The Pervasive Nature of Variability in SOC

Ateeq Khan*, Christian Kästner†, Veit Köppen*, Gunter Saake*
*University of Magdeburg,Magdeburg, Germany
ateeq,vkoeppen,saake@iti.cs.uni-magdeburg.de
†Philipps Universität Marburg,Marburg, Germany
kaestner@informatik.uni-marburg.de

***Abstract*—Service-oriented computing has gained attention from researchers and industry due to various benefits. Enterprises are supported by SOC-based solutions. Enterprise requirements change with the customer needs, market conditions, and variability is often needed in SOC. Research usually focuses at variability on the business process level and ignores variability at other layers. In this paper, we show what kind of variability can occur and how it spans across all layers of SOC. We present some example scenarios to discuss different kinds of variability and their reasons in SOC environments. Finally, we classify variability to show that variability is not an isolated problem in a single layer.**

***Keywords*-service-oriented computing; SOA; web services; business processes; variability; customization; variability approaches.**

I. Introduction

Demands of customers change; adapting software accordingly is challenging for industry and researchers. Software must have the flexibility to change itself to satisfy these demands. Service-oriented computing (SOC) is a paradigm to create information systems, and provides flexibility, interoperability, and other benefits [1], [2].

In SOC, services are basic units. Different services collaborate with each other to perform business operations and processes. These business processes requires different executions depending on the customer's needs. Service providers suggest industry best practices [2], but they cannot cover all types of business scenarios or specific customer's needs.
Service-oriented architecture (SOA) is an approach which uses services to provide IT solutions [1]. Generally, services are exposed for the whole world (not only for specific customers), so customisation/variation of services is necessary for customers. Customer demands are subjected to change because of several factors, e.g. change in requirements, solving new problems, availability of involved resources, user preferences or context, usage scenario, data formats, legal requirements, protocols, and user interfaces.

The scope of requirement modification also varies. It maybe small, e.g. to add a method in a service, or large, to change a whole workflow of an application. Even a small change in a service, sometimes, spans to the whole application due to dependencies on other services and involved factors (e.g. corresponding changes in user-interface, service parameters, or at the infrastructure level). Service adaptation motivates to find out and classify various types of possible variability.
Variability management is not a new area in software engineering; there are several approaches to manage variability. It is worthwhile to apply such approaches in SOC domain. An interdisciplinary study is necessary to find useful solutions, which provide and manage variability in SOC. There are approaches in SOC literature, which address and discuss variability but variability is often implicit and only some aspects are considered (mostly at the business process level [3]–[5]). In this paper, we use the terms adaptability and variability interchangeably, although adaptability is more general and variability is a technique to provide adaptability. In variability, we explicitly provide different options to change a service or system, so clients can choose options according to needs.

We argue that variability in SOC should be considered carefully and explicitly in all layers instead of providing ad hoc solutions. We motivate the need of variability in services exemplary. Often, variability is considered as an isolated problem on business processes, workflows, or at the middleware level. Here, we demonstrate that variability is not isolated and spans to all SOA layers. We show the possibilities of service variability into various layers and variability types in each layer. We discuss how variability affects other involved processes, services, layers, and service components. Once the impact is known, it can be used for adaptation in an optimised and cost-effective way. We classify variability by means of an example case study and generalise subsequently into a classification.

II. Case study: Sports ERP System

This paper illustrates how variability can pervade all parts of a system by a case study. We implement a simple *sports information system* for demonstration purpose, domain inspired from [6], based on services with different variability options. We find different types of variability prevailing in all parts of the system and its architecture, in the sport scenario, to meet the stakeholder's demands. Involved stakeholders of the system range from sports federations, clubs, members, coaches, trainers, players, volunteers, to fans. The variations in services and stakeholder's demands are due to their business requirements, company size, location, service level

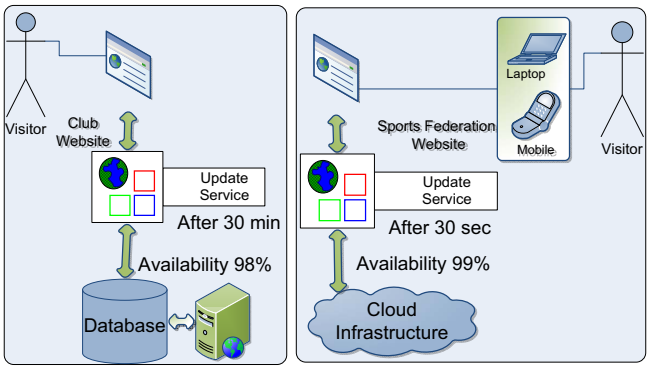

Figure 1. Motivation Scenario: Process Latest Match Result

agreements, support, pricing models, privacy, and security perspective.

We show different types of variability, for example consider a business process *Latest Match Result* in which, a latest match result or short text commentary is displayed on the consumer web site through a web service. A federation or club consumer uses the process to display the results on their website. Depending on the match, infrastructure is provided to this part. This scenario is depicted in Figure 1 and discussed further.

A. Process Latest Match Result

Let us consider that two different consumers for two distinct matches call the above-mentioned process. One match is of international importance played between two countries, *International Match*, and the sports federation website displays the result. The other match is a local *league match*, and the club website displays the result. Clearly, an international match has more importance compared with the local league match. More visitors and applications access the sports federation website as compared to the club website during a match. Different types of events (in terms of matches and service consumers) lead to different requirements. Some of the variability options for the above-mentioned process are depicted as a feature diagram [7] in Figure 2 (the notation is quite simple and standard in the product line community: boxes illustrate features; features connected by empty circles are optional, features connected with filled circles are mandatory, and arcs are used to describe alternatives) and discussed below.

- Stakeholder's variability: Due to different nature and size of involved stakeholders (federation and club), their requirements are different. A federation's requirement (requires more features, different variabilities are possible) is different from the local club's requirement, so variability is provided accordingly.
- User-interface variability: For the *international match*, the result is displayed at mobile devices, web pages, and can be further integrated in other portals. For the *league match*, the result is displayed only on a website.
- Service variability: Information update service on these websites will run at different frequencies. For an *international match*, the result is updated more frequently (every thirty seconds) as compared for the *league match* (after half an hour). This can be achieved by using different services or business rules.
- Variability of non-functional properties: Different customer requirements lead to vary non-functional properties (e.g. availability and security). A federation needs high availability of their solutions (99%), while for a club availability of 98% is sufficient in terms of service level agreements. A federation website would be more vulnerable to denial of services or other attacks, therefore security requirements would be higher. A federation may requires secure SSL connection from an employee over HTTPS protocol to login, while for a club website, employees can login using HTTP protocol.
- Infrastructure variability: The infrastructure varies for both federation and club websites. For a sports federation website, the number of website visitors can grow rapidly during the match, so infrastructure must be scalable (by adding/removing the hardware or software resources for optimal utilisation, e.g. cloud computing techniques). In contrast to the federation website, for a website of local club fixed hardware would be enough. Similarly, one federation may prefer services from a hybrid cloud model and other stakeholders prefer private cloud due to countries or federation policies.

We introduced a motivating scenario that shows different types of variability in a single scenario. We analyse variability at different levels in our sports system and consider that stakeholders can have various requirements at each level. Identification and discussion of all types of variabilities are not needed. Our purpose is to illustrate that variability is a recurring theme, without providing a complete domain analysis of the sports information system. Variability demand grows rapidly and will become more complex if we extend this case study to other specific sports domains. We discuss variability in the sports domain only but variability also exists in other domains and systems.

III. Variability Classification

After providing variability examples, now we generalise them in the form of a variability classification for SOC. This classification is depicted in Figure 3 which provides a holistic view. Our classification consists of different layers. Each layer consists of sub-layers for more granularities of detail. We discuss general variabilities from a technical point of view in each layer. Granularity of layers depends on the

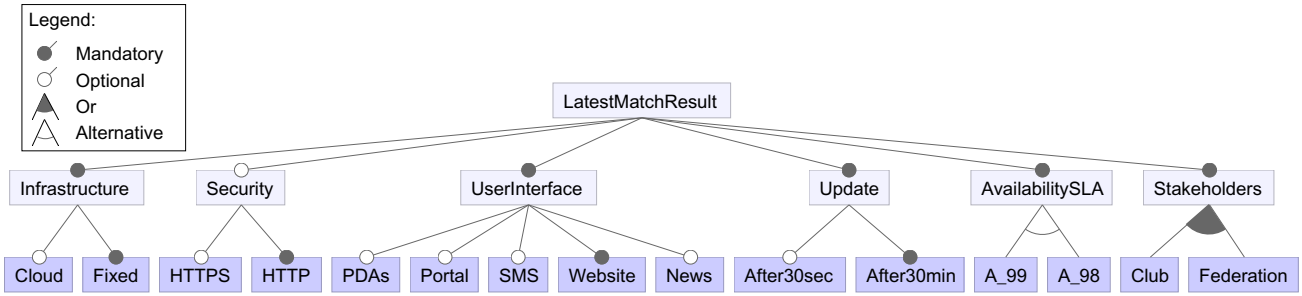

Figure 2. Feature diagram of Latest Match Result

applications and the environment where they are used. We provide variability classification and explicit representation of variability points which exist in each layer. Again, our goal is not a complete survey of all possible variability options, but we want to illustrate that variability is also pervasive throughout all technical layers.

A. Service Consumer Layer

Service consumers have same or different requirements in terms of interfaces. Services are composed of one another to achieve a common goal. Variation in requirements is due to provide flexibility, optimised execution of processes and for the reduction in costs. Variability in this layer is further decomposed into user-interface and service interface or configuration variability.

1) User-Interface Variability: Service consumers have different requirements to view an output of service or process in terms of graphical user interfaces (GUI). These requirements are specific to technologies, e.g. HTML, Java client, document formats (like PDF), email, and legacy applications. Examples of user-interface variability are shown in the *Latest match result* process in our case study.

2) Service Interface Variability or Configuration Variability: In SOC, a service can use other services. In case of service signature mismatch (service input and output parameters), the usage of available service is limited. Configurations are generated automatically or adapted based on the requirements of the calling system or service, e.g. match fixture service input parameters are country, city, zip code, or only country and zip code. Providing variability in the service signature can be used as a way, for reducing costs by using the same service from service providers and fulfilling consumer's needs.

B. Service Middleware Layer

The middleware layer provides support for the application integration, service aggregations, transactions, service monitoring, and infrastructure intelligence (business rules, policy driven behaviour, and business process support). Some requirements can also be implemented at interface or service provider layers according to scenarios and company size. There exist different solutions in industry to manage variations at middleware layer. An example is SAP Process Integration [8] as a middleware solution. It has different components for integrated communication, services routing, transformation, parameters mapping, BPM engine, and business rules. These components help to provide and manage variability. Application-specific adapters are developed to integrate services with other organization environments. This heterogeneity exists due to existing applications, different application protocols, and legacy applications. Message or protocol transformations are also used to overcome mismatch. In the *Organize match* process, ground availability activity is managed by a legacy system, and we have to integrate this activity in our service-based system.

1) Process Variability: Different consumer requirements lead to process variability. Even if industries' best practices are followed, not all consumer needs can be forecasted. Main reasons for deviations are consumer locations, business rules, policies, legal rules, and market conditions. Based on heuristics, best workflows (paths consisting of a set of services) can be suggested to consumer to improve their processes [9]. According to specific business rules, different workflows are executed, e.g. in our case study, team availability activity can be performed in start or after ground activity in *Organize match* process.

2) Service Orchestration: In service orchestration, services are arranged together to fulfil some common goal. These service orchestrations can be extended to integrate other business processes or fulfilling other requirements. During service orchestration, there are different points of variability, e.g. communication types (synchronous/asynchronous communication of services), handling exceptions (by introducing other services), consumer preference for service selection/replacement from specific service provider, service pricing, QoS, and service level agreements. In our case study, the *Organize match* process consists of activities in which some activities are performed by synchronous communication (match scheduling algorithm) and others are by asynchronous communication (team availability).

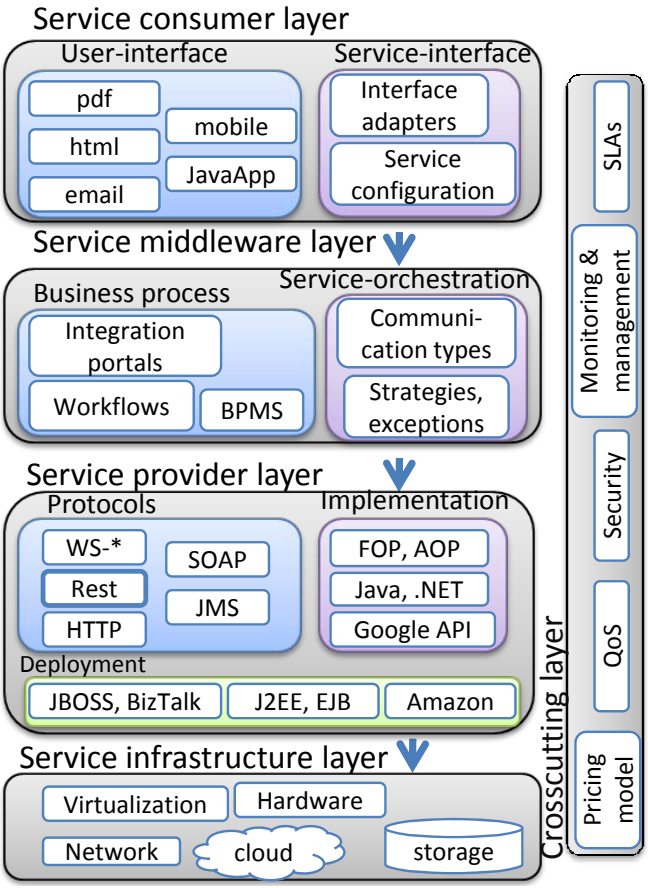

Figure 3. Variability Layers

C. Service Provider Layer

Service provider must customise the services to meet the consumers' specific requirements. In this layer, variabilities are mainly from protocols, deployment, and implementation perspective and we discuss them further.

1) Protocols Variability: Service protocols are one of the variability reasons in this layer. SOA is introduced in the market to solve interoperability and integration problems in the enterprise. Unfortunately, due to the lack of standardisation, interoperability becomes difficult. Solution providers introduced proprietary protocols and used non-standards specifications or drafts to meet requirements. A long list of different protocols and specifications exist in SOA at different layers. Some of them are WS-Security, WS-Transaction (WS-* specifications) or WS4BPEL, SOAP, WSDL, XML, JMS, JDBC, and Java remote method invocation.

The Middleware integration solutions are introduced to handle these variations. Some integration solutions do not support specific protocols or provide their own proprietary protocols. Message exchange formats used at middleware or integration portals are also a factor for variability, various integration portals support different message format specifications, which may lead to failure of a whole business process or a message. Enterprises have their preferences when selecting protocols or standards, e.g. SOAP 1.1 or SOAP 1.2. Therefore, there are possible variations between protocols and their compatibility with consumers and service providers.

2) Service Implementation: Different programming languages or implementation techniques can be used to host a service at the service provider level, e.g. Java, .NET, JavaScript with AJAX, and other platform specific techniques. Cloud service provider offers platform and technology to implement the services e.g. Microsoft Azure offers .NET related services on their platform. Service consumer may want to use platform for J2EE applications.

3) Service Deployment: Service deployment depends on the development language and target server (platform). For servers, different servers have their own recommended methods and preferences for deploying services and application, e.g. Apache, JBOSS, IBM WebSphere, Microsoft BizTalk, and Bea WebLogic. There are also different ways to host a service, e.g. as EJB, J2EE, .NET, CMP, and CCM. Each deployment approach has its own advantages and disadvantages. Similarly, different cloud providers have different requirements to host services on their platform.

D. Service Infrastructure Layer

This layer provides necessary infrastructures to service consumers and service providers. In this layer, variability can occur in terms of data storage, network facilities, hardware virtualization, software configuration, and legal issues.
At infrastructure layer, variability exists from selecting the infrastructure according to specific application needs. For example, to store data in a cloud according to country laws or consumer requirements (geographical locations and storage infrastructure, security needs). Consumers can select database management systems in the cloud, according to their application. Consumers may have more concerns about privacy of their user's data outside of their organizational control and want to save it in an encrypted way. Encryption techniques to store data in a cloud also vary. Such solutions are useful for Small-Medium Enterprises (SMEs) where the reduction of the cost and customisation are of main focus. We introduced an example in the *Latest match result* scenario, where hardware infrastructure is scalable according to usage.

E. Crosscutting Layer

Variability requirements also crosscut other layers. Examples of such variability are QoS, service level agreements (performance, reliability, competence, availability, and fault tolerance), service pricing, security, monitoring and management, and applications for legacy systems support. In some organizations, where security

is a priority requirement, additional security protocols are used or proprietary protocols are introduced, e.g. SAML [10], Kerberos [11], SSL, and SOAP message security. Variability in pricing models is also needed from the consumer perspective. Different pricing models and usage plans are offered according to infrastructure usage, e.g. pay per use of a service or monthly. Other examples of crosscutting variability are in the *Latest match result* process. In our proposed classification, some variabilities can exist at the different level of granularities, e.g. scalability (or some non-functional requirement) is a requirement, and it has not any effect on the presentation layer as compared to the infrastructure layer.

We discuss variability issues in SOC and provide a classification of variability. In each layer, we first motivate that why we need variability and listed some variability approaches and techniques. Each layer contains different variability granularities, which can satisfy customer demands and enables an integrated way for communication with different organizations. We show how variability spans and effects other layers. For simplicity, here, we limit the discussion of standards and protocols for business process modelling language, service monitoring and management, transactions, and workflows.

IV. Related Work

SOC adaptability is an active research area. There are approaches in literature addressing adaptable systems, service matchmaking [12], [13], and intermediate solutions [3], [13], [14]. Existing approaches focus toward service interfaces, service composition [14], message mismatch [15], and business processes [3], [4]. Typically, one kind of variability is considered at a time and other variabilities are not considered.

In [12], Sam et al. propose a framework for automatic web service customisation and focus only on the service mismatch between inputs and desired outputs at the service interfaces using intermediate or conflict resolution service. In [16], Zambonelli and Viroli argue that we need a completely new architecture to deal with adaptability and evolution of the services. In [17], Papazoglou suggests that we need to extend basic SOA and presents an extended SOA, which consists of different layers but variability was out of scope of the paper. In [18], Apel et al. motivate to use feature-based approach for the development of SOA and to provide variability. They highlight expected benefits and future research challenges of features and services. In [19], [20], authors introduces service variability patterns and different implementation techniques used for variability in SOC.

Examples of variability approaches on middleware layer are discussed in [3], [5], [13], [14]. In [3], Rahman et al. propose three types of variability at process level namely, inter-activity, intra-activity, and business process level. They intercept SOAP messages and use business rules to provide variability and consider only process variability. In [5], Mazzoleni and Srivastava consider a business processes customisation perspective. They also discussed the impact of the new requirements. In [13], Jiang et al. use a pattern-based approach to identify and managing variation points in web services. Different variation points are identified in WSDL documents, service endpoints, and business logic. In [14], behaviour patterns are used for dynamic service management in devices. In [21], four types of service variability are discussed and authors purpose solutions for these variabilities. The main difference is that we discuss other variabilities and effect of variation from one layer to other. We discuss at a more detailed level. In [22], Segura et al. presented variability in web service flow and provide variation points in web service flow. In [23], Mietzner et al. extend the service component architecture with variability descriptors for configurable services. Aspect-oriented adaptation mechanisms are used in [3], [4], which focus on certain types of service adaptation on different layers.

All these works address isolated variability issues, but do not integrate them into a bigger picture.

V. Summary and Outlook

We discuss why and what kind of variability is needed in SOC environment. We show different types of variability exist using a real world scenario in the case study. We propose a classification of variability in SOC environment. Such classification helps to model and address variability issues for adaptation of the system in a systematic manner. We highlight variabilities in different layers.

In ongoing work, we are working on a framework, which addresses variability issues at different layers for adaptation. We provide a variability classification for dealing variability at different layers of SOA. We have to further elaborate this classification to know what kind of dependencies exists to manage variability and what are the effects of variability on other layers, which may lead to further modifications. How we can model it into a tool, which gathers consumer requirements and provides variability accordingly with minimum effect on other layers. Providing variations may lead to inconsistency in whole application, how we can check these inconsistencies in case of SOA environment. Approaches like type checking in SOC environment are needed to check consistency of whole application.

Acknowledgement

Ateeq Khan is supported by a grant from the federal state of Saxony-Anhalt in Germany. This work is partially supported by the German Ministry of Education and Science (BMBF), within the ViERforES-II project No. 01IM10002B.

REFERENCES

[1] N. Josuttis, *SOA in Practice: The Art of Distributed System Design*. O'Reilly Media, Inc., 2007.

[2] M. P. Papazoglou and W.-J. van den Heuvel, "Service oriented architectures: approaches, technologies and research issues," *VLDB*, vol. 16, no. 3, pp. 389–415, 2007.

[3] S. S. ur Rahman, A. Khan, and G. Saake, "Rulespect: Language-Independent Rule-Based AOP Model for Adaptable Context-Sensitive Web Services," in *36th Conference on Current Trends in Theory and Practice of Computer Science (Student Research Forum)*, vol. II. Institute of Computer Science AS CR, Prague, Jan. 2010, pp. 87–99.

[4] A. Charfi and M. Mezini, "AO4BPEL: An aspect-oriented extension to BPEL," *World Wide Web*, vol. 10, no. 3, pp. 309–344, 2007. [Online]. Available: http://www.springerlink.com/index/10.1007/s11280-006-0016-3

[5] P. Mazzoleni and B. Srivastava, "Business driven SOA customization," in *ICSOC*, ser. Lecture Notes in Computer Science, vol. 5364, 2008, pp. 286–301. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-89652-4_23

[6] A. Schoneveld and M. Philip, "The service oriented club," white paper, Feb. 2007. [Online]. Available: http://www.slideshare.net/mphilip/the-service-oriented-club

[7] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-oriented domain analysis (FODA) feasibility study," Software Engineering Institute, Carnegie Mellon University, Tech. Rep. CMU/SEI-90-TR-21, Nov. 1990.

[8] "SAP NetWeaver Process Integration," SAP last accessed 15.03.2011. [Online]. Available: http://www.sdn.sap.com/irj/sdn/nw-pi71

[9] A. Lodhi, V. Köppen, and G. Saake, "An extension of bpmn meta-model for evaluation of business processes," *Scientific Journal of Riga Technical University, Computer Science*, vol. 46, pp. 27–34, 2011, presentation in the 10th International Conference on Perspectives in Business Informatics Research (BIR). [Online]. Available: https://ortus.rtu.lv/science/en/publications/10971/fulltext

[10] E. Maler, "Assertions and Protocol for the 2 OASIS Security Assertion Markup 3 Language (SAML) 4," *LANGUAGE (SAML)*, vol. 1, no. 17, p. 17, 2002.

[11] J. Steiner, C. Neuman, and J. Schiller, "Kerberos: An authentication service for open network systems," in *Proceedings Winter USENIX Conference*. Citeseer, 1988, pp. 191–201.

[12] Y. Sam, O. Boucelma, and M.-S. Hacid, "Web services customization: a composition-based approach," in *Proceedings of the 6th international conference on Web engineering*. New York, NY, USA: ACM, 2006, pp. 25–31.

[13] J. Jiang, A. Ruokonen, and T. Systa, "Pattern-based variability management in web service development," in *ECOWS '05: Proceedings of the Third European Conference on Web Services*. Washington, DC, USA: IEEE Computer Society, 2005, p. 83.

[14] S. Jiang, M. M. Shiaa, and F. A. Aagesen, "An approach for dynamic service management," in *Proceedings of IFIP WG 6.3 Workshop and EUNICE 2004 on Advances in fixed and mobile networks*, 2004.

[15] B. Benatallah, F. Casati, D. Grigori, H. R. M. Nezhad, and F. Toumani, "Developing adapters for web services integration," in *CAiSE*, ser. Lecture Notes in Computer Science, vol. 3520. Springer, 2005, pp. 415–429. [Online]. Available: http://dx.doi.org/10.1007/11431855_29

[16] F. Zambonelli and M. Viroli, "Architecture and metaphors for eternally adaptive service ecosystems," in *Intelligent Distributed Computing, Systems and Applications, Proceedings of the 2nd International Symposium on Intelligent Distributed Computing - IDC 2008, Catania, Italy, 2008*, ser. Studies in Computational Intelligence, vol. 162. Springer, 2008, pp. 23–32.

[17] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-oriented computing," *Communications of the ACM*, vol. 46, pp. 25–28, 2003.

[18] S. Apel, C. Kästner, and C. Lengauer, "Research challenges in the tension between features and services," in *Proceedings of the ICSE Workshop on Systems Development in SOA Environments (SDSOA)*. New York, NY, USA: ACM, May 2008, pp. 53–58. [Online]. Available: http://wwwiti.cs.uni-magdeburg.de/~ckaestne/sdsoa2008.pdf

[19] A. Khan, C. Kästner, V. Köppen, and G. Saake, "Service variability patterns," in *Advances in Conceptual Modeling. Recent Developments and New Directions, ER 2011 Workshops*, ser. Lecture Notes in Computer Science, O. De Troyer, C. Bauzer Medeiros, R. Billen, P. Hallot, A. Simitsis, and H. Van Mingroot, Eds., vol. 6999. Springer, Nov. 2011, pp. 130–140.

[20] A. Khan, C. Kästner, V. Köppen, and G. Saake, "Service variability patterns in SOC," School of Computer Science, University of Magdeburg, Magdeburg, Germany, Tech. Rep. 05, May 2011. [Online]. Available: http://wwwiti.cs.uni-magdeburg.de/iti_db/publikationen/ps/auto/KKKS11.pdf

[21] S. H. Chang, H. J. La, and S. D. Kim, "A comprehensive approach to service adaptation," in *SOCA '07: Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 191–198. [Online]. Available: http://dx.doi.org/10.1109/SOCA.2007.2

[22] S. Segura, D. Benavides, A. Ruiz-Cortés, and P. Trinidad, "A taxonomy of variability in web service flows," in *Service Oriented Architectures and Product Lines (SOAPL - 07)*, Kyoto. Japan, Sep. 2007.

[23] R. Mietzner, F. Leymann, and M. P. Papazoglou, "Defining composite configurable SaaS application packages using SCA, variability descriptors and multi-tenancy patterns," in *ICIW*. IEEE Computer Society, 2008, pp. 156–161. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/ICIW.2008.68