2016

Cloud Computing Adoption: An Effective Tailoring Approach

Mahdi Fahmideh Gholami
School of Information Systems, Technology, and Management, UNSW Business School, UNSW Australia, m.fahmidehgholami@student.unsw.edu.au

Farhad Daneshgar
School of Information Systems, Technology, and Management, UNSW Business School, UNSW Australia, f.daneshgar@unsw.edu.au

Fethi Rabhi
School of Computer Science and Engineering, The University of New South Wales,, f.rabhi@unsw.edu.au

Cloud Computing Adoption: An Effective Tailoring Approach

Mahdi Fahmideh
School of Information Systems and Technology Management, University of New South Wales (UNSW), Sydney, Australia
Email: m.fahmideh@unsw.edu.au

Farhad Daneshgar
School of Information Systems and Technology Management, University of New South Wales (UNSW), Sydney, Australia
Email: f.daneshgar@unsw.edu.au

Fethi Rabhi
School of Computer Science and Engineering, University of New South Wales (UNSW), Sydney, Australia
Email: f.rabhi@unsw.edu.au

Abstract

Many organisations are currently moving their legacy systems to the cloud as it offers reduced cost, improved operational efficiency, on-demand, and pay-as-you-go service models. While any cloud migration scenario to make legacies cloud-enabled may have their own characteristics, there is no universally superior or applicable method for all scenarios. In situations like this, designing customisable methods that fit characteristics of migration scenarios would be pivotal for successful adoption of cloud computing. The literature review reveals that issues surrounding method tailoring for the cloud migration have so far been missing and this is despite the call made by previous researches to adopt a tailoring perspective to the cloud enablement. To effectively address this shortcoming, this study applied the idea of situational method engineering to develop a framework which facilitates the design and maintenance of bespoke cloud migration methods. The paper demonstrates the applicability of the proposed framework via presenting two scenarios.

Keywords: Cloud Computing, Cloud Migration, Legacy Systems, Situational Method Engineering, Method Tailoring, Information System Development Methods.

1 Introduction

Cloud computing has received a great deal of attention amongst IT industry and academia as it empowers enterprise systems to perform highly computational tasks, provides efficient resource utilisation, and reduces infrastructure cost and maintenance effort. Cloud computing offers clients a wide range of services which are ubiquitously accessible, acquirable, releasable, and payable in a dynamic manner based on the amount of usage (Mell and Grance 2009). These advantages have motivated IT-based organisations to either move their existing legacy systems to the cloud, or build new systems using cloud services. According to Gartner's account, *by 2020, a corporate 'No-Cloud' policy will be as rare as a 'No-Internet' policy is today* (Gartner 2016). Legacies are key assets that deliver everyday services of organisations and contain massive data over a long period of time. A poor cloud migration may cause failing business processes, raise security issues, incur excessive costs, and resulted in unwanted maintenance overhead (Babar and Chauhan 2011). Therefore, an effective method is crucial for successful cloud migration. Researches pinpoint that such a transition needs to be organised, anticipated, and viewed from the methodological perspective, which specifies steps involved, techniques to support these steps, and definition of any required models and roles involved during the migration process (Andrikopoulos et al. 2013; Chauhan and Babar 2012; Jamshidi et al. 2013; Mohagheghi et al. 2010).

The multi-faceted cloud computing technology introduces many potential challenges such as security, the location of data, changes in legacy systems, interoperability, legislation, and vendor lock-in (Armbrust et al. 2010) as well as many situational factors that should be properly addressed when this technology is used in an organisation. According to (Louridas 2010) there is a difference between the US and the EU in addressing the ultimate data protection in the cloud. In the US, a cloud provider is responsible for complete data protection whilst in the EU cloud consumer is responsible to ensure cloud provider satisfies data protection requirements. Any misunderstanding of this fact may affect the systems security, and will raise exposure to vulnerabilities. Therefore, activities attuned to securing legacy data should to be incorporated into the migration process. To this end, researchers are have starting to appreciate the potential value that paid a great attention to engage method tailoring can offer to the for the cloud adoption process endeavours. Wang states that *cloud computing definitely has many attractive benefits to offer but there isn't a one size fits all formula for all customers' needs* (Wang 2015). Additionally, Banerjee (Banerjee 2012) explored existing migration methods and identified challenges that impede effective utilisation of cloud services. He concludes that *there is no one-size-fits-all cloud, and it is up to each business to decide how much change is tolerable and to decide how far into the cloud to step*. Furthermore, a call for designing configurable cloud migration methods is reported by Mahmood (Z.Mahmood 2013) as a response to *an immense need to identify correct process model for the deployment of cloud-centric environments in order to meet changed business requirement of clients* (p.64).

Despite these contentions, researches relevant to method tailoring for the cloud adoption are intangible and difficult to find. Although various cloud migration methods have been proposed for cloud enablement, the majority of them have been designed on the basis of a *one-size-fits-all* assumption. Consequently, the need still remains for a foundation which enables creating customisable methods based on characteristics of cloud migration scenarios/projects. In addressing abovementioned shortcoming, this paper contributes to the cloud computing literature by underpinning a framework for designing, maintaining, and sharing situational method construction for moving legacies to the cloud in a systematic manner. The framework borrows the idea of *method engineering approach* (Brinkkemper 1996; Harmsen et al. 1994) suggested in the Information System Development (ISD) literature. Thus, the principle objective of this study is *to develop a framework which supports the creating, maintaining, and sharing of customised methods for conducting cloud migration scenarios*.

The paper is structured as follows: Section 2 provides a background on the related work on cloud migration and method engineering. Section 3 describes the research methodology adopted for the study. Section 4 illustrates the applicability of the framework via presenting two migration scenarios. Finally, Section 5 provides a summary of the current study, its research implications, and future research on the topic.

2 Research Background

Organisations may set an IT plan to move their legacies to the cloud. However, its actualisation is left to choose and perform a step-by-step method which guides developers or organisations to carry out

such a transition to meet migration goals. A dozen cloud migration methods have been already proposed both by academia such as Cloud-RMM (Jamshidi et al. 2013), ARTIST (Menychtas et al. 2013), and REMICS (Mohagheghi et al. 2010) as well as industry such as Oracle (Laszewski and Nauduri 2011) and Amazon (Varia 2010). When viewed collectively, these methods define a collection of fixed activities for planning, feasibility analysis, identifying candidate cloud services, re-engineering, testing, and monitoring legacies which are to utilise cloud services. All these methods agree on aforementioned activities however, they differ in their focus, scope, and the application domain. Each method may have weaknesses in certain situations and strengths in others. A method might be suitable for moving large and distributed workloads from legacy data centres to public IaaS whilst another method might be best suited for enabling legacies to work as a SaaS. Accordingly, methods should be configurable, consolidate existing methods, and responsive towards the specific characteristics of given cloud migration scenarios.

Recent research has started to argue that cloud migration methods need to be tailored prior to their enactment, although a scholarly effort is scant. Among the numerated methods in the above, only methods REMICS (Mohagheghi et al. 2010) and ARTIST (Menychtas et al. 2013) consider the notion of method tailoring. REMICS method is stored as a set of building blocks that can be selected and assembled to create a method that match characteristics of a project at hand. The constructs of ARTIST method are presented in a textual format without a direct support for method customisation. ARTIST method, on the other hand, provides a tool which allows the customisation and instantiation of the base method with respect to results of a cloud migration feasibility analysis. Nevertheless, this method is associated with common reengineering activities and there is no focus on defining activities relevant to different service delivery models.

The current research believes that combining existing methods into a comprehensive migration method cannot be a proper solution since a set of locally-based activities in each method cannot be linearly integrated or sometimes they become irreconcilable and inconsistent. The current study proposes a novel framework based on the idea of *(situational) method engineering* (Brinkkemper 1996; Harmsen et al. 1994) for the creation of configurable and maintainable migration methods by reusing and enhancing existing migration methods. Such a perspective to cloud computing adoption has not been a feature of previous researches in the field.

Method engineering has been introduced as a way to design flexible and situated ISD methods. Under method engineering, the raw materials that a method engineer uses for method construction and configuration are called *method fragments*. They are atomic and reusable pieces of methods which can be either obtained from existing methods, or can be created from scratch as a result of experience gained from past system development effort (Ralyté 2004). Method fragments are stored in a specialised database called *Repository* (or *Method base)*. A method engineer can then identify and analyse the above repository to seek optimal method fragments so that the best possible method can be constructed for particular needs and further enactment by software developers. This paper undertakes the development of an innovative framework for designing, maintaining, and sharing situational cloud migration methods. The main advantage of the proposed framework is that it to provides a repository of reusable method fragments of all stages of the cloud migration process obtained from a wide-range of cloud migration research along with a general and semi-automated procedure with a tool support for utilisation of the repository to construct bespoke methods, which fit a given cloud migration scenario, and to publish them in a standardised and interoperable manner.

3 Research Methodology

We viewed the proposed framework as a specific *artefact* and thus followed the design science research guidelines (Hevner et al. 2004; Peffers et al. 2006; von Alan et al. 2004). The research was comprised of two phases including *Design* and *Validate*. The design phase had two steps. In the first step, a consolidated repository including common method fragments for incorporation into the cloud migration process was developed. This was achieved by conducting a systematic literature review based on guidelines in (Kitchenham et al. 2009). The literature review included identifying all studies related to the cloud migration and then extracting method fragments, harmonising their definitions and reconciliation of differences, and organising them into generic migration steps. The basic idea behind the development of method fragments has been to be sufficiently generic to incorporate to a variety of cloud migration scenarios and be platform and application independent. In doing so, eight high-level concerns related to the cloud migration were identified from the current literature concerning the cloud migration as a basis for identifying method fragments. These concerns were *Understanding Organisational Context*, *Understanding Cloud Migration Objectives and Requirements*, *Migration Planning*, *Identifying Legacy Systems*, *Target Cloud Platform/Service*

Selection, *Re-Architecting Legacies*, *Network Configuration*, and *Testing*. A detailed description of these concerns can be found at (Fahmideh et al. 2016). Originated from the above concerns, forty-six distinct method fragments were identified from the literature. The repository was enriched by a set of guideline specifying when a method fragment should be incorporated into the migration process regarding a selected migration type (e.g. IaaS, SaaS, and PaaS). There was an emphasis on iterative design, validation, and refinement of the method fragments. In this regard, the importance and validity of the method fragments were assessed through a Web-based survey of 106 practitioners and academia in the cloud computing community from 32 countries. All fragments were found sound, important, and relevant for the incorporation into a reference cloud migration method. In addition, the repository was extended by addition of the new fragments suggested by survey participants. The duration of above step was between February 2013 and September 2015. The resultant repository distils the knowledge of the cloud migration that is represented and maintained in the form of a generic process metamodel.

The second step of design phase was to develop a general procedure to use the repository of the method fragments for two objectives (i) *Allowing method designers to represent and maintain cloud migration methods/knowledge in a standardised, interoperable, and well-structured format* and *(ii) Creating and configuring customised cloud migration methods through reusing and enhancing method fragments stored in the repository*. The proposed tailoring procedure is viewed as a class of the *paradigm-based method tailoring* (Ralyté et al. 2003). In the Paradigm-based technique, a new method is configured by instantiation from an existing method or metamodel, herein the repository, related to a particular domain or abstraction of an existing method. The procedure starts with specifying the parameters of a cloud migration scenario and then identifying relevant method fragments, commencing with a high-level method and recursively configuring it to meet requirements expected by a migration scenario. The output of the design phase is a conceptual framework (artefact) comprised of the repository of atomic and reusable method fragments and the tailoring procedure for creating, maintaining, and sharing customised cloud migration methods.

The second phase was to validate the developed framework in previous phase. To this end, a prototype of the framework was implemented to show its efficacy in standardising of migration methods across the cloud community, designing, configuring, and sharing situation-specific migration methods.

4 Implementation and Demonstration of Proposed Framework

This section presents the implementation and demonstration of the proposed framework named MLSAC, which stands for *Migration Legacy Software Applications to the Cloud*. MLSAC semi-automates the tailoring procedure. This is realised through the provision of interactive forms and the method fragments stored in the database. Figure 1 shows the harness of method tailoring. The sourced input migration parameters such as migration phases and migration types are used to retrieve the classes of relevant method fragments from the repository. The method designer can configure the initially created method using functionalities offered by MLSAC such as method extension, method concretisation, method export, and sharing. Once the method configuration is finalised, the method designer can export it as an XML (Bray et al. 1998) document. Developers will concretise and enact this method to carry out a migration scenario. An important goal of MLSAC is to be compatible with as many data stores and modelling tools as possible. XML is considered as a common vendor-neutral format to represent models. It facilitates interoperability and machine-readability of models. Using XML format provides method interoperability across cloud computing communities and different modelling platforms.

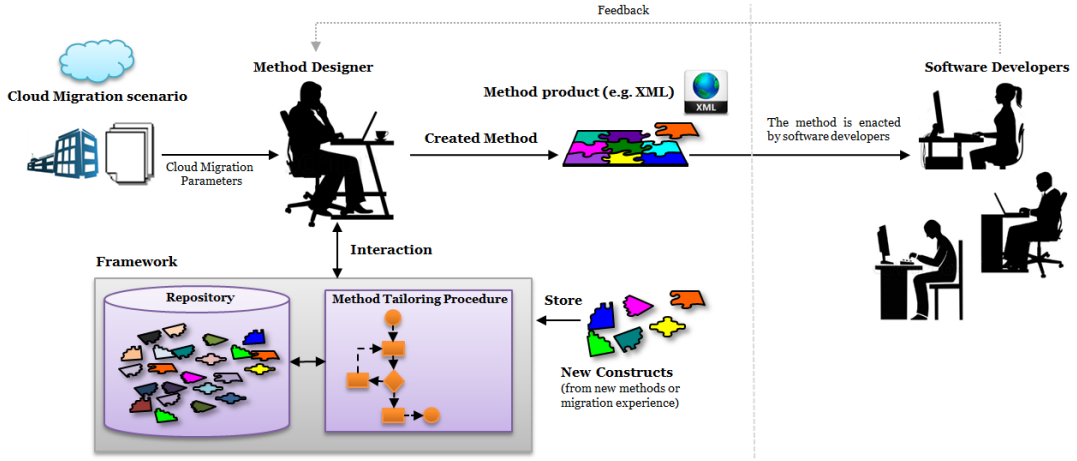

Figure 1. The method tailoring framework for cloud migration

Table 1 shows an excerpt of the method fragments stored in the repository. A full description of developed method fragments can be found in (Fahmideh 2015). MLSAC includes a total of nineteen forms to browse the repository for the creation of migration methods.

In the following subsections, the functionality of the framework for creating and maintaining cloud migration methods is demonstrated using two scenarios. The first one demonstrates how the method fragments can be reused or enhanced to represent an existing off-the-shelf migration method. From this angle, MLSAC facilitates the standardisation, knowledge transfer, and sharing experience about the cloud migration methods. In the second scenario, a new customised method is created for moving legacy systems to the EC2 Amazon cloud environment.

Method Fragment	Method Fragment	Method Fragment
Analyse Business Requirements	Enable Elasticity	Test Network Connectivity
Analyse Migration Cost	Encrypt/Decrypt Database	Test Performance
Analyse Migration Feasibility	Handle Transient Faults	Test Scalability
Analyse Network Change	Isolate Tenant Availability	Test Security
Analyse Organisational Changes	Isolate Tenant Customisability	Test Interoperability
Analyse Stakeholders Change	Isolate Tenant Data	Test Multi-tenancy
Analyse Technical Requirements	Isolate Tenant Performance	Make Application Stateless
Define Plan	Encrypt/Decrypt Messages	Decouple Application Components
Recover Legacy Application Knowledge	Obfuscate Codes	Adapt Data
Choose Cloud Platform/Provider	Rebalance Application	Develop Integrators
Design Cloud Solution	Refactor Codes	Deploy Application Component
Identify Incompatibilities	Re-configure Network	Replicate Application Components

Table 1. An excerpt of the method fragment repository

4.1 Maintaining and Sharing Existing Migration Methods

Like other ISD methods, cloud migration methods may be documented in textual format using natural language. Although this way of method maintenance might be suitable for some situations, it can also be problematic in others. If an element in this textual format is changed, then the entire document

should be manually updated and controlled for consistency and accuracy. As such, maintaining a good-sized textual method can be cumbersome. The proposed framework improves the maintainability and interoperability of methods across the community in a consistent manner. The following scenario is an example of representation and standardization of an off-the-shelf cloud migration method.

Australian Government Information Management Office (AGIMO) in consultation with Australian agencies has developed and published a generic method for crafting strategies to move systems to the cloud. The method defines activities that aid Australian agencies to implement cloud services throughout the migration lifecycle. The method has been documented in the form of thirty pages text and is publically available at (AGIMO 2012). Following a simple *tracing technique* (Sargent 2005), the method designer transfers and represents the paper-based AGIMO into a modular and structured document. The technique examines if there is a correspondence between AGIMO's activities and the existing method fragments in the repository. MLSAC allows the method designer to navigate through the repository to identify the appropriate mappings for the AGIMO's activities. Also, it provides flexibility to the designer for either reusing existing method fragments in MLSAC's repository or defining new method fragments to store AGIMO's activities. In the following scenario, it is assumed that the designer prefers to reuse the method fragments and, if required, specialise them in light of AGIMO's activities. If the method designer finds that the repository is deficient regarding some domain-specific AGIMO's activities, s/he can enhance the repository by defining new method fragments. Figure 2 shows the final representation of AGIMO document in MLSAC. MLSAC can process the method structure and automatically produce the corresponding XML document of the method. The XML document is then exported across regional agencies (Figure 3). Accordingly, developers in each agency can use the MLSAC and import this base method and then reuse and tailor it to target their agency's migration scenario. Developers in a local agency can specialise the method through new method fragments or technical implementation details using complementing methods or previous cloud migration experience.

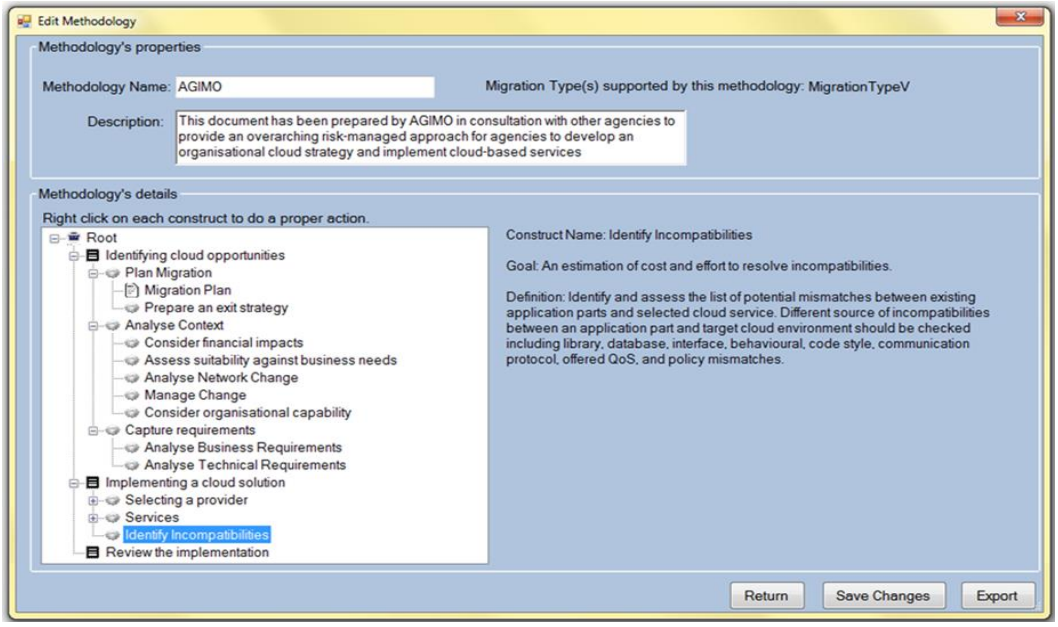

Figure 2. Representation and maintaining of AGIOM (an off-the-shelf cloud migration method) through reusing method fragments in MLSAC

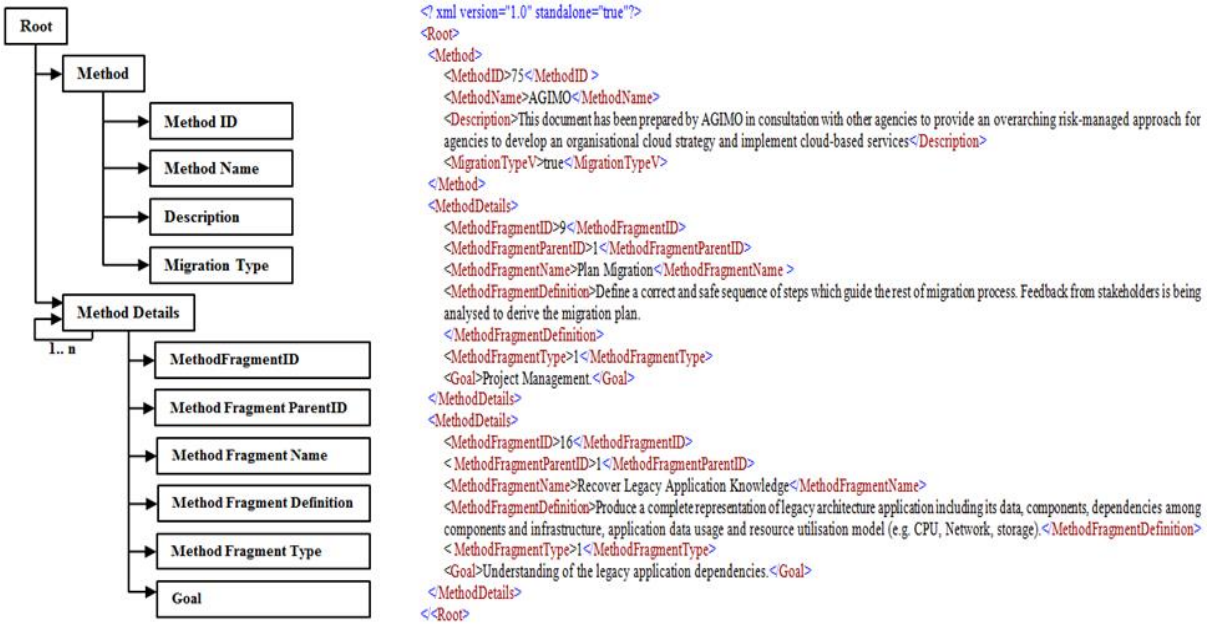

Figure 3. (Left hand) a general schema for method representation, (right hand) a snapshot of XML instance of AGIMO

The following example shows how AGIMO's activities are mapped onto the method fragments in the repository. In AGIMO, the activity called *Assess suitability against business needs* is to identify business services that are to gain the advantages of cloud services and assess the impact of migrating them to the cloud. In AGIMO this activity is defined as *"Agencies should identify the information types, services, and associated business processes which stand to gain the most from cloud-based services and assess the impact of moving them to the cloud. The agency's enterprise architecture will provide a useful place to start this analysis"* (AGIMO 2012). The most suitable method fragment in the repository to encapsulate this activity is the *Analyse Migration Feasibility*. The designer modifies this method fragment to store activity *Assess suitability against business needs* (Figure 4).

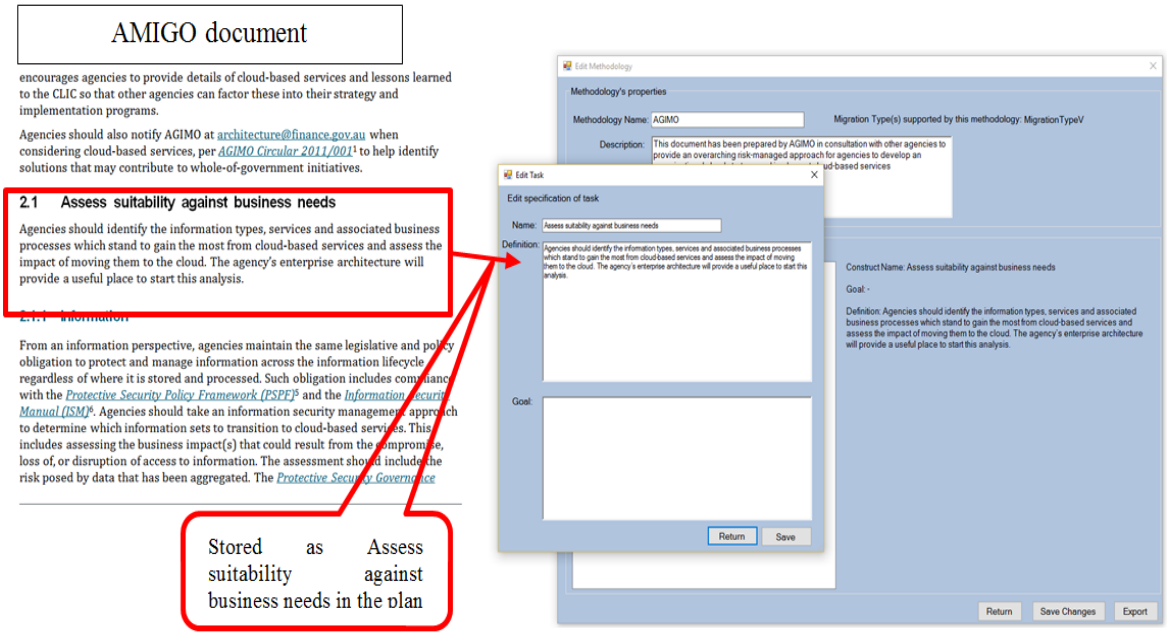

Figure 4. A snapshot for storing activity "Assess suitability against business needs"

4.2 Situational Method Tailoring

An organisation plans to deploy and run a legacy system in EC2 Amazon cloud severs. A method designer is responsible for defining a well-structured method that informs developers of relevant

activities that should be carried to move this legacy to EC2 Amazon. In this scenario, the cloud migration has already been found feasible in the organisational context and the method designer focuses on creating a customised method which only covers the implementation phase.

Setting input migration parameters. Inputs to the MLSAC are migration types and migration phases. This scenario is a virtualisation of a legacy system using IaaS model and the designer chooses the migration type V (Figure 5). Accordingly, MLSAC retrieves from the repository all method fragments that are relevant to the IaaS migration. The following method fragments are suggested by the MLSAC for the created method: *Resolve Incompatibilities* (including subclasses *Refactor Codes*, *Develop Integrator*, and *Adapt Data*), *Encrypt/Decrypt Entities* (including subclasses *Encrypt Database*, *Obfuscate Codes*, and *Encrypt/Decrypt Messages*), *Enable Elasticity*, *Test Application* (including all test subclasses), *Configure Network*, and *Deploy Application Components* (Figure 6). MLSAC specifies which method fragments are *Mandatory*, *Situational*, and *Unnecessary* to be carried out in the created method for a general IaaS migration. As shown in Figure 6, *Test Application* is suggested as mandatory and is denoted by √. On the other hand, the incorporation of method fragments related to incompatibility resolution is subjected to the choice of a cloud platform as denoted by (√). It means if legacy components are to be bound to specific cloud services, potential incompatibilities between the legacy and these services should be identified and resolved accordingly through adaptation mechanisms.

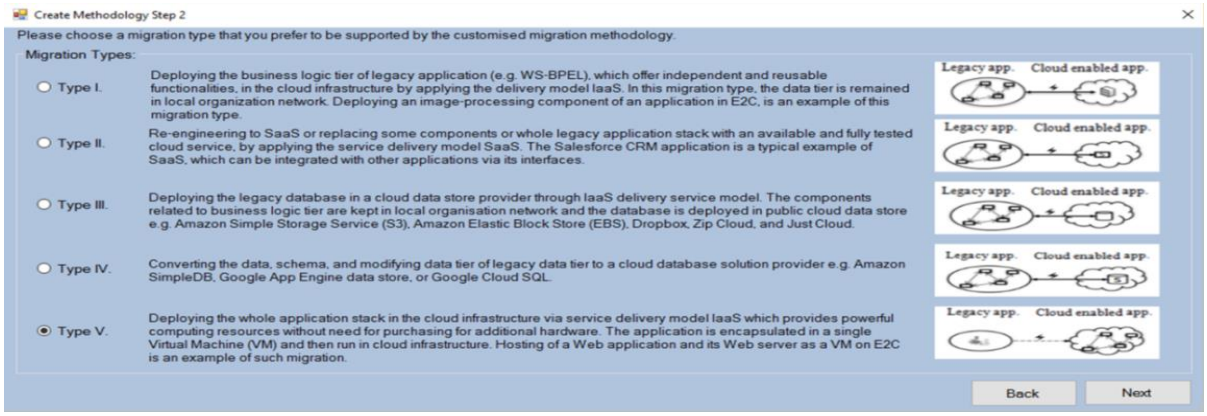

Figure 5. Selecting a cloud migration service delivery model

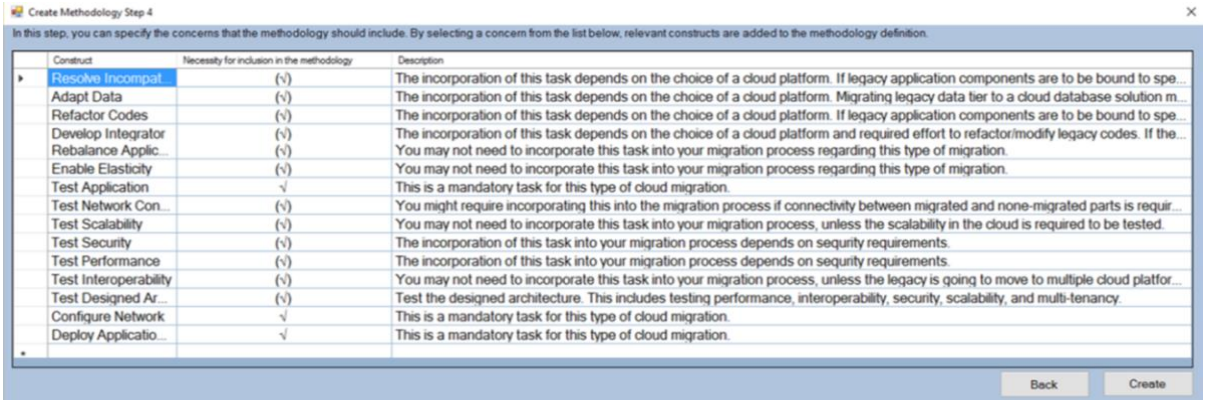

Figure 6. Suggested method fragments for incorporation into the migration method regarding the migration scenario parameters

Method Configuration. In some situations, there is no need to include all method fragments in the method as suggested by MLSAC. For example, based on the collected information about existing legacies, the method designer knows that the legacy data is compatible with Amazon Relational Database Service and its APIs. Hence, there is no need to incorporate specific fragments for the data adaptation and *Adapt Data* is removed from the method.

Furthermore, through the notion of inheritance originated from the object-oriented software development, the designer can extend existing method fragments. Let us assume that the designer wants to define a particular type of code refactoring where developers should be aware of updating the legacy APIs to make them compatible with EC2 APIs. Such a potential incompatibility might be

syntactical or semantical in nature. With respect to this, the method designer can add a new method fragment named *Update APIs and Framework* under the method fragment *Refactor Codes*.

Method Concretisation. An advantage of MLSAC is that it provides the ability of defining implementation techniques for the operationalisation of method fragments. Defined techniques are stored in the repository and can be further reused in methods. For example, the network adaptation is required to connect other local organisational systems to the system that was migrated to EC2 Amazon. Therefore, the method designer defines the following technique to carry out *Reconfigure Network*: "*The network should be divided into a total of 4 subnets across 2 Amazon Web Service availability zones in a single Amazon Web Service region. Each availability zoom should have a public subnet that users connected to, a private subnet for the application tier, a second private subnet for the database tier, and finally a separate subnet for the bastion hosts. Also, the local firewall should be set to open new ports. These ports allow developers and administrator to connect to Amazon Web Service. The connection string to the database should also be updated to connect Amazon server*".

Furthermore, the method designer defines alternative resource provision techniques related to the method fragment *Enable Elasticity*. This gives a freedom to developers who will enact the method to compare and choose the most appropriate implementation techniques that suit the scenario. The method designer can borrow many existing resource provision techniques available in the cloud computing literature. S/he defines three scaling techniques: (i) *Reactive scaling* where developers define a set of threshold-based scaling rules for the resource acquisition and release that requires a deep knowledge of the system resource utilisation patterns, (ii) *Proactive scaling* where developers use observation and prediction techniques to anticipate workload, and (iii) *Hybrid scaling* where a combination of reactive and proactive techniques are used to determine when to get resource during short and long period of system execution. MLSAC can support adding these techniques to the repository. Figure 7 shows defining these three scaling techniques which are to assign to *Enable Elasticity*.

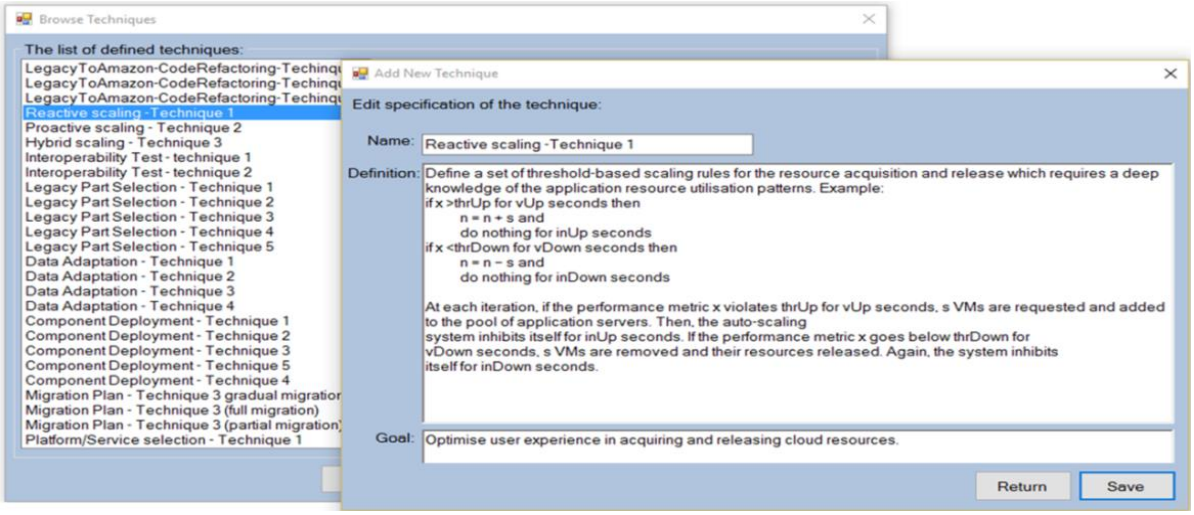

Figure 7. Adding new supportive techniques to method fragment Enable Elasticity

5 Conclusion and Future Research

Through a review and analysis of the current literature, this study identified the lack of an approach for tailoring methods to be fit for cloud computing adoption albeit few researches have already raised the issue in the literature. By applying method engineering approach, a framework was developed that offers a repository of reusable method fragments, obtained from the existing literature. The framework also defined a procedure for method creation and tailoring. MLSAC prototype demonstrated how the framework can be operationalised for method tailoring when entering input migration parameters and reusing method fragments from the repository.

The study contributes to the cloud computing literature by applying method engineering approach as a novel approach for effective cloud computing adoption. The suggested framework provides a substrate for situational-based adoption of cloud computing technology in IT-based organizations and is applicable in situations where a unique constructed method is required to address specific characteristics of a cloud migration scenario. As such, the proposed framework is a domain-specific

(i.e., cloud computing) application of method engineering. Such harness is also applicable to other sub-domains of cloud computing such as mobile cloud applications. The practical contribution of this research is to inform IT practitioners about core method fragments that should be incorporated into the migration process when moving legacies to the cloud. The framework also helps practitioners to create and share the knowledge of cloud computing migration in a consistent and standardised manner. Through example scenarios, we showed how the framework helps organizations to tune and update methods during transition to the cloud.

Currently, we are working on the qualitative evaluation of the framework in various method tailoring scenarios regarding other service delivery models. The framework is currently under evaluation by a panel of domain experts who have experience in the cloud migration for the extension with more method fragments and refinement of method customisation procedure. A further work of this research is to extend the framework to include inter-dependency among input migration parameters during method tailoring effort. This involves method tailoring with making trade-offs among different migration parameters and goals which may be in contradiction with one other. The situation may become complex when there are dependencies among different goals constrained with factors such as cost and effort. Finally, the focus of the framework is currently on method tailoring rather than method quality. Therefore, the evaluation of the rationale and suitability of cloud migration methods regarding migration goals is also another further direction of this research.

6 References

AGIMO. 2012. "A Guide to Implementing Cloud Services," *Australian Government Information Management Office*), p. 30.

Andrikopoulos, V., Binz, T., Leymann, F., and Strauch, S. 2013. "How to Adapt Applications for the Cloud Environment," *Computing* (95:6), pp. 493-535.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., and Stoica, I. 2010. "A View of Cloud Computing," *Communications of the ACM* (53:4), pp. 50-58.

Babar, M. A., and Chauhan, M. A. 2011. "A Tale of Migration to Cloud Computing for Sharing Experiences and Observations," *Proceedings of the 2nd International Workshop on Software Engineering for Cloud Computing*: ACM, pp. 50-56.

Banerjee, J. 2012. "Moving to the Cloud: Workload Migration Techniques and Approaches," *High Performance Computing (HiPC), 2012 19th International Conference on*: IEEE, pp. 1-6.

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., and Yergeau, F. 1998. "Extensible Markup Language (Xml)," *World Wide Web Consortium Recommendation REC-xml-19980210. http://www. w3. org/TR/1998/REC-xml-19980210* (16).

Brinkkemper, S. 1996. "Method Engineering: Engineering of Information Systems Development Methods and Tools," *Information and software technology* (38:4), pp. 275-280.

Chauhan, M. A., and Babar, M. A. 2012. "Towards Process Support for Migrating Applications to Cloud Computing," *Cloud and Service Computing (CSC), 2012 International Conference on*, pp. 80-87.

Fahmideh, M., Daneshgar, F., Low, G., and Beydoun, G. 2016. "Cloud Migration Process—a Survey, Evaluation Framework, and Open Challenges," *Journal of Systems and Software* (120), pp. 31-69.

Fahmideh, M., Low Graham, Ghassan Beydoun. 2015. "Conceptualising Cloud Migration Process," *Twenty-Fourth European Conference on Information Systems (ECIS), İstanbul,Turkey, 2016*:In Press).

Gartner. 2016. *[WWW document] http://www.gartner.com/newsroom/id/3354117 (accessed 22 September 2016).*).

Harmsen, A. F., Brinkkemper, J., and Oei, J. H. 1994. *Situational Method Engineering for Information System Project Approaches*. University of Twente, Department of Computer Science.

Hevner, A. R., March, S. T., Park, J., and Ram, S. 2004. "Design Science in Information Systems Research," *MIS quarterly* (28:1), pp. 75-105.

Jamshidi, P., Ahmad, A., and Pahl, C. 2013. "Cloud Migration Research: A Systematic Review," *Cloud Computing, IEEE Transactions on* (PP:99), pp. 1-1.

Kitchenham, B., Pearl Brereton, O., Budgen, D., Turner, M., Bailey, J., and Linkman, S. 2009. "Systematic Literature Reviews in Software Engineering – a Systematic Literature Review," *Information and software technology* (51:1), pp. 7-15.

Laszewski, T., and Nauduri, P. 2011. *Migrating to the Cloud: Oracle Client/Server Modernization*. Elsevier.

Louridas, P. 2010. "Up in the Air: Moving Your Applications to the Cloud," *IEEE software*:4), pp. 6-11.

Mell, P., and Grance, T. 2009. "The Nist Definition of Cloud Computing," *National Institute of Standards and Technology* (53:6), p. 50.

Menychtas, A., Santzaridou, C., Kousiouris, G., Varvarigou, T., Orue-Echevarria, L., Alonso, J., Gorronogoitia, J., Brunelière, H., Strauss, O., and Senkova, T. 2013. "Artist Methodology and Framework: A Novel Approach for the Migration of Legacy Software on the Cloud," *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2013 15th International Symposium on*: IEEE, pp. 424-431.

Mohagheghi, P., Berre, A., Henry, A., Barbier, F., and Sadovykh, A. 2010. "Remics- Reuse and Migration of Legacy Applications to Interoperable Cloud Services," in *Towards a Service-Based Internet,* E. Nitto and R. Yahyapour (eds.). Springer Berlin Heidelberg, pp. 195-196.

Peffers, K., Tuunanen, T., Gengler, C. E., Rossi, M., Hui, W., Virtanen, V., and Bragge, J. 2006. "The Design Science Research Process: A Model for Producing and Presenting Information Systems Research," *Proceedings of the first international conference on design science research in information systems and technology (DESRIST 2006)*, pp. 83-106.

Ralyté, J. 2004. "Towards Situational Methods for Information Systems Development: Engineering Reusable Method Chunks," *Procs. 13th Int. Conf. on Information Systems Development. Advances in Theory, Practice and Education*, pp. 271-282.

Ralyté, J., Deneckère, R., and Rolland, C. 2003. "Towards a Generic Model for Situational Method Engineering," in *Advanced Information Systems Engineering,* J. Eder and M. Missikoff (eds.). Springer Berlin Heidelberg, pp. 95-110.

Sargent, R. G. 2005. "Verification and Validation of Simulation Models," *Proceedings of the 37th conference on Winter simulation*: Winter Simulation Conference, pp. 130-143.

Varia, J. 2010. "Architecting for the Cloud: Best Practices," *Amazon Web Services*).

von Alan, R. H., March, S. T., Park, J., and Ram, S. 2004. "Design Science in Information Systems Research," *MIS quarterly* (28:1), pp. 75-105.

Wang, D. W. 2015. *Cash in on Cloud Computing*. Meghan-Kiffer Press.

Z.Mahmood. 2013. in *Cloud Computing Methods and Practical Approaches*. Springer-Verlag London p. 64.