REASONS FOR INTEGRATING SOFTWARE COMPONENT SPECIFICATIONS IN BUSINESS PROCESS MODELS

Marie-Therese Christiansson
Division for Information Technology, Information Analysis,
651 88 Karlstad, Sweden
Email: Marie-Therese.christiansson@kau.se

Benneth Christiansson
Division for Information Technology, Information Analysis,
651 88 Karlstad, Sweden
Email: Benneth.Christiansson@kau.se

Keywords: Software Component Specification, Business Process Modelling, Acquisition

Abstract: Organisations, business processes and co-workers are in a ”never-ending change-mode”. It is therefore unrealistic to expect any definitive requirements for computer based information systems. In this paper we argue for the need of bridging the gap between business process modelling and software component specification. By using a core business process model that integrates both essential knowledge concerning business processes and their possible improvements and also integrates software component requirements in the form of software component specifications; IS professionals should be able to judge the potential, development and management of component-based information systems. This implies the need of an “informal” software component specification that is grounded in business processes and created for the people who are best suited to model requirements, i.e. people who run and perform business. This “close to business” specification can be expressed on a high-level and in an informal manner due to the fact that the specification does not have to serve as software development requirements because the software component already exists, the difficulty is acquiring, identification. With an integrated core business process model we have the possibility to perform modelling more effective and achieve more benefits and also use it as a foundation for software component acquisition. We can focus on business actions and their constant changes and at the same time identify the corresponding changes in software requirements.

1 INTRODUCTION

Well known phenomena in information systems development is the communication gap between business modelling professionals and IT professionals. We believe business models should be used for the understanding and change of organizations and as requirements on the software system. Nellborn (1999, p. 199) describe this work as the “Berlin wall” dilemma; *Both sides making assumptions about the knowledge, needs and context of the other side.”* Langefors (1993, p. 142) describes the development of software systems as finding the solutions to: *”Two fundamental problems with information systems were pinpointed at the outset: (1) The "infological” problem of how to define the information to be made available to the information system user, and how to design data that may represent the information to the user; and (2) The ”datalogical” problem of how to organize the set of data and the hardware so as to implement the information system.”*. We believe that emphasis during software development needs to be on both these problem areas. In today’s software engineering community the focus is on the “datalogical” side. We believe that component-based systems development delivers an opportunity to focus more on the “infological” part of software systems development. This is due to the fact that the software already is developed. Software components already exist, we do not need to express requirement in such details that we can construct the software based on them, we only need enough details to enable acquisition. This situation is more similar to acquisition of application packages rather than traditional software development. We argue for a software component specification in the 'infological' side. This specification should be grounded in business processes that captures the "what, why and for whom" the system is developed. We also believe this approach to be an informal strategy that captures requirements based on business practice and describe them in such a way that people, who are best suited to model requirements, find them usable. Our approach for specifying software components is based on the integration of results from two research fields; 1) Component Based Software Engineering and 2) Business Process Modelling.

We believe it is possible to short-circuit the extensive and time consuming modelling in traditional software development process when using software components. Langefors (1993, p. 73) describe the system development process; as starts by working out a requirement specification of goals to be satisfied by the system. *"Thereupon one […] will look for a perceptibly small enough structure of subsystems having such external properties and interconnections, as would generate the specified external properties of the whole system. If these subsystems already exist, on the market for instance, the design is finished."* As software components are already constructed, the problem of software development and to formally break down requirements into more and more detailed software specifications is solved, however we have a partly new problem with acquisition of the existing components (Christiansson, 2000).

We have chosen to describe our research effort as: *"The meeting between business processes and software components – in a well founded specification."* This approach is focused mainly towards capturing requirements of COTS (Components Off The Shelf) software components, and creating a specification close to business that can be used for acquisition of software components. Our opinion is that we will never be able to tear down "the wall" between professionals and different stakeholders in business- and systems development if we don't use integrated models. A study of 20 approaches in software component specifications shows that the main focus in software component specification is towards the 'datalogical' side mainly concerned with construction and somewhat regarding assembly (Christiansson & Christiansson, 2003 b). This does not imply any "integration" effort. We argue for the need of specification strategies that takes advantage of the new possibilities and handles some of the challenges such as enabling acquisition of software components.

2 PROBLEMS WITH COMPONENT-BASED SOFTWARE DEVELOPMENT

Component-based software development should mean that software systems are created through the assembly of more or less standardized software components into unique software solutions. *"Although each bought component is a standardized product, with all the attached advantages, the process of component assembly allows the opportunity for significant customization."* (Szyperski, 2002, p. 6). A key reason for the interest in component-based software development is the possibility to reuse. In many approaches, reusability is not inherent in the development process. What can be reused and what cannot be reused is not precisely defined, and how the changes can be introduced in the reusable parts is not formalized. The basic idea is that software systems are built by assembling components already developed and prepared for integration. This approach have several promises or possible advantages like more effective management of complexity, reduced time to market, increased productivity, improved quality, a greater degree of consistency, and a wider range of usability (Crncovic & Larsson, 2002; Casanve, 1995). However, this strategy has several problems that need to be addressed. One problem is time and effort required for development of components. Reusability requires generality this requires increased time and effort to develop this also implies that to be widely reusable, a component must be sufficiently general, scalable, and adaptable; it will therefore be more complex (and thus more complicated to use) (Steel, 1996; Basili & Caldiera, 1991). There are also problems with the nature of software development *"...developing software is essentially a creative activity. There is no single right solution, and consequently each individual architect or team naturally devises a solution that is structurally different from any other solution."* (Whitehead, 2002, p. 8)

Another problem is unclear and ambiguous requirements. In general, requirements management is an important and complex phase in the development process, its main objective being to define consistent and complete component requirements. One of the major problems of software development in general comes from unclear, ambiguous, incomplete, and insufficient requirement specifications (Pressman, 1997; Beck, 2000; Christiansson, 2000; Heineman & Councill, 2001). Reusable components are, by definition, to be used in different applications, some of which may yet be unknown and the requirements of which cannot be predicted. This applies to both functional and non-functional requirements. This makes it more difficult to identify the requirements properly and hence to design and build components successfully (Sommerville, 2001; Pfleeger, 2001). Yet another problem is component maintenance costs. Although application maintenance costs can be lowered, component maintenance costs can be very high since the component must be able to respond to the different requirements of different applications running in different environment (Steel, 1996). One approach that eliminates many of the above described problems is ***acquiring application packages***. This strategy implies the purchase of a standardized complete solution that can be slightly adjusted through the use of parameters for some aspects of the software. The strategy is finding a standardized solution that is 'close enough' to what is required. The burden of development, maintenance and product evolution is left to the vendor of the application package and as Szyperski (2002, p. 5) states *"To cover the time-to-market risk, there is a strong trend toward using standard software – that is, software that is only slightly adjusted to actual needs."* But there are a number of problems with the acquisition of application packages, when acquiring software that only to some part cover the actual business processes, the organization needs to change their business processes according to the software instead of the opposite (Szyperski, 2002; Christiansson, 2000). Application packages are standard software; this means that the system in itself can not give any competitive edge (Christiansson, 2000). Whitehead (2002, p. 7) also points out that application packages can overlap with in-house systems and create a problem regarding from which system the functionality should be used. *"The functionality of a package will inevitably overlap with a company's in-house applications. And where more than one package is bought, these will overlap."* Another problem is cultural differences within organizations. Whitehead also (2002, p. 8) points out the fact that people like to keep doing things the way they already are *"...software is used to support processes that diverge naturally because different organizations simply achieve a similar external result in a different way. People are likely to protect the divergences not just where they give genuine competitive advantage but also because people simply prefer to do things their own way."* Finally there is a problem with vendor dependency. When acquiring application packages a dependency towards the vendor is created regarding, support, configuration, maintenance and product evolution (Christiansson, 2000). An additional problem is identifying the application package to acquire. How do we express requirements? Where do we find requirements? How do we know if an application package fulfils requirements? These are all problems that occur when acquisition is at hand. These problems also apply to acquisition of software components. In our research we want to integrate essential knowledge about business processes and requirements on an 'infological' level of software components to use in an acquisition of already existing software components. By using an integrated core business process model we enable the possibility to have a source of knowledge to be used when:

– Software systems need to be integrated or at least connected with each other to enable structural changes within business practices.
– Software systems overlap with in-house systems and create a problem regarding from which system the functionality should be used.
– We need accurate, correct and usable documentation other than the source-code in itself.
– Predicting the feasibility of the final implementation.

3 A SOFTWARE COMPONENT SPECIFICATION - CLOSE TO BUSINESS

Today, the need for well functioning and economically justifiable information systems is crystal clear. The quality of a company's information systems has become recognized as a strategic corporate advantage. Information systems and information technology can be viewed as the backbone of the modern enterprise and as such crucial to its supporting and providing new possibilities of running business operations. Information technology should, like Hammer (1990) states, be used as an enabler to perform innovations which makes a difference to business customers/clients. Business modelling is the work of reconstructing and describing how business operations have been run, are being run, may or should be run to achieve a basis for observing and comprehending business operations (Tolis & Nilsson, 1996).

We need an approach to specify requirements on software components to identify actual needs. Communications patterns need to be clarified assessed and developed, to achieve efficient business operations, internally and in co-operation with others. Business actions and their conditions and requirements of co-workers, knowledge, machinery; flows of information, materials and payments; must be distributed and coordinated. Part of this work is to establish the roles and responsibilities needed to perform the task, and the necessary resources (Christiansson M-T., 2001). Another part is to determine if the tasks should be supported or run by a software system, to efficiently provide people with information that is to be used in performance of operational actions (Langefors, 1973). A boundary-crossing quality is one of several features inherent in processes. Other features are that processes should (Davenport, 1993):

– have a beginning and an end;
– consist of structured, coherent activities, that are operated by actors which may be humans or machines, and consume resources of different kinds;
– contribute to fulfil the business goals; and
– help to create value for external customers/clients.

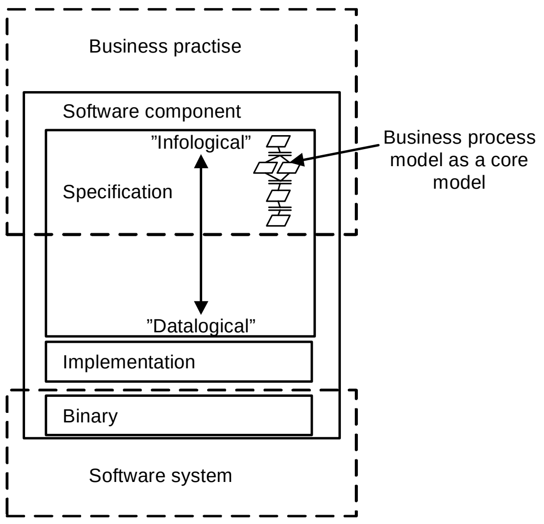

Figure 1: A software component specification in an integrated core model

To use a process oriented perspective in analysis may help us to capture essential knowledge of business practices to be used in system development. Business models are founded on two basic needs, namely *"… trying to understand and trying to change"* business operations in organizations (Tolis & Nilsson, 1996, p. 10). Understanding and changing organizations signifies that the starting point should be what is actually carried out in these. Taylor (1993, p. 112) underlines this as follows:*"… communication (and therefore organization) is grounded in action, not in information transmission, nor even in the transfer of knowledge."* In other words, the focus in system development should be on the way we carry out tasks and on the communication pursued as prerequisites for, and results of, these activities. Business models aim to communicate different phenomena in organisations, i.e. to identify and bring up matters for discussion. The purpose is not to describe the 'truth', as this is a highly subjective quantity, but rather to provide a basis for discussions, by which people can communicate differing viewpoints to arrive at an inter-subjective view. An obvious strategy would be to choose business models that correspond to the motives of business process mapping, reflecting the desired knowledge about organisations. Cheung and Bal (1998, p. 274) states that *"a methodology for business improvement is really only as good as the tools and techniques that support it"*. All kinds of process models, software based or paper based are aimed to support business analysis and to enable improvements. This means in very short terms, what you [are able to] see is what you get. Perspectives that a process model is able to present are bounded by techniques of representation, i.e. the language (terms) used in modeling and its syntax and semantic (Biazzo, 2002). A "software component specification – close to business" needs a process framework to indicate which essential matters of business operations and software components to describe in an integrated core model (Christiansson & Christiansson, 2003a), figure 1

Our hypothesis is that well functioning component-based information systems should be founded in business processes. Breton & Bézivin (2000) agree on an approach to share few sets of basic concepts between heterogeneous models to integrate a product model (i.e. UML, CORBA, Java) with a process model to show which software component will perform an automated task, or to analyze the data flow through a process.

4 A SOFTWARE COMPONENT SPECIFICATION - CLOSE TO BUSINESS CHANGE

The software industry is challenged by the new claims made by its customers. According to Peisl (2002, p. 2) there are many reasons for these new claims *"To be successful in today 's rapidly changing market, your business requires a solution that allows you to integrate your business processes across different vertical markets and various operating environments. Effective business integration enables disparate business resources —both inside and outside an enterprise —to work together to support a company's business strategy. A business process management (BPM) solution is the collection of instruments with which you can model your enterprise business processes, perform activity-based costing, simulate the processes and deploy them."*

In today's global market, organizations have to continuously adjust and improve their business practices to maintain a competitive edge. This conveys that the demands and requirements on the organizations software systems change at the same rate. The role of software in information systems is of vital importance. Bosch (2000, p. 3) states that *"…organizations increasingly often distinguish themselves in their ability to exploit the possibilities facilitated by software rather than on the basis of other factors."* This leads to two things 1) software is increasing in size and complexity and 2) the software systems in themselves can define the competitive edge for companies.

The challenge is to increase the productivity of software system development and to augment the flexibility of software systems to react to business process changes. One approach to achieve this is taking clues from traditional production techniques. Software systems should be constructed from prefabricated, easily identifiable software components (Szyperski, 2002, Christiansson, 2000 that can be widely used. To develop software systems with a component-based approach is one of the newer trends within the software development community. However for component-based software development to be successful in organizations, the software developers must give close attention to the design of components as independent abstractions with well-specified behaviours. Without well-specified behaviours the possibility to distribute and acquire software components will be limited (Christiansson, 2000).

Szyperski (2002) presents software component-based development as a middle path between ERP-systems acquisition and traditional tailoring of software systems. We regard component-based systems development as being closer to the field of acquiring application packages rather than tailoring of software systems. Vigder et al. (1996 , p. 13) claims that *"… in order to realize the benefits of COTS software a procurement process must be in place that defines requirements according to what is available in the marketplace, and that is flexible enough to accept COTS solutions when they are proposed."*. These indicate the fact that the modelling should be more general than reflected. In this case we believe in a middle path that we call a focused modelling Christiansson & Christiansson, 2003b). Problems with application packages such as software that only to some part cover the actual business processes or the need to adjust business processes in accordance to the software instead of adjusting software according to business can be avoided. With a more directed and close to business specification, a priority in modelling and acquisition is possible to create a business edge trough acquisition of "smaller" software components to be composed "one step at a time".

With an established motive with modelling namely software component acquisition, you don't want to or need to describe all possible variants of business processes at present ("is-processes"). Focus can be on desired business processes with the software components as enablers ("should-processes"). Like Hammer (1990, p.104) states; *"use computers to redesign - not just automate existing processes... companies tend to use technology to mechanize old ways of doing business. They leave the existing processes intact and use computers simply to speed them up."* The purpose is not to explain all kinds of performance in a business process, rather to describe desired business processes in more focused terms of "to what result and for whom". If we can use the same business process model to capture the business situation at the present time (the "Is"-process) and at the same time use the model to capture requirements for the desired business situation (the "Should"-process) we can get to the systems development faster and produce fewer documents along the way, This will in the end lead to an increased possibility to reach the "should"-process description at all, which in today's business modelling is far from always the situation.

Our approach does not rest on the notion that it is possible to define and describe all system requirements and software features beforehand, or at least very early in the development process. (Christiansson & Christiansson, 2003b). We believe that software requirements should be defined in the description of the actual business actions it is intended to support. Thereby we enable the possibility to cope with the problems of missing and/or inaccurate requirements as well as coping with constant business change. By using one core model changes can be updated and tested "on a paper based process" without the time consuming job of updating several related documents. Beck (2000, p. 3) illustrates this problem as *"Business misunderstood – the software is put into production, but it doesn't solve the business problem that was originally posed. Business changes – the software is put into production, but the business problem it was designed to solve was replaced six months ago by another, more pressing, business problem."*

By using an integrated core business process model we enable the possibility to:

- Determine if actions should be supported or run by a software system, to enable desired ways of doing business and performing business actions.
- Handle that people like doing things the same "old" ways by visualizing and comparing the "Is-processes" with the "Should-processes", i.e. improve business processes supported by software system.
- Handle the rapid changes in business practices that results in new and changed requirements on software, as well as further reorganizations of business processes.
- Eliminate the notion that it is possible to define and describe all system requirements and software features beforehand by an ongoing business process

improvement which handles new or changed requirements.

5 A SOFTWARE COMPONENT SPECIFICATION - CLOSE TO PEOPLE

Our opinion is that we need to delimit the information amount in a software requirement specification to bare essentials with a pragmatically language in an integrated core model which different stakeholders can relate to. *”The various stakeholders ought to be allowed to specify desires or requirements in a language that they could understand… this language ought to be efficient for their talking about their own reality.”* (Langefors, 1993, p. 71). We do agree on the fact that it is important to let the people who know the business state their requirements in their own business language. This may though lead to communication problems as Nellborn (1999, p. 201) states it*; “…business development and information systems development are seen as separate issues performed by different people and not as two components of the same solution to a problem.”* One recent approach of bridging this gap is using so called business-process management systems. *“The business-process management system (BPMS) is a framework for building adaptable processes”* (Smith & Fingar, 2002, p. 2). This approach has in our view advantages and is certainly a step in the right direction. But we believe some of the efforts as being misguided. *“To make BPM a reality, its underlying business-process language needs to be rich enough to describe the process of hosting a dinner party, yet also precise enough to describe how computer system “A” talks to computer system “B”…”* (ibid. p. 2) This indicates a complex language that handles informal requirements on a high level of abstraction to very formal actions such as computer transactions between systems. This is the major drawback with this approach. We do not believe it is necessary or even possible to grasp everything in one model (language). Capturing “everything” will lead to large models with too many details; this will lead to document overload and the need of atomization. We do not regard this as the right way to go when capturing software requirements. Smith & Fingar (2002, p. 3) also states that *“…creating a new process is no different from putting a new customer record in a database”* and in this sense they also claim that *“…business people should be able to implement changes to live business processes, meaning that the lifecycle for the process design and modification needs to be where the process is used, not in the data centre”* In essence they claim that the business people themselves should be the ones developing the systems. And if it was just as easy as putting a new customer record in a database, to create a system this would be a great approach. The problem is that putting a customer record in a database is not only about static data. We need functionality that enables the actual “putting”. This functionality is software and this software needs to be created and understood and maintained somewhere and by someone. We do not believe that business people are the ones to do this. We believe business people want to do business not software.

A rewarding solution is to be able to capture requirements based on the desired business processes expressed by people who run and perform business practice in a core model. A more focused modelling might give more value in fewer models. This assumption is based on the nature of development processes. A system development is dependent on knowledge from different stakeholders who want to spend as little time as possible in a useful manner. Christiansson & Christiansson, 2003a)

By using an integrated core business process model we enable the possibility to:

- Ease the difficulty to state and to communicate requirements by found those in daily practice.
- Handle the natural uncertainty and lack of understanding regarding software requirements by found those in daily practice.

6 CONCLUSIONS

In the increasingly competitive software industry the need for new innovative techniques to deliver satisfying software systems is greater than ever. This may in some sense explain the great belief and adaptation of new techniques, strategies and/or tools that are made out to be the great solution to the different problems that the software industry is facing. We believe that by combining business process modelling with software component specifications, we have a solution that addresses and handle many fundamental problems with software development. This core model can be used as an enabler of acquisition. The barriers between business analysis and systems development can be mended by embedding focus on software components within the business process model. By using a business model that integrates essential knowledge concerning actions in business processes and software components ability to support, gives new innovative ways to perform business actions, and new ways to handle management of component-based information systems. With our approach we avoid treating business development and information systems development as separate issues, and instead regard them as one integrated task. We believe that using an integrated core model we can capture essential knowledge concerning software component requirements based on the desired business processes expressed by people who run and perform business practice. We can also facilitate the fact that software requirements change at the same time as the business change and a core business process model can be used without the time consuming job to update a lot of related documents.

REFERENCES

Basili, V. R. Caldiera, G. (1991) Identifying and Qualifying Reusable Software Components, IEEE Computer, Vol. 24, Issue 2, pp. 61-70.

Beck, K. (2000) Extreme Programming Explained, Addison Wesley Longman, Inc., California, Menlo Park, USA.

Biazzo, S. (2002) Process mapping techniques and organizational analysis – Lessons from sociotechnical system theory, Business Process Management Journal, Vol. 8 No. 1, pp. 42-52.

Bosch, J. (2000) Design & Use of Software Architectures Adopting and evolving a product-line approach, Addison Wesley Longman, Inc., California, Menlo Park, USA

Breton, E. Bézivin, J. (2000) An Overview of Industrial Process Meta-Models, in proceedings of ICSSEA ‘2000, 13th International Conference on Software & Systems Engineering and their Applications, Paris, December 5-8.

Casanve, C. (1995) Business Object Architectures and Standards, Proc. Data Access Corporation, Miami.

Cheung, Y. Bal, J. (1998) Process analysis techniques and tools for business improvements, Business Process Management Journal, Vol.4 No. 4, pp. 274-90.

Christiansson, B. (2000) Att komponentbasera informationssystem - Vad säger teori och praktik?, Licentiatavhandling, Institutionen för datavetenskap, Linköpings universitet.

Christiansson, M-T. (2001) Business modeling in inter-organisational cooperation – what to describe, why and how? in Nilsson, A. G. & Pettersson, J. S. (Eds.) On Methods for Systems Development in Professional Organisations – The Karlstad University Approach to Information Systems and its Role in Society, Studentlitteratur, Lund.

Christiansson, M-T. Christiansson, B. (2003 a) Focused and Reflective Modeling – State of the Art (submitted paper)

Christiansson, B. & Christiansson, M-T. (2003 b) The Missing Approach for Component Specification, in Assar, S., Semmak, F. & Barkhi, R. (Eds.) Proceedings of 1:st International Workshop on Component-Based Business Information Systems Engineering (CBBISE'03), 2 September, Geneva.

Crnkovic, I. Larsson, M. (2002) Introduction, In: Crnkovic, I. & Larsson, M. (eds.) Building Reliable Component-Based Software Systems, Artech House, Boston.

Hammer, M. (1990) “Reengineering Work: Don’t Automate, Obliterate”, Harvard Business Review, Vol. 59, No. 4, pp.104-112.

Heineman, G. T. Councill, W. T. (2001) Component-Based Software Engineering Putting the Pieces Together, Addison Wesley Longman, Inc., California, Menlo Park, USA.

Langefors, B. (1973) Theoretical Analysis of Information Systems, 4:th ed., Studentlitteratur, Lund.

Langefors, B. (1993) Essays on Infology – Summing up and Planning for the Future, Gothenburg Studies in Information Systems, Report 5, University of Göteborg.

Peisl, R. (2002) Delivering business agility and service level management with business process integration, WebSphere Business Integration Marketing IBM Software Group, USA.

Pfleeger, S. L. (2001) Software Engineering, Theory and Practice, Upper Saddle River, NJ; Prentice Hall.

Pressman, R. S. (1997) Software Engineering a Practitioner’s Approach, McGraw-Hill Companies, Inc. New York, USA.

Smith, H. & Fingar P. (2002) A New Path To Business Process Management. Optimize Magazine, CMP Media LLC, USA, October 2002, Issue 12.

Sommerville, I. (2001) Software Engineering 6th edition, Addison Wesley Longman, Inc., California, Menlo Park, USA.

Steel, J. (1996) Component Technology Part I An IDC White Paper, Proc. Int. Data Corporation, London.

Szyperski, C. (2002) Component Software Beyond Object-Oriented Programming, Addison Wesley Longman, Inc., California, Menlo Park, USA.

Taylor, J. R. (1993) Rethinking the Theory of Organizational Communication: How to Read an Organization, Ablex Publishing, Norwood, New Jersey.

Tolis, C. Nilsson, A. G. (1996) Using Business Models in Process Orientation, in: Lundeberg, M. & Sundgren B. (eds.). Advancing Your Business – People and Information Systems in Concert, Studentlitteratur, Lund.

Vigder M. R, Gentleman W. M. & Dean J. (1996). COTS Software Integration: State of the art. National Research Council of Canada., Toronto, Canada.

Whitehead, K. (2002) Component-based Development Principles and Planning for Businness Systems, Addison Wesley Longman, Inc., California, Menlo Park, USA.