MIGRATING LEGACY SYSTEMS TO A SERVICE-ORIENTED ARCHITECTURE WITH OPTIMAL GRANULARITY

Saad Alahmari, Ed Zaluska and David De Roure
School of Electronics and Computer Science University of Southampton, Southampton, U.K.

Keywords: Service-Oriented Architecture, Service Granularity, Software Evolution, Legacy Systems, Architecture Modelling.

Abstract: The enhanced interoperability of business systems based on Service-Oriented Architecture (SOA) has created an increased demand for the re-engineering and migration of legacy software systems to SOA-based systems. Existing approaches focus mainly on defining coarse-grained services corresponding to business requirements, and neglect the importance of optimising service granularity based on service reusability, governance, maintainability and cohesion. An improved migration of legacy systems onto SOA-based systems requires identifying the 'right' services with an appropriate level of granularity. This paper proposes a novel framework for the effective identification of the key services in legacy code to provide such an optimal mapping. The framework focuses on identifying these services (based on standardized modelling languages UML and BPMN) and provides effective guidelines for identifying optimal service granularity over a wide range of possible service types.

1 INTRODUCTION

The difficulty of accommodating technology evolvement and rapid business changes is a significant problem with existing legacy software systems. In the past, software systems were typically developed with embedded business rules and logic, scattered and duplicated code, unstructured modules, and tightly-coupled functions. Such legacy systems nevertheless often represent a considerable investment by the underlying business which will frequently rely on the software for many day-to-day business activities. Service-Oriented Architecture (SOA) is a modern approach to implementing (and re-implementing) such systems as a set of robust and interoperable services. There are many different interpretations of the 'best' approach to build a SOA system (Kontogiannis et al, 2007).

Typical SOA modelling lifecycles include the phases of service identification, service specification, and service realization (Arsanjani, 2004). Our research emphasizes the importance of the service identification phase for defining the right service because mistakes made here can lead the overall failure of the resulting SOA-based system. The success critically depends on the correct identification, presentation and definition of the key services, because the exposed functionalities in a service define the service granularity. It is important to appreciate that achieving an optimal level of service granularity requires a compromise between many elements, both technical and non-technical. In particular, the optimal granularity of key services can be expected to vary at various layers with different service types (Kulkarni et al, 2008) or service layers (Reldin et al, 2007; Ramollari et al, 2007).

Recent research conducted by (Kohlborn, 2008) on thirty modern service analysis approaches showed that 76% of those approaches introduced two types of services (e.g. business service, software service or generic service). The other approaches defined between one and eleven types of services. All of the approaches studied applied one of the development strategy techniques (e.g. top-down, bottom-up and meet-in-the-middle), although the services identified were significantly different from one approach to another. This means that defining service granularity, which is a very challenging task, requires considering not only service characteristics but also provisional service types. For example, an infrastructure service that is concerned with providing heterogeneous underlying capabilities to other services should be fine-grained for high

Alahmari S., Zaluska E. and De Roure D. (2010).
MIGRATING LEGACY SYSTEMS TO A SERVICE-ORIENTED ARCHITECTURE WITH OPTIMAL GRANULARITY.
In *Proceedings of the 12th International Conference on Enterprise Information Systems - Databases and Information Systems Integration*, pages 198-207
DOI: 10.5220/0002896401980207

reusability and encapsulation. On the other hand, a business service will typically be implemented as a coarse-grained service for maximizing business value and traceability of business processes.

Our approach analyses legacy systems using two formal models: firstly a Unified Modelling Language (UML) representation and secondly a Business Process Modelling Notation (BPMN) representation. Although these present two different views of complex software systems, we started initially with a transformation from a static model (e.g. UML class diagram) to a behaviour model (BPMN) for two reasons. First, a significant number of current legacy systems are Object-Oriented systems. Second, there is already a wide acceptance of business process modelling for service-oriented systems [Rolland et al, 2007; Erradi et al, 2006; Erradi et al 2007] . However, BPMN and UML can be expected to overlap when considering workflow patterns and also to share significant deficiencies such as informal semantics and imprecise resource definitions. This is an important motivation for the investigation of a combined approach in this research, because relying on a single approach cannot be expected to produce the required 'optimal' result. The key contributions of this paper are:

- To provide a framework and guidelines for identification of services from legacy code with proper level of granularity.
- To propose an automated and simple transformation from a static model (UML) to a behaviour model (BPMN).

The rest of this paper is organized as follows: in section 2 related work is discussed. In section 3 the service types and proposed approach are introduced. Section 4 presents an example of an application, followed by an example evaluation in section 5. Sections 6 presents the conclusions.

2 RELATED WORK

Research conducted in the arena of migrating and integrating legacy systems for support of SOA has proposed several different approaches from different perspectives: the technical domain, the business domain, and the conceptual (Ramollari et al, 2007). Reference (Zhang et al, 2004) proposes a hierarchical clustering algorithm to extract independent services from procedural software systems into an object-oriented (OO) model. This approach uses a grey-box strategy which is a combination of system wrapping with the key business logic. However, the drawback of this approach is that it identifies target services at two different architectural levels without a reconsolidation process. In addition, it requires human intervention to identify and assess the appropriate services.

Reference (Chen et al, 2005) discusses the transforming of legacy systems developed with Object-Oriented Design (OOD) or Component Based Design (CBD) into SOA applications using feature analysis. The feature analysis consist of three stages: identifying system features, constructing the feature models, and tracing the relationship between the defined services. The identified service operations are exposed by class delegations using a tool called a Web Service Wrapper. The authors claim that feature analysis bridges the gap between abstraction architecture and source code, whereas business requirements are excluded. In addition, this approach does not cover service classifications and granularity aspects. Reference (Zou et al, 2001) provides a framework to transform legacy systems into a web-enabled environment by means of a CORBA wrapper (CORBA IDL, SOAP, WSDL, and UDDI). This approach is accomplished in three stages. This approach does not provide enough detail on how to identify services along with new business requirements and the targeted service characteristics.

Reference [Jianzhi et al, 2005] develops a framework ICENI (Imperial College e-Science Network Infrastructure) to leverage components of legacy systems into a grid environment. It applies reverse engineering techniques to components using a Java Native Interface (JNI) wrapper to encapsulate code and Commerce eXtensible Markup Language (CXML) to describe specifications for communication with the ICENI workflow. The drawback of this approach is that they lack an understanding of the problem domain. It migrates independent blocks of code directly into services that cannot interoperate using a bottom up approach. Reference (Zhang et al, 2005) proposes an architecture-based service-oriented reengineering approach that uses a hierarchical clustering method to identify services from legacy systems based on mapped requirements in UML models. This approach requires human supervision to assist in determining the optimal service granularity along with the clustering technique.

Reference (Galster et al, 2008) proposes a graph-based framework that discovers service granularity according to business goals during the design phase. However, this approach does not define fine-grained services and quantifies coarse-grained services using

a non-technical description. Reference (Suntae et al, 2008) focuses mainly on how to define the right services in the analysis phase considering business change factors and goals. Reference (Rolland et al, 2007) introduces an approach that depends on exploring the purposes of a business process in order to identify a service. As a result, this approach defines a new type of service called an "Intentional Service" which considers business goals, pre- and post-conditions, and variant interpretations instead of the technical aspects of interfaces, behaviour, and composition of services respectively.

Reference (Sneed et al, 2006) refers to the importance of granularity in defining web services without suggesting any particular solution. Reference (Fareghzadeh, 2008) attempts to solve the gap between service provider and requester regarding to the service agreements. A Unified Service Model (USM) is proposed along with a service operational model to specify business services from a business people perspective. Although the authors of this approach claim that the USM defines business services at multiple levels of granularity, no metrics or guidelines are provided to identify the service granularity. Reference (Arsanjani et al, 2006) outlines set of activities in the analysis phase that lead to adequate broad foundation for service identification. The author classifies service types in three layers; orchestration layers (process service), business layer (task and entity service), and application service layer. Even though the author emphasizes the importance of the key principles of service-oriented, no details have been given as to how those principles can be applied to guaranteed optimized services

In a real-world project in an Energy Management System (ESM), reference (Xiaofeng et al, 2007) utilizes specific enterprise service hierarchy patterns for selected business processes to determine the service granularity. These patterns failed to provide effective guidelines to enhance the service granularity. References (Arsanjani, 2004; Lawson, 2009) propose a comprehensive approach with a high-level service architectural classification and iterative processes. It consists of three phases: identification, specification, and realization; the later reference added components and flow. Although these approaches were successful in highlighting the broadly architectural aspects, they failed to provide detailed guidance.

Reference (Erradi et al, 2006) introduces the Service Oriented Architecture framework (SOAF) with five conceptual levels: information elicitation, service identification, service definition, service realization, and roadmap with planning. Each level requires inputs to proceed with a set of activities that deliver outputs as inputs for the next layer. For service identification, it defines the design tasks to be performed (e.g. specifying service policy). It also scales service granularity levels by means of the grouping of the number of invoked components or services via an operation on a service interface and the number of updated sources. Finally, transformation strategies are defined along with the plan for service implementation. Reference (Erradi et al, 2007) extends the service design concepts of the SOAF framework (Erradi et al, 2006) with a business-driven approach based on a meta-model to define service granularity. It highlights broad guidelines for enhancing the service granularity such as reusability, business-alignment, designing for assembly, and reducing the ripple effects of application changes.

Reference (Dwivedi et al, 2008) introduces a semi-automated approach to identify services on process-oriented systems. It converts the UML based business process models into XMI. The XMI reader "NSUML" is used to produce the MOF (Meta Object Facility) for mapping XMI meta-model. The algorithm used runs over an XMI meta-model developed a statistic based approach which helped in creating APIs to query candidate services. Although this approach provides a good definition of the service identification issues and presents an interesting tool, it fails to demonstrate how the tool will integrate the service hierarchy layers and properties. Reference (Shirazi et al, 2009) attempts to categorize services based on operational state of services and logical presentations, i.e. differentiating between application and business services. However, this approach is incomplete because authors do not consider important elements that effect service identification phase such as granularity and complexity.

By analysing all of this related work, it is found that none of the previous approaches has enabled accurate service identification, in terms of when services should be coarse-grained and fine-grained. Although the approaches studied usually conclude with various service design principles, they do not provide well-defined and effective mechanisms to accomplish these principles. However, all of the references agree on the complexity of considering all applicable factors to fulfil both the business and the technical aspects. We believe that service granularity is a key aspect of service design that has a significant impact on other aspects such as reusability, maintainability, performance and

flexibility. As confirmed by this literature review, there are no close approaches to the work reported in this paper.

3 SERVICE IDENTIFICATION FRAMEWORK

3.1 Defining Service Types

The functional scope of different services is a key element required to construct service taxonomy. A service should accomplish certain goals that can be quantified correspond to a business or a technical requirement. Our classification is concerned primarily with defining right level of granularity for every service type. This classification will guide the service identification framework to define roles and properties for each service type. It will be also used along with the proposed metric in indentifying the right service with optimal granularity. The closest service type classification to that in our approach is the work by (Kulkarni et al, 2008a, 2008b), who identify eight generic service types from a hierarchical perspective of the enterprise layers. In contrast, our classification focus is on defining the purpose of the service, i.e. what the service is expected to expose to the service consumer. Table 1 shows service types with different levels of granularity from the functional perspective. We define seven service types; process service, business service, composite service, transactional-data service, master-data service, utility service, and infrastructure service. The services are defined from the most granular (e.g. process service) to the least granular (e.g. infrastructure service).

The purpose of the service can be defined as CRUD (create, retrieve, update, delete) functions or business logic or specific domain functions (e.g. a service for customer credit check) or infrastructure capabilities. The data nature is also considered as a function of the frequency of data modification. This consideration facilitates data exposure and process access functions through flexible and reusable services (Lawson, 2009). Based on the concept of the Business Intelligence (BI) meaning of data we define two new types of services: master-data service and transactional-data services. For example, master data (such as customer data name and address) is not likely to change on a frequent basis, hence few data parameters are manipulated, and there is no verification or compensation mechanism required (Haesen et al, 2008). In contrast, with a transactional data (such as “items ordered”) always changes (e.g. when the customer places a new order), hence such a service requires more communications to process.

Table 1: Service types.

Service type	Functional scope
Process Service	Processing sequenced tasks in a pre-defined flow of business process (coarse-grained).
Business Service	Presenting business logic of a transaction or a business entity (coarse-grained).
Composite Service	Aggregating several services in the enterprise (coarse-grained).
Transactional-data Service	Processing CRUD functions of transitional data (fine-grained).
Master-data Service	Processing CRUD functions of master-data (fine-grained).
Utility Service	Providing business domain functionalities to other services (fine-grained).
Infrastructure Service	Providing essential technical functionalities for other services and the architectural enterprise (fine-grained).

3.2 Proposed Approach

A service definition is inevitably derived from business processes or business functions (Steghuis, 2006). This paper aims to develop an SOA architectural framework to assist service identification for migrated legacy systems with optimal service granularity. The framework is based on process portfolios that are derived from the UML and BPMN standards in addition to an optional knowledge portfolio (as shown in figure 1). In the case of new system requirements, business processes in BPMN models are updated. Then, a cluster metrics is used to generate services based on the proposed previously service types classification. The SOA meta-model provides a comprehensive description of service types integrating the semantic of business processes. It provides effective standards for service granularity quantification after capturing potential services.

In many practical legacy system migrations, there is no available information about the legacy system apart from the code. Thus, assuming that only the code is available (as an extreme scenario); our approach consists of three main stages as follows:

First stage: the analysis and reengineering stage has the following activities:

1. If any documentation, expertise, or staff interviews are available, an analysis model can be produced to provide a knowledge portfolio (optional).
2. Transforming the legacy system codes into a formal representation using UML models by reverse-engineering techniques to obtain a static model automatically (using modelling tools such as IBM Rational Rose). The reverse-engineering process results in class diagrams which can then be produced to activity diagrams of the legacy system.
3. Transforming the UML models into BPMN models automatically using model driven development (MDD) techniques in order to define a business processes portfolio. The Relation Definition Language (RDL) describes the transformation rules as part of the Eclipse Modelling Framework (EMF) which utilizes the IBM Model Transformation Framework (MTF) running on the IBM WebSphere Business Modeler. Standards for mapping elements between UML activity diagram and BPMN diagrams are identified according to the pattern-based BPMN analysis (Russell et al, 2006; Griffen et al, 2007).

Second stage: the services elements identification stage has the following activities:

4. Defining atomic processes and business entities (Teale et al, 2004) from the processes portfolio; applying a clustering metrics on atomic processes to define fine-grained services based on CRUD functions (Griffen et al, 2007) and business logic metrics.

Third stage: the services evaluation stage has the following activities:

5 Evaluating coarse-grained and fine grained services against the SOA meta-model.

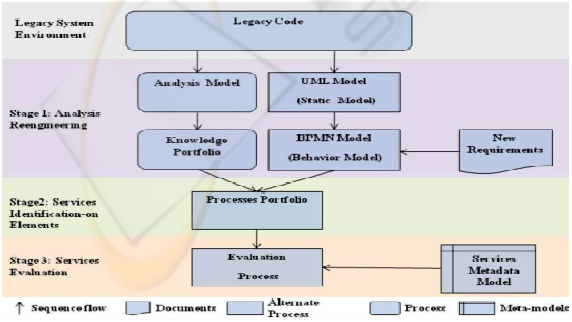

Figure 1: Service identification framework.

3.2.1 Analysis and Re-engineering Stage

To construct a sufficient underpinning for the service identification, the analysis and re-engineering stage consists of three models. First, the analysis model provides high-level system understanding and support for the business requirements. Secondly, the activity portfolio provides the structure of the legacy system. Thirdly, the process portfolio describes the behaviour of the legacy system.

The analysis model is based on understanding what the system does from the legacy code only is not sufficient. Legacy software systems were typically developed with embedded business rules and logic, scattered and duplicated code, unstructured modules, and tightly-coupled functions. The aim is to discover the system main functions and components via workshops, questionnaires, interviews and available documents. The outputs will complement the modelling of the atomic processes in BPMN and also capture additional information of non-functional requirements such as performance and reusability.

The UML model ensures decomposed units are relative and coarse-grained. Here our aim is to partition the class diagrams in terms of business functions in order to determine primitive functions and then atomic processes; the class diagrams are derived from the legacy code using IBM Rational Rose and reverse-engineering techniques. Business functions are defined from the knowledge portfolio process. We describe the structural and behaviour of the legacy system using UML class diagrams and activity diagrams respectively. Firstly, the reverse-engineering process results in a class diagram that shows the defined classes of the legacy code providing a static model. Secondly, after automatically generating the class diagram of the legacy code, use cases and activity diagrams are defined manually. It is important to guarantee consistency, completeness, and correctness when devising a behavioural diagram (e.g. activity diagram) from a structural diagram (e.g. class diagram), otherwise all subsequent diagrams would inherit the same issues. Several approaches attempt to check the rightness of UML diagrams using different techniques such as verification rules and meta-models (Kyu et al, 2003), profiling inconsistency and evaluation (Egyed, 2007). However, no tools in the industry yet are sufficient to solve these issues and in addition current approaches are time-consuming and affected always by model changes (Egyed, 2007). To resolve this

issue in our approach, the legacy code is reviewed manually line by line after identifying targeted classes.

The BPMN model is used to depict the business processes of the legacy system and also new requirements. The adoption of process modelling using BPMN as a modelling language in this research is motivated by several factors. First, relevant research agrees that process-oriented modelling provides a good basis for SOA (Rolland et al, 2007; Erradi et al, 2006; Erradi et al 2007). Secondly, UML 2.0 Activity Diagrams for business process modelling introduced by OMG suffers from limitations on modelling related resource and representing various types of control-flow constructs (Russell et al, 2006). It is important to note that BPMN has specific advantages over UML in describing complex scenarios such as richer constructs (Recker et al, 2009).

The use of MDD is a growing trend, as it provides capabilities to transform a model notation to other models (Griffen et al, 2007; Kalnins et al, 2006). Our transformation uses the concept of MDD in terms of converting a model to different level of abstraction using an alogrithm. Since most (if not all) legacy systems are based on functional decomposition, the UML standards are adequate for capturing the detailed design of legacy systems by the reverse-engineering technique. Reference (Griffen et al, 2007) introduces the capabilities of using WebSphere Business Modeler to automatically transform UML activity diagrams into business processes. This capability is used to achieve the transformation between UML activity diagrams and business processes in BPMN standards by modifying the proposed XMI content elements. Figure 2 shows the entire process of the transformation using MTF and EMF models.

The IBM WebSphere Business Modeler tool is used to perform the conversion (Griffen et al, 2007). It includes a general XML schema which supports the mapping of an XML file to business processes.

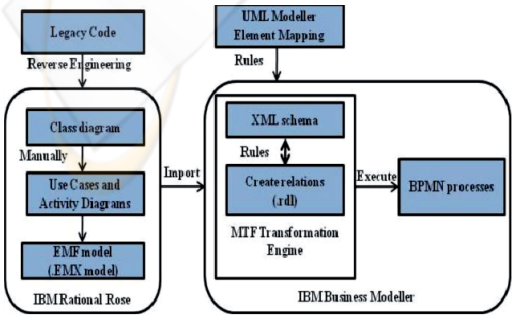

Figure 2: UML and BPMN transformation method.

This paper utilizes the structure of the XML schema provided with different interpretation of the business processes. After defining our EMF model using the Rational Rose tool, the transformation rules are developed in RDL. The MTF (a set of tools facilitating transformations between EMF models) parses the written relations rules. The transformation engine then executes the rules to match both models: the source (UML) model and target (BPMN) model.

3.2.2 Service Elements Identification Stage

The service identification stage is an essential step for the architecture of service-oriented systems. The term optimal service or 'right service' refers to a service that offers an acceptable size of functionalities without interfering with the pre-defined service properties. The service properties will be established according the service agreements between service provider and consumer. The properties can be expressed in terms of absolute number (e.g. execution time of service), or scale (e.g. the security of a service). For example, a service that requires a very strong security policy is likely to have lower performance. In this example, the service properties should include a high scale for the security and a rational execution time. We need to deduce the underpinning elements that assist in identifying 'optimal' services from the three models (e.g. UML, BPMN and the analysis model). Those elements are atomic processes and relative data entities (objects). A bottom-up analysis is conducted to reveal functions and data structure as part of the knowledge portfolio. The results of this stage will be a set of services implementing various service types derived from the activity and process portfolios respectively.

Transforming UML activity diagrams to business processes in BPMN provides the basis for capturing existing business activities. It is also a shift from an object-oriented architecture (in this case study) or functional decomposition to a process-oriented methodology for SOA. Using BPMN fills the gap between business process design and process realizations. As a basis for the BPMN diagrams, transformed activity diagrams have been refined with the detailed flow of processes using information from the knowledge portfolio. BPMN models help to describe granular business activities in details or sub-processes implying cohesively the optimal granularity. For example, the "pool" notation in BPMN represents a business entity or business role in a process which is linked to another entity with "message flow" arrows. In SOA, this can

help to identify initially the size of exchange messages among business entities in a service or different services.

In order to define optimal services, a set of distinct metrics is designed. Metrics include number of atomic processes and relative data entities (objects). Reference (Jamshidi et al, 2008) uses CRUD functions to define the affinity between assumed elementary business processes and business entity. Our metrics are derived from the metric proposed by reference (Jamshidi et al, 2008) with several main changes. Firstly, our atomic processes are automatically driven from a legacy system using revere re-engineering technique. Secondly, in the cluster technique distinct CRUD functions and business logic are used. Thirdly, we establish rules for the identification of optimal services. Finally, our metric is service type classification dependent. Eventually, this classification will provide guidance to an intensive description of each service (e.g. messages data types). We focus on what the atomic processes are, not how they are performed. Gathering similar behaviour atomic processes leads to high cohesion and loose coupling among services. Encapsulating atomic processes and relevant data entities increases reusability and eliminates redundancy. The aim is to disunify groups of atomic processes and data entities into services according to specific rules. These rules will be applied as follow:

- A service can have CRUD and executed business logic for either one entity and any number of processes or any number of entities and one process.
- A process that has CRUD functions for many entities can be encapsulated in a service with other similar processes and entities.
- A service can not cover CRUD functions and executed business logic for one entity if they are provided by different processes. Those processes which provide CRUD functions will be defined by a service.
- Every data entity must have at least one creation operation and each atomic process presents a coherent functionality. Every atomic process invokes at least one entity.

The CRUD functions and business logic identify the relationship between an atomic process and an entity as following:

- "C" means this atomic process CREATES an instance of this data entity.
- "R" means this atomic process READS an instance of this data entity.
- "U" means this atomic process UPDATES an instance of this data entity.
- "D" means this atomic process DELETES an instance of this data entity
- “BL” means this atomic process has business logic belongs to an instance of this data entity.

3.2.3 Service Evaluation Stage

Classification and categorizing service types establish a basis to define the service granularity. Elements such as a service contract, an implementation, and an interface (Steghuis, 2006). They can all contribute to this classification and definition. Every service type is inherently associated with a specific level of abstract and implementation descriptions. Defining service types is always an independent task from the service identification process, if it is considered. Although the SOA architecture view of the enterprise logically presents potential service types, our proposed methodology need to comprise all service scenarios to be complete. We develop a SOA meta-model that describes service types along with the semantics of our approach. The SOA meta-model shown in figure 3 presents the relationship between business process characteristics and different service types. We have assumed that the optimal granularity level of the service can be identified based on our classification and method together.

The SOA meta-model provides a comprehensive understanding of two major characteristics: a business process and a service. When we apply our SOA meta-model (see case study), we will have a new set of services with optimal level of granularity. To classify the identified services in a process-oriented system, we define the following rules:

- Breaking down activities of every process to atomic activities (only systematic activities are considered).
- Specifying the service type depends on the process functionality (purpose) and interoperability with other processes.
- Every business process can be modelled as one operation or more of a service or as a service itself.
- A high-level business process can be modelled as a service process type.
- A service can be equal to one or more operations.
- An operation encapsulates either business logic or CRUD functions.
- Business logic can be implemented by a business service or by many business services as part of service composition.

- One or many business service is orchestrated by a process service and supported by one or many utility services.
- CRUD functions can be implemented by either a transactional service or a master-data service depend on the type of the data entity or as part of collaborative similar services in service composition.
- One or more infrastructure service can be invoked directly or as part of a service composite. Any type of service can be part of one or many composite service.

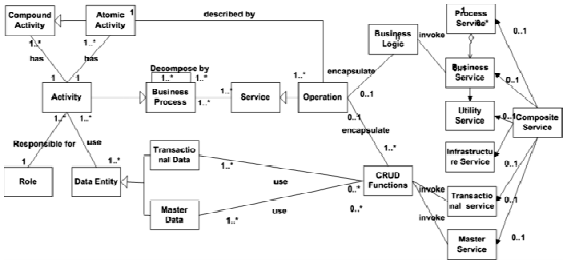

Figure 3: SOA meta-model.

4 EXAMPLE CASE STUDY

This section describes and discusses a pilot application of our framework to re-engineer a legacy code implementing an e-commerce project (Alahmari, 2002). The aims of the project were to build a servlet-based, object-oriented, three-tier architecture, dynamic e-commerce, customizable website that offers online customers and an internal administrator a wide range of functionalities. The webstore consists of six java packages which have a total of twenty-eight classes. For brevity, we selected the main four java classes (as a sample) from the legacy code to apply our approach.

In the analysis and reengineering stage: after the legacy code is reversed to java class diagrams, the activity diagrams are created manually as part of the UML model. The activity diagrams are transformed automatically to business processes in BPMN (as described previously). The transformation results in two main business processes with twenty one sub-processes and tasks.

In the service elements identification stage: the proposed method (section 3.2.2) is applied to the atomic processes and data entities. A set of a coarse-grained services (business and process services) are identified implementing cohesive functions with wide range of granularity. Table 2 shows the aggregation of relative atomic processes and data entities against candidate identified (coarse-grained) services.

In the service evaluation stage, the identified services from the second stage are evaluated using the SOA meta-model definitions and rules. This stage produces a new set of various types of candidate services with optimal range of coarse-grained and fine-grained services. Table 3 includes two columns: The first column shows processes identified from the processes portfolio after adopting the BPMN standards. The second column lists candidate services with the optimal level of granularity. For example, Pro9 has previously only one service (service_3), after evaluation it has a composite service which includes a business service (service_3_A) and a transactional service. Both defined services present optimal level of granularity.

Table 2: The cluster matrix.

	Process/Entities	Customers	Orders	Product	Items Ordered
Pro9	select products from itemlist	--	--	R	Service_3
Pro10	submit selected products	--	--	R	--
Pro11	create a shopping cart	--	BL	R	Service_4
Pro12	view an order	Service_5	R	--	R
Pro13	submit an order	U	C	--	U
Pro14	delete shopping cart	--	--	U	U
Pro15	cancel an Order	--	D	U	U
Pro16:	modify an order	Service_6	U	U	U/BL
Pro17:	generate a bill		U/BL	Service_7	
Pro18:	provide shipping information	--	U/BL	--	--

Table 3: Optimal candidate services.

Service type	Functional scope
Pro9: select products from a list	business service (service_3_A) transactional-data service (service_3_B)
Pro10:submit selected products	business service (service_3_C) transactional-data service (service_3_D)
Pro11:create a shopping cart	business service (service_4_A) transactional-data service (service_4_B)
Pro12: view an order	Transactional-data service (service_5_A)
Pro13: submit an order	business service (service_5_B)
Pro14: delete shopping cart	Transactional-data service (service_5_C)
Pro15: cancel an Order	business service (service_5_D) transactional-data service (service_5_E)

5 CASE STUDY EVALUATION

As the case study illustrated, to define the right services with optimal granularity from legacy code, knowledge portfolio and distinct functionalities need to be captured in underlying formal representations

(e.g. UML and BPMN). Having defined business processes and data entities for such system, a metric method can now assist the process of service identification. Because there are no agreed definitions and standards for a service, it is appropriate to use metric measurements. The metric assists in deriving optimal services considering the service purpose (e.g. CRUD functions, and business logic) and granularity rules. Because of the different interpretation of a business process and level of abstraction, we define an additional different set of rules for the process portfolio.

The SOA meta-model provides a foundation for defining the optimal granularity by encapsulating a service type classification and also the method metric. Through our evaluation process indicates that the purpose of the service can assist in defining the right service for a process or activity, we must also note that there are other elements can be considered along with the purpose of the service. Adding those elements (e.g. service contract, and service interface) will provide a better foundation for defining the service granularity. The technique used shows that having a well-defined meta-model considering all service definition aspects (e.g. service types) and process definitions can enhance the service identification stage. An outcome and a possible limitation of this approach is that it was not possible to introduce automatic mapping from UML activity diagram to BPMN models. It is a subject of further research aimed at creating a bidirectional MDD transformation between UML class diagrams and activity diagrams.

6 CONCLUSIONS

With the increased need to expose disturbed system functions as interoperable services across enterprises, research has been focusing on providing a general approach for different service-oriented modelling lifecycle phases. Our research emphasizes the importance of the service identification phase for defining the right service, because any faults at the service definition stage can lead to failure of the entire SOA project. The main contribution of this paper is a framework and guidelines for identification of specific services from legacy code with the appropriate levels of granularity. It also introduces an intensive meta-model that defines uniquely the characteristics of business processes and service types in the way that definitions are mapped. The paper also emphasizes the importance of the classification of service types to define service properties. Our future work will be to expand the service to automate the service identification process and hence generate the optimal level of service granularity automatically.

ACKNOWLEDGEMENTS

We acknowledge helpful discussions with Rob Phippen and Kim Clark (IBM Hursley Park, UK).

REFERENCES

Arsanjani, A., 2004. *Service-oriented modeling and architecture: How to identify, specify, and realize services for your SOA*, [Online], Available at: http://www.128.ibm.com/developerworks/webservices /library/ws-soa-design1/ [Accessed 30/12/2009]

Arsanjani, A., Allam, A., 2006. Service-oriented modeling and architecture for realization of an SOA. *Proc. 2006 IEEE International Conference on Services Computing*, pp.1, IEEE Computer Soc.

Chen, F., Li, S., Yang, H., Wang, C., Chu, W., 2005. Feature analysis for service-oriented reengineering. *Proc. 12th Asia-Pacific Software Engineering Conference*, IEEE Computer Soc.

Dwivedi, V., Kulkarni, N., 2008. A model driven service identification approach for process centric systems. *Proc. 2008 IEEE Congress on Services Part II (SERVICES-2)*, pp. 65-72. IEEE Computer Soc.

Egyed, A., 2007. UML/analyzer: a tool for the instant consistency checking of UML models. *Proc. 29th International Conference on Software Engineering* (ICSE'07), pp.787-790. IEEE Computer Soc.

Erradi, A., Anand, S., Kulkarni, N., 2006. SOAF: An architectural framework for service definition and realization. *Proc. IEEE Services Computing Workshops*, pp. 151-158. IEEE Computer Soc.

Erradi A., Kulkarni N, Naveen., Maheshwari, P., 2007. Service design process for reusable services: financial services case study, *ICSOC*, pp.606-617.

Fareghzadeh,N. 2008. Service identification approach to SOA development. *Proceedings of World Academy of Science, Engineering and Technology*, 35, ISSN 2070-3740.

Galster, M., Bucherer, E., 2008. A Business-Goal-Service-Capability Graph for the Alignment of Requirements and Services. *Proc. IEEE Congress on Services*, pp.399-406. IEEE Computer Soc.

Griffen, C., Huang, R., Sen, Z., Fiammante, M., 2007. Transforming UML «Activity» diagrams to WebSphere Business Modeler processes. *IBM WebSphere Developer Technical Issue 10.6,* Journal July 18.2007.

Haesen,R., Snoeck, M., Lemahieu, W., Poelmanset, S., 2008. On the definition of service granularity and its architectural impact. Proc. Advanced Information

Systems Engineering. *20th International Conference*, *CAiSE 2008*, pp. 375-389. Springer-Verrlag.

Jamshidi, P., Sharifi, M., Mansour, S., 2008. To establish enterprise service model from enterprise business model. *Proc. IEEE International Conference on Services Computing,* pp. 93-100. IEEE Computer Soc.

Jianzhi, L., Zhang, Z., Yang, H., 2005. A grid oriented approach to reusing legacy code in ICENI framework. *Proc. Proceedings of the 2005 IEEE International Conference on Information Reuse and Integration*, IEEE Computer Soc.

Kalnins, A., Vitolins, V., 2006. Use of UML and model transformations for workflow process definitions. Vilnius Gediminas Technical Univ Press. *Proc. 7th International Baltic Conference on Databases and Information Systems,* pp. 3-14. Technika,

Kohlborn, T., 2008. *A consolidated approach for service analysis*. Master's thesis. Westfalische-Wilhelms University Munster.

Kontogiannis, K., Lewis, G.A., Smith, D.B., Litoiu, M., Muller, H., Schuster, S., Stroulia, E., 2007. 'The landscape of service-oriented systems: a research perspective'. *2007 International Workshop on Systems Development in SOA Environments.* Minneapolis, MN, IEEE Computer Soc.

Kulkarni, N., Dwivedi, V., 2008. The role of service granularity in a successful SOA realization - A Case Study. *IEEE Congress on Services.* Honolulu, HI, IEEE Computer Soc.

Kyu, H., L-K., Kang, B-W., 2003. Meta-Validation of UML structural diagrams and behavioral diagrams with consistency rules. *Proc. of IEEE Pacific Rim Conf on Communications, Computers and Signal Processing,* PACRIM, 2, pp. 28-30.

Lawson, J., 2009. Data services in SOA: maximizing the Benefits in enterprise architecture. *Oracle published*, [Online] April, 2009. Available at: http://www.oracle.com/technology/pub/articles/j_lawson_soa_data.html [Accessed 30/12/2009].

Ramollari, E., Dranidis, D. ,Simons, A.J.H., 2007. A Survey of Service Oriented Development Methodologies. *2nd European Young Researchers Workshop on Service Oriented Computing*.

Recker, J., Muehlen, M., Siau, K., John, J., Indulska ,M., 2009. Measuring method complexity : UML versus BPMN. *15th Americas Conference on Information Systems, San Francisco*, California.

Reldin, P., Sundling, P., 2007. *Explaining SOA Service Granularity: How IT-strategy shapes services*. Master's thesis.

Rolland, C., Kaabi, R.S., 2007. An intentional perspective to service modeling and discovery. *Proc. 31st Annual International Computer Software and Applications Conference*, IEEE Computer Soc. Institute of Technology Linkoping University.

Russell, N., Van der Aalst, W.M.P., Ter Hofstede, A.H.M., Wohed, P., 2006. On the suitability of UML 2.0 activity diagrams for business process modelling. *BPM Centre Report BPM-06-03*, BPMcenter.org.

Shirazi, H.M., Fareghzadeh, N., Seyyedi, A., 2009. A combinational approach to service identification in SOA. Journal of Applied Sciences Research, INSInet Publication, 5(10): pp.1390- 1397.

Sneed, H.M., 2006. Integrating legacy software into a service oriented architecture. Proc. *10th European Conference on Software Maintenance and Reengineering*, IEEE Computer Society.

Steghuis, C., 2006. *Service granularity in SOA projects: a trade-off Analysis*. Master's thesis. University of Twente.

Suntae,K. , Kim, M., Park, S., 2008, Service identification using goal and scenario in service oriented architecture. *Proc. 2008 15th Asia-Pacific Software Engineering Conference*, IEEE Computer Soc.

Teale, Ph., Jarvis, R., 2004. Business patterns for software engineering use, Part 2. *The Architecture Journal*, [Online], Available at: http://msdn.microsoft.com/enus/arcjournal/aa480036.aspx, [Accessed 30/12/2009].

Xiaofeng, W., Hu, S., Haq, E., Garton, H., 2007. Integrating legacy systems within the service-oriented architecture. *Proc. 2007 IEEE Power Engineering Society General Meeting*, pp.7. IEEE Computer Soc.

Zhang, Z., Ruimin, L., Yang, H., 2005. Service identification and packaging in service-oriented reengineering. *Proceedings of the 17th International Conference on Software Engineering and Knowledge Engineering,* IEEE Computer Soc.

Zhang, Z., Yang, H., 2004. Incubating services in legacy systems for architectural migration. Proc. Proceedings. *11th Asia-Pacific Software Engineering Conference*, IEEE Computer Society.

Zou, Y., Kontogiannis, K., 2001. Towards a Web-centric legacy system migration framework. *The 3rd International Workshop on Net-Centric Computing (NCC): Migrating to the Web, International Conference on Software Engineering (ICSE'01),* Toronto, Canada.