablers via the platform operator. Relevant feedback information can be derived from analysis of network structure, service interactions, and service consumer preferences. In using our method, optimization of individual service offerings and of the network as a whole is facilitated through the platform operator, while retaining the autonomy of each service provider in the network.

Ulrich Scholten, Robin Fischer, Christian Zirpins (2009):

Perspectives for Web Service Intermediaries: How Influence on Quality Makes the Difference, Proceedings of the 10th International Conference on Electronic Commerce and Web Technologies (EC-Web 09) Linz, Austria.

Abstract:

In the service-oriented computing paradigm and the Web service architecture, the broker role is a key facilitator to leverage technical capabilities of loose coupling to achieve organizational capabilities of dynamic customerprovider- relationships. In practice, this role has quickly evolved into a variety of intermediary concepts that refine and extend the basic functionality of service brokerage with respect to various forms of added value like platform or market mechanisms. While this has initially led to a rich variety of Web service intermediaries, many of these are now going through a phase of stagnation or even decline in customer acceptance. In this paper we present a comparative study on insufficient service quality that is arguably one of the key reasons for this phenomenon. In search of a differentiation with respect to quality monitoring and management patterns, we categorize intermediaries into Infomediaries, e-Hubs, e-Markets and Integrators. A mapping of quality factors and control mechanisms to these categories depicts their respective strengths and weaknesses. The results show that Integrators have the highest overall performance, followed by e-Markets, e-Hubs and lastly Infomediaries. A comparative market survey confirms the conceptual findings.

References

[1] P. Mell and T. Grance. The NIST Definition of Cloud Computing. 2009, http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc, retrieved 2013-02-08.

[2] B. P. Rimal, E. Choi, and I. Lumb: A taxonomy and survey of cloud computing systems. In INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on, pages 44-51. IEEE, 2009.

[3] A. Lenk, M. Klems, J. Nimis, S. Tai, and T. Sandholm. What's inside the Cloud? An architectural map of the Cloud landscape. Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, pages. 23-31, IEEE Computer Society, 2009.

[4] A. Holt, K. Weiss, K. Huberty, E. Gelblum, S. Flanner, S. Devgan, A. Malik, N. Rozof, A. Wood, P. Standaert, F. Meunier, J. Lu, G. Chen, B. Lu, K. Han, V. Khare, and M. Miyachi. Cloud Computing Takes Off - Market Set to Boom as Migration Accelerates. Morgan Stanley Blue Paper Morgan Stanley, 2011.

[5] H. Chesbrough. Open Innovation: Where Weve Been and Where Were Going. Research-Technology Management, 55(4):20-27, 2012.

[6] S. M. Lee, T. Kim, Y. Noh, and B. Lee. Success factors of platform leadership in web 2.0 service business. Service Business, 4(2):89-103, 2010.

[7] L. Cherbakov, G. Galambos, R. Harishankar, S. Kalyana, and G. Rackham. Impact of service orientation at the business level. IBM Systems Journal, 44(4):653-668, 2005.

[8] J. D. Sterman. Business Dynamics: Systems Thinking and Modeling for a Complex World. McGraw Hill Higher Education, Boston, MA, 2000.

[9] T. De Wolf and T. Holvoet. Emergence versus self-organisation: Different concepts but promising when combined. Engineering self-organising systems:77-91, 2005.

[10] S. Keswani, A. Krans, E. H. Henlin, J. Mirandi, M. Casey, and K. Gagnon. Market Landscape – Public Cloud 4/2011. CLOUD BUSINESS QUARTERLY, 2011.

[11] R. Hirschheim, H. K. Klein, and K. Lyytinen. Information Systems Development and Data Modeling: Conceptual and Philosophical Foundations. Cambridge University Press, Cambridge, 1995.

[12] J. Becker and D. Pfeiffer. Konzeptionelle Modellierung: Ein wissenschaftstheoretischer Forschungsleitfaden. In F. Lehner and S. Zelewski (eds.) Wissenschaftstheoretische Fundierung und wissenschaftliche Orientierung der Wirtschaftsinformatik, pages 1-17. Gito, Berlin, 2007.

[13] S. K. Langer. Feeling and Form: a theory of art developedfrom philosophy in a new key, London: Routledge & Kegan Paul, 1953.

[14] B. Shneiderman: The eyes have it: A task by data type taxonomy for information visualizations. In Visual Languages, 1996. Proceedings., IEEE Symposium on, pages 336-343. IEEE, 1996.

[15] J. Mackinlay. Automating the design of graphical presentations of relational information. ACM Transactions on Graphics (TOG), 5(2):110-141, 1986.

[16] C. Alexander, S. Ishikawa, and M. Silverstein. A Pattern Language: Towns, Buildings, Constructions. Oxford University Press, New York, NY, 1977.

[17] E. Gamma, R. Helm, R. Johnson, and J. Vlissides: Design Patterns: Abstraction and Resuse of Object-oriented Designs. In ECOOP '93, pages 406-431. Springer, Heidelberg, 1993.

[18] E. Turban. Decision support and expert systems: management support systems. Prentice Hall PTR, 1990.

[19] T. Eisenmann, G. Parker, and M. Van Alstyne. Platform envelopment. Strategic Management Journal, 32(12):1270-1285, 2011.

[20] G. Parker and M. V. Alstyne. Innovation, openness and platform control. Proceedings of the 11th ACM conference on Electronic commerce, pages. 95-96, Cambridge, Massachusetts, USA, ACM, 2010.

[21] A. Parker and T. Pohlmann. The Emerging IT Ecosystem: The Line between Technology and Service Will Blur at a Faster Pace. Forrester Research, Cambridge, MA, 2007.

[22] T. Eisenmann, G. Parker, and M. W. Van Alstyne. Strategies for two-sided markets. Harvard business review, 84(10):92, 2006.

[23] J. Cardoso: Modeling Service Relationships for Service Networks. In 3rd International Conference on Exploring Services Sciences. Porto, Portugal: LNBIP, 2013.

[24] J. Cardoso, C. Pedrinaci, and P. Leenheer: Open Semantic Service Networks: Modeling and Analysis. In 3rd International Conference on Exploring Services Sciences. Porto, Portugal: LNBIP, 2013.

[25] J. Cardoso, C. Pedrinaci, T. Leidig, P. Rupino, and P. De Leenheer: Open semantic service networks. In International Symposium on Services Science (ISSS'12), Leipzig, Germany, 2012.

[26] M. Bitsaki, O. Danylevych, W. J. Van Den Heuvel, G. Koutras, F. Leymann, M. Mancioppi, C. Nikolaou, and M. Papazoglou: Model transformations to leverage service networks. In Service-Oriented Computing–ICSOC 2008 Workshops, pages 103-117. Springer, 2009.

[27] M. Bitsaki, O. Danylevych, W. van den Heuvel, G. Koutras, F. Leymann, M. Mancioppi, C. Nikolaou, and M. Papazoglou: An Architecture for Managing the Lifecycle of Business Goals for Partners in a Service Network. In 1st European Conference ServiceWave. Lecture Notes in Computer Science vol. 5377, pages 196-207, Madrid, 2008.

[28] O. Danylevych, D. Karastoyanova, and F. Leymann. Service Networks Modelling: An SOA & BPM Standpoint, Springer, 2010.

[29] V. Kartseva, J. Hulstijn, J. Gordijn, and Y.-H. Tan. Control patterns in a health-care network. European Journal of Information Systems, 19(3):320-343, 2010.

[30] H. Akkermans, Z. Baida, J. Gordijn, N. Peña, A. Altuna, and I. Laresgoiti. Using Ontologies to Bundle Real-World Services. IEEE Intelligent Systems, 19(4):57-66, 2004.

[31] Z. Baida, J. Gordijn, and H. Akkermans. Service Ontology, 2001.

[32] S. de Kinderen and J. Gordijn: e^{3}Service: A Model-based Approach for Generating Needs-driven E-service Bundles in a Networked Enterprise. In 16th European Conference on Information Systems (ECIS), pages 1-12, Galway, 2008.

[33] A. R. Hevner, S. T. March, J. Park, and S. Ram. Design Science in Information Systems Research. MIS Quarterly, 28(1):75-105, 2004.

[34] T. Wilde and T. Hess. Methodenspektrum der Wirtschaftsinformatik: Überblick und Portfoliobildung. Arbeitspapiere des Instituts für Wirtschaftsinformatik und Neue Medien, LMU München, 2, 2006.

[35] A. Osterwalder. The Business Model Ontology: a proposition in a design science approach. Institut d'Informatique et Organisation. Lausanne, Switzerland, University of Lausanne, Ecole des Hautes Etudes Commerciales HEC, 173, 2004.

[36] E. Paslaru. Introduction to Design Research, A Methodological Background for Scientific Work. 2004.

[37] H. Plattner, C. Meinel, and L. Leifer. Design thinking: understand-improve-apply. Springer Publishing Company, Incorporated, 2010.

[38] M. Sein, O. Henfridsson, S. Purao, M. Rossi, and R. Lindgren. Action design research. MIS Quarterly, 35(1):37-56, 2011.

[39] R. Davison, M. G. Martinsons, and N. Kock. Principles of canonical action research. Information Systems Journal, 14(1):65-86, 2004.

[40] S. Scholten, U. Scholten, and R. Fischer. Composite Solutions for Consumer-Driven Supply Chains: How to Control the Service-enabling Ecosystem? Proceedings of the 3rd academic symposium on Supply Management, Würzburg, Gabler-Verlag, 2010.

[41] U. Scholten, R. Fischer, D. Bojkov, and N. May: Supply Chain Control building on Emergent Self-Organizing Effects. In Supply Management Research, pages 311-335, 2011.

[42] U. Scholten. Service Level Management in Platform Ecosystems. Informatik 2010, Leipzig, Gesellschaft für Informatik, 2011.

[43] L. S. Kirsch. Portfolios of control modes and IS project management. Information Systems Research, 8(3):215-239, 1997.

[44] U. Scholten, N. Schuster, and S. Tai: A pattern language and repository for service network management. In Service-Oriented Computing and Applications (SOCA), 2012 5th IEEE International Conference on, pages 1-9. IEEE, 2012.

[45] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee. A design science research methodology for information systems research. Journal of Management Information Systems, 24(3):45-77, 2007.

[46] K. M. Eisenhardt. Building Theories from Case Study Research. Academy of Management Review, 14(4):532-550, 1989.

[47] C. Legner. Do Web Services Foster Specialization?-An Analysis of Commercial Web Service Directories. Business Services: Konzepte, Technologien, Anwendungen-9. Internationale Tagung Wirtschaftsinformatik Wien, 1:67-76, 2009.

[48] U. Scholten, R. Fischer, and C. Zirpins. Perspectives for Web Service Intermediaries: How Influence on Quality Makes the Difference. E-Commerce and Web Technologies:145-156, 2009.

[49] Trello. Trello Homepage. 2013, www.Trello.com, retrieved 03.02.2013.

[50] N. May, U. Scholten, and R. Fischer: Towards an Automated Gap Analysis for E-Service Portfolios. In, pages 274-281. IEEE, 2011.

[51] J. Van Bon. Foundations of IT Service Management basierend auf ITIL. Van Haren Publishing, 2008.

[52] J. R. Commons. INSTITUTIONAL ECONOMICS. American Economic Review, 21:648-657, 1931.

[53] J. Rohlfs. A theory of interdependent demand for a communications service. The Bell Journal of Economics and Management Science:16-37, 1974.

[54] S. S. Oren and S. A. Smith. Critical mass and tariff structure in electronic communications markets. The Bell Journal of Economics:467-487, 1981.
[55] M. L. Katz and C. Shapiro. Technology adoption in the presence of network externalities. The journal of political economy:822-841, 1986.
[56] C. Shapiro and H. R. Varian. VERSIONING: THE SMART WAY TO. Harvard business review:107, 1998.
[57] J. C. Rochet and J. Tirole. Platform competition in two-sided markets. Journal of the European Economic Association, 1(4):990-1029, 2003.
[58] T. Eisenmann, G. Parker, and M. Van Alstyne. Opening platforms: How, when and why? Harvard Business School Entrepreneurial Management Working Paper(09-030), 2008.
[59] G. Nicolis and I. Prigogine. Self-organization in nonequilibrium systems: from dissipative structures to order through fluctuations. John Wiley and Sons, New York, 1977.
[60] I. Prigogine, I. Stengers, and H. R. Pagels. Order out of Chaos. Physics Today, 38:97, 1985.
[61] T. De Wolf and T. Holvoet. Towards a methodology for engineering self-organising emergent systems. Frontiers in Artificial Intelligence and Applications, 135:18, 2005.
[62] B. C. Neuman. Scale in distributed systems. Readings in Distributed Computing Systems. IEEE Computer Society Press, 1994.
[63] C. Binnig, D. Kossmann, T. Kraska, and S. Loesing: How is the weather tomorrow?: towards a benchmark for the cloud. In Proceedings of the Second International Workshop on Testing Database Systems, pages 9. ACM, 2009.
[64] R. D. Smith. The chief technology officer: Strategic responsibilities and relationships. Research-Technology Management, 46(4):28-36, 2003.
[65] I. van de Weerd, S. Brinkkemper, R. Nieuwenhuis, J. Versendaal, and L. Bijlsma: Towards a reference framework for software product management. In Requirements Engineering, 14th IEEE International Conference, pages 319-322. IEEE, 2006.
[66] S. Trenner. Der CIO in Unternehmen. Seminar Corporate Governance, Potsdam, 2009.
[67] D. Tapscott. The digital economy: Promise and peril in the age of networked intelligence. McGraw-Hill New York, 1996.
[68] A. Meier and S. Ullrich. Zur Klassifikation von Geschaeftsmodellen im Market Space. HMD-Praxis der Wirtschaftsinformatik, 261:7-19, 2008.
[69] R. C. Lewis and B. H. Booms. The marketing aspects of service quality. Emerging perspectives on services marketing, 65(4):99-107, 1983.
[70] S. Frølund and J. Koistinen. Qml: A language for quality of service specification. HP Laboratories Technical Report HPL No. 1368-6798, 1998.
[71] A. Parasuraman, V. A. Zeithaml, and L. L. Berry. A conceptual model of service quality and its implications for future research. The Journal of Marketing:41-50, 1985.
[72] A. Keller and H. Ludwig. The WSLA framework: Specifying and monitoring service level agreements for web services. Journal of Network and Systems Management, 11(1):57-81, 2003.
[73] J. O'Sullivan. Towards a Precise Understanding of Service Properties. Dissertation, Brisbane, 2006.
[74] E. M. Maximilien and M. P. Singh. Reputation and endorsement for web services. ACM SIGecom Exchanges, 3(1):24-31, 2001.

[75] L. H. Vu, M. Hauswirth, F. Porto, and K. Aberer. A search engine for QoS-enabled discovery of semantic web services. International Journal of Business Process Integration and Management, 1(4):244-255, 2006.

[76] S. Kalepu, S. Krishnaswamy, and S. W. Loke: Reputation= f (user ranking, compliance, verity). In Web Services, 2004. Proceedings. IEEE International Conference on, pages 200-207. IEEE, 2004.

[77] J. Cardoso and A. Sheth. Semantic e-workflow composition. Journal of Intelligent Information Systems, 21(3):191-225, 2003.

[78] E. Kim and Y. Lee. Quality Model for Web Services (WSQM). Organisation for the Advancement of Structured Information Standards (OASIS), 2005.

[79] F. M. Feller. PayPal—Globales Zahlungssystem mit Kompetenz für lokale Zahlungsmärkte. Handbuch E-Money, E-Payment & M-Payment:237-247, 2006.

[80] C. Janiesch, M. Niemann, and N. Repp: Governance in the Internet of Services: Governing Service Delivery of Service Brokers. In (Pre-)ICIS SIG SVC Conference, pages 1-2, Paris, 2008.

[81] B. S. Frey and F. Oberholzer-Gee. The cost of price incentives: An empirical analysis of motivation crowding-out. The American economic review:746-755, 1997.

[82] A. Jøsang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. Decision support systems, 43(2):618-644, 2007.

[83] L. Mui, M. Mohtashemi, and C. Ang: A probabilistic rating framework for pervasive computing environments. In Proceedings of the MIT Student Oxygen Workshop (SOW'2001), 2001.

[84] Facebook. Homepage. 2013, www.Facebook.com, retrieved 01.02.2013.

[85] K. Chard, S. Caton, O. Rana, and K. Bubendorfer: Social cloud: Cloud computing in social networks. In Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on, pages 99-106. IEEE, 2010.

[86] E. M. Rogers. Diffusion of innovations. Simon and Schuster, 1995.

[87] M. Claypool, P. Le, M. Wased, and D. Brown: Implicit interest indicators. In Proceedings of the 6th international conference on Intelligent user interfaces, pages 33-40. ACM, 2001.

[88] S. Fox, K. Karnawat, M. Mydland, S. Dumais, and T. White. Evaluating implicit measures to improve web search. ACM Transactions on Information Systems (TOIS), 23(2):147-168, 2005.

[89] Dropbox. Homepage. 2013, www.Dropbox.com, retrieved 03.02.2013.

[90] C. Bizer, T. Heath, and T. Berners-Lee. Linked data-the story so far. International Journal on Semantic Web and Information Systems (IJSWIS), 5(3):1-22, 2009.

[91] T. Berners-Lee. Long Live the Web. Scientific American, 303(6):80-85, 2010.

[92] A. Hors and M. Nally. Using read/write Linked Data for Application Integration–Towards a Linked Data Basic Profile. Linked Data on the Web (LDOW2012), 2012.

[93] C. Haas, S. Caton, K. Chard, and C. Weinhardt: Co-operative infrastructures: An economic model for providing infrastructures for social cloud computing. In Proceedings of the 46th Annual Hawaii International Conference on System Sciences (HICSS), 2013.

[94] S. Tai, N. Desai, and P. Mazzoleni: Service communities: applications and middleware. In Proceedings of the 6th international workshop on Software engineering and middleware, pages 17-22. ACM, 2006.

[95] I. Silva-Lepe, R. Subramanian, I. Rouvellou, T. Mikalsen, J. Diament, and A. Iyengar. Soalive service catalog: A simplified approach to describing, discovering and

composing situational enterprise services. Service-Oriented Computing–ICSOC 2008:422-437, 2008.
[96] A. G. Kleppe. Software language engineering: creating domain-specific languages using metamodels. Addison-Wesley Professional, 2009.
[97] N. Chomsky. Three models for the description of language. Information Theory, IRE Transactions on, 2(3):113-124, 1956.
[98] J. Lyons. Introduction to theoretical linguistics. Cambridge university press, 1968.
[99] N. Chomsky. Syntactic structures. Walter de Gruyter, 2002.
[100] J. H. Larkin and H. A. Simon. Why a Diagram Is (Sometimes) Worth Ten Thousand Words. Cognitive Science, 11(1):65-99, 1987.
[101] H. A. Simon. On the forms of mental representation. Perception and cognition: Issues in the foundations of psychology, 9:3-18, 1978.
[102] Object Management Group Inc. Business Process Modeling Notation, V1.1. 2008, http://www.bpmn.org/Documents/BPMN%201-1%20Specification.pdf, retrieved 2008-10-20.
[103] R. W. Lewis. Programming industrial control systems using IEC 1131-3. Iet, 1998.
[104] OMG. UML 2.0 specification. 2005, retrieved 03.03.2012.
[105] B. Shneiderman. Designing the User Interface: Strategies for Effective Human-Computer Interaction. 4th edn. Addison-Wesley Amsterdam, 2004.
[106] J. Bertin. Semiology of graphics: diagrams, networks, maps. 1983.
[107] T. Clark, A. Evans, and S. Kent. Engineering modelling languages: A precise meta-modelling approach. Fundamental Approaches to Software Engineering:242-260, 2002.
[108] É. André. Metamodeling and Language Engineering, pages. 1-15, Fakultät für Theoretische Informatik, Technische Universität Dresden, 2006.
[109] D. Howe. Abstract Syntax. Free Online Dictionary of Computing, 1998.
[110] OMG. Meta Object Facility (MOF) Core Specification 2.0. 2006, http://www.omg.org/mof/, retrieved 03.03.2012.
[111] OMG. 2.0 OCL Specification. 2003, retrieved 02.03.2013.
[112] J. W. Forrester. Industrial dynamics. 1961. Pegasus Communications, Waltham, MA, 1961.
[113] M. Rysman. The economics of two-sided markets. The Journal of Economic Perspectives:125-143, 2009.
[114] F. M. Bass. A new product growth model for consumer durables. Management Science, 15:215-227, 1969.
[115] F. Bass. Frank M. Bass Official Website. 2012, http://www.bassbasement.org/BassModel/, retrieved 11.10.2012.
[116] Ventana. VenSim PLE Plus - User's Guide Version 4. Ventana Systems Inc, Harvard, 2011.
[117] E. Ries. The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses. Crown Business, 2011.
[118] M. A. Schilling. 8. Protecting or diffusing a technology platform: tradeoffs in appropriability, network externalities, and architectural control. Platforms, Markets and Innovation:192, 2009.
[119] K. Boudreau. Open platform strategies and innovation: Granting access vs. devolving control. Management science, 56(10):1849-1872, 2010.
[120] A. Hagiu and R. S. Lee. Exclusivity and Control. Journal of Economics & Management Strategy, 20(3):679-708, 2011.

[121] O. Föllinger. Regelungstechnik, Einführung in die Methoden und ihre Anwendungen. Aufl. Hüthig, Heidelberg, 1994.
[122] N. Wiener. Cybernetics, or Communication and Control in the Animal and the Machine. MIT Press, Boston, 1948.
[123] K. M. Eisenhardt. Control: Organizational and economic approaches. Management science:134-149, 1985.
[124] W. G. Ouchi. A conceptual framework for the design of organizational control mechanisms. Management science:833-848, 1979.
[125] S. J. Ashford and A. S. Tsui. Self-regulation for managerial effectiveness: The role of active feedback seeking. Academy of Management Journal:251-280, 1991.
[126] S. Bradner. Key words for use in RFCs to Indicate Requirement Levels, 676 http://www. ietf. org/rfc/rfc2119. txt, IETF RFC 2119. 1997, retrieved 14.09.2012.
[127] Salesforce.com. Homepage. 2013, www.Salesforce.com, retrieved 02.02.2013.
[128] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman. Reputation systems. Communications of the ACM, 43(12):45-48, 2000.
[129] O. M. G. OMG. Object Constraint Language (OCL). 2012, http://www.omg.org/spec/OCL/2.3.1, retrieved 03.03.2012.
[130] E. Guldentops, Steven De Haes, Gary Hardy, Jacqueline Ormsby, Daniel Fernando Ramos, Jon Singleton, and Paul A. Williams. Board Briefing on IT-Governance, Rolling Meadows, IT Governance Institute, 2003.
[131] S. K. Card, J. D. Mackinlay, and B. Shneiderman. Readings in information visualization: using vision to think. Morgan Kaufmann, 1999.
[132] Z. Whittaker. How far do Google Drive's terms go in 'owning' your files? 2013, http://www.zdnet.com/blog/btl/how-far-do-google-drives-terms-go-in-owning-your-files/75228, retrieved 08.02.2013.
[133] C. Rupp, J. Hahn, S. Queins, M. Jeckle, and B. Zengler. UML 2 glasklar. Hanser, 2005.
[134] R. B. Ferguson. Salesforce.com Unveils Force.com Cloud Computing Architecture. eWeek, Entreprise IT Technology New, Opinion and Reviews, Chicago, Ziff Davis Enterprise, 2008.
[135] N. Schuster, C. Zirpins, and U. Scholten. How to Balance Flexibility and Coordination? Service-oriented Model and Architecture for Document-based Collaboration on the Web. International Conference on Service Oriented Computing & Applications SOCA 2011, Irvine, IEEE, 2011.
[136] T. Raffelsieper, J. Becker, M. Matzner, and C. Janiesch. Requirements for a Pattern Language for Event-driven Business Activity Monitoring. Münster, 2011.
[137] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow Patterns. Distributed and Parallel Databases, 14(1):5-51, 2003.
[138] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, MA, 2005.
[139] S. Scholten and U. Scholten. Platform-based Innovation Management: Directing External Innovational Efforts in Platform Ecosystems. Journal of the Knowledge Economy:1-21, 2012.
[140] M. Cusumano and A. Gawer. The Elements of Platform Leadership. MIT Sloan Management Review, 43/3:51-58, 2002.
[141] F. Keller and S. Wendt: FMC: An Approach Towards Architecture-centric System Development. In 10th IEEE International Conference and Workshop on the

Engineering of Computer-Based Systems (ECBS), pages 173-182, Huntsville, AL, 2003.

[142] M. Czuchra, N. Peters, W. Tscheschner, M. Kuntze, and G. Decker. Oryx-Editor: Web-based Graphical Business Process Editor. . 2012, http://code.google.com/p/oryx-editor/, retrieved 15.05.2012.

[143] G. Decker, H. Overdick, and M. Weske. Oryx – An Open Modeling Platform for the BPM Community. In M. Dumas, M. Reichert, and M.-C. Shan (eds.) Business Process Management, vol. 5240, pages 382-385. Springer Berlin / Heidelberg, 2008.

[144] G. Decker, A. Grosskopf, and A. Barros: A Graphical Notation for Modeling Complex Events in Business Processes. In 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC), pages 27, Annapolis, MD, 2007.

[145] C. Cabanillas, M. Resinas, and A. Ruiz-Cortés. RAL: A High-Level User-Oriented Resource Assignment Language for Business Processes. In F. Daniel, K. Barkaoui, and S. Dustdar (eds.) Business Process Management Workshops, vol. 99, pages 50-61. Springer, Heidelberg, 2012.

[146] B. Hoehrmann. RFC 4329 Scripting Media Types. 2006.

[147] D. Crockford. The application/json Media Type for JavaScript Object Notation (JSON). 2006, http://tools.ietf.org/html/rfc4627, retrieved 15.05.2012.

[148] J. Ferraiolo, F. Jun, and D. Jackson. Scalable vector graphics (SVG) 1.1 specification. 2003, retrieved 14.02.2013.

[149] U. Scholten and M. Reimchen. DynoAnalyzer, pages. Analyzer expanding DynoCloud.org, Karlsruhe, Google Code, 2012.

[150] B. Naveh. JGraphT. 2008, http://jgrapht. sourceforge. net, retrieved 11.10.2012.

[151] M. Josek. Erweiterung der Dynamic Network Notation und des DynoCloud Editors um ein ordinales Gewichtungsverfahren von Base-Value Alternativen. Bachelor Arbeit, Karlsruhe, 2012.

[152] N. A. Schuster. Coordinating Service Compositions: Model and Infrastructure for Collaborative Creation of Electronic Documents. Dissertation, Karlsruhe Institute of Technology, Karlsruhe, 2013.

[153] S. L. Pfleeger and B. A. Kitchenham. Principles of survey research: part 1: turning lemons into lemonade. SIGSOFT Softw. Eng. Notes, 26(6):16-18, 2001.

[154] B. A. Kitchenham and S. L. Pfleeger. Principles of survey research part 2: designing a survey. SIGSOFT Softw. Eng. Notes, 27(1):18-20, 2002.

[155] B. Kitchenham and S. L. Pfleeger. Principles of survey research: part 5: populations and samples. SIGSOFT Softw. Eng. Notes, 27(5):17-20, 2002.

[156] B. Kitchenham and S. L. Pfleeger. Principles of survey research part 4: questionnaire evaluation. SIGSOFT Softw. Eng. Notes, 27(3):20-23, 2002.

[157] U. Scholten, R. Fischer, C. Zirpins, and S. Scholten. DYNO: A Notation to Leverage Dynamic Network Effects in PaaS Ecosystems. International Conference on Service Oriented Computing & Applications SOCA 2011, Irvine, IEEE, 2011.

[158] S. T. Pai and A. Fellah. India 4G and Cellular Market Analysis and Forecasts, 2010-2015. Maravedis Wireless Market Research & Analysis, pages. 01.09.2010, New Delhi, Maravedis, 2010.

[159] J. Becker, M. Rosemann, and C. v. Uthmann. Guidelines of Business Process Modeling. In W. van der Aalst, J. Desel, and A. Oberweis (eds.) Business Process Management: Models, Techniques, and Empirical Studies. Lecture Notes In Computer Science vol. 1806, pages 30-49. Springer, Berlin, 2000.

[160] B. A. Kitchenham and S. L. Pfleeger. Principles of survey research: part 3: constructing a survey instrument. SIGSOFT Softw. Eng. Notes, 27(2):20-24, 2002.

[161] E. Wittern and C. Zirpins. Service Feature Modeling - Modeling and Participatory Ranking of Service Design Alternatives, Karlsruhe, Karlsruhe Institute of Technology, 2012.

[162] Loughborough-University. Questionnaire Design. 2012, http://www.lboro.ac.uk/media/wwwlboroacuk/content/library/downloads/advicesheets/questionnaire.pdf, retrieved 25.10.2012.

[163] S. S. Stevens. On the theory of scales of measurement, Bobbs-Merrill, College Division, 1946.

[164] S. Siegel. Nonparametric statistics. The American Statistician, 11(3):13-19, 1957.