Towards Intent-Driven Systems

Johan Silvander
Abstract

Context: Software supporting an enterprise’s business, also known as a business support system, needs to support the correlation of activities between actors as well as influence the activities based on knowledge about the value networks in which the enterprise acts. This can be supported with the help of intent-driven systems. The aim of intent-driven systems is to capture stakeholders’ intents and transform these into a form that enables computer processing of them. Only then are different machine actors able to negotiate with each other on behalf of their respective stakeholders and their intents, and suggest a mutually beneficial agreement.

Objective: When building a business support system it is critical to separate the business model of the business support system itself from the business models used by the enterprise which is using the business support system. The core idea of intent-driven systems is the possibility to change behavior of the system itself, based on stakeholder intents. This requires a separation of concerns between the parts of the system used to execute the stakeholder business, and the parts which are used to design the business based on stakeholder intents. The business studio is a software that supports the realization of business models used by the enterprise by configuring the capabilities provided by the business support system. The aim is to find out how we can support the design of a business studio which is based on intent-driven systems.

Method: We are using the design science framework as our research framework. During our design science study we have used the following research methods: systematic literature review, case study, quasi experiment, and action research.

Results: We have produced two design artifacts as a start to be able to support the design of a business studio. These artifacts are the models and quasi-experiment in Chapter 3, and the action research in Chapter 4. The models found during the case study have proved to be a valuable artifact for the stakeholder. The
results from the quasi-experiment and the action research are seen as new problem solving knowledge by the stakeholder.

Conclusion: The synthesis shows a need for further research regarding semantic interchange of information, actor interaction in intent-driven systems, and the governance of intent-driven systems.
Acknowledgments

To everyone who have contributed to the one I am today, for good and for worse.
No one mentioned and no one forgotten.
Preface

Papers in This Thesis

This compilation thesis includes the following five papers.


Chapter 5: Johan Silvander and Mikael Svahnberg, “Towards Executable Business Rules”, Intended for a conference (Accepted as an appendix to Chapter 3).


Contribution Statement

Johan Silvander is the lead author of all the papers in this thesis. As a lead author, he took the main responsibility in designing the studies, collecting and analyzing

---

1Papers marked with “Under Review” are currently considered for publication. Papers marked with “Under Submission” have been submitted but were not reviewed yet at the time of writing.
data, and reporting the findings in peer-reviewed publications. Furthermore, he is the sole author of Chapter 1, the overview. The co-authors’ contribution are described below.

Chapter 2: Mikael Svahnberg contributed with valuable methodology support during the whole study. He reviewed and commented on intermediate versions and the final draft off the paper.

Chapter 3: Magnus Wilson contributed with valuable comments during the analysis of the results. Magnus Wilson, Krzysztof Wnuk and Mikael Svahnberg reviewed and commented on intermediate versions and the final draft off the paper.

Chapter 4: Magnus Wilson wrote parts of the Introduction and Background sections. Magnus Wilson and Krzysztof Wnuk reviewed and commented on intermediate versions and the final draft off the paper.

Chapter 5: Mikael Svahnberg reviewed and commented on intermediate versions and the final draft off the paper.

Chapter 6: Mikael Svahnberg reviewed and commented on intermediate versions and the current draft off the paper.

Related Papers Not Included in This Thesis

- Magnus Wilson, Krzysztof Wnuk, Johan Silvander and Tony Gorschek, “Towards effective and efficient business modeling - A systematic literature review” journal (Under Submission).

Other Papers Not Included in This Thesis


Funding

The research presented in this paper was partly funded by the Swedish Knowledge Foundation, the project “Professional Licentiate of Engineering Research School”,

```
conducted at Blekinge Institute of Technology, Software Engineering Research Lab - Sweden.
# Contents

Abstract  i

Acknowledgments  iii

Preface  iv

1 Overview  1
  1.1 Introduction  1
    1.1.1 The use of Pask’s conversation theory  3
    1.1.2 Background and related work  5
    1.1.3 Research questions  8
  1.2 Methodology  9
    1.2.1 Research settings and data collection  9
    1.2.2 Limitations  11
  1.3 Research Framework and Research Execution  11
    1.3.1 Research framework  11
    1.3.2 Research execution  13
  1.4 Conclusion and Further Research  16
    1.4.1 Conclusion  16
    1.4.2 Produced design artifacts  17
    1.4.3 Suggested new design problems  18

2 A Systematic Literature Review on Intent-Driven Systems  20

3 Supporting Continuous Changes to Business Intents  22

4 Encouraging Business Flexibility by Improved Context Descriptions  24

5 Towards Executable Business Rules  25
Chapter 1

Overview

1.1 Introduction

Knowledge creation in enterprises is obtained through interactions between actors [24]. Intents are the aim and purpose resulting from knowledge creation regarding internal or external influences. The SECI-model [24] and Pask’s conversation theory [30] are models which can be used for knowledge creation when enterprises are interacting with each other.

We define intent as a subject or type of possible behavior, i.e. something that can be interpreted to have significance. Any actor can have intents. When an actor publicly declare an intent in a certain context, it becomes a stated intent. A stated intent could be a declaration of capabilities an actor promise to provide, or a requirement an actor try to impose on another actor. A true cooperation between actors is based on the capabilities promised by the interacting actors. A decision about whether an actor has kept its promise or not, can be done by the promising actor as well as any actor which is observing the behavior of the promising actor.

Definition 1.1: Our definition of intent

Today’s enterprises are part of value networks. A value network refers to “A set of connections between organizations and/or individuals interacting with each other to benefit the entire group [17]”. Enterprises in a value network can be seen as parts in a compositional system and are by themselves compositional
Chapter 1. Overview

systems, also known as a system of systems. Compositionality refers to “The evident ability of humans to represent entities as hierarchies of parts, with these parts themselves being meaningful entities, and being reusable in a near infinite assortment of meaningful combinations [13]”. Cyber-physical systems can be seen as a vital part of a compositional system.

The construction of a compositional system requires methods to achieve a holistic collective benefit through the individual systems’ participation and cooperation when each system adopts a solution that maximizes its own self-interest [10]. The construction of such a system has to support changes of a system’s policies and rules in a way that are effective and efficient [12]. Achieving the changes in an effective and efficient way requires knowledge about the affected intents and the correlation between intents. The correlations between intents are affected by factors like value network structure, decision process and the actors’ responsibilities.

The actors in a compositional system may be humans or machines. By using software agents as machine actors enterprises can bring customers closer to suppliers of products and services, support their continued demand for change, inject further intelligence into enterprises and simplify the environment for both customers and employees [1]. Software supporting an enterprise’s business, also known as a business support system, needs to support the correlation of activities between actors as well as influence the activities based on knowledge about the compositional systems the enterprise acts in.

An actor’s intent has to be communicated to other actors. During the interaction about the intent the actors have to prove their understanding of the intent in order to gain a common understanding and knowledge about the intent. This interaction can continue in several steps and might re-shape the original intent. Together with Ericsson we are using Pask’s conversation theory [30] as a model to describe intent-driven systems.

The remaining sections in the overview of this thesis is organized as follows. In Section 1.1.1 we present how we are using Pask’s conversation theory, followed by the presentation of the background and related work in Section 1.1.2. The research questions are presented in Section 1.1.3. The research methodology is presented in Section 1.2, followed by a presentation of the research framework, and the execution of the research in Section 1.3. Finally, in Section 1.4, the conclusion and further research are presented.
Chapter 1. Overview

We define an intent-driven system as a compositional system of actors where the actors declare, negotiate, and assess intents made by actors. The intents can be declared, negotiated, or assessed, on-behalf of an actor’s stakeholder or an actor’s self-interest.

The aim of intent-driven systems is to capture stakeholders’ intents and transform these into a form that enables computer processing of them. Only then are different machine actors able to negotiate with each other on behalf of their respective stakeholders and their intents, and suggest a mutually beneficial agreement. This requires a separation of concerns between the parts of the system used to execute the stakeholder business, and the parts which are used to design the business based on stakeholder intents.

Definition 1.2: Our definition of intent-driven systems

1.1.1 The use of Pask’s conversation theory

The minimal structure of the conversation theory is shown in Figure 1.1. Figure 1.1 illustrates how two actors (A and B) interact on the same domain D with the help of a common language L. In order to construct, express, and validate a topic, the two layers of procedures with feedback loops (FB) and feedforward loops (FF) exist. The P(0) layer procedures operate upon the domain in order to bring about or explain topic relations. The P(1) layer operate on P(0) procedures in order to construct or reconstruct them.

![Figure 1.1: The minimal structure of the conversation theory.](image)

Two key components in Pask’s conversation theory [30] are language and domain. A language L is defined as:
Chapter 1. Overview

“L may be a natural, written or spoken, symbolic language, but it need not be. It may be a system of symbolic behaviors such as dance or actions such as key pressing. It may be formalized, as in, mathematics and higher level programming languages, but it need not be. It must however have many of the qualities of a natural language, with possibilities to express and interpret commands, questions, answers, obediences, explanations, or descriptions.” [21].

A domain could be described as:

“A domain is a collection of topic’s, and a topic is essentially a relation. This may be a very concrete relation (a relation between alphabetic characters and the keyboard positions in type writing) or it may be an abstract relation (a relation between smugglers and the countries they operate in): To learn or solve a problem is to ‘bring about’ such a topic relation.” [21].

Pask [30] stresses the fact that the different actors have obtained their specific domain information through several different interactions which makes the model recursive. This means that each actor obtains its specific domain information in different contexts. We define a context frame as the total domain information for a specific domain an actor has obtained. A context description is what gives a context frame a scope (boundary) and defines a meaning (semantics).

The idea of an intent-driven system as chains of interacting loops with one or several cycles is shown in Figure 1.2. Figure 1.2 shows how two human actors (A and B) interacts about an intent, using language L regarding domain D. The interaction results in some sort of common understanding. Each human actor interacts with its respective machine actor to translate the intent from domain D to domain $D'$, using language $L_{hm}$. The two machine actors ($A'$ and $B'$) interact on the intent, using Language $L'$ regarding domain $D'$, and the outcome is fed back to their respective human actor, using language $L_{mh}$. One or both of the human actors may not be satisfied with the outcome. This will render the start of a new cycle. This indicates that actor interaction between several actors of different types are vital for intent-driven systems. The languages are built upon semantics and a collaboration between actors might need a interchange of semantics. Since an intent-driven system is a compositional system, governance is needed to guide the different actors to achieve a holistic collective benefit.

When building a business support system it is critical to separate the business model of the business support system itself from the business models used by the enterprise which is using the business support system. The business studio will support the realization of business models used by the enterprise by configure the capabilities provided by the business support system.
The idea of a business studio is shown in Figure 1.3. Figure 1.3 shows how two human actors (A and B) interacts with a business studio, which is the dotted part in the figure. The human actor A represents the enterprise using the business support system an the human actor B represents the owner of the business support system. Actor A and actor B interact using language L to formulate common understanding of the needed capabilities. Actor B interacts with the component (B’) in the business support system to configure basic capabilities according to the understanding obtained during the interaction with actor A. The interactions between actor B and component B’ are regarded as a configuration of the business studio, and is not part of the business studio itself. Actor A interacts with component A’ to obtain its business intents. Component A’ is responsible to translate the business intents and composing the needed functionality with the help of the basic capabilities (B’). Since the business studio is not a monolithic actor, and there are more actors than one in an enterprise, the business studio will be a composition of what is described in Figure 1.2.

1.1.2 Background and related work

The research project is done in collaboration with Ericsson and is based on the design science methodology [14]. At the time of the project, Ericsson was in an early pre-study phase of a business studio. The business studio is part of a business support system and will act in the area of planning and monitoring business intents. The idea with the business studio is to deliver support for a 360-degree view of an enterprise’s business. The view includes both the actual execution of an enterprise’s business and the intended changes to this execution.
Chapter 1. Overview

Figure 1.3: An explanation model of the business studio concept based on intent-driven systems.

The business studio will support and govern the decisions and actions needed to maintain or change the way an enterprise does its business. Ericsson’s customers will be able to buy the business studio as a product or a service. With the help of the business studio, Ericsson’s customers will get support and knowledge about how to configure, monitor, and redesign their products and business. The software could be used in different business areas, for example charging, billing, customer relationship management, partner relationship management, product management, order management, etc. The business studio supports the idea of continuous business-requirement engineering. Intent-driven systems are one of the cornerstones for the business studio product.

The business intent realization builds upon collaborations in the form of interactions between the different actors. The interactions between the actors are negotiated since each actor has its own view of a business intent. The negotiation may result in a, to some extent, desired outcome. The outcome of an interaction between human actors results in conclusions stated in natural language. From a software engineering perspective the conclusions can be made executable in the form of policies and rules but this would require Natural Language Processing, suggested in [37], and a formal way of expressing the policies and rules, e.g. using Semantics of Business Vocabulary and Rules (SBVR) [26]. Since SBVR is business agnostic semantics and ontologies are needed to give meaning to the policies and rules. The appropriate semantics and ontologies are not available in today’s enterprises [15].

Combinations of interactions between more than one real world actor exists in the literature [8, 16, 18], but it is not evident that the combinations presented in the literature can be used in other domains due to the tight coupling between the realization and the information in each solution. Interactions are not only
Chapter 1. Overview

taking place between individual actors. The interactions between groups are an integral part of the SECI-model which is used for knowledge creation [24]. The interactions between groups are supported in Pask’s conversation theory [30].

When constructing a compositional system, it is necessary to understand the information in a system’s context frames as well as having temporal separation [11], and for systems that are part of a compositional system to coordinate and cooperate in uncertain environments [9]. In a compositional system, multi-context capabilities are needed [29] since an actor’s context frame might have to act in several roles.

Intent-driven systems have a need for governance since new or modified business intents introduce changes in the correlations between business intents. To govern all the business intents requires each involved party to share a common understanding of the used governance model. Since several actors have to collaborate to fulfill a business intent it is not realistic to assume the use of one homogeneous governance model. To achieve a common understanding of heterogeneous governance models, collaboration between different governance models could be used. This collaboration requires appropriate semantics and ontologies which are not available today [15]. Governance models are discussed in, e.g. Wiesner et al. [37], Beigi et al. [3], and Lewis et al. [20].

Creating software components that can be orchestrated and bringing value to the relevant stakeholders in business ecosystems, and timely respond to frequent changes remains the main challenge. This is partly addressed by Software product lines [4] and industrial Product-Service systems [23] which focused on changeability [31], as ways to create flexible, adaptable and efficient component-based software architectures. To obtain the possibility of a high level of automation of decision and action selection [28] the information acquisition has to be effective and efficient. This requires the possibility to automatically apply rules and constraints during the information acquisition to improve the sender’s and the receiver’s understanding of the information.

To fully support business flexibility, we need to better understand and define the business context. Modeling context is also critical for developing context-aware software systems [5]. Baldauf et al. [2] summarized context-aware systems including methods to achieve context-awareness. Despite several similarities, context-aware software systems focus on dynamically discoverable services rather than dynamically changing business opportunities.

Supporting business flexibility requires support for agile business policies and agile business rules [6] which are used to govern how an enterprise does its business [25]. It is desired to have a common governance structure and a standardized way
of handling the business rules. Rosca et al. have contributed valuable knowledge in the area of common governance of business rules [33, 34].

1.1.3 Research questions

According to Runeson et al. [35] research questions are statements about the knowledge that is being sought, or is expected to be discovered, during the study. The discovery or attainment of this knowledge demonstrates that the study has achieved its intended objectives.

In design science a research question is a knowledge-question supporting a design problem. The knowledge-question-answering activity returns knowledge to the design-problem-solving activity. Since design problems can create new problems, this generates an iteration over design problems and knowledge questions in design science [36].

In order to understand how we can support the design of a business studio we formulate a research question (\textbf{mRQ: What design artifacts are needed in order to construct a business studio which is based on intent-driven systems?}) which will serve as the main knowledge question for this design science study.

We formulate two knowledge questions in order to understand the state-of-the-art with respect to intent-driven systems. These knowledge questions are stated as research questions and are investigated in Chapter 2. The first research question (\textbf{RQ1: What methods/techniques supporting intent-driven systems have been presented in literature?}) is used to find evidence in the literature, of methods/techniques supporting intent-driven systems, i.e. supporting the construction, expressing and validation of business intents. RQ1 is divided into six sub-questions which are mapped to intent-driven systems with the help of Figure 1.4. Each pair of research questions in Figure 1.4 are covering how a business intent can be constructed, expressed, and validated. RQ1.1, RQ1.3, and RQ1.5 is used to investigate how a business intent can be constructed and expressed. RQ1.2, RQ1.4, and RQ1.6 is used to investigate how a business intent is validated.

The second research question (\textbf{RQ2: What evidences for enabling flexible realizations of intent-driven systems have been presented in literature?}) is used to find evidence in the literature of how the different methods/techniques could be used together to enable flexible realizations of intent-driven systems. We focus on four aspects; semantics, interchange of semantics, governance, and actor interaction.

The synthesis of RQ1 and RQ2 do not provide all the needed answers to construct a business studio. The aspects of semantics, interchange of semantics, governance,
Chapter 1. Overview

and actor interaction are not provided to the extent that a business studio can be constructed. Instead we formulate a new knowledge question which is stated as a research question (RQ3: How can support for continuous changes to business intents be realized in business support systems?) and investigated in Chapter 3. The focus of the knowledge question is on the aspects of semantics, interchange of semantics, governance, and actor interaction.

The research questions RQ1-RQ3 support Ericsson in their early pre-study phase of a business studio. The results from Chapter 2, 3, and 4 generate design artifacts and new design problems which make it possible for Ericsson to produce a first proof-of-concept of a business studio. This will make it possible for us to fully answer mRQ in the future.

Figure 1.4: Mapping of RQ1’s sub-questions to the intent-driven systems model.

1.2 Methodology

In this section we discuss the research methodology used in this thesis. In Section 1.2.1 we discuss the research settings and data Collection, and Section 1.2.2 describes the limitations of this thesis.

1.2.1 Research settings and data collection

Each chapter details research settings and methods for data collection. However, we would like to highlight the main points for each chapter.

The systematic literature review in Chapter 2 use a methodology based on Kitchenham and Charters’ guidelines [19]. We start to investigate the need for a systematic literature review and conduct a pilot study. During the pilot study we refine the search strings, specify a study selection approach, develop a search strategy, define study quality assessment, define a data extraction process, and create the review protocol.
Chapter 1. Overview

Since the ideas span a wider area than computer science we select SCOPUS as the primary source, with subsequent additional searches in ISI/Web of science, ACM and IEEE Xplorer. In all, we investigate eleven different subject areas (Multidisciplinary, Economics, Environmental Science, Social Sciences, Arts and Humanities, Psychology, Engineering, Mathematics, Business, Decision Sciences, or Computer Science.).

A reference management system\(^1\) is used to import the papers and the first full text reading selection was done in the tool. The selected papers are then imported to a tool named Atlas.ti\(^2\) supporting the grounded theory approach [7]. With the help of Atlas.ti, the coding is done as an iterative process. We apply open coding to each of the first fifteen papers. Selective coding is done on all the read papers. With the help of memos we create the first iteration of the extraction table. This process continues until all papers are read. The papers from the former sets are included in new iterations. A coding and recoding is applied on papers when the understanding of the content is improved. The theoretical codes are supported by using network views (a feature in Atlas.ti). The network views based on codes are used to find the relation between the papers. The extracted data are analyzed and synthesized based on the memo concept in grounded theory [7].

In Chapter 3 we use a case study methodology. The design of this case study is based on Runeson et al. [35] using a focus group approach [38, 32] in order to mimic the workflow used in the studied company. Information is captured in documents, drawings, and photos. We apply open coding technique [7] to analyze the collected qualitative data.

The data are collected during focus group [38] interviews with the appointed persons. During the interviews the information is captured on whiteboards, in PowerPoint documents, and directly in a document. The interviews are approximately two hours in length and are based on semi-structured focus group interviews [32]. Data triangulation is achieved by using interviews, informal meetings, continuous member checking [32].

We apply open coding technique [7] to analyze the collected qualitative data. Open coding help us to find common terms and concepts as well as to find synonyms and hyponyms which are used by the study subjects. The first and second authors perform open coding and iteratively discuss its results. The aim with this coding approach is to agree upon and to find new terms or concepts

\(^1\)Mendeley https://www.mendeley.com

\(^2\)Atlas.ti is a system for conducting grounded theory analysis of texts and voice recordings. In this tool you can mark text and apply codes, and you can generate different views (e.g. a network view, where different relationships between papers can be investigated.) of your analysis. http://atlasti.com
Chapter 1. Overview

during each meeting. These terms or concepts are linked to existing ones and their meaning are revised, if needed.

For the quasi-experiment in Chapter 3 and the action research in Chapter 4, we are provided with data from the practitioners. In both cases we use machine learning pipe-lines as the test and validation instruments.

1.2.2 Limitations

For each chapter we discuss validity threats according to Robson [32] and perform the countermeasures we find appropriate. However, we would like to highlight the main validity threats for each chapter.

The semantics is a problem when interpreting data. In Chapter 2 we investigate eleven different subject areas. Depending on the subject area and the problem context the semantics sometimes is different. This might lead to an ineffective search string but we believe that the pilot study and the refinement of the search string are mitigating this problem. In Chapter 3 we have the possibility to talk to the participants and correct misinterpretations. However, there is always a risk that an agreement is done on a too high abstraction level, leaving the interpretation of the details as an assumed consensus.

The data we use for the quasi-experiment in Chapter 3 and the action research in Chapter 4 are test data, which the practitioners provide to us. The lack of real customer data might impose a limitation on the results. The implementations are not part of the practitioners current solution.

1.3 Research Framework and Research Execution

In this Section 1.3.1 we give an overview of the framework used for our research and in Section 1.3.2 the execution of the research is described.

1.3.1 Research framework

The research framework we have use in this thesis is the design science methodology [14, 36, 22]. Design science is the design and investigation of artifacts in context. The artifacts are designed to interact with a problem context in order to improve something in that context. The problem context can be extended with
social context and knowledge context. The social context consists of stakeholders who may affect or may be affected by the project. The knowledge context consists of knowledge from natural science, design science, design specifications, useful facts, practical knowledge, and common sense. Figure 1.5 shows a framework for design science.

In Figure 1.5 one can see the two different parts of design science, design and investigation. These two parts correspond to two kinds of research problems, design problems and knowledge questions. Knowledge questions are used to understand the existing knowledge which is useful for the artifact in context. We expect to find one answer to a knowledge question. The answer should be evaluated by truth, which is not dependent on the stakeholder goals. However, it is possible that the answer is wrong or incomplete, depending on our understanding of the problem context or the knowledge context. Design problems call for a change in the real world and require an analysis of actual or hypothetical stakeholder goals. A solution is a design, and there are usually many different solutions. The solutions are evaluated by their utility with respect to the stakeholder goals, and there is not one single best solution.

How the different chapters relate to Design and Investigation is indicated in Figure 1.5.

![Figure 1.5: Design Science framework (adopted from Wieringa [36]).](image)
Chapter 1. Overview

1.3.2 Research execution

The problem context in this thesis is how Ericsson’s business studio can support continuous business-requirement engineering with the help of intent-driven systems. In order to design relevant artifacts, get answers to knowledge questions, and provide new answers to the knowledge context we perform three studies. These studies are presented in Chapter 2, Chapter 3, and Chapter 4. A brief overview of how we use the design science framework for the studies are given below.

In Chapter 2 we adopt the constructive worldview in order to generate a theory which can be used to create understanding about a problem context. Together with experts from Ericsson the ideas of what is needed to create an intent-driven system are assembled. These ideas form an understanding of what is needed and useful in order to enable flexible realizations of intent-driven systems. The core idea of intent-driven systems is the possibility to change behavior of the system itself, based on stakeholder intents. This requires a separation of concerns between the parts of the system used to execute the stakeholder business, and the parts which are used to design the business based on stakeholder intents. Based on this we formulate two knowledge questions (RQ1 and RQ2).

In order to answer RQ1 and RQ2 we perform a systematic literature review. We use Kitchenham and Charters guidelines [19] for systematic literature reviews and our purpose with the systematic literature review is in-line with what Robson states [32]. The systematic literature review help us to understand the state-of-the-art with respect to intent-driven systems. During this systematic literature review, we study available methods and techniques that may be useful for supporting intent-driven systems, as well as the existence of aspects needed to enabling flexible realizations of intent-driven systems. The data extraction process is based on the data analysis used in grounded theory.

Answers to the RQ1 and RQ2 exist. The existence of methods/techniques which can be used as building blocks to construct intent-driven systems exist in the literature. How these methods/techniques can interact with the aspects needed to enabling flexible realizations of intent-driven systems is not evident in the existing literature. The synthesis shows a need for further research regarding the aspects; semantics, interchange of information, actor interaction in intent-driven systems, and governance of intent-driven systems. The existing answers to the knowledge questions (RQ1 and RQ2) give valuable knowledge to the problem context. However, in order to design artifacts we need more knowledge about the how to fill the gaps identified in Chapter 2.

In Chapter 3 we investigate the gaps that are identified in Chapter 2. In Chapter 2
Chapter 1. Overview

we identify a gap in the literature regarding multi-actor interactions in intent-driven systems. This indicates the need for further research since business support systems need to support the correlation of activities between actors as well as influence the activities based on knowledge about the value networks in which the enterprise acts. These activities are used to fulfill or create business intents, which often are expresses as business rules. The life cycle of business intents needs to be governed and the a semantic interchange between different parts of the ecosystem must be supported. The aim of the study is to identify how a business support system can support continuous changes to business intents. The first step is to find a theoretical model which serves as a foundation for intent-driven systems. Based on this we formulate a new knowledge question (RQ3).

Once again we adopt the constructive worldview in order to generate a theory which can be used to create an artifact. The research methods we choose from are a case study or surveys and questionnaires. The problem with a semantic interchange, which we find in the systematic literature review, indicate a need for the possibility to clarify terms with the involved participants. We therefore conduct a study in which we can get knowledge from practitioners and contribute new answers to the knowledge context.

In order to answer the knowledge question we conduct a case study using a focus group approach with employees from Ericsson. We use the methodology defined by Runeson et al. [35]. The case study is influenced by the spiral case study process [35] with an iterative character in order to stepwise adjust the goal and scope to the iterative findings. In order to analyze the data we use the open-coding technique in grounded theory. Since we decide to validate the theoretical models, after the case study is performed, we use a type of mixed methods design called sequential exploratory design. The validation of the theoretical model was performed as a proof-of-concept, studied in a quasi-experiment.

The case study result in a model supporting continuous definition and execution of an enterprise. The model is divided into three layers; Define, Execute, and a common governance view layer. This makes it possible to support continuous definition and execution of business intents and to identify the actors needed to support the business intents’ life cycles. This model is supported by a meta-model for capturing information into viewpoints. We agree with Robson about the outcome [32], but instead of performing another case study in order to get deeper understanding of the models, we decide to validate the models with the help of a proof-of-concept.

In the proof-of concept, we borrow ideas from the gaming industry where a specific context gives the character the possibilities to, for example find specific treasures and stipulates how these treasures can be handled. The proof-of-concept
Chapter 1. Overview

shows that; the Design layer is supported by the possibility to, visually and logically, validate the correctness of the business rules before they are put in execution. The Execute layer is supported by the possibility to logically validate the correctness of the business rules before they are put in operation, and to deploy and operate the executable business rules as a context frame meta-model. The Governance views are supported by the fact that the executable business rules can be handled as immutable artifacts. During the proof-of-concept the need for visual and logical validation became evident. The automation of rules creation and distributed execution of the rules became vital aspects during the analyze of the proof-of-concept. The proof-of-concept was implemented with the help of a machine learning pipe-line.

In Chapter 4 we validate the theoretical models found in Chapter 3 which are supporting continuous changes to business intents. Supporting continuous changes to business intents can be seen as providing business flexibility. Business flexibility is the possibility to create new business models or change existing ones, in an effective and efficient way. This includes changes to business rules which supports tactics or strategies. We take an advocacy/participatory worldview and use action research as the research methodology.

We decided to stay technology-independent, in order to achieve a conceptual business model background. One way to achieve this is to investigate legal contracts. A business model is supported by a set of legal contracts, we start to derive the business rules from these type of contracts. Some of the information in a legal contract is not meaningful to translate into a business rule which should be executed in software, e.g. which laws should be used to solve a dispute. Many times the nature of the language used in legal contracts requires human interpretation. However, the majority of the terms and conditions in a legal contract can be translated into meaningful business rules which could be implemented in software. The legal contracts we have chosen to study are targeting support for value propositions, based on different business models. The business rules are based on following five parts of the Osterwalder canvas [27]: customer type, customer relationship, channels, revenue streams, and a specific area of the value propositions.

In Chapter 4 we extend the design artifact in Chapter 3 in order to support legal contracts. There are small changes to the visual and logical verification of the business rules. We use another algorithm than used in the proof-of-concept (Chapter 5) in Chapter 3, since we would like to enforce legal rules instead of performing recommendations of product offerings. The analogy with the gaming industry is even more evident in this proof-of-concept since we change the (business) state in the enterprise when handling legal contracts.
Chapter 1. Overview

Since we are doing the implementation and it is not part of the product, we call it a proof-of-concept instead of action research. This proof-of-concept provides initial results regarding software architectural mechanisms which can support context descriptions and the context description’s support for business-driven software architecture, and the required level of speed and business flexibility demanded by the business ecosystem.

The knowledge-question-answering activity returns knowledge to the design-problem-solving activity. This can create new problems which generates an iteration over design problems and knowledge questions in design science. We start to answer knowledge questions which lead to a new design problem. We conduct a proof-of-concept which gives more insight to the problem at hand. The artifact which results from our design activities is returned to the question-answering activity which is used to create new design problems. The new design problems are described in the Section 1.4.3.

1.4 Conclusion and Further Research

In Section 1.4.1 we summarize the answers to the thesis research questions based on the conclusions from Chapter 2 - Chapter 4, we describes the produced design artifacts in Section 1.4.2, and in Section 1.4.3 we give suggestions on new design problems which can be included in our further research.

Before we continue we would like to remind the reader of the three phrases which were carved into the major temple in Delphi: “nothing in excess”, “know thyself”, and “make a pledge and mischief is nigh”. Currently we have studied the available state-of-the-art and there is indeed very little in excess available. But, our research is about getting new knowledge and to be able to share this knowledge in order to improve problem contexts. By knowing ourselves, and armed with the knowledge we have gained during our research, we are quite confident of being able to improving the problem context at hand. By only presenting a part of all possible design problems as our further research, we hope to have avoided the third phrase.

1.4.1 Conclusion

RQ1 is aimed at finding evidence in the literature for the support of intent-driven systems. RQ1 is divided into sub-questions, RQ1.1-RQ1.6, each covering one vital aspect of intent-driven systems. The results indicate existence of methods/techniques which can be used as building blocks to construct intent-driven
systems. However, the majority of the papers found during the study use an instance centric view of the problem they solve, which introduce a tight coupling between the realization and the information in each method/technique. In the proof-of-concepts we took a small step in the direction of separating the realization from the information, by using generic algorithms to produce executable business rules.

The aim of RQ2 is to find evidence in the literature of how the result of RQ1’s sub-questions could be used to enable flexible realization of intention driven systems. The synthesis show no evidence of how these methods/techniques can interact with the aspects needed to enabling flexible realizations of intent-driven systems. For example, the governance of intent-driven systems as well as possibilities for semantic interchange of information is not evidently available in the literature. In the proof-of-concepts we took smalls step in the direction of improving the semantic interchange aspect and the governance aspect. The implementation makes it possible to, visually and logically, validate the correctness of the business rules before they are put in production. The possibility to generate executable code representing the model of the business rules, makes it possible to execute the same model in different components without the need of re-implementation.

A continuous definition and execution of a business intent’s life cycle in an enterprise and its value networks, could not be found in the existing literature. Nor did we find a meta-model supporting a context frame aware realization of a business intent’s life cycle in a compositional way. RQ3 is addressed by suggesting a solution supporting continuous re-definition and execution of an enterprise as a model of value architecture layers and business functions, divided into the Defined and the Execute layers. This model mimics the reality of an enterprise. To support the model we build upon Pask’s conversation theory [30] by introducing the context frame. This makes it possible to support continuous definition and execution of business intents supporting the enterprise and its value networks. To the best of our knowledge this ability is not presented in the literature. With the help of the proof-of-concepts we have realized the models in a limited way. The main contributions of the models is the support the models give for the understanding of how to separate business models from the implementation of a business support system.

1.4.2 Produced design artifacts

We have produced two design artifacts as a start to be able to answer mRQ. The proof-of-concepts are seen as new problem solving knowledge by the stakeholder.
Chapter 1. Overview

The models in Chapter 3 have proved to be a valuable artifact for the stakeholder. The models in Chapter 3 are suggesting a solution supporting continuous redefinition and execution of an enterprise as a model of value architecture layers and business functions, describes the initial steps of intent-driven systems, and encapsulates the context frame meta-model.

In the next section we are suggesting new design problems to investigate, in order to get closer to the possibility of answering mRQ.

1.4.3 Suggested new design problems

The possibility of transforming business intents into executable business rules is vital in order to achieve an effective and efficient business. There is a need for a highly automative transformation process which will require effective and efficient algorithms for both for the translation of the legal contracts into business rules and the transformation of the business rules into executable business rules. To be able to find highly automative transformation processes we will continue to investigate how the business intents in legal contracts can be transformed into executable business rules.

Business rules can be evaluated and enforced as part of business processes. Many of the business processes in an enterprise can have a high level of automation. By encapsulate business processes in executable containers it can be possible to obtain an effective and efficient business by supporting dynamic, thin, and throwaway processes. In order to support encapsulation of business processes in executable containers, we will continue to investigate how context frames can be used as executable containers.

The transformation of business intents into executable business rules requires the possibilities to find needed capabilities, and to compose executable containers from these capabilities. Since there can exist a multitude of different stakeholder and their intents can be expressed in a none technical language, a fuzzy search mechanism might be needed. The methodology proposed in Chapter 6, and its suggested improvements, are small steps in the direction of an interactive fuzzy search mechanism. In order to be able to create compositions of capabilities, matching functionalities between the capabilities are needed. We will investigate how technical proposals to business intents can be exposed as compositions of context frames.

All the design problems described in this section share the need for governance. As mentioned before, the core idea of intent-driven systems is the possibility to change behavior of the system itself, based on stakeholder intents. This requires
Chapter 1. Overview

a separation of concerns between governance of the parts of the system used to execute the stakeholder business, and governance of the parts which are used to design the business based on stakeholder intents. Since the execute and the design parts are related there are interactions between the different governance structures. From the described design problems we can conclude, business intent, business rules, container capabilities, and execution of containers need their own governance structures which are interacting with each other. The described structure forms a compositional system of governance structures. We need to extend our research to cover the governance of business rules for both the design and the execute parts, as well as their interactions. We will use this research as a leap-stone for further studies.
Chapter 2

A Systematic Literature Review on Intent-Driven Systems

Abstract

Context: The aim of intent-driven systems is to capture stakeholders’ intents and transform these into a form that enables computer processing of the intents. Only then are different computer-based agents able to negotiate with each other on behalf of their respective stakeholders and their intents, and suggest a mutually beneficial agreement. This requires a separation of concerns between the parts of the system used to execute the stakeholder business, and the parts which are used to design the business based on stakeholder intents.

Objective: The aim is to find out which methods/techniques as well as enabling aspects, useful for an intent-driven system, that are covered by research literature.

Method: As a part of a design science study, a Systematic Literature Review is conducted.

Results: The existence of methods/techniques which can be used as building blocks to construct intent-driven systems exist in the literature. How these methods/techniques can interact with the aspects needed to enabling flexible realizations of intent-driven systems is not evident in the existing literature.
Conclusion: The synthesis shows a need for further research regarding semantic interchange of information, actor interaction in intent-driven systems, and the governance of intent-driven systems.
Chapter 3

Supporting Continuous Changes to Business Intents

Abstract

Context: Software supporting an enterprise’s business, also known as a business support system, needs to support the correlation of activities between actors as well as influence the activities based on knowledge about the value networks in which the enterprise acts. This requires the use of policies and rules to guide or enforce the execution of strategies or tactics within an enterprise as well as in collaborations between enterprises. With the help of policies and rules, an enterprise is able to capture an actor’s intent in its business support system, and act according to this intent on behalf of the actor. Since the value networks an enterprise is part of will change over time the business intents’ life cycle states might change. Achieving the changes in an effective and efficient way requires knowledge about the affected intents and the correlation between intents.

Objective: The aim of the study is to identify how a business support system can support continuous changes to business intents. The first step is to find a theoretical model which serves as a foundation for intent-driven systems.

Method: We conducted a case study using a focus group approach with employees from Ericsson. This case study was influenced by the spiral case study
Chapter 3. Supporting Continuous Changes to Business Intents

process.

Results: The study resulted in a model supporting continuous definition and execution of an enterprise. The model is divided into three layers; Define, Execute, and a common governance view layer. This makes it possible to support continuous definition and execution of business intents and to identify the actors needed to support the business intents’ life cycles. This model is supported by a meta-model for capturing information into viewpoints.

Conclusion: The research question is addressed by suggesting a solution supporting continuous definition and execution of an enterprise as a model of value architecture components and business functions. The results will affect how Ericsson will build the business studio for their next generation business support systems.
Chapter 4

Encouraging Business Flexibility by Improved Context Descriptions

Abstract

Business-driven software architectures are emerging and gaining importance for many industries. As software-intensive solutions continue to be more complex and operate in rapidly changing environments, there is a pressure for increased business flexibility realized by more efficient software architecture mechanisms to keep up with the necessary speed of change. We investigate how improved context descriptions could be implemented in software components, and support important software development practices like business modeling and requirement engineering. This paper proposes context descriptions as an architectural support for improving the connection between business flexibility and software components. We provide initial results regarding software architectural mechanisms which can support context descriptions as well as the context description’s support for business-driven software architecture, and the business flexibility demanded by the business ecosystems.
Chapter 5

Towards Executable Business Rules

Abstract

Context: In today’s implementations of business support systems, business rules are configured in different places of the system, and in different formats. This makes it hard to have a common view of what is defined, and to execute the same logic in different parts of systems. It is desired to have a common governance structure and a standardized way of handling the business rules.

Objective: To investigate if it is possible to support visual and logical verification of business rules and to generate executable business rules.

Method: Together with practitioners we conducted an experiment.

Results: We have implemented a machine learning pipe-line which supports visual and logical verification of business rules, and the generation of executable business rules. From a machine learning perspective, we have added the possibility for the ID3 algorithm to use continuous features.

Conclusion: The experiment shows that it is possible to support visual and logical verification of business rules, and to generate executable business rules with the help of a machine learning pipe-line.
Chapter 6

Uncovering Implicit Rules in Medicine Diagnosis

Abstract

Context: Decisions taken by experts may be based on explicit and implicit rules. By uncovering the implicit rules the expert may have the possibility to explain its decisions in a better way, both for itself and the person which the decision is affecting. In the area of medicine, laws are enforcing the expert to be able to explain its decision when a patient is complaining about a decision. Another vital aspect is the ability of the expert to explain to the patient why a certain decision is taken, and the risks associated with the decision.

Objective: To investigate if it is possible for a machine learning pipe-line to find implicit rules used by experts, when they decide if a patient could be operated or not.

Method: We conduct an analysis of a data set, containing information about patients and the decision if an operation should be performed or not.

Results: We have implemented a machine learning pipe-line which supports detection of implicit rules in a data set. The detection of the implicit rules are supported by an algorithm which implements an agglomerative merging of feature values. We have improved the original algorithm by showing the boarders of the feature values of a discretization bin.

Conclusion: The analysis of the data set shows it is possible to find implicit
rules used by the experts with the help of an agglomerative merging of feature values.
Bibliography


