Proceedings of the International Workshop on COTS Terminology and Categories: Can We Reach a Consensus?
1-4 February, Rendondo Beach, California, USA

An informal COTS Specification Model –

Enabling Component Acquisition.

Benneth Christiansson and Marie-Therese Christiansson
Information Technology Department
Karlstad University
Sweden

Introduction

In today's global market, the need for well functioning and robust information systems is greater than ever. Organizations also have to continuously adjust and improve their business practices to maintain a competitive edge at the same rate the information systems have to be adapted in accordance with these changes. This is a big challenge for the software development community, and has been a big issue in the software engineering field for at least two decades. 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, 2001) 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 behaviors. Without well-specified behaviors the possibility to distribute and acquire software components will be limited. Langefors (1995, 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 take our point of departure in this description. We believe that emphasis during software development needs to be on both these problem areas also regarding component-based software development or even more so regarding component-based development. We believe this is accurate due to two facts firstly software component development should be focused on assembly rather than about construction. This means that we do not have to focus on how the actual development is done; the software component is an existing artifact. Secondly software component development is about acquisition, we need to be able to identify which components we need when assembling systems; this conveys the need for a specification of the components behavior (Szyperski, 2002, Heineman & Councill, 2001). This acquisition we believe needs to be based on an 'infological' specification as well as a "datalogical" one, using Langefors (1995) terms. In this paper we describe an informal "infological" model to specify software components used by the largest software component distributor Componentsource (2003). We argue for the need of an informal approach for addressing the 'infological' problem of software development that enables acquisition rather than construction. We have previously shown (Christiansson & Christiansson, 2003) that such approaches are missing in the research community. In this paper we focus on an existing solution created in practice. We believe an optimal approach probably can be found somewhere in the middle between "unreflected" practitioners problem solving ad-hoc solutions and "reflected" theorists "unused" efforts.

We have chosen to describe our research effort as: "The meeting between business processes and software components – in a well founded specification." Our approach is focused mainly towards the process of capturing software requirements, and the creation of a specification model that can be used as a tool in the acquisition phase of component-based systems development. Approaches towards the assembly phase already exist and is a major area of research in the component-based software engineering community (Christiansson & Christiansson, 2003). 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 to be possible to integrate 1) business process models; 2) the capturing of software requirements in an informal manner and; 3) software component specifications into one type of document. This document can be used as a tool for software component acquisition. Acquisition and assembly is based on the knowledge of what is needed to acquire and assemble and also on what is possible to acquire. 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.*". What is needed is an approach to specify requirements on software components to identify actual needs. To be able to capture requirements based on the desired business processes expressed by people

who run and perform business practise. This is not the reality of today Beck (2000, p. 3) illustrates this 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."*

We believe requirements should be captured close to the business practice to cope with the problem of missing and/or inaccurate requirements as well as constant business change, see figure 1. The figure illustrates the connection between business practice and software system as being software components. A software component's "infological" part exists in the business practice and the "datalogical" part exists in the software system. The figure also shows our approach as a part of the components "infological" side.

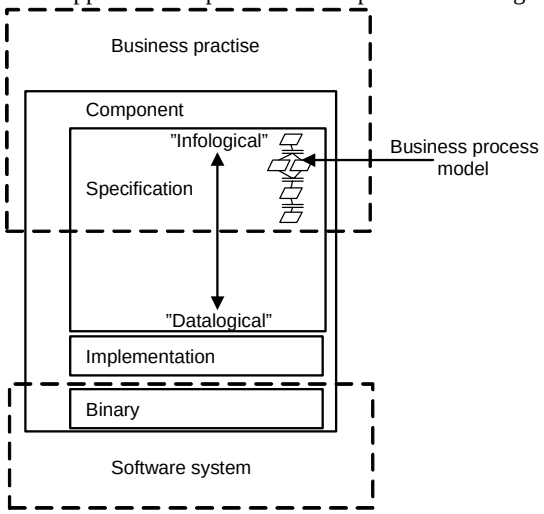

Figure 1. Our approach to a well founded component specification (Christiansson & Christiansson, 2003).

Componentsource Component Specification Model

We have chosen to study one of the largest marketplaces for COTS acquisition Componentsource (2003) this is an on-line marketplace for component acquisition. Componentsource was founded in 1995 under the parole "Making Reuse a Reality". And according to their own information (Componentsourcee, 2003) the "*Marketplace of choice for technical decision makers everywhere - home to the world's largest source of software components for all platforms (9,000 components and expanding)"*. In our experience component acquisition is oftentimes hindered due to lack of tools to aid identification of components. There are no standard strategies for identifying components, or any standard language for expressing requirements on components. *"A standard approach to building and using components must be set and universally practiced if the software engineering community is to reap the benefits…of reusable software components…*" (Vaughn, 1990, p. 1). Since Vaughn expressed this need things have happened, but mainly regarding the building part or in Langefors (1993) terms the "datalogical" side. Regarding component usage there are still no clear standards and we have in another study (Christiansson & Christiansson, 2003) shown that this is a low priority area in the component engineering research community. Componentsource however have a big clientele and offers over 9000 components in their repository. This indicates the need for a structured way of presenting these components and it also indicates that they developed successful tools to aid customers and vendors in enabling component acquisition. By studying their offer procedure we have reconstructed their component specification model. To identify the underlying generic structure used at Componentsource we have identified the "find a component" process and documented it. The next step is to analyze the information presented and needed throughout the process and by describing it we have described the Componentsource Component Specification Model.

Find a component process

The "find a component" process starts out with a free-text search based on keywords that describe the components functionality on a very high abstraction level (figure 2). These keywords are not regulated in any way. The only directions suggested is that the keywords used for component identification can be product name or description of the service the component provides e.g. the keywords "credit card verification" implies a search for components from any vendor with any name that performs a credit card verification service. After these initial informal search criteria filtering of the repository the result can be refined by applying different additional filters. These filters are based on platform decisions e.g. .NET, DLL, ActiveX/COM etc. The possibility to choose which platform implementation that is desired is available. The amount of components to choose from is

now narrowed to be vendor-specific. The next step is to choose one specific component and filter it based on editions and versions of releases. At this last stage of the "find" process the component is presented with the following information to enable decision regarding acquisition:

- a list of features presented in natural language in an informal manner called overview
- pricing and licensing information regarding actual costs and legal details regarding usage and responsibilities
- evaluations and downloads if an evaluation version of the component exists it can be downloaded from here
- compatibility information regarding possible operating systems, architecture of product, component type (eg. COM, javabeans, ActiveX etc.), creation tools (e.g. Visual Basic, C++, Java, etc.), glue-code compatibility, product class (the vendors classification of the component e.g. Business component, SQL component etc.), test and review status and prerequisites for component usage
- publisher details containing information about the vendor
- customer reviews if available
- support forum if available

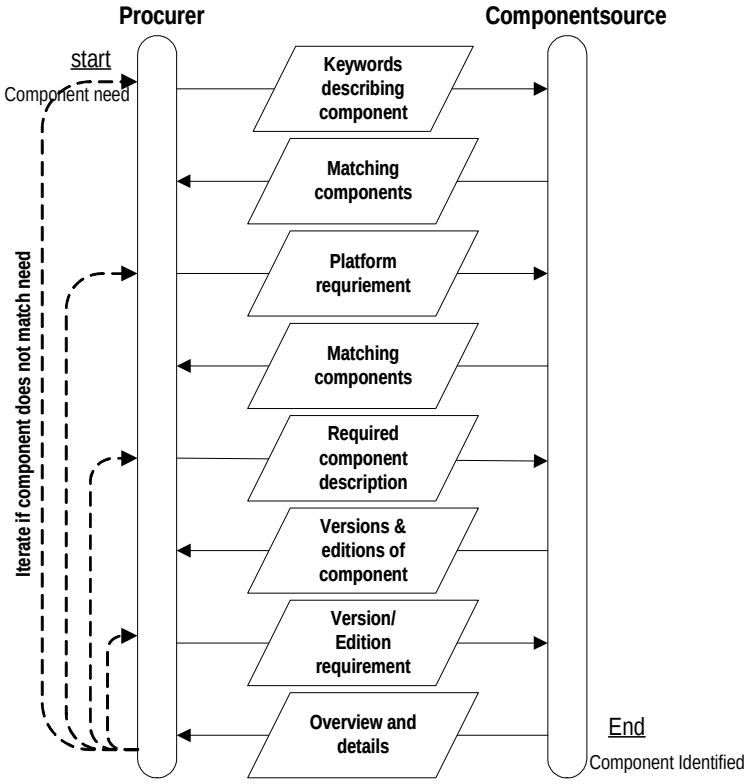

Figure 2. The Find a component process.

The content of the component specification model

The component specification content at Componentsource consists of different parts, these parts handle different aspects of a components specification and are also presented at different levels of abstraction and with different levels of informality. The most important part of the specification is however the initial keywords used as search criteria in the Componentsource repository. These keywords are the only part of the specification used for identification of possible component candidates. The rest of the specification is only used to delimit this initial quantity of component candidates until only one component is left. This component is hopefully what the procurer was looking for. If the procurer fails with identifying a component matching the needs, the procurer can try to identify new candidates (se figure 2) and iterate the process by using new keywords. During the search of the repository Componentsource uses a free-text approach scanning through the documents specifying components supplied by the component supplier. The degree of success for procurers in finding components is thereby based on their ability to choose "correct" keywords and the supplier's ability to use "correct" keywords when supplying information about components to Componentsource. This indicates the need of support when specifying components on a very high level of abstraction in an informal way. The next section of a component

specification covers pricing and licensing. This information is provided to enable investment decisions. It is difficult to decide the importance of this part of the specification but of course the component needs to be price worthy to the procurer. Componentsource also provides evaluations and download possibilities before the final acquisition. This download enables the potential procurer to try the features in the actual environment the component is supposed to exist in after acquisition. The specification also contains compatibility information regarding operating systems, used architecture for component construction, information about the tools used to construct the component and product class. Product class is a classification scheme used throughout Componentsource to classify components. It consists of 106 different categories (see appendix 1). The next part of the specification contains details regarding the component supplier. This information contains a brief description of the vendors business, other components, contact information etc. Componentsource have also chosen to include a section with customer reviews in the specification. In this section component-users can express their experiences and comments about the component. The last section of the specification contains a support forum. This part can consist of frequently asked questions, sample code, usage examples etc. In table 1 we summarize the specification model used at Componentsource.

Table 1 Short summary of component specification

Section of specification	**Containing**
Overview	Overview containing of a short definition of key functionality
Pricing and licensing	Price and license information
Evals. and downloads	Possible downloads of evaluation versions
Compatibility	Restrictions regarding component usage
Publisher details	Information about supplier
Customer reviews	Gathered experience
Support forum	Information to ease usage

Conclusions and summary

Componentsource is one of the largest component distributors. This indicates a successful strategy in aiding procurer in their acquisition process. By reconstructing this process we have described the component specification model that is implied in this support of the acquisition process. The specification model is based on an informal keyword identification process based on the procurer and suppliers ability to express functionality in similar terms. The specification model does not aid the actors in this crucial task and still the process works. The only support in this initial identification process is the product class classification scheme (presented in appendix A) provided by Componentsource. The rest of the component specification contains different details and aspects of the component to aid the procurer in the decision making. It does not aid in the actual identification of component. We believe that the support in the identification process can be enhanced and better supported to make this process more accurate and precise and not so much depending on the actors experience and implicit knowledge.

References

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

Christiansson, B. (2001) ”Component-Based Systems development” In: Nilsson A.G. & Pettersson, J.S. (eds.).*On Methods for Systems Development in Professional Organisations*, Studenlitteratur, Lund.

Christiansson, B. & Christiansson, M-T. (2003) 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), September 2003, Geneva

Componentsource (2003) www.componentsource.com (2003-11-05)

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. (1995) Essays on Infology – Summing up and Planning for the Future. Studentlitteratur, Lund.

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

Vaughn T. (1990) Issues in Standards for Reusable Software Components. http://ruff.cs.umbc.edu. 2000-12-16

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

Appendix A

The component classification scheme used at Componentsource (Componentsource, 2003)

3D Modeling Components
ActiveX/COM Repair Tools
Addressing, Postcode & ZipCode Components
Artificial Intelligence Components
Artificial Intelligence Tools
Audio, Sound & MIDI Components
Barcode Components
Barcode Tools
Bio Recognition/Authentication Components
Business Rules Components
Button and Cursor Design Components
CAD Components
Calendar & Scheduling Components
Charting & Graphing Components
CNS International Code Components
Code Tools Component
Creation Tools
Credit Card Authorization Components
Data Compression Components
Data Compression Tools
Data Conversion Components
Data Entry Components
Data Entry Verification Components
Data Storage Components
Database Connectivity Components
Database Management Components
Database Management Tools
Database Reporting Components
Debugging & Testing Components
Debugging & Testing Tools
Deduplication & Data Cleansing Components
Development Tools
Device Driver Components
Diagramming Components
DirectX Components
Distributed Component Management
eCommerce Components
eCommerce Tools
Email Components
Encryption Components
Explorer Components
Facsimile Components
File Handling Components
File Upload Components
Financial Components
Find & Replace Components
Find & Replace Tools
Function & Source Libraries
Grid Components
Help Authoring Tools
Help Desk Management Tools
Imaging Components
Installation Tools
Instrumentation Components
Internet Communication Components
Internet Communication Tools
Localization Components
Localization Tools
Manufacturing Components
Mapping & GIS Components
Maths and Stats Components
Messaging Components
Modeling Tools
Multimedia Components
Network Administration Components
Network Administration Tools
Network Communication Components
Online Analytical Processing Components
Print & Preview Components
Print & Preview Tools
Product Suites
Productivity Tools
Reporting Components
Reporting Tools
Resizing Components
Search Components
Search Tools
Security & Administration Components
Serial Communication Components
Software Licensing Components
Software Licensing Tools
Software Upgrade Components
Source Code Generators
Speech Recognition Components
Spelling Components
Spread Components
SQL Components
SQLTools
Telephony, Paging & SMS Components
Text Components
Tool Suites
Toolbar Components
Training Courses
Treeview & List Components
User Interface Collections
User Interface Components
VBA a and Scripting Components
Version Control Tools
WebSite Components
WebSite Tools
Windows API Components
Windows Registry Components
Windows Registry Tools
XML Components
XML Tools