Analyzing the Business of Software: A Modelling Technique for Software Supply Networks

Slinger Jansen[1] Anthony Finkelstein[2] Sjaak Brinkkemper[1]

[1] Utrecht University
Information and Computing Sciences Institute
Utrecht, The Netherlands
{s.jansen, s.brinkkemper}@cs.uu.nl
[2] University College London
Dept. of Computer Science
London, United Kingdom
a.finkelstein@cs.ucl.ac.uk

Abstract. *One of the most significant paradigm shifts of software business management is that individual organizations no longer compete as single entities but as complex dynamic supply networks of interrelated participants. Understanding these intricate software supply networks is a difficult task for decision makers. This paper outlines a modelling technique for representing and reasoning about software supply networks. Modelling software supply networks allows managers to visualize liability and responsibilities and identify new business opportunities in a software supply network.*

1 Software Supply Networks

Individual businesses no longer compete as single entities but as supply chains [5]. This holds for the software industry as well, where software products and services are no longer monolithical systems developed in-house, but consist of complex hardware and software system federations [3] produced and sold by different organizations. This development has lead organizations to combine their business and components into complex software supply networks (SSNs), from which they supply end-users with integrated products. As these SSNs grow more complex, it becomes harder for the participants of SSNs to make informed decisions on development strategy, responsibility, liability, and market placement. It also becomes harder to manage the risk associated with these decisions [4].

A software supply network is defined as *a series of linked software, hardware, and service organizations cooperating to satisfy market demands*. SSN management is different from physical goods supply chain management (SCM) in two ways. First, software is adaptible after release and delivery, giving rise to the need for extensive maintenance. Secondly, products delivered to end-users in SSNs are tolerated with much lower quality levels than physical products [1]. As SSNs grow more complex, organizations require more insight into SSNs [2]. Such insight is required by all the participants dealing with the supply network,

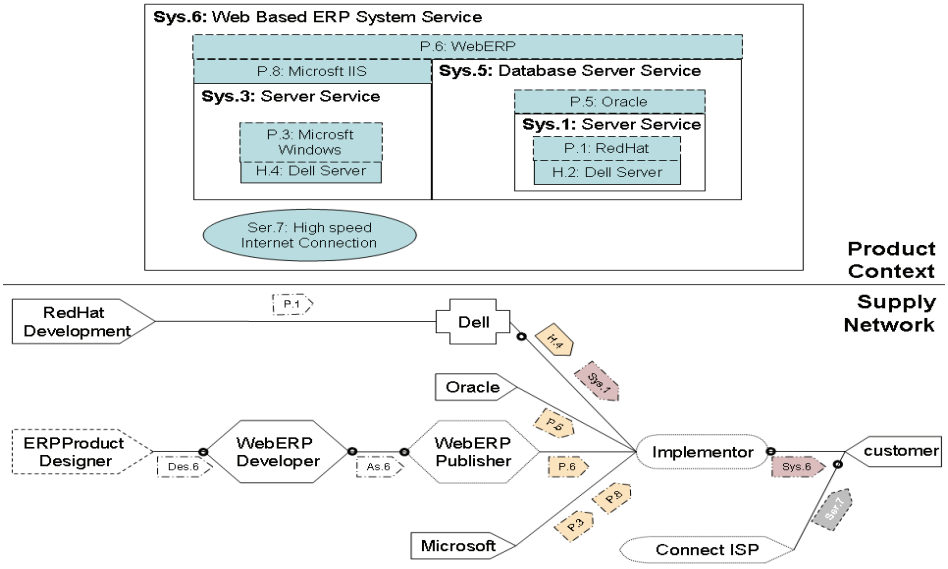

Fig. 1. SSN Model of WebERP

such as a customer trying to determine who to address when an interface does not work, a vendor considering strategic alliances, or a judge trying to determine liability within a SSN. We present a method for modelling the complex relationships between participants in the supply networks of composite products and services. Such models enable participants in the SSN to evaluate risks and architectural decisions for software products, by use of a product context model.

A result of the difference between conventional supply networks and SSNs is that literature on collaboration in supply networks [7] does not discuss maintenance and how it requires information flow through the supply chain. The same holds for other work on SCM, such as [6], which groups horizontal ties between firms (such as manufacturers and suppliers), but fails to recognize the importance of leveraging feedback in such networks, or Lambert and Cooper [5], who provide a conceptual framework for SCM without maintenance.

2 Software Supply Network Models

The model for SSNs consists of two parts, the software product context and the supply network. The product context describes the context in which a software service operates, and the software products, hardware products, and software services that are required to provide the software service. A supply network displays all participants in a SSN, the connections between these participants, and the flows describing the type of product that flows down these connections.

The two diagrams are related in that the product context shows all products that are traded in their different forms in the supply network.

A software service is the provision of one or more functions by a system of interest to an end-user or another software service. A system is a combination of independent but interrelated software services, software components, and hardware components that provides one or more services. There are three types of entities in the software product context, being (1) white-box services and their systems, (2) black box services, and (3) software and hardware components making up systems. Dependencies are modelled by drawing the required component under the first component. At the bottom of all software components will be the hardware and services that are required to create a system that provides a service.

With respect to the supply network, there are seven different types of participants (represented as entities). Between these entities different products and services are distributed (represented as labels on arcs between entities). When looking at all participants, seven categories of organizations arise that participate in the value chain of software products and software services.

2.1 An Example: WebERP

In figure 1 the example SSN model is presented for a customer requiring a Web Enterprise Resource Planning (ERP) service. To supply this service internally, the customer has a partner implementer organization who implements a product WebERP on a newly purchased local database server and a local web server. The software product context displays that to supply **Sys.6**, **P.6** is required. To run **P.6** a server is required that supplies WebERP through a web server application, in this case Microsoft IIS. On the other side a database server (**Sys.5**) is required that manages all the data for WebERP. Both servers, supplied by Dell, run a different operating system. To provide the WebERP service, a high speed internet connection is required, such that people working outside of the organization can access WebERP. As products transition from source code to product to system, they generally retain the same number, such as for WebERP; **Des.6**, **As.6**, **P.6**, and **Sys.6** are all instances of the same product.

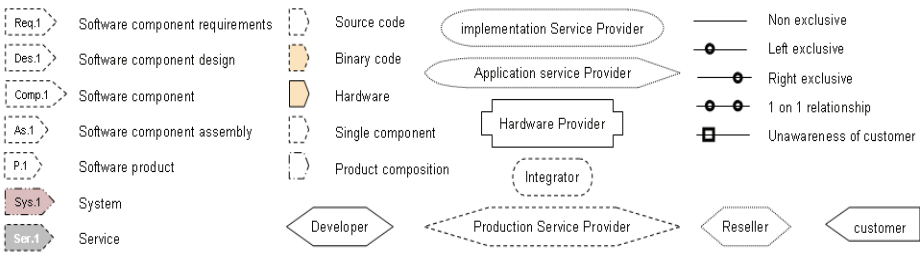

Fig. 2. Software supply network model legend

With all services and products laid out, the supply network can be created. The customer requires **Sys.6** from the implementer. The implementer purchases two servers from Dell, one with RedHat installed, the Oracle database management system, WebERP, and Microsoft Windows and IIS for the web server. The implementer combines all components into a new system **Sys.6** that is delivered to the customer.

WebERP, the application, is designed by an external ERP product designer. Their design is the blueprint for WebERP, which is developed by WebERP developer and sent to the WebERP publisher. The WebERP publisher compiles the components and productizes the WebERP components, explaining the color change of the flows and the transition from component assembly to product. Finally, the internet service provider (ISP) provides **Ser.7**, which is black box because it is not relevant how this service is provided.

With this example some applications of SSN models can be demonstrated. In figure 1 the implementer could decide to provide the service **Ser.6** itself directly to the customer. Furthermore, the Oracle could decide to deliver *ready to run* systems by installing their product onto a server **Sys.5** and sell them as black boxes to the implementer. These two examples, however, provide a small portion of the myriad of applications of SSN models.

This paper has presented the concept of a SSN model, and its two parts, the product context and the supply network. Part of our future work is to find out whether the SSN models can be used to establish economic viability of a business model.

References

1. L. Baxter and J. Simmons. The software supply chain for manufactured products: reassessing partnership sourcing. In *International Conference on Management of Engineering and Technology*, 2001.
2. R. Colville and P. Adams. It service dependency mapping tools provide configuration view. In *Gartner Research News Analysis*. Gartner, 2005.
3. C. Ghezzi and G. P. Picco. An outlook on software engineering for modern distributed systems. In *Proceedings of the Monterey workshop on Radical Approaches to Software Engineering, Venice (Italy), October 8-12*, 2002.
4. S. Jansen and W. Rijsemus. Balancing total cost of ownership and cost of maintenance within a software supply network. In *proceedings of the IEEE International Conference on Software Maintenance, Philadelphia, PA, USA, September*, 2006.
5. D. M. Lambert and M. C. Cooper. Issues in supply chain management. In *Journal of Industrial Marketing Management*, 2002.
6. S. G. Lazzarini, F. R. Chaddad, and M. L. Cook. Integrating supply chain and network analyses: the study of netchains. In *Journal on Chain and Network Science*. Wageningen Academic, 2001.
7. J. Patosalmi. Collaborative decision-making in supply chain management. In *Seminar in Business Strategy*, 2003.