Software Process as a Service? Bridging Service Design and the Software Process

Miltiadis Lytras[1] and Miguel-Ángel Sicilia[2]

[1]American College of Greece
Gravias 6 St., Aghia Paraskevi (Greece)
mlytras@acgmail.gr
[2]*Information Engineering* Research Unit
Computer Science Dept., University of Alcalá
Ctra. Barcelona km. 33.6 – 28871 Alcalá de Henares (Madrid), Spain
msicilia@uah.es

Abstract. Software has become an essential component in many service systems, either as an enabler of a more efficient and cost-effective interaction or becoming part of the value co-creation activities themselves. However, software processes that result in the development or evolution of service-support systems do not provide explicit elements or considerations that link with the models and design processes of the services they are intended to support. Since the arrangement, change and improvement of services determine how supporting software should be developed and changed, there is a need to bridge software process models and service design models. In a radical position this would entail that the software process itself becomes a service for the design and evolution of services. This paper conceptualizes a preliminary approach for that purpose. .

Keywords. Software process, service design, service systems, process modeling.

1 Introduction

The importance of the service sector has increased in industrialized economies, accounting for a higher portion of national GDPs than in the last years (Wölf, 2005). Consequently, the interest in services has grown rapidly and it has led to the emerging concept of a *service science* (Chesbrough and Spohrer, 2006). As services are pervasive as the "front stage" in economic activities of any kind (Teboul, 2006), the role of software systems adequately supporting services is also becoming more important with the widespread use of the Internet for e-commerce (Feigenbaum, Parkes and Pennock, 2009; Chen & Tsou, 2007). This raises several questions, including how software supporting services differentiate from other kinds of software, how software that better supports service design can be developed, and in general, how the software process can be adapted to the service design process.

Different kinds of services[1] require different kinds and intensities of human interaction, from completely automated services (e.g. those provided by a software system that acts on behalf of the provider) to services in which software support mediates the relationship only to some extent, and they also diverge in the extent to which these systems support more customized or more standardized services. We consider here customer interaction-intensive service systems with a high degree of customization and that evolve driven by interdisciplinary service innovation (Beirne, and Cromack, 2009). For that case, good service-supporting software needs to be *highly configurable* to better adapt to the different needs and profiles of the users. Examples of such high configurability can be found in the field of mass customization (Berger et al., 2005). Further, services need to be constantly re-designed to face changes in customer needs and behavior, and also as a way to achieve competitiveness through innovation (Berry et al., 2006). Then, software supporting services need to be *highly evolvable*. This view affects both some quality attributes related to the internal structure of software, and also to the software process itself, that needs to be reconsidered to face the cross-boundary collaboration and joint analysis that is critical to bring together the perspectives of social science and engineering. This suggests a change in the traditional role of Software Engineers to a more proactively involved one that mixes service design techniques with agile software development methods.

This paper discusses some conceptual aspects of bridging software process models and interaction-intensive service design requirements, speculating about some possible directions for a new understanding of software and service co-design. The research problem can be stated as: *how (highly interactive and customizable) service design can be better integrated with the software process?* The paper sketches some preliminary directions resulting from ongoing work in a new integrated service-software co-design process.

The rest of this paper is structured as follows. Section 2 approaches the problem of combining service design and the software process. Then, Section 3 discusses how service evolution can be translated into attributes of the software product. Finally, conclusions and outlook are provided in Section 4.

2 Introducing service design in the software process

A service can be defined as “a non physical product manufactured by suppliers and consumers at the point of delivery”. This definition emphasizes value co-creation and recognizes that services are assemblies of previously designed components such as people, processes, prior information, knowledge and skills and tools for the service, including information technology. Ideally, software design should follow service design, so that the software is built to support the front stage interaction, and this determines the required attributes of the software to be built. This leads to a first important aspect that is related to the richness and intensity of interaction. A high

[1] Here we use the term “software service” to differentiate services as computer-based artifacts from the general notion of service as a process of co-creation of value involving customers and providers.

level of interaction will lead to a requirement for software to be more customizable to the service needs. Figure 1 shows a service-intensity matrix adapted from Teboul (2006), depicting the relation of software configurability needs in relation to higher levels of interaction that lead to more customized services (as opposed to standardized ones).

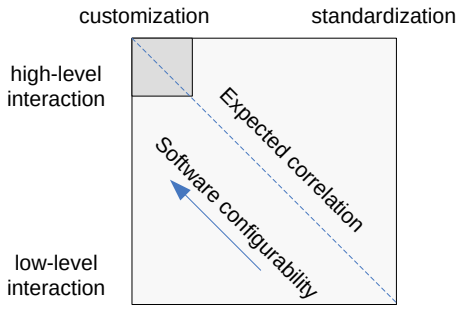

Fig. 1. Relationship of software configurability and the service intensity matrix.

As service innovation is related to change and experimentation with customer interaction, we consider here software in the upper left extreme of Figure 1. Configurability entails delaying the composition of particular product features to a later moment in the software development or deployment process, and this is known to impact testing, as integration testing is also delayed (Jaring, Krikhaar and Bosch, 2008). For practical purposes we define here configurability as an internal software attribute related to the capabilities of the software design to accommodate changes in the way its parts are aligned with user interaction with a low effort[2]. For example, Yang and Hsiao (2009) describe a process of service innovation in the healthcare domain using action research to analyze service design. Teamwork produced changes in service design requirement impacting the supporting software, for example, requiring the reconfiguration of the process of delivering medicine to patient's homes or the addition of preservation condition information for some medicines.

The service design process (Goldstein et al., 2002) requires first a concept of the service to be developed, which usually comes from marketing studies. Then, the engineering of the service can be approached methodically (Bullinger, Fahnrich, and Meiren, 2003), considering the *structural*, *process* and *outcome* aspects. Process models describe how the outcomes of a service are achieved, as services are *intangible* in nature, the process aspect takes a prominent relevance. The various processes are documented from the concept phase onwards to ensure process efficiency. The objective is always to eliminate non-value-adding activities at the earliest possible stage and to remove unnecessary interfaces and media discontinuities. Then, the service design model can be process-centric, and notations as the *Business Process Modeling Notation* (BPMN) (White and Miers, 2008) can be used for expressing it,

[2] Note this definition may be conflicting with other senses of "configurability" in the Software Engineering literature.

enriched with meta-information about the service concept. In the case of rich and customized interaction, the process models can be expressed as a set of models.

Service process specifications (SPS) then become the main input of the development process, and SPS breakdown becomes the natural breakdown for software development. It can be argued that use-case techniques for requirements analysis actually can be used to specify processes as scenarios. However, use cases are descriptions of a system's behavior as it responds to a request that originates from outside of that system. In contrast, SPS focuses on service activities composed to create value. Configurability is supported by a fine-grained detail and maximum decoupling of service activities, so that they can be reused and reconfigured in an inexpensive way.

The consideration of the service design process leads to some potential new values and changes in the software process, aligned with the idea of agile process models. These are discussed in the rest of this section, using the OpenUP model[3] as a framework. OpenUP builds on existing widely used process models and it is openly documented, so it provide a good vehicle to express emphasis for particular design processes.

Defining the use cases as service encounters

Use cases are defined in terms of main and alternative courses of action for a given functionality. Service supporting use cases can be considered a subset of the functionality of the application that critically determines value and it is informed by marketing and customer behavior studies. This need to be accounted for in the "`Find and Outline Requirements`" task, so that the customer concept and value can be traced from the development artifacts through the requirements.
The use of BPNM models allow for a more flexible modeling of this kind of interactions and could constitute a good option as a technique for the task `Detail Requirements`.

Planning the increments and releases based on service value

Agile methods emphasize incremental negotiation of the next requisites to be build and the quick establishment of a functional release. This fits well with service design, but need to be reconsidered, as full customer interactions should be the unit for defining the iterations. This has to be accounted for in the `Plan Project` and `Plan Iteration` tasks. Also, the overall project management approach needs to be aligned with both external users and the service-providing organization. Bygstad and Lanestedt (2009) concluded that "successful ICT based service innovation is not associated with a tightly run project (focused on cost, time and quality) or a professional project manager. Rather, successful service innovation is found in projects with a strong integration with the service providing organization and the external

[3] http://epf.eclipse.org/wikis/openup/

users of the services." This requires flexible planning and tracking based on strong user and customer involvement.

Develop an architecture based on the principle of configurability and flexibility

As discussed above, configurability is a main required attribute for changing services. This needs to be considered early in the architecture as the main building block. Approaches to *software product line development* provide a good source of techniques and ideas for such approach to delayed reuse.

Test early against real interaction

The `Create Test Cases` task in OpenUP takes place early in the Inception phase. However, these cases are mostly intended for correctness testing, while in the context of service design, it is user (customer) validation that has the emphasis. This requires early customer validation of the interaction (even if incomplete) ideally via experimental approaches as those that are common in usability testing, but with a marketing target. Also, the business model of the service-providing organization must be an objective of test cases. The economic nature of service interaction can be approached by using the dual modeling structure present in the REA ontology (Geerts and McCarthy, 1999).

3 How to introduce service evolution in the software process?

The challenge of change in software services has been recognized elsewhere (Papaglozou, 2008) but more is required to understand the nature of software supporting services and their evolution patterns. Service innovation processes make even more important the process of software evolution.

Configurability in an extreme leads to reduce the needs for software evolution, as easy configuration should allow make some changes become inexpensive or even user-configurable. However, configurability entails in general increased software complexity so software supporting services require a careful consideration of process models allowing for evolvable and configurable product construction. Also, in many cases service innovation requires some form of technological support that was not anticipated, or even the innovation process can be considered to be limited by the degree of knowledge on the possibilities and costs of technology of the participants in the service design process.

Software as a Service (SaaS) is a model of software deployment where an application is licensed for use as a service provided to customers on demand. That idea can be used for service design, as it might be that third parties provide the software support for the interaction, so that the service company does not need to invest in infrastructure. However, the notion of configurable software services conflicts with the

need for differentiation as a competitive advantage. Thus, it is in the flexibility for evolution of the software where the key advantage lies.

If service evolution comes from service innovation, the software process needs to be fitted to the innovation activities. For that purpose, the software process needs a redefinition that makes some software development activities part of a service of value co-creation with the service providing organizations. Service design is constrained by technology costs and feasibility, and also new technologies are a main driver of service innovation. Then, software process can be understood as a supporting and enabling "service for service design", requiring a profound knowledge of the technical structure, costs of evolution and architectural constraints of the existing software infrastructure. This requires a redefinition of the software engineer or software manager, which needs to transition to a role with mixed skills and becomes a facilitator of the service innovation process within a flexible project structure (Bygstad and Lanestedt, 2009). In a radical approach, the *Inception* phase of OpenUP can then be merged completely with service design methods, with the last activity, `Agree on the Technical Approach`, as the only activity that is the sole responsibility of the software engineering team.

This "Software Process as a Service" (SPaaS) concept represents a radical merging of service design and software engineering and profoundly affects the contractual relations of software development staff with other stakeholders. However, the main ingredients for this mix (e.g. flexibility in planning, customer involvement, value-orientation) were already available in agile methods.

4 Conclusions and outlook

Highly intense services that are continuously reconsidered as part of innovation processes place specific demands on the role of Software Engineers and the software process itself. The idea of agile software processes can be adapted to better fit the service design process. This paper has explored some possible values for a service-design aware software process, pointing to some possible directions for further inquiry. A more radical view of the process would be that of more tightly merging the service design and innovation process with the software process, so that the latter becomes actually a service for the former. This idea fits the emerging paradigm of service science as a interdisciplinary mixture of management and engineering.

All the ideas presented in this paper are obviously highly speculative, so that they are intended to stimulate discussion and as a previous step for a thorough integration of service design processes with software processes. Future work should progress in proposing new software-service co-design processes and contrasting them in practical cases or experiential reports.

References

Beirne, M., Cromack, C. (2009) Managing creative coalitions: Reflections on the social side of services innovation, European Management Journal, 27(2), pp. 83-89

Berger, C., Moeslein, K., Piller, F. and Reichwald, R. (2005) Co-designing the customer interface for customer-centric strategies: Learning from exploratory research, European Management Review, 2 (2005) 3: 70-87.

Bullinger, H.J., Fahnrich, K.P., Meiren, T. (2003) Service engineering--methodical development of new service products, International Journal of Production Economics, 85(3), pp. 275-287.

Bygstad, B. and Lanestedt, G. (2009) ICT based service innovation - A challenge for project management, International Journal of Project Management, 27(3), pp. 234-242

Chen, J.S. & Tsou, H.T. (2007). Information technology adoption for service innovation practices and competitive advantage: the case of financial firms" Information Research, 12(3) paper 314.

Chesbrough, H. and Spohrer, J. (2006). A research manifesto for services science. Communications of the ACM. 7, pp. 35-40.

Eriksson, M., Borstler, J., Borg, K. (2009) Managing requirements specifications for product lines - An approach and industry case study, Journal of Systems and Software, 82(3), pp. 435-447

Feigenbaum, J., Parkes, D. C., and Pennock, D. M. (2009). Computational challenges in e-commerce. Commun. ACM 52, 1 (Jan. 2009), 70-74.

Geerts, G. and McCarthy (1999). An Accounting Object Infrastructure For Knowledge-Based Enterprise Models. IEEE Intelligent Systems & Their Applications (July/August 1999), pp. 89-94.

Goldstein, S.M., Johnston, R., Duffy, J.A. and Rao, J. (2002) The service concept: the missing link in service design research?, Journal of Operations Management, 20(2), pp. 121-134

Jaring, M., Krikhaar, R.L., Bosch, J. (2008) Modeling Variability and Testability Interaction in Software Product Line Engineering,. Seventh International Conference on Composition-Based Software Systems, 2008. ICCBSS 2008, pp.120-129.

Papazoglou, M.P. (2008) The Challenges of Service Evolution", Keynote address, In Procs. Advanced Information Systems Engineering Conf.: CAISE 2008, Springer-Verlag, Lecture Notes in Computer Science, Montpellier, France.

Teboul, J. (2006). Service Is Front Stage: Positioning Services for Value Advantage. Insead Business Press. Palgrave Macmillan.

White, S.A and Miers, D. (2008). BPMN Modeling and Reference Guide. Future Strategies Inc..

Wölfl, A. (2005), The Service Economy in OECD Countries, STI Working Paper, 2005/03, OECD, Paris.

Yang, H.L., Hsiao, S.L. (2009) Mechanisms of developing innovative IT-enabled services: A case study of Taiwanese healthcare service, Technovation, 29(5), pp. 327-337