Towards Integration of Semantically Enabled Service Families in the Cloud

Marko Bošković[1,2], Ebrahim Bagheri[1,3], Georg Grossmann[4], Dragan Gašević[1,2], Markus Stumptner[4]

[1]Athabasca University, Canada
{firstname.lastname}@athabascau.ca
[2]Simon Fraser University Surrey, Canada
[3]University of British Columbia, Vancouver, Canada
[4]University of South Australia, Australia
{georg|mst}@cs.unisa.edu.au

Abstract. Success of a Software Product Line (SPL) typically induces increase of requirements that expand over the expertise of its initial company. In the context of cloud computing, where SPLs are deployed in the form of business process families that are offered over the Internet, this expansion requires partnering with other available families. With the increasing number of companies that offer their solutions in the cloud, there is a need for tools and methods for integration of configurable business processes. In this position paper, we propose a methodology for integration that employs ontologies and Semantic Web technology, and propose a tool support that supports the proposed methodology.

1 Introduction

Motivated by the fact that different stakeholders have similar requirements, Software Product Line Engineering [1] (SPLE) argues the development of similar software systems as a whole, herewith sharing many assets and increasing reuse ability. An SPL is customized for every customer by selecting the set of most desirable features. Beside SPLE, Service-oriented Computing is another computing paradigm that promotes reuse where services enable rapid and easy composition of loosely coupled distributed software applications, and provide general computational elements that can be reused across different domains [2]. At this moment, there is a significant research for integrating these two software engineering paradigms, e.g. [3,4,5,6,7,8]. Recently, benefits of this synergy have been seen in the context of cloud computing [9], where synergistic solutions for service-oriented applications and SPLs are delivered over the Internet in the form of Business Process Families (BPFs) that are being configured for each user independently, while keeping BPFs, supporting systems software, hardware and maintenance, away from her [10].

The success of SPLs usually leads to their expansion that reaches a level that exceeds the innovation capabilities of one organization [11]. In such an expansion, companies converge different domains, often those that were not their primary business. In the context of BPFs in the cloud, this requires partnering of already existing BPFs. Therefore, there is a need for methods and tools for integration of BPFs.

Contemporary methods for integration of SPLs are mostly formal, and assume only feature equivalents across different families, e.g. [12,13,14,15]. However, in practice, because features are typically not equivalents, we consider integration of families as

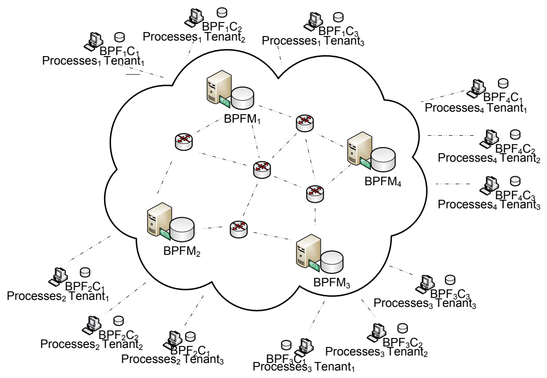

BPF$_1$C$_1$ Processes$_1$ Tenant$_1$
BPF$_1$C$_2$ Processes$_1$ Tenant$_2$
BPF$_1$C$_3$ Processes$_1$ Tenant$_3$
BPF$_4$C$_1$ Processes$_4$ Tenant$_1$
BPF$_4$C$_2$ Processes$_4$ Tenant$_2$
BPF$_4$C$_3$ Processes$_4$ Tenant$_3$
BPFM$_1$
BPFM$_4$
BPFM$_2$
BPFM$_3$
BPF$_2$C$_1$ Processes$_2$ Tenant$_1$
BPF$_2$C$_2$ Processes$_2$ Tenant$_2$
BPF$_2$C$_3$ Processes$_2$ Tenant$_3$
BPF$_3$C$_1$ Processes$_3$ Tenant$_1$
BPF$_3$C$_2$ Processes$_3$ Tenant$_2$
BPF$_3$C$_3$ Processes$_3$ Tenant$_3$

Fig. 1. Families of business processes in cloud computing (inspired by van der Aalst [10]). (BPFM=business process family model, BPFC=configuration)

an engineering task that cannot be fully automated. Therefore, we provide a method and propose a tool support that heavily uses ontologies [16] and Semantic Web technologies [17] for semantic annotation of BPFs, that can be used to automatically derive interdependencies and allows for semi-automated integration.

2 The Proposed Method

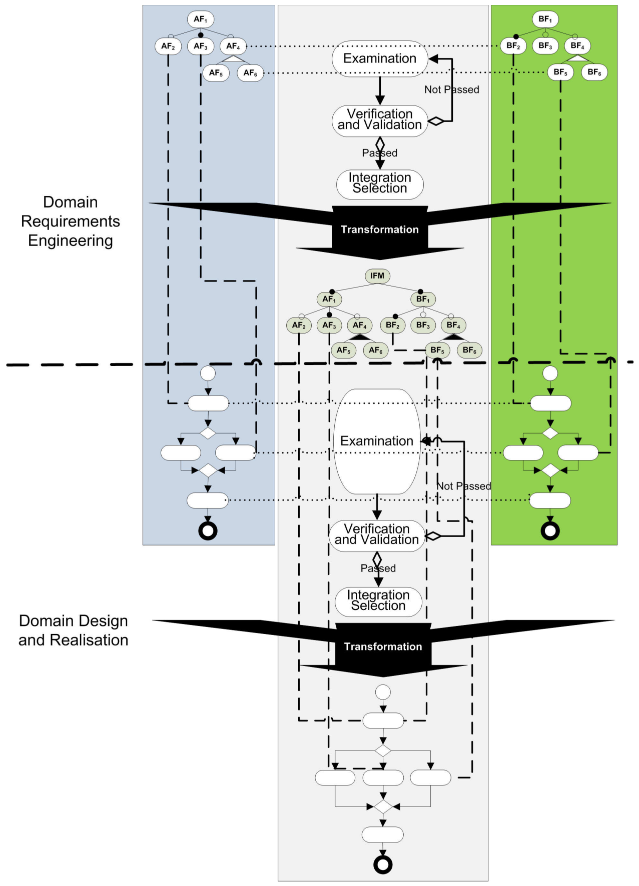
Cloud computing is an emerging computing paradigm that promotes delivery of applications to users as services over the Internet while keeping the hardware, systems software and system maintenance away from her [9]. Therefore, each BPF in the cloud is distributed and independently deployed [10], as illustrated in (Figure 1).

Each BPF is specified with Business Process Family Models (BPFMs) consisting of artifacts specifying the problem space, the solution space, and the mappings between problem and solution spaces [18]. The solution space is typically a Business Process Model Template (BPMTs) [19], i.e., superimposition of all business process variants. The problem space, on the other hand, represents all possible features of family members and typically is captured with feature models, a tree-like structure [20]. A BPF is configured for each user by selecting the desired features of the family. A feature selection, with the help of mappings, forms the final business process for a particular user. In the context of BPFs in the cloud, problem, solution, and mapping models are deployed to an external location on the Internet, while each tenant has his own customized configuration of the family, as shown in Figure 1.

SPLE generally consists of two life-cycles: Domain Engineering (DE) and Application Engineering (AE) [21][1]. In short, DE aims at the development of common assets (e.g. models, components, documentation) and configuration knowledge (typically feature models and mappings). AE is dedicated to the selection of appropriate features.

Our integration methodology considers integration of BPFs as a form of the DE. It builds upon the framework proposed by Linden et al. [21] and is depicted in Figure 2.

DE consists of *requirements engineering*, *domain design*, *domain realization* and *domain testing*. In our methodology, *requirements engineering* results in a fully inte-



Examination
Not Passed
Verification and Validation
Passed
Integration Selection
Domain Requirements Engineering
Transformation
IFM
Domain Design and Realisation

Fig. 2. Integration of families of service-oriented systems

grated feature model, while the *domain design and domain implementation* are one phase that results in integrated BPF. *Domain testing* is out of the scope of this paper.

In the *requirements engineering* phase we propose the following activities:

1. Examination of relationships between features of independent families. For example, in the integrated feature model, we can have features of different families that represent identical business processes by intention, but their actual realizations (extensions) are different. Some other examples of relationships are that can be found is that features represent business processes of different families with the same intension and extension (they use the same service), or that they are history related, meaning that one business process must be executed before the other one. We base our relationships on the ones identified by Grossmann et al. [22,23] (More on the relationships and their integration options can be found at: https://files.semtech.athabascau.ca/public/TRs/TR-SemTech-03052011.pdf). To automate this recognition we employ ontologies and Semantic Web technologie;

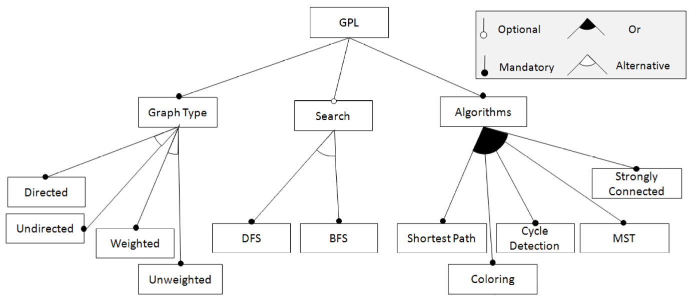


Fig. 3. A feature model of Graph Product Line

2. Verification and Validation of relationships. In the process of defining integrated feature models, there is a need for the validation of relationships between features with target customers and developers of different families, and verification whether the relationships are well specified, e.g., to recognize whether there are inconsistencies in the integrated feature model;
3. Integration selection is an activity where an integration engineer selects the appropriate choices for integration. Every relationship between features does not uniquely specify the configuration relationship, but rather provides a set of possible choices. For example, the integration engineer might choose to have in the integrated feature model, two features that are identical by intension but different by realization (extension). In such a case, the integration engineer might also choose to allow for mutually exclusive configuration, or that both can appear in the final application. The integration engineer selects this relationship from the set of available configuration relationships ^{0};
4. Transformation is the final activity, where selected integration patterns and initial feature models are inputs, and output is a feature model of the integrated families.

In the context of integration, *domain design* and *domain implementation* are the same phase, because the outcome is a business process template of already implemented families. We propose the same activities as in the *requirements engineering* model, namely: 1)**Examination** of relationships between business processes in business process models as proposed by Grossmann et al [22,23]; 3) **Verification and Validation** of relationships for semantic and well-formedness; 3) **Integration selection**, i.e., the selection of predefined integration options(e.g., the services with the same intention but different extension can be integrated in a way that at runtime their results are accumulated or that exactly one can be executed); 4) **Transformation** from input BPMTs to the integrated BPMT.

3 Foundations

3.1 Feature Modeling

A feature model is a means for representing the possible configuration space of all the products of a system product line (system family) in terms of its features. Typically, feature models are represented with feature diagrams in the form of a tree whose root node represents a domain concept, e.g., a domain application, and the other nodes concept property, e.g., domain application functionality, modeled in a way to capture commonalities and variability among product family variants. The rest of features are classified as:

- **Mandatory** feature: the feature must be included in a product if its parent feature is selected.
- **Optional** feature: the feature may or may not be included if its parent is selected.
- **Or feature group**: from a set of Or feature group, any non-empty subset of features can be included if their parent feature is selected.
- **Alternative feature group**: from a set of alternative features, only one feature can be included if their parent feature is selected.

Additional constraints are defined on the feature models, named integrity constraints. Two main constraints are: **includes** – selection of a given feature requires the inclusion of another feature; and **excludes** – that specifies mutual exclusion of two features. An example of a feature model of Graph Product Line is given in Figure 3.

3.2 Semantically-enhanced Business Process Model Templates

As previously stated, a Business Process Model Templates is a superimposition of all members of a BPF [19]. Web services are seen as main means for operationalization of business processes and accordingly, BPMTs [2,24].

The main characteristic of Web services is that they can be deployed over large scale networks such as the Web; hence, they need to and indeed carry machine processable descriptions that properly inform other programs of their operations and how they can be properly invoked. One of the limitations of contemporary Web services is that their description lacks meaningful explanations or in other words semantic descriptions. Semantic Web services add capability of describing structural and behavioral semantics to Web services by providing the means to expressively annotate Web services with shared conceptualizations in the form of ontological concepts [25]. Ontologies provide agreed upon and formal domain specifications [16] based on Semantic Web markup languages such as OWL and DAML and are shared by different software systems and applications. Not only does this sharing of knowledge allow software systems to search for suitable Web services based on syntactical matches, but to also consider semantic relevance within the matchmaking process. BPMTs that use Semantic Web Services as operationalizations, are called Semantically-enhanced Business Process Model Templates.

4 The proposed tool support

4.1 Feature Model Representations

Several different formats for storing and manipulating feature models have been proposed in the software product line community including XToF, SXFM and TVL [26]. Although the representations of these serializations are different, the semantics of the languages are quite similar and they can be easily transformed to one another. Within our framework, Figure 4, we model feature models with *Semantic Annotations for Feature Model Description Language* (SAFMDL), and serialize them as profile for feature

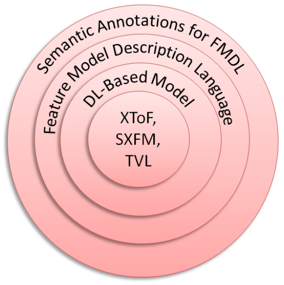

Semantic Annotations for FMDL
Feature Model Description Language
DL-Based Model
XToF,
SXFM,
TVL

Fig. 4. The Onion Architecture for SAFMDL

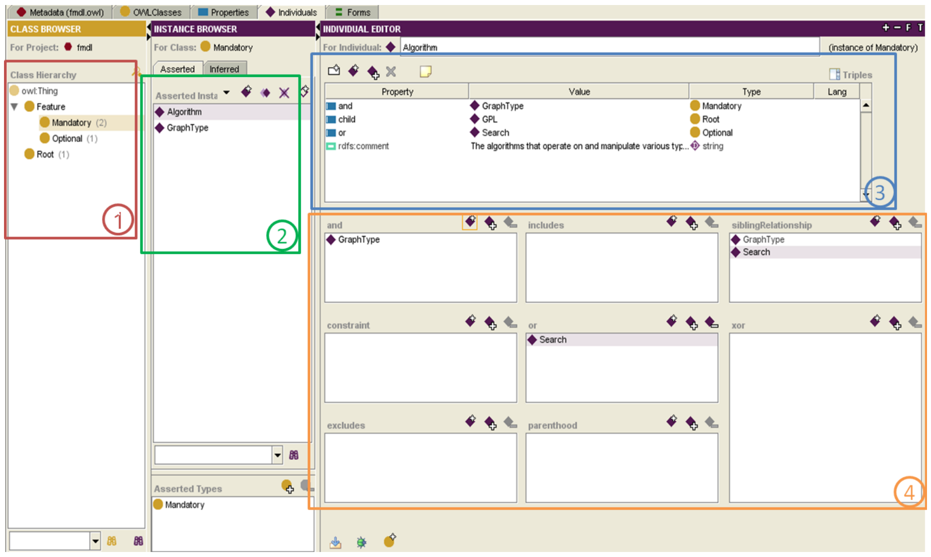
modeling based on Web Ontology Language (OWL). However, AUFM Suite that supports our framework, provides mechanisms to convert to and from other serialization into SAFMDL.

As shown in Figure 4, the core of SAFMDL is a Description Logic based model specified with Feature Model Description Language (FMDL). FMDL is a feature modeling profile that provides the standard concepts for developing a feature model. It is modeled based on OWL and can essentially be seen as an ontology for feature modeling. The structure of FMDL consists of the required concept and property definitions for instantiating a feature model, which corresponds to the feature modeling meta-model. The instantiation of the feature modeling meta-model is performed by providing ontology individuals (concept instances) for the FMDL concept definitions. Figure 5 depicts the details of FMDL in the Protégé ontology editor. FMDL feature models can also be developed within our AUFM Suite, which is a graphical Eclipse plugin for feature modeling.

As shown in Figure 5, the structure of a feature model is based on the two main concepts of Root and Feature, and two of its sub-concepts Mandatory and Optional. These concepts are shown on the Class Hierarchy panel (Box 1). A new feature model can be instantiated by providing individuals for each of these concepts. For instance, the Algorithm and GraphType features have been added to this feature model as mandatory features (Box 2). The relationships between the features are modeled through properties. The list of possible relationships between the features of a fetture model is shown in Figure 3 (Box 4). For instance, it can be seen that *Algorithm* has a *siblingRelationship* with both *GraphType* and *Search* features. It can also be seen that the *Root* of the feature model is named *GPL* (Graph Product Line) and that *Algorithm* is one of the direct children of the *Root* (Box 3).

The benefit of using DL-based feature models is that standard DL reasoning mechanisms can be used to derive and validate feature model configurations and also extended DL algorithms can even be used to detect and resolve inconsistencies within feature models. Besides the exact syntax and semantics of FMDL, it provides an additional advantage of providing grounds for being extended with additional capabilities without requiring structural changes. Since, FMDL is based on OWL, additional information or data can be added to it through the introduction of new *Class* or *Property* definitions. This has been exploited to further extend FMDL to support the semantic annotation of its elements, referred to as SAFMDL.

SAFMDL profile introduces three new properties that reference concepts within external shared ontologies. These additional properties, *selfModelReference*, *preCon-*



Property	Value	Type	Lang
and	GraphType	Mandatory	
child	GPL	Root	
or	Search	Optional	
rdfs:comment	The algorithms that operate on and manipulate various typ...	string	

Fig. 5. The Structure of FMDL in Protégé

```
<fmdl:Mandatory rdf:ID="Algorithm">
  <safmdl:selfModelReference rdf:resource="http://monet.nag.co.uk/algorithm#Algorithm"/>
  <fmdl:or rdf:resource="#Search"/>
  <fmdl:and rdf:resource="#GraphType"/>
</fmdl:Mandatory>
```

Fig. 6. Semantically annotating feature model elements

ditionModelReference, and *postConditionModelReference* allow each feature to be further described by presenting what concept or notion the feature represents in the domain of discourse, what other notions it relies on for being realized, and which other concepts will be impacted by this feature, respectively. Given this capability, SAFMDL allows designers to qualify their design with meaning and hence avoid ambiguity and enhance communication, model sharing, better model realization, and finally integration. Figure 6 depicts an example where the Algorithm feature is grounded/annotated using the *Algorithm* concept within the MONET ontology. MONET is an ontology for describing and provisioning web-based mathematical services. With this annotation, the Algorithm feature is now confined with the semantic meaning attached to Algorithm in MONET and its scope is restricted by what defined clearly in that ontology.

4.2 Connecting Problem and Solution Spaces

As discussed earlier, we will need to move from the problem space (i.e., the feature model) into the suitable solution space (i.e., BPMTs). The main challenge towards the operationalization is to find the right Web services that both syntactically and semantically implement features that are available in a feature model.

Given that SAFMDL provides the means for describing feature models with semantic descriptions, it is possible to create a correspondence between the problem space feature models and solution space semantic Web services. The semantic descriptions shared between both spaces can be seen as glue that can enhance the discovery of the most appropriate services for realizing the abstract software applications. In order to operationalize abstract product representations of the problem space, here are three sources of information that need to be completely integrated, namely 1) semantically annotated feature models; 2) semantically annotated Web services; 3) the sources of the semantic information,

These three sources of information are either expressed in a valid XML format or through some extensions of the RDF triple format; therefore, appropriate XSPARQL [27] queries can consolidate these sources of information and provide for the realization of problem space models using Semantic Web services. If we return to the example from Figure 6, and assume that a set of Web services is available to us that are annotated using SAWSDL [28] with concepts from subsets of the MONET ontology. In the example in Figure 7, we show that the Search feature from GPL can be operationalized using XSPARQL. As seen in the process shown in Figure 7, the first step is to extract the semantic annotation that describes the feature of interest (♣). This will provide the basis to search for Web services that are also annotated similarly. The valuable aspect of ontological semantic descriptions is that they provide meaningful hierarchical relationships; therefore, even if two concepts are not identical, they can still be related lower down or higher up the subsumption hierarchy. Concepts below another concept in the hierarchy can be seen as further specializations of that concept and can hence be relevant in the matchmaking process. For this reason, it is reasonable to look for Web services that are either directly annotated with the semantic annotation of the feature of interest or other concepts that are below it in the hierarchy (♥). The last step is to explore the set of available Web services that are annotated with acceptable ontological concepts (♠) using a suitable query. The outcome of this query is a list of Web services that have appropriate matching semantic descriptions to the feature of interest (Search).An expert designer or software developer would then need to review the matches and select the best one to operationalize that feature. In Figure 7, we have only checked for matches based on *sfmdl:selfModelReference* to save space but in reality checks also should be put in place for pre and postconditions as well.

4.3 Recognizing Relationships

As previously mentioned, by employing ontologies and Semantic Web technologies, all of our artifacts (feature models and business process templates) are annotated with semantic descriptions. These semantic descriptions can also be used to automatically derive the integration relationships between different features. For example, a similar query to the one presented in Figure 6, can be used to search for the features representing the identical business process. The only change in the query is that it should search for the exact match of the *Algorithm* concept.

In order to provide automatic recognition of interrelationships between feature models, we intend to provide a library of XSPARQL Queries, that automatically recognize relationships between features in feature models and business processes in business process templates. These queries are intended to be triggered in the *Elicitation* phase of the *Domain Requirements Engineering* for identifying of relationships between features in different feature models and in *Domain Design and Implementation* for finding of relationships between business processes in business process templates. Furthermore, more soffisticated ontology based techniques for automatic recognition can be employed, like the ones used for service matchmaking [29,30] and business process matchmaking [31] based on similarity metrics proposed by Dijkman et al. [32].

4.4 Implementation Aspects

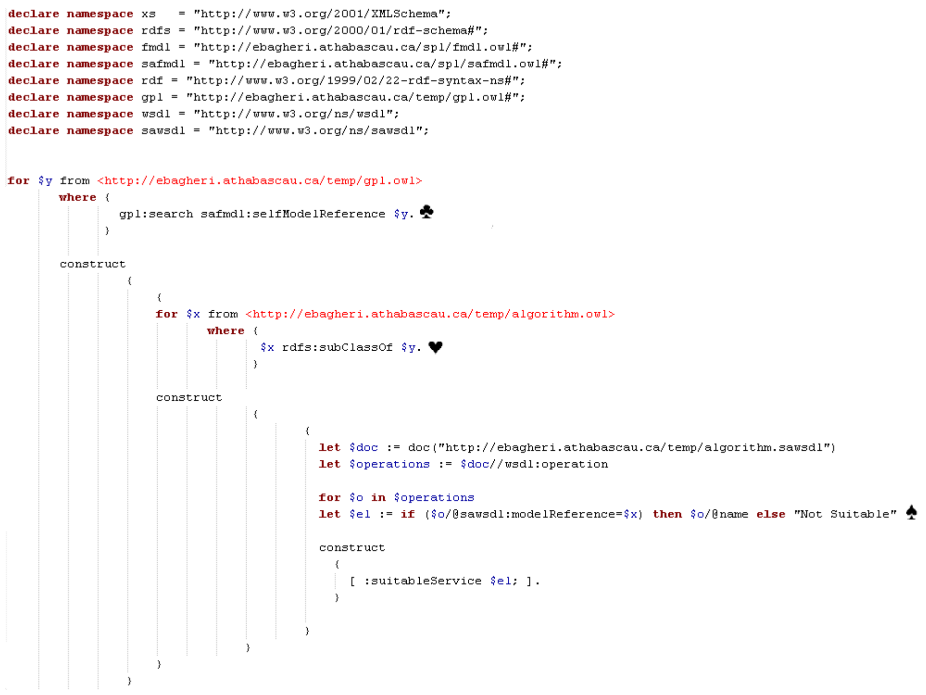
To support software developers for working with our proposed framework, we have started to implement the AUFM Suite - a chain of Eclipse based tools for development of Semantically-enhanced Families of Business Processes. So far, we have implemented the following tools:

- SAFDML Editor: This tool provides ontology representation of feature models, as described in Section 4.1
- rBMPN tool: for modeling the composition of features (represented as activities) using Business Process Modeling Notation 2.0 (BPMN2). Additionally, the tool provides facilities for modeling business rules over BPMTs.
- S-AHP tool: This tool goes beyond the work presented in this paper, and is used in AE phase of integrated BPF. The tool captures stakeholders' preferences in the terms of relative importance, and ranks features according using the implementation of our S-AHP algorithm [33].
  For the next stage of our development, we are working to integrate the XSPARQL language with our tooling support for formulating and executing queries on the repository of semantic Web services.

5 Related Work

Up to this day, several formal approaches exist for composition of feature models and solution space models. Feature model composition has been a topic of Acher et al. [15] and Segura et al. [14]. Acher et al. [15] introduce a domain-specific language for integration of feature models with operators for merging and inserting. Segura et al. [14], introduce an approach for automated merging of feature models, using graph transformations. In their work they provide a set of rules, and with the means of graph transformation, they perform the merge. Beside the fact that both of these approaches are focused only on feature models, they are formal and assume that there is a semantic equality between features that are merged. Our work takes an engineering perspective and assumes that there can be also different levels and semantics of equivalence. Due to this fact, our approach is semi-automated, and does not take the developers out of the process of integration. Rather it is an interactive process, where developers specify the semantic interrelationships and choose between different integration options.

Similar to formal composition of feature models, there are several approaches for formal composition of features in solution space models. Batory et al. [34] has introduced also an algebraic framework for specification of composition of features. Similarly, Erwig et al [12] also introduces a formal calculus for composition of different features in solution space models. However, our work, goes beyond these approaches and provides a semantics-based composition. Furthermore, Batory et al. and Erwig et al. focus on composing features of a single SPL, while in our work we focus on integration of SPLs. Finally, Apel et al. [13][35] introduces a feature algebra for language independent feature compositions. A feature is represented as a feature structure tree (FST), a language independent representation of a subset of the abstract syntax trees.



```
declare namespace xs   = "http://www.w3.org/2001/XMLSchema";
declare namespace rdfs = "http://www.w3.org/2000/01/rdf-schema#";
declare namespace fmdl = "http://ebagheri.athabascau.ca/spl/fmdl.owl#";
declare namespace safmdl = "http://ebagheri.athabascau.ca/spl/safmdl.owl#";
declare namespace rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#";
declare namespace gpl = "http://ebagheri.athabascau.ca/temp/gpl.owl#";
declare namespace wsdl = "http://www.w3.org/ns/wsdl";
declare namespace sawsdl = "http://www.w3.org/ns/sawsdl";


for $y from <http://ebagheri.athabascau.ca/temp/gpl.owl>
       where {
               gpl:search safmdl:selfModelReference $y. ♣
              }

       construct
                 {
                     {
                     for $x from <http://ebagheri.athabascau.ca/temp/algorithm.owl>
                            where {
                                   $x rdfs:subClassOf $y. ♥
                                  }

                     construct
                              {
                                      {
                                       let $doc := doc("http://ebagheri.athabascau.ca/temp/algorithm.sawsdl")
                                       let $operations := $doc//wsdl:operation

                                       for $o in $operations
                                       let $el := if ($o/@sawsdl:modelReference=$x) then $o/@name else "Not Suitable" ♠

                                       construct
                                         {
                                           [ :suitableService $el; ].
                                         }

                                      }
                              }
                     }
                 }
```

Fig. 7. A sample of XSPARQL Query for mapping problem and solution spaces

With this algebra, when two features are composed, they are merged only in the case when they have the same name and type. Our integration, goes beyond just a name and type based integration and facilitates semantics based integration.

To our knowledge, van Ommering was the first to observe composition (integration) of SPLs from the engineering perspective. In his work [36][37][38], he has introduced a notion of product populations, a set of SPLs whose members share many commonalities. In such context, (semi-) independent SPLs are developed by separated intra-organizational teams and later integrated into one variant rich product population. To support development of product population, van Ommerling introduces a lock-step process and a component model. The component model supports integration with the means of glue code. Our specification of interrelationships goes beyond glue code, and enables semantic based specification of interrelationships and semi-automated integration based on these semantic correspondences.

Recently, Bosch et al. [39] have proposed different process models for development and integration of SPLs in various global software engineering contexts. Our work focuses on the technical level of integration and can be applied in all engineering processes proposed by Bosch et al.

6 Conclusions and Future Work

In this paper, we have described a semantically enabled approach to the integration of Service Families in the Cloud. This task is a challenge specific to a leading edge environment where software engineering techniques are currently breaking new grounds along multiple dimensions: business processes evolve into service processes dynamically deployed in the Cloud; software product lines evolve into service families, with feature models being used to describe a more dynamic and flexible architectural style; integration technologies developed for business processes need to be extended to fit the service environment and so provide high level tool support in situations where traditional methods could not keep up.

We have described how a business process integration technology based on the semantic classification of correspondences and selection of integration patterns can be adapted to service families by using a process fragment classification approach for the extended feature models describing the services. Furthermore, we demonstrate how ontologies and Semantic Web technologies can be employed to automatically identify correspondences between business processes and features. We have given an example and described the tool support that can be employed for these tasks.

In the future, we are going to focus on completing the tool support and evaluation of the approach by applying it on realistic case studies.

Acknowledgments. This research was in part supported by Alberta Innovates – Technology Futures through the New Faculty Award program,

References

1. Pohl, K., Böckle, G., van der Linden, F.J.: Software Product Line Engineering: Foundations, Principles and Techniques. Springer (2005)
2. Papazoglou, M.: Whats in a service? In Oquendo, F., ed.: Software Architecture. Volume 4758 of LNCS. Springer (2007) 11–28
3. Lee, J., Muthig, D., Naab, M.: A feature-oriented approach for developing reusable product line assets of service-based systems. J. of Systems and Software **83**(7) (2010) 1123–1136
4. Medeiros, F.M., de Almeida, E.S., Meira, S.R.L.: SOPLE-DE: An Approach to Design Service-oriented Product Line Architectures. In: Proc. of the 14th Int. Conf on SPLsd. SPLC'10, Springer (2010) 456–460
5. Koning, M., Sun, C.a., Sinnema, M., Avgeriou, P.: Vxbpel: Supporting variability for web services in bpel. Inf. Softw. Technol. **51** (2009) 258–269
6. van der Aalst, W.M.P., Dreiling, A., Gottschalk, F., Rosemann, M., Jansen-Vullers, M.H.: Configurable process models as a basis for reference modeling. In: Proc. of BPM 2005 Workshops. Volume 3812 of LNCS. (2005) 512–518
7. Boffoli, N., Cimitile, M., Maggi, F.M.: Managing business process flexibility and reuse through business process lines. In: ICSOFT 2009 - Proc. of the 4th Int. Conf. on Software and Data Technologies, Vol. 2, Springer (2009) 61–68
8. Schnieders, A., Puhlmann, F.: Variability mechanisms in e-business process families. In: Proc. of the 9th Int. Conf. on BIS. Volume 85 of LNI., GI (2006) 583–601
9. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: A View of Cloud Computing. Communications of ACM **53**(4) (April 2010) 50–58
10. van der Aalst, W.: Configurable services in the cloud: Supporting variability while enabling cross-organizational process mining. In: On the Move to Meaningful Internet Systems: OTM 2010. Volume 6426 of LNCS. Springer (2010) 8–25
11. Bosch, J.: The challenges of broadening the scope of software product families. Communications of ACM **49** (December 2006) 41–44
12. Erwig, M., Walkingshaw, E.: The choice calculus: A representation for software variation. ACM Trans. on SE and Methodology (to appear)

13. Apel, S., Lengauer, C., Mller, B., Kstner, C.: An algebra for features and feature composition. In: Alg. Meth. and Softw. Technology. Volume 5140 of LNCS. Springer (2008) 36–50
14. Segura, S., Benavides, D., Ruiz-Cortés, A., Trinidad, P.: Automated merging of feature models using graph transformations. In Lämmel, R., Visser, J., Saraiva, J.a., eds.: Gen. and Transf. Techniques in SE, Springer (2008) 489–505
15. Acher, M., Collet, P., Lahire, P., France, R.: Composing feature models. In: 2nd Int. Conf. on SLE (SLE 2009). Volume 5969 of LNCS., Springer (2010) 62–81
16. Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing. Int. J. on Human-Computer Studies **43** (December 1995) 907–928
17. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American (May 2001)
18. Czarnecki, K., Eisenecker, U.W.: Generative programming: methods, tools, and applications. ACM Press/Addison-Wesley Pub. Co. (2000)
19. Czarnecki, K., Antkiewicz, M.: Mapping features to models: A template approach based on superimposed variants. In: Proc. of the Int. Conf. on GPCE. LNCS, Springer (2005) 422–437
20. Kang, K., Cohen, S., Hess, J., Novak, W., Peterson, A.S.: Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute (1990)
21. Linden, F.J.v.d., Schmid, K., Rommes, E.: Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering. Springer (2007)
22. Grossmann, G., Ren, Y., Schrefl, M., Stumptner, M.: Behavior based integration of composite business processes. In: BPM. Volume 3649 of LNCS. Springer (2005) 186–204
23. Grossmann, G., Ren, Y., Schrefl, M., Stumptner, M.: Definition of business process integration operators for generalization. In: ICEIS (3), Springer (2005) 510–517
24. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer (2007)
25. McIlraith, S.A., Son, T.C., Zeng, H.: Semantic web services. IEEE Int. Sys. **16** (2001) 46–53
26. Boucher, Q., Classen, A., Heymans, P., Bourdoux, A., Demonceau, L.: Tag and prune: a pragmatic approach to software product line implementation. In: Proc. of the IEEE/ACM Int. Conf on Aut SW Eng. ASE '10, ACM (2010) 333–336
27. Akhtar, W., Kopecký, J., Krennwallner, T., Polleres, A.: Xsparql: traveling between the xml and rdf worlds - and avoiding the xslt pilgrimage. In: Proc. of the 5th European Sem. Web Conf. ESWC'08, Springer (2008) 432–447
28. Kopecký, J., Vitvar, T., Bournez, C., Farrell, J.: Sawsdl: Semantic annotations for wsdl and xml schema. IEEE Internet Computing **11** (November 2007) 60–67
29. Kiefer, C., Bernstein, A.: The creation and evaluation of isparql strategies for matchmaking. In: Proc. of the 5th European Sem. Web Conf. ESWC'08, Springer (2008) 463–477
30. Klusch, M., Kaufer, F.: Wsmo-mx: A hybrid semantic web service matchmaker. Web Intelli. and Agent Sys. **7** (January 2009) 23–42
31. Kiefer, C., Bernstein, A., Lee, H.J., Klein, M., Stocker, M.: Semantic process retrieval with isparql. In: Proc. of the 4th E. Conf. on the Sem. Web. ESWC '07, Springer (2007) 609–623
32. Dijkman, R.M., Dumas, M., van Dongen, B.F., Käärik, R., Mendling, J.: Similarity of business process models: Metrics and evaluation. Inf. Syst. **36**(2) (2011) 498–516
33. Bagheri, E., Asadi, M., Gašević, D., Soltani, S.: Stratified analytic hierarchy process: Prioritization and selection of software features. In: The 14th Int. SPL Conf., Springer (2010)
34. Batory, D., Sarvela, J.N., Rauschmayer, A.: Scaling step-wise refinement. In: Proc. of the 25th ICSE. ICSE '03, IEEE Computer (2003) 187–197
35. Apel, S., Lengauer, C., Möller, B., Kästner, C.: An algebraic foundation for automatic feature-based program synthesis. Sci. Comp. Prog. **75**(11) (2010) 1022–1047
36. van Ommering, R., Bosch, J.: Widening the scope of software product lines from variation to composition. In: Software Product Lines. Volume 2379 of LNCS. Springer (2002) 31–52
37. van Ommering, R.: Building product populations with software components. In: Proc. of the 24th ICSE Conf. ICSE '02, ACM (2002) 255–265
38. van Ommering, R.: Software reuse in product populations. IEEE Transactions on Software Engineering **31** (July 2005) 537–550
39. Bosch, J., Bosch-Sijtsema, P.: From integration to composition: On the impact of software product lines, global development and ecosystems. J. of Sys. and Softw. **83**(1) (2010) 67–76