Service Architecture Design for E-Businesses: *A Pattern-Based Approach*

Veronica Gacitua-Decar and Claus Pahl

School of Computing, Dublin City University, Dublin 9, Ireland
{vgacitua,cpahl}@computing.dcu.ie

Abstract. E-business involves the implementation of business processes over the Web. At a technical level, this imposes an application integration problem. In a wider sense, the integration of software and business levels across organisations becomes a significant challenge. Service architectures are an increasingly adopted architectural approach for solving Enterprise Applications Integration (EAI). The adoption of this new architectural paradigm requires adaptation or creation of novel methodologies and techniques to solve the integration problem. In this paper we present the pattern-based techniques supporting a methodological framework to design service architectures for EAI. The techniques are used for services identification, for transformation from business models to service architectures and for architecture modifications.

1 Introduction

E-commerce, and more generally E-businesses, involve the implementation of business processes over the Web. The processes could span different organisations and include several software applications. At a technical level, the implementation of e-businesses imposes an enterprise applications integration (EAI) problem.

Nowadays, Service-oriented Architecture (SOA) is considered a promising architectural approach for EAI. Several methodologies such as [1],[2],[3] have been proposed for service architecture development. However, even though they provide useful guidelines, they are still maturing in aspects such as the providing of techniques promoting the automation of the architecture design process, and an adequate meta-modelling framework for capturing both, business and software aspects.

In [4], we developed a framework for designing service architectures for EAI. The framework is driven by business models and exploits patterns to support the design of architectures. Software patterns are considered in the software community as architectural abstractions representing encapsulated practical design knowledge [5]. Similarly, business reference models and business patterns provide encapsulated modelling knowledge [6]. The methodological framework guides the creation of architectures while incorporating the business and software dimensions of the integration problem. The approach provides explicit traceability between business and software modelling elements and uses patterns as central elements to provide improved changeability characteristics to the resultant architectures.

The objective of this paper is to describe the pattern-based techniques required for such a development framework. The framework is structured in a layered architecture,

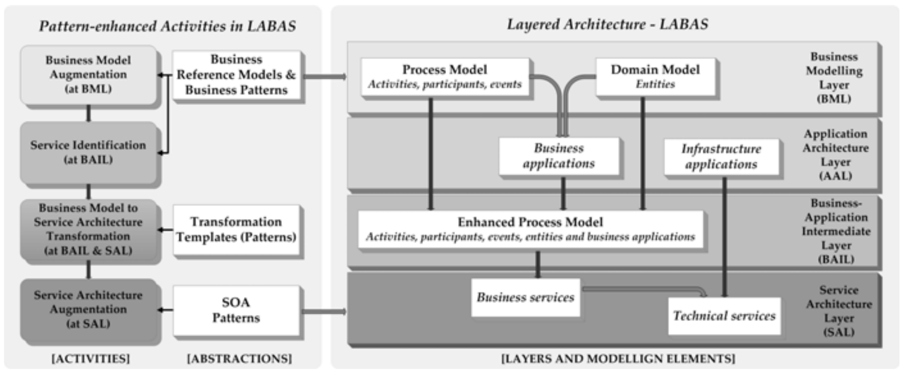
Pattern-enhanced Activities in LABAS
Business Model Augmentation (at BML)
Business Reference Models & Business Patterns
Service Identification (at BAIL)
Business Model to Service Architecture Transformation (at BAIL & SAL)
Transformation Templates (Patterns)
Service Architecture Augmentation (at SAL)
SOA Patterns
[ACTIVITIES]
[ABSTRACTIONS]
Layered Architecture - LABAS
Process Model
Activities, participants, events
Domain Model
Entities
Business Modelling Layer (BML)
Business applications
Infrastructure applications
Application Architecture Layer (AAL)
Enhanced Process Model
Activities, participants, events, entities and business applications
Business-Application Intermediate Layer (BAIL)
Business services
Technical services
Service Architecture Layer (SAL)
[LAYERS AND MODELLIGN ELEMENTS]

Fig. 1. LABAS and pattern-enhanced activities of the design process

their main elements are summarised in section 2. The pattern-based techniques are described in section 3. We finalise the paper with a discussion, revision of related work and conclusions in sections 4 and 5.

2 Layered Architecture: Layers and Architecture Abstractions

The framework provides a layered architecture called LABAS[1] for structuring the EAI problem [4]. The development of architectures solutions with LABAS involves the incremental transformation from business models into service architectures. The transformation is supported by architectural abstractions and pattern-based techniques. Fig.1-right side depicts the architecture layers and its elements. On the left side, the involved architecture abstractions and the main steps of the development processes are shown.

Layers. Layers in LABAS are the Business Modelling Layer (BML), the Application Architecture Layer (AAL), the Business-Applications Intermediate Layer (BAIL) and the Service Architecture Layer (SAL).

– *BML* is the container for the business model that represents the business context of the integration problem. Models in BML are expressed in an enhanced BPMN notation that includes domain model elements.
– *AAL* is the container for the application components supporting the business processes in BML. AAL is structured in an applications architecture model.
– *BAIL* is the container for an enhanced business process model that integrates the elements from BML and AAL, supported an explicit traceability model.
– *SAL* is the container for the service architecture that solves the integration problem. Services are categorised in two main types: *business services* and *technical services*.

[1] Layered Architecture for Business, Applications and Services.

Architectural Abstractions. In LABAS, business reference models, business patterns and SOA patterns are the main considered architectural abstractions.

Business reference models provide a standard model and decomposition of a business domain. They are originated from experience and combined with business patterns, they can constitute business reference architectures.

Business patterns are micro-models describing standard decompositions of business reference models. Similarly to software reference architectures, the business reference architectures constrain the composition of business patterns. Two main business patterns kinds are considered: *process patterns* and *domain patterns*. A further categorization regarding specific domains might be also considered in LABAS, but it is not mentioned in the paper.

SOA patterns are software design patterns that provide solutions for technical aspects of service architectures, such as service invoking, service composition, security, among others.

Business and SOA patterns have a semantic dimension and a structural dimension, however process patterns add a behavioural dimension. In this paper, we will refer only to structural aspects of patterns.

Layered architecture implementation. A meta-model and profile for the layers in LABAS is provided. Most LABAS constructs are mapped to UML 2.0 constructs. Traces in BAIL are an integral part of the LABAS profile and follows the trace-tagged traceability meta-model from [7]. The LABAS meta-model and profile for SAL are based on a proposal for the UML Profile and Meta-model for Services (UPMS) provided by the OMG.

Patterns are implemented and organised in pattern catalogues. Each pattern contains information organised in a *pattern template*, which includes sections such as intent, motivation, participants, consequences, associated quality attributes, among others. Usually, pattern templates involves textual descriptions [8]. We add the models representing the patterns into the pattern templates. The models representing the patterns use elements from the LABAS profile. This approach allows the interchange of the pattern catalogue and its elements, encouraging in this manner, the using of patterns as tool-supported modelling constructs.

Note that the implementation of the pattern-based techniques -described in the next section- will be part of a plug-in for a standard UML modelling tool. The plug-in is complemented with the LABAS profile, compliant with the LABAS meta-model.

3 Pattern-Based Techniques to SOA Design

The pattern-based techniques in LABAS provide support to business analysts and software architects to incrementally design service architectures. The main activities supported by pattern-based techniques are depicted in Fig.1-right side, and encompass:

– *Business model augmentation* involves the addition of business patterns into the business model. Recommendation of the using of patterns, pattern instantiation and patterns combination are techniques supporting this activity.

- *Business and technical service identification* involves the analysis of business models and their relations with the supporting software applications to define the services of the architecture solution. Patterns identification and patterns matching are two techniques aiding this activity.
- *Business model to service architecture transformation* incrementally generates the service architecture solution. This activity is supported by pattern-based transformation templates.
- *Service architecture augmentation* incorporates SOA patterns to provide solutions to technical issues such as communication between services, security, services distribution, among others. This activity is analogous to business model augmentation.

3.1 Business Model and Architecture Augmentation

Patterns represent experience-based solutions that can be applied to improve the quality of designs. The LABAS framework provides a repository of patterns in the form of pattern catalogues and the techniques that allows their use. This section describes the techniques allowing the use of patterns to augment business models and architectures.

Pattern recommendation. Less experienced designers might not be aware that a pattern might be applied to improve the quality of their designs. The utilisation of patterns into designs requires the recognition of the associated design problem. Tool support for pattern recommendation requires to increase the degree of automation for the pattern-problem identification[2]. If the pattern problem is expressed in terms of elements and relations of the design model, then a design problem could be systematically searched, and once located, it can be suggested to the designer to allow the subsequent instantiation of the associated pattern-solution. The modelling of the pattern problem is a key issue for this task [9].

In LABAS, business models, architectures and patterns are represented as graphs to support the business model and architecture augmentation activity. From a structural point of view, the identification of the pattern-problem is supported by identifying the structural features that individually characterise each pattern-problem model. These structural features are represented as subgraphs of the graph representing the pattern-problem model. The systematic searching of the structural features is supported by graph matching techniques.

Pattern comparison. A design problem could have more than one pattern solution associated. In this case, two or more patterns require comparison. Comparison is supported in LABAS with information about quality attributes associated to patterns. This information is encapsulated in the pattern consequences section of the pattern template.

Pattern modification. The documentation of a pattern-solution provides a *generic* solution. In order to instantiate a pattern in a specific model, the generic pattern-solution might require modifications to become suitable for the specific model. The preservation of the pattern properties requires that only allowed modifications can be done. The

[2] Note that a pattern description in a pattern template includes a generic problem-solution pair and its variations.

allowed modifications are implemented in a set of techniques that can be applied to modify patterns. For instance, a basic technique to modify the *mediator* pattern [8] involves the increasing or decreasing of *colleague* elements. After modification of the pattern, validation techniques are applied.

Pattern instantiation. Pattern instantiation involves the creation of elements and relations from the pattern model into a design model, and/or the merging of pattern elements with design model elements. In LABAS, patterns can be instantiated to augment models at BML and SAL layers.

In order to increase the degree of automation, once selected a pattern, its instantiation can be carried out through the implementation of a graph transformation rule $p : L \longrightarrow R$ [10], where L corresponds to the graph that represents the pattern-problem, and R corresponds to the graph that represents the pattern-solution. The transformation rule p allows the transformation from a graph that represents the model or architecture with a design problem, into a graph that represents the model or architecture with the instantiated pattern-solution.

Pattern combination. Architecture designs and business models can contain more than one pattern instance. Patterns can be combined to provide a design solution with a larger scope. Let us say that PAT_i and PAT_j are two patterns in a pattern catalogue, and $G_i = (V_i, E_i)$, $G_j = (V_j, E_j)$ are graphs[3] representing those patterns. Different types of pattern combinations might occur, depending on if $V_i \cap V_j = \emptyset$, where $\emptyset$ is an empty set; or if $V_i \cap V_j \neq \emptyset$. If the latter case occur and $PAT_i \subset PAT_j$, then PAT_i is *embedded* within PAT_j. In other cases the *union* of patterns is utilised. Union and embedding are to basic operations for pattern combination [11].

Requirements to combine patterns could exceed the capabilities of basic techniques. Only as an illustration, we use an analogy with relational algebra. Basic operations as projection (π) and selection (σ) in relational algebra are not enough in some practical uses for data base queries. Composition of operators is a solution to the restrictions of the basic operations, e.g. $\pi_A(\sigma_P(r))$ [4] allows the projection on a selected set of tuplas of a relation r. Analogously, the combination of pattern techniques provides a medium to satisfy more complex requirements.

Combination of patterns could interfere with the expected contributions that each pattern provides separately. An important issue in pattern combination is to verify that individual pattern consequences are preserved. This is difficult to ensure before implementation. However, at design-time, it is possible to analyse the potential interferences between associated quality attributes of each pattern.

3.2 Pattern-Based Service Identification and Incorporation

The explanation of this section uses an example extracted from a case study in [4]. The case study involves a billing and payment process, representing a typical process where customers and businesses interact. The example is focused on a payment transaction

[3] V_i and V_j are the set of graph vertices. E_i and E_j are the set of graph edges in G_i and G_j.

[4] A: attributes where the relation r is projected. P: logic predicate satisfied by tuplas in r.

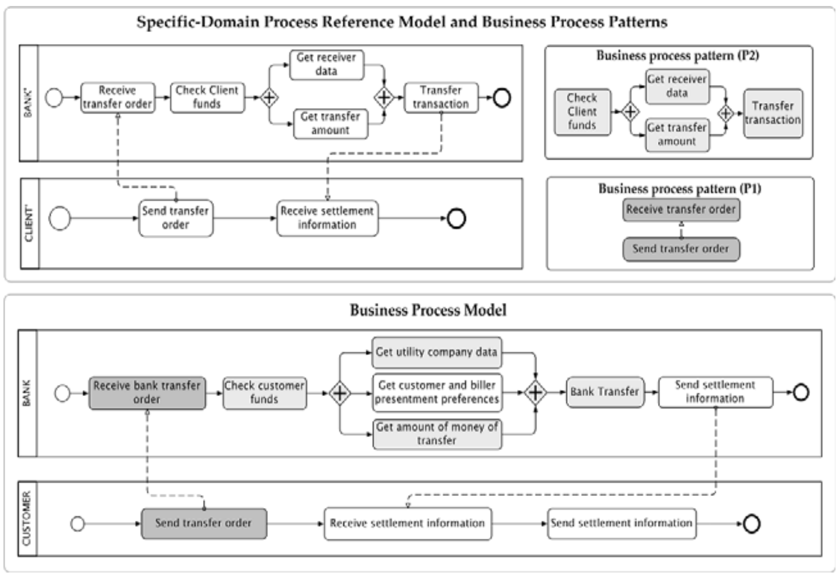

Fig. 2. Bank's transfer process modelled with the BPMN notation - right side. Business process reference model and patterns - left side.

- bank's transfer. Customers can send a bank's transfer order to pay their bills. Afterwards, the bank verifies the funds in the payer's account and executes the transaction. After the transaction is completed, the bank sends the settlement information to the payer. The customer in turn sends the settlement information to the biller. Fig.2-bottom shows the high level business process. A simplified reference model and two simplified examples of business process patterns (P1 and P2) are shown at the top of the figure.

Business Service Identification. The identification of business patterns facilitates the identification of reusable business services. Business patterns are reusable sections of business models, and they are a common denominator among different organisations and within the same organisation that is changing over time. Changeability is related to the ease of an architecture to change, but also with the ability of the architecture to remain invariant after a change agent acts [12]. The definition of business services in LABAS takes into account the latter characteristic.

In order to foment the automation of business pattern identification, *pattern matching* techniques are utilised. From a structural point of view, the pattern matching technique is based on the matching of a graph (G_{PAT}) representing the pattern over a graph G_{BM} representing the business model. The granularity of the pattern is such that $G_{PAT} \subseteq G_{BM}$. To identify a business pattern, and consequently a business service, an algorithm searches for the sub-graph G_{PAT} within the graph G_{BM}. Fig.2 shows a simplified schema of a reference process model, patterns and a business model containing those patterns.

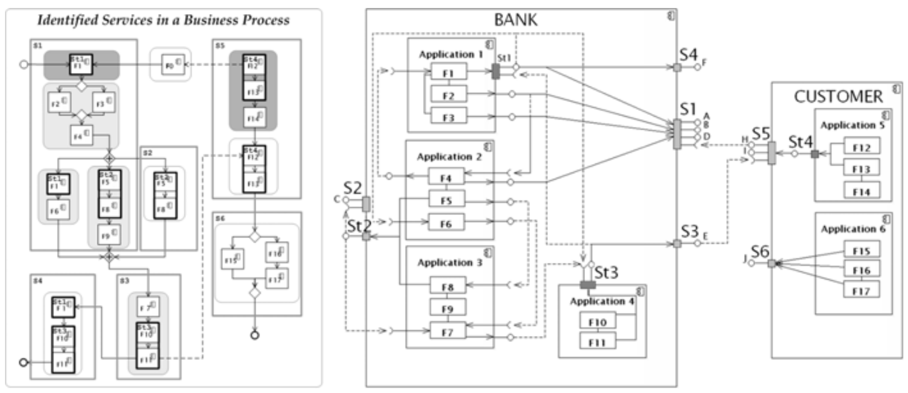

Fig. 3. Business and technical services identified in a enhanced business process model - left side. The related service architecture - right side.

Technical Service Identification. The identification of common flow structures in process models from BAIL is the basis for the identification of technical services. In order to identify common flow structures the process models from BAIL are decomposed until the activities with a one to one relation with applications components from AAL are reached. The flow structures have explicit traces with application components from AAL. The traces represent invocations to functionality of applications from AAL or responses from applications to fulfill information requirements of steps in the process flow. The identification of technical services across process models pursues the fundamental concept of reuse in SOA. The identification of common control flow structures is supported in LABAS by graph partitioning techniques. Graph representing enhanced process models have information about the types of their elements, therefore control flow structures involving certain types of elements can be further categorised as different technical services types, for instance: data aggregation, calculations, among others.

The Fig.3 sketches a simplified decomposition of the process model of the Fig.2. Note that the proper modelling notation of LABAS is not used because of space constraints. S1 to S6 represent the identified business services. S1 and S5 were defined through business pattern matching. St1 to St4 are the defined technical services through the identification of common flow structures in BAIL. Technical services can be reused by business services. Note that in order to simplify the illustration of technical services, only simple flow structures were depicted in the Fig.3. More complex structures involving other control flow structures, for instance decisions, joins, among others, can also be considered.

3.3 Business Model to Service Architecture Transformation

The transformation from business models into service architectures involves the tranformation of identified software services in BAIL to service elements in SAL. The transformation uses a pattern-based *transformation template* that maps, at meta-modelling level, architectural abstractions from BAIL into SAL. The relations among BAIL elements are preserved after the transformation. These relations provide information about the flow dependencies between business services. Fig.3-right side shows the software architecture model generated from the BAIL model depicted in the left side.

Note that several model driven development approaches have followed a strategy of direct translation from business modelling constructs to software constructs, e.g. direct transformation from BPMN-to-BPEL constructs. However, business models could contain sections that cause deadlocks and other problems for the process execution [13]. As mentioned previously, in LABAS, the transformation from business models to service architectures is based on pattern-based transformation templates. An advantage of using transformation templates is that they can be designed to provide only error-free transformations. Nevertheless, this require a previous step to refine the business process model to match with the business model section of the transformation template. In [14], a control-flow pattern approach for BPMN-to-BPEL translation is presented. The transformation templates in LABAS follows a similar approach, but beyond control-flow structures.

4 Related Work and Discussion

Different pattern-based techniques have been proposed for analysis, design and evolution of architectures. In [15], a systematic method to select patterns using language grammars and design space analysis is introduced. In a similar approach, a method to design solutions for transactional workflows for SOA is presented [16]. The design method models alternative solutions to recurrent architectural decisions, as patterns and primitives. These architecture abstractions are, in turn, mapped to technology-based solutions. Both approaches follow a manual-based approach for pattern selection and instantiation. In our case, the aim is provide support for (semi-)automation.

Sets of patterns are normally part of organised collections named pattern languages. Pattern languages allow regulated combinations that extends the reach of individual patterns [17]. In [18] a pattern language for process-oriented integration of software services is presented. We focus not only on patterns at software level, but also at business level and we investigate the relation of patterns at those two levels.

After software implementation, changes on it might interfere with the previously applied design patterns. In [19] a graph-transformation approach to pattern level design validation and evolution is presented. Only structural aspects are reviewed. In [20], current limitations of patterns on evaluating their impact on quality attributes is presented, however the interaction of the pattern's consequences have not been discussed. In [21], an ontological-based approach for modelling architecture styles is presented. Style modifications and combinations among them are introduced. Relations between quality requirements and modelling of styles are investigated. We are currently extending the ideas in [21].

A key step in the SOA development processes is to define what are the involved software services [1]. We presented an approach to service identification based on business models and related reference models. Development and maintenance of business and software architecture models, together with associated reference architectures and reference models, have been encouraged with the increasing use of enterprise architecture frameworks, such as for example the Zachman framework [22].

In this paper we have neglected process simulation and semantic considerations for pattern matching and pattern identification techniques. However, the integration of behavioral aspects and semantic aspects will be investigated. The integration of structural, behavioral and semantic aspects could consider the directions followed in [23] and [24]. To evaluate the architecture solutions created with LABAS we have considered the use of the Architecture-Level Modifiability Analysis (ALMA) method [25]. In [4] we have demonstrated the use the proposed layered architecture and discussed the use of ALMA for a modifiability analysis.

5 Conclusion

Traditionally, the creation of architectures have only focused on structural descriptions. Instead, we focus on processes and constrained architectural descriptions. The continual rise of abstraction in software engineering approaches was a central driver, placing the notion of patterns at business domain level and focusing on its subsequent transformation to a service architecture.

In this paper we have presented the notational elements and pattern-based techniques used in our methodological framework (LABAS) to design service architectures for EAI. The presented pattern-based techniques are utilised for software service identification, for business model to service architecture transformations and for architecture modifications. The LABAS approach has as one of its ultimate goals, the creation of service architecture solutions with improved changeability characteristics, while maintaining coherence between the business model and the software architecture. The explicit traceability between elements of the different layers of LABAS contributes to the coherence between the business level and the software level. Changeability characteristics of the architecture solutions are improved by using patterns and the pattern-based techniques described in this paper.

From the point of view of designing architectures, the integration of business aspects and software aspects required to implement E-businesses involves three main dimensions: structure, behaviour and semantic. The consideration of these three dimensions in a single framework to design architectures is challenging since these three dimensions have been mostly investigated separately and with different formalisms. We aim to integrate these dimensions to provide an integral support to the designers interested in the developing of service architectures.

References

1. Erl, T.: Service-oriented architecture: Concepts, Technology, and Design. Prentice Hall, Englewood Cliffs (2004)
2. Papazoglou, M.P., van den Heuvel, W.J.: Service-oriented design and development methodology. Int. J. of Web Engineering and Technology (IJWET) 2, 412–442 (2006)
3. Arsanjani, A.: Service-oriented modeling and architecture (2004)
4. Gacitua-Decar, V., Pahl, C.: Business model driven service architecture design for enterprise application integration. In: ICBIIT 2008 (2008)

5. Bass, L., Clements, P., Kazman, R.: Software Architecture in Practice, 2nd edn. Addison-Wesley Professional, Reading (2004)
6. Fettke, P., Loos, P.: Reference Modeling for Business Systems Analysis. IGI (2006)
7. Baelen, V.v., Berbers, J.: Traceability as input for model transformations. In: ECMDA Traceability Workshop (ECMDA-TW), Haifa, Israel (2007)
8. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional Computing Series (1995)
9. Kim, D.K., Khawand, C.E.: An approach to precisely specifying the problem domain of design patterns. J. of Visual Languages and Computing 18(6), 560–591 (2007)
10. Corradini, A., Montanari, U., Rossi, F., Ehrig, H., Heckel, R., Lowe, M.: Algebraic approaches to graph transformation. Handbook of Graph Grammars and Computing by Graph Transformation 1, 163–245 (1997)
11. Gomes, M.C., Rana, O.F., Cunha, J.C.: Pattern operators for grid environments. Sci. Program. 11(3), 237–261 (2003)
12. Ross, A., Rhodes, D., Hastings, D.: Defining changeability: Reconciling flexibility, adaptability, scalability, modifiability, and robustness for maintaining system lifecycle value. Journal of Systems Engineering (accepted, 2008)
13. Koehler, J., Gschwind, T., Kuster, J., Pautasso, C., Ryndina, K., Vanhatalo, J., Volzer, H.: Combining quality assurance and model transformations in business-driven development. In: AGTIVE 2007. LNCS. Springer, Heidelberg (2008)
14. Ouyang, C., Dumas, M., ter Hofstede, A.H.M., van der Aalst, W.M.P.: Pattern-based translation of bpmn process models to bpel web services. International Journal of Web Services Research (2007)
15. Zdun, U.: Systematic pattern selection using pattern language grammars and design space analysis. Software Practice and Experience 37(9), 983–1016 (2007)
16. Zimmermann, O., Grundler, J., Tai, S., Leymann, F.: Architectural decisions and patterns for transactional workflows in soa. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 81–93. Springer, Heidelberg (2007)
17. Buschmann, F., Henney, K., Schmidt, D.C.: Pattern-Oriented Software Architecture: On Patterns and Pattern Languages. Wiley and Sons, Chichester (2007)
18. Hentrich, C., Zdun, U.: Patterns for process-oriented integration in service-oriented architectures. In: EuroPLoP 2006, Irsee, Germany, pp. 1–45 (2006)
19. Zhao, C., Kong, J., Dong, J., Zhang, K.: Pattern-based design evolution using graph transformation. J. of Visual Languages and Computing 18(4), 378–398 (2007)
20. Harrison, N., Avgeriou, P.: Leveraging architecture patterns to satisfy quality attributes. In: Software Architecture. LNCS, pp. 263–270. Springer, Heidelberg (2007)
21. Pahl, C., Giesecke, S., Hasselbring, W.: An ontology-based approach for modelling architectural styles. In: Oquendo, F. (ed.) ECSA 2007. LNCS, vol. 4758, pp. 60–75. Springer, Heidelberg (2007)
22. Sowa, J.F., Zachman, J.A.: Extending and formalizing the framework for information systems architecture. IBM Syst. J. 31(3), 590–616 (1992)
23. Ehrig, M., Koschmider, A., Oberweis, A.: Measuring similarity between semantic business process models. In: APCCM 2007, Australia, vol. 67, pp. 71–80 (2007)
24. Martens, A.: Simulation and equivalence between bpel process models. In: Proc. of the Design, Analysis, and Simulation of Distributed Systems Symposium (DASD 2005) (2005)
25. Bengtsson, P., Lassing, N., Bosch, J., van Vliet, H.: Architecture-level modifiability analysis (alma). Journal of Systems and Software 69(1-2), 129–147 (2004)