Service-Oriented Software Engineering (SOSE) Framework

Harri Karhunen[1], Marko Jäntti[1], Anne Eerola[1]

[1] Department of Computer Science, University of Kuopio, Kuopio, Finland
[1] {harri.karhunen, marko.jantti, anne.eerola@uku.fi}

Abstract - **The gap between business decision making and software engineering causes inefficiency and quality problems in software development. Software engineers do not understand organization's value creation objectives and their influence on software production and structure. For this reason software does not fulfil the requirements of business and software quality is inadequate too often. Our objective in the service-oriented software engineering project (SOSE) is to develop methods and tools to improve quality and profitability of software development. In this paper we describe SOSE framework and clarify with examples its phases, utility, and application in pilot projects.**

SOSE framework's first activity is to create a well-defined business case. Then, business processes and data concepts are identified, to meet business requirements of the business case, and modelled with informal diagrams like UML and BPML. Finally, the refinement continues with use case maps, system-level services, and business service components. We propose that service, process, entity, and utility components are used as design elements of the business service component. In implementation we utilize platform independent and platform specific models. This study has been carried out in co-operation with ICT companies and their customers in electricity domain in Finland.

Keywords: **Business Case; Software Architecture; Service-Oriented Architecture; Enterprise Application Integration**

I. INTRODUCTION

Business decision-makers want adaptive, flexible, and cost-effective solutions. Software projects compete with other projects inside a company when investments are made, thus cost-benefit analysis is important when selecting development targets. The business domain and requirements of that domain need to be modelled in an appropriate way as well as existing systems. The first activity of any software project should be creating a well-defined business case that helps organization to justify the software project [21]. In previous studies a business case is used for various purposes; to support the software product line approach [6], to support exploiting open source software [14] and to reduce costs of performance failures [24]. Some researches have addressed that business decisions need to be combined with software development decisions [23].

Various stakeholders take part to the development project which has many phases guided in several methodologies [16], [12], [20]. A component-centric approach is based on the use of components working together [10]. Component-based development offers means to define platform independent component kit architecture with component connectors and a path from business goals to the program code [8]. Components offer through endpoints (for entry and exit) interfaces for external communication [5].

The Service-oriented architecture (SOA) extends the component-based development methods. It is an architectural style and design principle for building software applications which promote loose coupling between components. Services are components with published interfaces. The consumers can dynamically discover these interoperable service interfaces [19]. The goal-driven approach, for enterprise component identification and specification, is both component-based and service-oriented. It analyzes and structures the business model to the enterprise components [15]. Services offer a way to build system and process integration and cooperation inside an enterprise (Enterprise Application Integration, EAI) and between enterprises (Enterprise Service Architecture, ESA) [25], [22]. The service-oriented approach in a business chain between enterprises comprises an Enterprise Service Bus (ESB), which is a connector between request and response actions in business processes [1]. The architectural decisions are always a compromise, thus architectural evaluation and transformations are needed [2].

The former methodologies focus too much on existing information system implementation and introduction. The business domain and its requirements, i.e. the business goals, visions, and strategies, should be a starting point when software development decisions are made. In our research we emphasize software business and software engineering and improve former methods according to following goals: First, we construct a business-, service-, and component-oriented software design framework, which is advantageous to several heterogeneous design problems. Second, we develop a method, which facilitates moving from business requirements to business cases and further from these to implemented service- and component-based software. Third, we create a design model for distributed software, emphasizing communication and software integration with interoperability, and illustrate it with documented template solutions for stakeholders.

In this paper, we propose design of a software business case and derivation of software starting from that business case (see Fig. 1). The method is based on a top-down approach, where the business domain area is considered as one

problem domain. The work list consists of the following steps:

– divide the business domain into subsets and analyse business cases leading to business processes and concepts,

– move from business requirements to software requirements, which are described with business processes, use cases and service modelling, and

– design and implement business service components and distributed communication.

This research has been carried out in co-operation with ICT companies and their customers. As a concrete case we consider business case definition and service oriented software design at the electricity domain.

The rest of this paper is organized as follows. The rationale for building a business case and business case definitions is described in section II. The path from the business case to service-oriented software architecture is described in section III. In section IV we describe a case study, where business processes at the electricity market were utilized. Discussion and Conclusion section V summarises results and gives future plans to further validate our method in practice.

II. RATIONALE FOR A SOFTWARE BUSINESS CASE

This section clarifies the rationale for using a business case in software development by answering following questions: **why** is a business case useful for software development, **what** is a software business case, **when** is it established and by **whom**. Here we use both the results from a literature review and a case study that was performed in SOSE project. Previous studies in this research field support our objectives that building a business case should be a crucial part of any software project or an IT service development project.

We define a software business case as a collection of business related information that can be used to define critical stakeholders and their success criteria, the scope of the software product, a service or a project, market analysis and competitors, business opportunities, costs, benefits and risks.

Reasons for using a software business case (Why)

The first point of our rationale defines **why** the business case is useful for software development. We suggest that a software business case is an efficient tool for minimizing the gap [4] between software engineers and business decision makers. A business analysis is needed to create a bridge between technology-oriented software engineers and strategy-oriented business decision makers.

The relationship of a business case to the service management is described in IT Infrastructure Library ITIL standard. A business case for a new IT service helps organizations in IT-budgeting, accounting and charging services [17].

Because previous studies e.g. [21] give a strong support for using a business case in software development, we conducted a case study [13] to explore why and how three case companies use a business case in their software projects. Main empirical finding was that case companies used the business case to support sales activities and pricing decisions, to allocate resources between concurrent projects within the organization, and to identify interoperability issues between customer's product platform and products offered by the software company.

Definition for a software business case (What)

The second point of our rationale is to define **what** a software business case is. Building a business case for software projects is one possible way to increase the quality of system development. A business case typically includes partly the same features than business plans and feasibility studies.

There are a lot of differences in the contents of a software business case between organizations and project guides [18], [11]. For example, the recent study of Robertson [21] considers the SGS (Scope, Goals, Stakeholders) model as a useful basis for the business case. The business case based on the SGS model should describe investment's goals, business benefits, the stakeholders that act as a source of the requirements, the investigation's scope and expectations on the desired product's scope.The following list gives an overview of basic issues that a software business case should comprise:

- the scope of the software product or the project
- critical stakeholders and their business requirements
- project's resource requirements
- revenue projections and return-on-investment, break-even, and cost/benefit calculations
- expected levels of service from the new services, components or technology
- market analysis, competitors and risks
- business benefits of the solution
- evaluation of alternative suppliers
- estimation of software lifecycle
- the technical platform used by customers

Our case companies mentioned a couple of business case issues that were not emphasized in previous studies such as the evaluation of alternative suppliers in terms of reliability and estimation of software lifecycle (how long time software can be used). One of our case companies recorded information about customers' product platforms to the business case. They also stated that business benefits play an important role in the business case document. In two of our case companies the business case was well-integrated to the software development process and frequently used. One company stated that they use business case if they have enough time in their project.

The position of a software business case in software development process (When)

The third point of our rationale defines **when** a software

business case is established. Fig. 1 describes the role of the business case in a software development process.

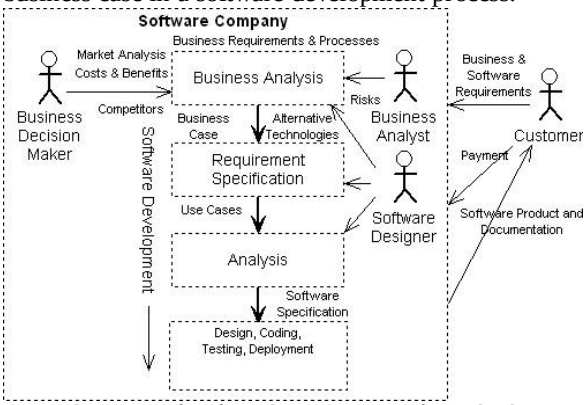

Fig. 1. The position of a software business case in software development process

Phase 1: A business analysis is the first step in establishment of requirements. A business analyst collects all relevant business requirements, software requirements, and possible descriptions of customer's business processes into a business case document. A business decision maker, for example, a production manager supplements the business case document with e.g. a market analysis, a cost & benefit analysis and information about direct and indirect competitors. A software designer can provide a business case with his/her view on alternative technology choices.

Phase 2: A business analyst and a software designer translate business requirements into software requirements using use cases of UML [11]. Later in software projects UML use cases are useful in identifying reusable functional modules for a new system, and deriving and organizing test cases. The business case should be an open document for all software project members and it should remind workers during the project of the things that create value for a customer.

Phase 3: In the analysis phase, use cases are analyzed in more detail including preconditions, post-conditions; basic, alternative and exceptional flows.

Phase 4: The development process continues with design, coding and testing. Finally, the software and documents are deployed to the customer including in most cases installation and training services. The customer checks whether the product or a project has met all required business requirements.

A business case is usually established before the requirement specification phase in the waterfall model and in the inception phase of the Rational Unified Process. In the Cooper's Stage-Gate process model, building a business case is done before a development stage and it is one of the most important stages [7]. According to Gilpert [9] a discovery phase is a key to the success of any software project. In other product development models or in software development lifecycle models the business case could be positioned in stages called exploration, preliminary analysis etc.

The target group of the software business case (Who)

The last point of our rationale defines persons **who** create and use the business case. Our answer is that a business case can be used by business decision makers, business analysts, project managers, IT service managers, software designers, product managers etc. This part of the rationale needs more research efforts, for example empirical studies investigating who are the persons in software industry that create or use business cases.

III. MOVING FROM BUSINESS CASE TO SOFTWARE ARCHITECTURE

In this section we describe service and component design, which is accomplished after the business case creation and which forms a traceable path from business requirements to software fulfilling those requirements (see Fig. 2).

First, business processes and use cases are specified using the *functionality line*. The service structure is derived and described as a system level service map. Second, the *information line* is followed. The business concepts are used when business entities are created. Both of these lines are accomplished intertwined. Third, in the final design the business entities and services are mapped to *business service components*. In refinement the communication, interfaces, and dependencies between components are defined.

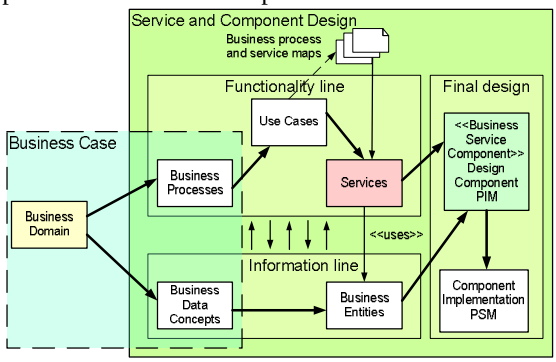

Fig. 2. SOSE Framework development process

Functionality line

The *service and component* design has a top-down perspective. First the system breakdown is made and the sub-systems are analyzed. As a mediating tool from business architecture to software architecture use cases and service oriented design are used.

The SOSE framework phases for service modelling are:

- The requirements of the business domain are described as business processes.
- Every business process is described as a use case, which acts like a business process and can have several sub use cases.
- Main business processes of the business domain constitute a *business process map* with system level granularity.
- The system level *service map* is derived from a busi-

ness process map. Each process in the business process map needs at least one service, which fulfils the requirements of the use case.
- Business level services utilize the information model of the company (Business Entities). The behaviour of each service is modelled more specific using scenarios.
- Services consist of a set of processes, sub-services or legacy systems.

The use case model depends on the business model and reflects the business vision, which changes during the time. For this reason, we need the current business model and the target business model. Software development is costly and needs a lot of work. For saving workload and expenses we need a balance between requirements of today and tomorrow.

Information line

Herzum et al. have proposed the Enterprise Object Model (EOM) as fundamental building blocks of business applications [10]. Correspondingly, we propose the business information model (BIM) to describe business data concepts with a long lasting nature. BIM has an influence to the whole software system and it is described as early as possible.

The SOSE framework phases for information line are:
- Business data and resource concepts are elicited and specified so that they support business services.
- Business entities are built according to business concepts and placed into business service components as building blocks. The coordination inside the entity component is designed, too.

Final design

The SOSE framework phases for final design are:
- Every service is modelled as a *business service component (BSC)*.
- A set of business service components constructs system level components needed in business processes.
- The work of BSC is accomplished using other components (classified as process, entity, and utility component), and proxies to legacy systems.
- Entity components are derived so that requirements of the business information model (BIM) are fulfilled, whereas process components support business processes.
- The platform independent model (PIM) is built using these components so that integrity of data remains and a needless replication is avoided.
- The platform specific model (PSM) is built according to PIM definitions.

The SOSE component model has three levels of granularities: *system level component (SLC)*, *business service component* (BSC), and *component level*. Each system level component is constructed from business service components.

The SOSE model and Herzum's [10] component model have three levels of the granularity. But the SOSE model is service-based whereas Herzum's model is component-based. The system level component of the SOSE framework is a set of services. It has only one network addressable service component. Thus, the system is more manageable because distributed objects are not used.

The business service component has 4-layer architecture [3]. Each tier is separated of each other and the communication occur using well defined interfaces. Components construct hierarchical layers, where components depend on another component forming a component dependency graph (for example, see Fig. 5):
- The upper most is the service layer with one service component (SC) offering an external interface to other participants in a bus. The service component uses other business service components, utility components, and legacy systems through adapters, and distributed systems using appropriate protocol.
- Second layer is the process layer consisting of process components.
- The next layer downwards is the entity layer with business entities and other resources like legacy systems' adapters referred as entity components, which are conducted by the service component.
- The lowest level is the utility layer with supporting components. For instance, user authentication, distributed transaction management components, and distributed service agents locate here. The BSC uses the coordinated interaction mode between layers. The coordination style is recommended, if persistence resources are provided by other services or resources outside this business component. It is possible that there is a persistence data provider, which offers information as a service used in many processes. The coordination gives simplicity and controllability.

The platform independent model (PIM) is transformed to platform specific model (PSM). The later this transformation is made the better portability is achieved. The SOSE PIM-model is a collection of design components (i.e. business service components), which are building blocks when the PSM-models are created. The goal of the SOSE is to work both as an abstract design framework and as an implementation tool when arranging services as code packages.

IV. EXPERIMENTAL RESULTS

Introduction

In this section we present experimental results got in our case study in the Electricity Market domain. First, business processes in electricity market were specified. Second, we specified main processes and system level use cases. We described the current system and designed the goal, i.e. the target system, according to the business goals and processes.

Third, we specified the business information model. The above phases were accomplished intertwined, but here we present the results separately to increase readability. Finally, the business service component was designed to fulfil required services and business entities.

Business Processes in the Electricity Market

Our example case is from power market. It is a description of electricity supplier business activities. The electricity supply business had a "closed market model" with no competition in the past. The company, who owned the electric wires, could operate in a monopoly position. There was no need to be open and have co-operative systems, but this all changed rapidly. The companies needed cooperation with other electricity operators, but software solutions were monolithic, hence one started to integrate software using one-to-one interfaces. This was costly and the software architecture looks now like a spaghetti with many dependencies between business software programs and systems.

Because of the closed market model the information systems are not co-operative. The business process planning using the existing information application context is very inflexible, expensive and has a lot of point-to-point integrations between applications.

Functionality line

In our case study the electricity business main process consisted of following sub-processes:

- Marketing: CRM, Sales, Invoicing, Metering
- Network Monitoring Process
- Network Service and Maintenance Process
- Network Planning Process
- Electricity Marketplace Brokering Process

The marketing is responsible on customers marketing activities (custom-sales-chain) like offer, order, order confirmation, invoicing. The Network Monitoring process monitors the state of a electricity local network and balances with other networks. Furthermore, it offers information about the electricity consumption (kWh) in the network, what is the state of the global electricity network, where and what kind of alarms exist in the network. The Network Service and Maintenance Process uses the information produced by the network monitoring process and offers the failure service. In the network planning process the network structure is designed, evaluated and stored in a digitalized form. This process offers e.g. graphical presentations of the network and makes calculations of the network load. The Electricity Marketplace Brokering Process buys and sells the electricity in the Nordic electricity exchange.

The next phase of the functionality line was to define a business process map, which describes the main business processes and their external dependencies and stakeholders and which was refined in service maps. Both current and target business use cases are taken into account when the service map is defined. For example, in Fig. 3 the upper diagram describes the existing situation of marketing and the lower diagram describes the strategic target and the services needed for customer service strongly simplified.

Next the services and their behaviour were studied further. The contracts between services were defined using collaboration diagrams. For instance, the connection contract use cases were mapped to a business service component. The behaviour of this component was described using a sequence diagram.

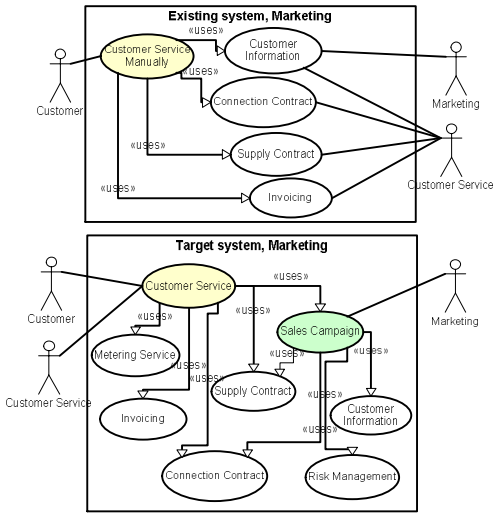

Fig. 3. Service map: Marketing of the existing and future system.

Information line

The business information model (BIM) consists of customer and contract data, the facility data, the network planning data, metering data of the electric network, and risk management data. The marketing system component uses legacy systems in order to get metering data and network planning data (see Fig. 4).

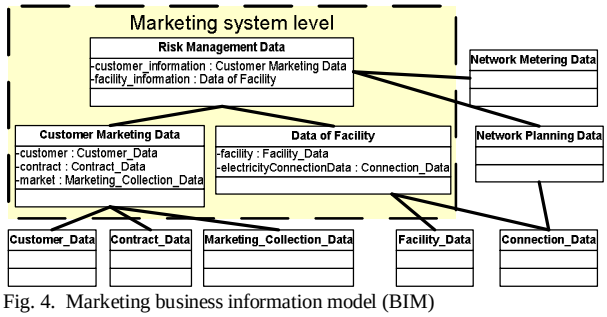

Fig. 4. Marketing business information model (BIM)

Final design

In defining the business service component (BSC) the offered interfaces of the BCS must be defined including contract information (pre/post conditions, prising, user rights, and the quality requirements of service). BSC offers interfaces for external communication. As an example, the connection contract business service component is presented in figure 5. The service component acts as a coordinator. It is possible that each process component has its own islands of

data. The persistence of data is guarantied by entity components which use appropriate resources like databases, other services, application interfaces, and legacy systems (see Fig. 5).

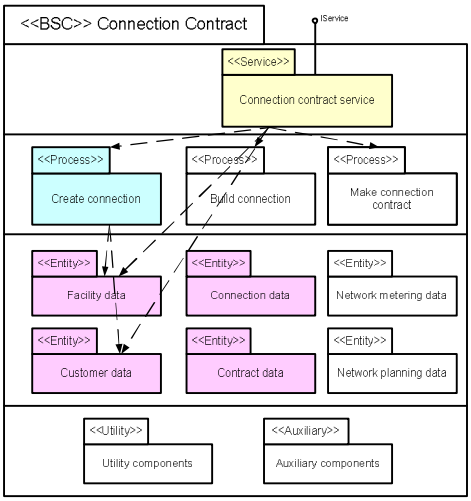

Fig. 5. Connection Contract Business Service Component (BSC).

V. DISCUSSION AND CONCLUSION

The Service-Oriented Software Engineering (SOSE) framework compromises different business requirements in order to make the software development and integration process easier. In this paper, we proposed that software projects should always start by creating a business case to justify the project implementation. The service and component design are mapped together in order to build the business services component fulfilling requirements specified in business case. However, the business process and business information separation offers more flexibility in defining business entities and processes.

SOSE is a mediator between the business and information system developers. It is informal enough with graphical descriptions for discussions and new ideas. It separates the actual, current information system implementation from the target system (business vision) offering tools to design a path to fulfil business visions. The development process is iterative and cycled.

Quality attributes of the customers, such as security, performance, scalability and throughput, must be compromised. Therefore, the architectural transformations and refinement are needed. It is easier to make improvements to early sketches than to implementation.

Using the PIM and PSM models the early design late binding goal realizes. Together with loose coupled component system of SOSE framework the business and system developers can achieve reusable, adaptive, and portable system development.

We are going to further validate and develop the method in practice during years 2005-2006 by building an open source implementation and studying how our principles meet the needs of the practical implementations in software industry.

ACKNOWLEDGMENT

This paper is based on research in the SOSE project (2004-2006), funded by the National Technology Agency TEKES, European Regional Development Fund (ERDF), TietoEnator Corp., Atro Oyj, and Softera Solutions Oy.

REFERENCES

[1] A. Bond, “ODSI: Enterprise Service Co-ordination,” DOA'01, Italy, 2001, pp. 0156-0164.
[2] J. Bosch , Design and Use of Software Architectures, Adopting and Evolving a Product-Line Approach. Addison-Wesley., 2000.
[3] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal, Pattern-Oriented Soft-ware Architecture: A System of Patterns. Wiley & Sons Ltd, 1996.
[4] B. Boehm and K.J. Sullivan, “Software Economics: A Roadmap, ” in Finkelstein, ed., The Future of Software Engineering, 2000 International Conference on Software Engineering, Limerick, Ireland, 2000.
[5] D. Chappell, Enterprise Service Bus. O'Reilly, 2004.
[6] S. Cohen, “ Case Study: Building and Communicating a Business Case for a DoD Product Line, ” Technical Note, CMU/SEI-2001-TN-020, Software Engineering Institute, 2001.
[7] Cooper R.G, Winning at New Products, 3rd ed., Perseus Publishing, Cambridge, Mass, 2001.
[8] D. D'Souza, A. Wills, Objects, components, and frameworks with UML: the catalysis approach. Addison-Wesley, 1998.
[9] S. Gilpert, “Wearing Two Hats: Analyst-Managers for Small Software Projects, ” IT Professional, July/August 2004, pp. 34-39.
[10] P. Herzum, O.Sims, Busines Component Factory. OMG Press, 2000.
[11] I. Jacobson, G. Booch, J. Rumbaugh, The Unified Software Development Process. Addison-Wesley, 1999.
[12] I. Jacobson, M. Christerson, P. Jonsson, G. Övergaard, Object-oriented Software Engineering. Addison-Wesley, 1998.
[13] M. Jäntti, H. Karhunen , A. Eerola, “The Rationale for Building a Business Case in Software Development Project, ” Proceedings of the 28th Information Systems Research Seminar in Scandinavia 2005, unpublished.
[14] C.A.Kenwood. (2001, July). A Business Case Study of Open Source Software. Technical Paper, MITRE, [Online]. Available: http://www.mitre.org/work/tech_papers/tech_papers_01/kenwood_software/
[15] K. Levi, A. Arsanjani, “A Goal-Driven Approach to Enterprise Component Identification and Specification,” Communication of the ACM, vol. 45, no.10, pp. 45-52, 2002.
[16] L. Maciaszek, B. Liong, Practical Software Engineering, a case Study Approach. Addison-Wesley, 2005.
[17] Office of Goverment Commerce, ITIL Service Delivery, The Stationary Office, UK, 2002.
[18] Office of Goverment Commerce, ITIL Infrastructure Manage-ment, The Stationary Office, UK, 2002.
[19] Panda, D. (2005). An Introduction to Service-Oriented Architecture from a Java Perspective [online]. Available: http://www.onjava.com/pub/a/ onjava/2005/01/26/soa-intro.html
[20] C. Raistrick, P. Francis, P., J. Wright, C. Carter, I. Wilkie, Model Driven Architecture with Executable UML. Cambridge University Press, 2004.
[21] S. Robertson, “Requirements and the Business Case,” IEEE Software, September/October, 2004, pp. 93-95.
[22] W. Ruh, F. Maginnis, W. Brown, Enterprise Application Integration. Wiley, 2001.
[23] C. Wallin, F. Ekdahl, S. Larsson, “Integrating Business and Software Development Models,” IEEE Software, November 2002, pp. 28-33.
[24] L.G. Williams, C.U. Smith (2003). Making the Business Case for Software Performance Engineering, Software Engineering Research and Performance Engineering Services [Online]. Available: http://www.perfeng.com/papers/buscase.pdf
[25] D. Woods, Enterprise Service Architecture. O' Reilly,2003.