FREMA method for describing Web Services in a Service Oriented Architecture.

David E. Millard, Yvonne Howard, Ehtesham-Rasheed Jam, Swapna Chennupati, Hugh C. Davis, Lester Gilbert, Gary B. Wills

Electronics and Computer Science,
University of Southampton, Southampton, UK
{dem,ymh, erj2, cs, hcd, lg3, gbw}@ecs.soton.ac.uk

ECSTR-IAM06-002, dated 01 March 2006
ISBN: 085432 851 3

Abstract. Service-Oriented Architectures (SOAs) are increasingly deployed to achieve distributed systems that are modular, flexible and extensible. Designing for a SOA can be difficult, however. There are issues involving the granularity of the cooperating services, and there are no currently accepted conventions for describing a service or its interactions at an abstract level. This paper presents the Service Responsibility and Interaction Design Method (SRI-DM), a light-weight agile method for engineering a Web service design, based on capturing a scenario as a use-case, factoring this into a set of Service Responsibility and Collaboration Cards (SRCs), and constructing a Sequence diagram illustrating their interactions in fulfilling the scenario. The paper presents the notation for each step and describes with the aid of an example how this process is being used to create a service design within the domain of e- assessment.

Keywords: Web Service Design, SRC, Sequence diagrams, Assessment

1 Introduction

Engineering widely distributed systems has long been a challenge for the software engineering community, in the last few years a trend has emerged towards Service-Oriented Architectures (SOA) that aims simplifying this problem. SOAs are an attempt to modularise large complex systems in such a way that they are composed of independent software components that offer services to one another through well-defined interfaces. The service approach is ideally suited to more loosely coupled systems, where individual parts may be developed by different people or organizations. Wilson *et al.* [15] describe the three main advantagaes of such a system as Modularity (dynamic coupling), Interoperability (standard interfaces), and Extensibility (encapsulation).

Service-orientation is a philosophical approach to creating distributed systems, but there are a number of standards and approaches to providing them at an implementation level (including Web Services based on SOAP, GRID Services based on OGSI, and REST services based on HTTP and XML).

Because of the difference in these approaches, and due to a lack of common notation and engineering experience, developing a service-oriented system can be difficult. Decisions must be made about how to divide a problem into logical services, how those logical services should be interfaced to maximize reuse, how they should be gathered together to create composite services, and what service-oriented implementation is best suited to each service, or the design as a whole.

Agile software development is a software engineering technique that attempts to build software in a short timespan by emphasizing face-to-face communication and software deliverables over documentation. It is a development technique used with small teams that requires light-weight engineering methods and tools and results in rapid and flexible software development.

In this paper we present the Service Responsibility and Interaction Design Method (SRI-DM), an agile development method for the modeling of services at an abstract level that is independent of implementation. SRI-DM is agile as it enables a team of developers to quickly define a scenario and generate a number of services that will fulfill it. It is lightweight in that the documentation is minimal, and serves to drive the development forward as well as record it for others. SRI-DM:

1. Defines a scenario with a use case diagram
2. Uses the individual use cases to factor a set of services
3. Represents these at a high level using Service Responsibility and Collaboration Cards (SRCs)
4. Defines how SRCs can interact to fulfill the scenario using a sequence diagram

The rest of this paper is structured as follows. Section 2 presents related work and describes alternative approaches to engineering services. Section 3 presents the SRI-DM and the notation that we have adopted and adapted to capture each stage. Section 4 presents an example of SRI-DM being used to create a set of services in the domain of e-assessment. Section 5 reflects on our experiences and reports on some informal evaluation activities that we have undertaken on our notation and approach. Section 6 concludes the paper and describes our future plans for taking our approach out into the web services community, particularly within the domain of e-assessment.

2 Background

Service-orientation is a philosophical approach to creating distributed systems, but there are a number of standards and approaches to providing them at an implementation level.

Web services have received a great of recent attention, they are defined around a set of standards (such as SOAP, WSDL, UDDI) developed by the W3C to make functionality available over the web as simply as data [4]. A weakness of the early web service approach was that there was no security system built into the service infrastructure; this means that there is no standard mechanism for session control, and that web services were therefore mostly insecure and stateless. This certainly mirrors the web approach, and is good for non-sensitive information and ad-hoc systems, but is not capable of supporting a virtual organisation without additional non-standard security layers .

GRID services on the other hand assume a highly secure environment, and rely on certificates and authentication bodies to operate [7]. This heavyweight approach to security makes it possible to build virtual organisations (that exchange and manipulate sensitive information) but can be prohibitively heavyweight for developers wishing to build simpler services and applications.

These two technologies are becoming more closely defined and a new generation of Web Service standards (such as WS_Security) is now being introduced to add a standard layer of authentication and security to Web Services, this will make Web Services an attractive proposition for systems builders as it possible to build virtual organisations using relatively lightweight middleware.

A third approach to service provision is represented by Representational State Transfer (REST) [6], this is the name for a methodology rather than a set of standards, where HTTP and XML are used to send and retrieve data to a remote script or application living on a web server. Web sites such as Google who offer both a REST and SOAP interface report that most activity is through the REST interface, which indicates that REST may be good enough for much of current service-oriented practice. However, REST services are not secure enough to build virtual organisations and therefore cannot support the growing number of sophisticated service-based systems.

We believe that each approach is applicable in different situations, and that an agile methodology for service design should be agnostic about the service technology itself.

2.2 Establishing SOA

There are also issues to do with the take-up of web services within enterprises. Weatherley [14] suggests that in the educational domain there are a number of barriers that prevent the widespread use of Web services for delivering web-based educational materials. These barriers relate to the need for understanding Web service protocols and the dynamic nature of the communication with web browsers. In addition, in many institutions developers are prevented from installing or running dynamic application software on their servers. Mukhi et al. [10] believe that to see an increase in the adoption of SOA, some of the non-functional features such as security transactionality and reliability need to be improved. They have developed a framework that supports and uses transactional and reliable services; this is achieved by using a policy model, based on WSPolicy.

SOA specifications are progressing toward standardization in a variety of ways, including small groups of vendors and formally chartered technical committees. For example, an SOA Reference Model Technical Committee has been formed by OASIS members to encourage the continued growth of different and specialized SOA implementations whilst preserving a common layer of understanding about SOAs themselves.

We believe that a substantive barrier to the establishment of SOAs is that there is little shared understanding about how services should be developed, what granularity is appropriate for different problems, and no common notation to enable developers to share designs.

2.3 Modeling Services

Dijkman and Dumas [5] explain the need for particular Service Oriented Design strategies, based on a number of characteristics that differentiate Service from Component-based design: High Autonomy (of designers and developers), Coarse Granularity (of service interfaces), and Process Awareness (close relationship with business processes). Enterprise level service development is most effected by the later two characteristics. For example, Quartel *et al* [12] describe the use of design milestones to help develop web services from business practices, and Benatallah and Dumas [3] have created environments to ease the creation of composite services. Martin et al. [9] suggest that the best way to implement Web Services in an enterprise is to start with a component-based architecture that exposes business process level services as web services.

In more loosely coupled community efforts, such as the JISC e-Framework [11], the first characteristic of SOA design, High Autonomy, becomes the dominant problem, as services for the framework are being developed by a wide variety of institutions for a number of purposes. What is required is not just a common repository for services, but a community wide understanding of the domain, and how independently authored services fit within in.

Wilson et al. [15] present *Reference Models* as a potential solution. They describe a situation in which 'a Framework is used to derive a Reference Model, which is used in a particular Design which then results in an Artifact, such as a piece of delivered software'. Broadly speaking a Reference Model can be thought of as a description of how a set of services within a Framework collaborate to provide the necessary functionality for a particular domain.

Reference models are a way to help architects and software vendors make consistent logical divisions in their architectures and products. But they require a method for describing services and their interactions at an abstract, logical level. Wada *et al* [13] have taken a model driven approach to this problem, building a model of the domain and then using this to derive an object design; this kind of modeling has also been used with SOAs to validate a design [1].

The authors believe that the model-driven approach to service-design, while worthwhile in many domains where there is a consistent/constrained understanding of the processes, may be to heavyweight for situations where the domain is broader. In these situations an agile software engineering approach seems more appropriate.

3 SRI-DM

The Service Responsibility and Interaction Design Method (SRI-DM) separates abstract representations of Services from their implementation, it uses a collection of logical descriptions (called *Service Profiles*) to describe how a number of services, regardless of implementation, might be combined to fulfil a particular problem scenario defined as a Use Case. We have based our approach on the following principles:

- To facilitate and record a clear design path from a problem scenario to a software implementation.
- To be informed by agile software engineering practices:
  - Produce the simplest model that is useful
  - Draw on close relationship with domain experts to define scenarios
  - Enable developers to build the simplest service architecture with quality attributes of cohesion and loose coupling
- Use UML 2.0 as a modeling method where possible, to enable understandability and promote links to case tools
- Work at an abstract level that is non-prescriptive at implementation level

There is a tension when designing services between ensuring that services are atomic (to encourage reuse) and yet designing services that are appropriate building blocks for a higher purpose, enabling the services to be combined to create a larger system. In effect services are always created within a context, and yet must be described independently from that context to be fully reusable.

SRI-DM achieves this by treating individual Service Profiles as atomic, and placing the description of how they might be combined in a separate sequence diagram that is tied to a particular scenario. Therefore the method produces a design that has the following parts:

- **A Scenario**: presented as a Use Case Diagram and narrative that describes a problem for which a set of services can provide a solution.
- **Service Profiles**: a set of profiles that describe a number of services at an abstract logical level. These give an impression of the granularity of services within the scenario and describe the individual

capabilities of each one, they are designed to promote reuse and understanding of the design, while retaining flexibility in the implementation.

- **Sequence Diagram**: This describes one example of how the services can interoperate to fulfil the scenario.

Service Profiles are not concrete interfaces and so cannot be described using interface definition languages (such as WSDL), instead they set the granularity of the model, and describe in a semi-formal way the role of each service and the potential way in which they might rely on one another.

In the rest of this section we will look at each part of the SRI-DM, Scenario, Service Profiles, and Sequences, and describe their formal notation.

3.1 Scenarios

Our method takes as its starting point a scenario that describes a problem that is to be solved using a set of interacting services. We have chosen Use Case diagrams as our method of modeling because they are high level and implementation independent. From an agile pointy of view they are also useful in that they relatively informal, yet help to define and structure a problem space, without going into too much detail about the activities within that space. A brief narrative description is held alongside the diagram as a whole, as well as for each individual use case, this description helps disambiguate the use cases, explains the roles of the different actors associated with that use case and focuses at a high level on what each use case involves.

Scenarios are developed in a community or user focused manner in line with agile principles to ensure that they are relevant. Effectively the use case diagrams capture the practice of an existing user community.

3.2 Service Profiles

Service Profiles are abstract descriptions of a service that may be fulfilled by several different *Service Implementations* that potentially expose different concr$ete interfaces. We therefore needed to model Service Profiles in a high level way that does not prescribe a data model or dictate explicit methods. To do this we created Service Resource Cards (SRCs), based on an existing agile technique called Class Responsibilities/Collaborations first described by [2].

Our SRC models the capability of a service to realise a specific use case (a single bubble from a larger use case diagram). The aim of the cards is to help articulate a design, to guide refinement of that design, to model for understandability, and to model at an appropriate granularity. The SRCs do not show how services may be combined in a wider scenario, but do model possible collaborations with other services that might occur for this service to fulfill its own specific use case.

An SRC card is a small card (we use A5 address cards)

- The name of the service appears at the top of the card
- Down the left hand side of the card, we list the responsibilities of the service
- On the right hand side we list and group other services which collaborate to fulfill the responsibilities listed on the left hand side.

The responsibilities of a service describe at a high level the purpose of a service's existence: what is it for, what does it do, what can it provide to other components?

The guidance for CRC design is that a class should not have more than 3-4 responsibilities, as too many responsibilities corresponds to low cohesion in a class (a measure of a poor quality design). We have not undertaken a study to evaluate whether this is universally true for service design, although this guideline seemed appropriate for the SRCs we have developed in our e-assessment domain cases.

The use case from which the service was derived can help indicate where collaborations will be required. In particular, *include* relationships are a strong indicator that a collaboration should be used, although as too many collaborations create a tightly coupled design they should be suggested sparingly. In particular, Use Cases connected through a Use Case *actor* do not necessarily collaborate.

The Service Profile is atomic in that any connection with other services is described in terms of how that connection might help this service fulfil its own role. This is different from describing how a set of services might be used together for some purpose that is greater than any individual service. This requires a different kind of specification, with a separate narrative that defines the greater purpose, and a more detailed formal description of how a set of services co-operate to fulfil that purpose.

3.3 Service Sequence Diagrams

At the scenario level, services represented by SRCs must interact with each other to fulfil a wider purpose. These interactions are complex and include transactions, sequences and state. We looked at a number of UML2.0 diagrams for representing a dynamic model, including state transition and activity diagrams. We decided that if the scenario modeling was to maintain the high level of abstraction necessary for agile development then it would be inappropriate to declare a detailed data model, or to specify the logic of the communicating services. So we use Sequence Diagrams to represent the interactions, this shows which services should communicate and in which order, and contains enough description to show how the individual services are responsible for moving and processing data, without having to specify the detail of the data model or the decision making logic.

4 An Example Factoring using the SRI Design Method

We call the process of deriving a set of services for a given scenario *Service Factoring*. The philosophy behind our method is that this whole process is transparent and fully audited. It begins with a community consultation exercise that produces a number of scenarios. This are formalised as Use Case Diagrams, and each Use Case becomes the starting point for Service Factoring. From the Use Case an SRC, or set of SRCs, are created. Referring back to the Use Diagram of the Scenario allows us to specify a sequence diagram that describes how these services interact to fulfil the goal of the scenario. In this section we will briefly show the results of performing this process of the domain of e-Assessment.

4.2 Developing the Use Cases

The authors have been involved in a project called FREMA (the Framework Reference Model for Assessment) which has examined how a number of scenarios from the e-assessment domain might be supported via services. The first part of the project was to create a domain definition, information gathered about the domain in terms of projects, standards, software and services that were concerned with assessment. To do this we elicited practice from a number of members in the e-assessment community via workshops and semi-formal interviews, including the UK Centre for Educational Technology Interoperability Standards (CETIS), Qualification agencies such as SQA and Edexcel, and a number of Higher Education Institutions, including the UK Universities of Hull, Loughborough, Southampton, and Kingston, and the Spark University of Technology Sydney, Australia.

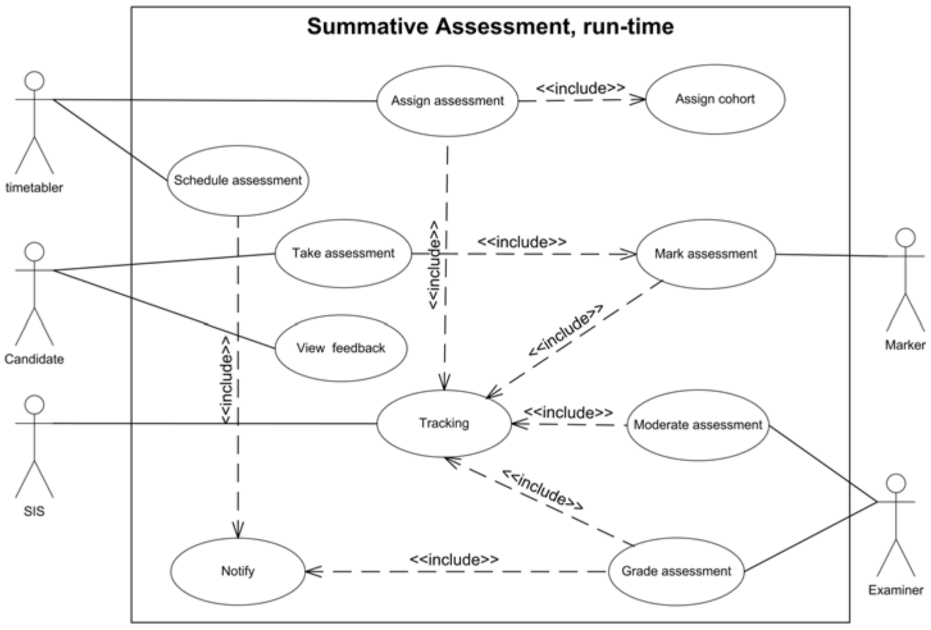

Fig. 1. The Use Case Diagram for the run-time portion of the End-to-end Summative Assessment Scenario (abridged for simplicity).

The resulting view of assessment was very broad, including systems that augment or facilitate traditional assessment processes. However, the most common scenario was one of Computer Aided Assessment (CAA), it concerns a lecturer or teacher who can set summative assessments digitally, so that they can be taken remotely, and possibly within a flexible timeframe. We call this the End-to-end Summative Assessment Scenario. Figure 1 shows a part of the Use Case diagram constructed for this scenario. The granularity of the use case translates directly to the granularity of the Service Profiles (although there is not necessarily a one-to-one mapping of Use Case to Service Profile). In the next section we will show how one of these use cases, Take Assessment, was converted into an SRC card.

4.2 Constructing the SRCs (Take Assessment)

Deriving SRCs from Use Cases is a complex process, often an earlier factoring must be revisited in order to pull out common collaborations. We use the following process:

1. Work through each use case. A traditional noun and verb analysis is a useful technique; verbs can indicate the responsibilities of the services that fulfil the use case, and the nouns imply a data model. From the verb analysis write down all of the operations needed to satisfy a use case.
2. Consider which responsibilities might be common with other SRCs and move them from responsibilities to collaborations
3. Group the operations into responsibilities
4. Identify which responsibilities would benefit from which collaboration
5. Test the completeness/accuracy of the design by working various scenarios
6. Re-visit the SRC and re-factor as necessary as other SRCs are developed, and common collaborations become apparent.

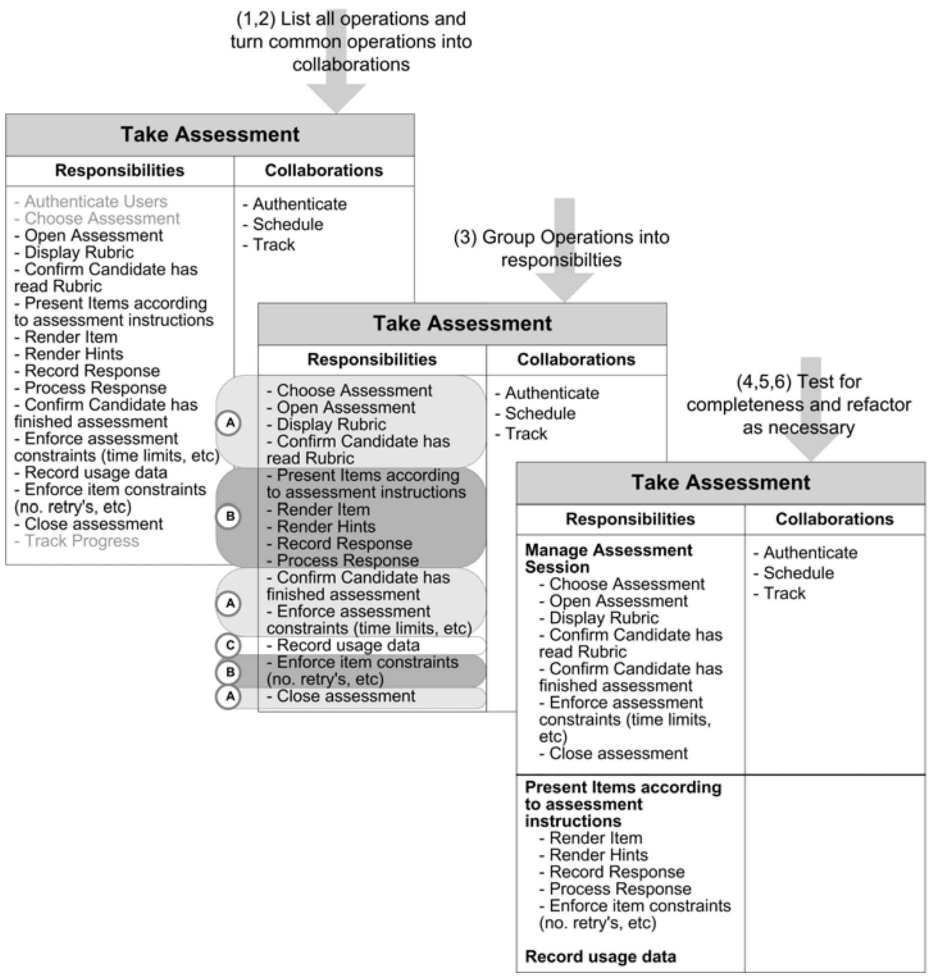
(1,2) List all operations and turn common operations into collaborations
Take Assessment
Responsibilities
Collaborations
- Authenticate Users
- Choose Assessment
- Open Assessment
- Display Rubric
- Confirm Candidate has read Rubric
- Present Items according to assessment instructions
- Render Item
- Render Hints
- Record Response
- Process Response
- Confirm Candidate has finished assessment
- Enforce assessment constraints (time limits, etc)
- Record usage data
- Enforce item constraints (no. retry's, etc)
- Close assessment
- Track Progress
- Authenticate
- Schedule
- Track
(3) Group Operations into responsibilties
Take Assessment
Responsibilities
Collaborations
A
B
A
C
B
A
- Choose Assessment
- Open Assessment
- Display Rubric
- Confirm Candidate has read Rubric
- Present Items according to assessment instructions
- Render Item
- Render Hints
- Record Response
- Process Response
- Confirm Candidate has finished assessment
- Enforce assessment constraints (time limits, etc)
- Record usage data
- Enforce item constraints (no. retry's, etc)
- Close assessment
- Authenticate
- Schedule
- Track
(4,5,6) Test for completeness and refactor as necessary
Take Assessment
Responsibilities
Collaborations
Manage Assessment Session
- Choose Assessment
- Open Assessment
- Display Rubric
- Confirm Candidate has read Rubric
- Confirm Candidate has finished assessment
- Enforce assessment constraints (time limits, etc)
- Close assessment
Present Items according to assessment instructions
- Render Item
- Render Hints
- Record Response
- Process Response
- Enforce item constraints (no. retry's, etc)
Record usage data
- Authenticate
- Schedule
- Track

Fig. 2. The factoring of the Take Assessment Use Case into a SRC

Figure 2 shows this process applied to the Take Assessment Use Case (the numbers above each card refer to the numbered stages, described above, that produced it). The Use Case description is used to derive the initial list of operations, which are consequently factored into a set of responsibilities and collaborations. Sometimes the operations that are moved to collaborations also remain as responsibilities (for example, *Choose Assessment* becomes a collaboration called *Schedule*, but remains as one of the responsibilities of the Service), this is because the service still has a responsibility to allow users to choose an assessment, even if this is done via a collaboration. Tracking on the other hand is removed as a responsibility because it is not something that this service offers to others. In general the collaborations should be considered to be recommendations only, and in need not be separated out into separate services at the implementation stage.

4.3 Building a Sequence Diagram

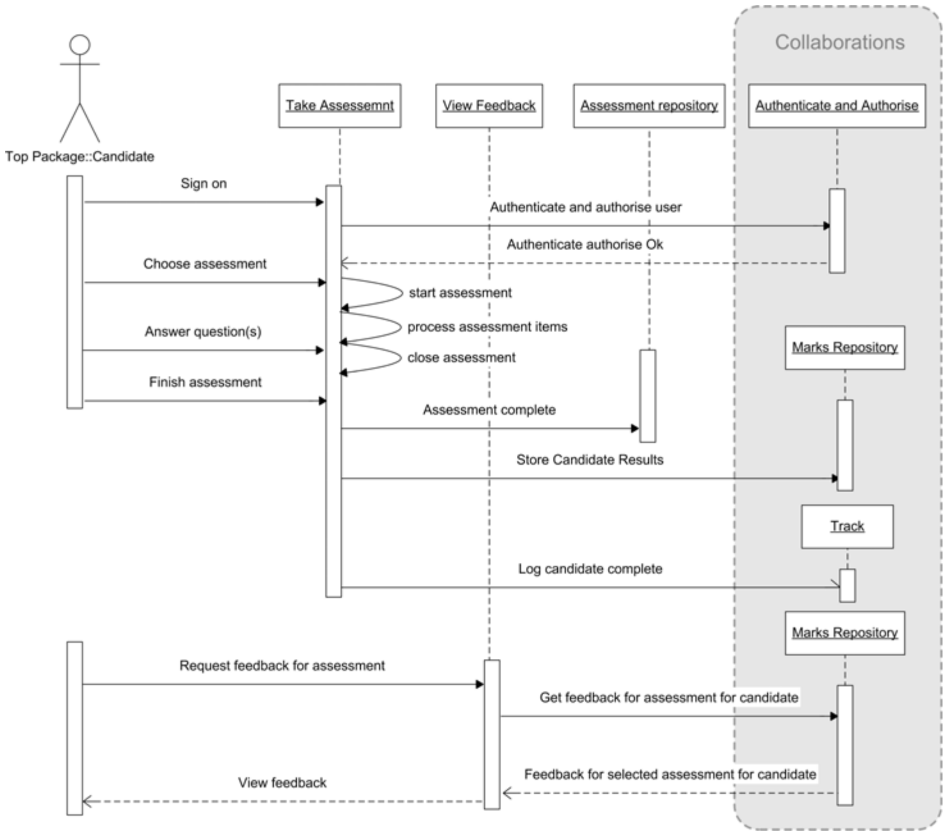

Fig. 3. Sequence Diagram from the *Take Assessment* Use Case shown in Figure 1.

The sequence diagrams demonstrate one way in which a group of services can interact to fulfill the original scenario. It cannot be a definitive representation of service interaction in general, as services as asynchronous, and some of the communication can be reordered without affecting the performance of the system as a whole. So the sequence diagrams act as a demonstration and validation of the SRCs, rather than a definitive template for service interaction. Figure 3 shows a sequence diagram for the part of the scenario related to Take Assessment, and in particular the interactions around the candidate. Collaborations are modeled, although in this paper they are grouped together into one column to aid clarity.

5. Validation and Discussion

Our validation strategy has been to ensure that the designs produced via SRI-DM are sensible, accessible and intuitive. To this end we have undertaken a formal evaluation of our scenarios and our methods of presenting them. For our formal evaluation we presented versions of our e-Assessment Domain Definition and resulting scenarios at the CETIS Assessment Special Interest Group (SIG). This is a self-selecting group which includes early adopters, developers and representatives of standards bodies; the Assessment SIG is especially knowledgeable about the domain and has a considerable stake in the evolution of interoperable, open services. The reaction of the group was encouraging, they believed that the scenarios that we had developed were accurate and important to the e-Assessment domain, and the use case diagrams that we presented captured the scenarios well.

We have also presented the CETIS group with the SRC and interaction diagrams that we have created of the CAA scenario. Reaction to the individual cards and the interaction diagram was good. All delegates agreed that this was a sensible granularity at which to model services, and that the SRC and interaction diagrams were complimentary. Many believed that this lightweight modeling would be useful in their existing service design practice. Based on this reaction we now in a position to engage more directly with community members, and allow them to use the SRCs and interaction diagrams that we have defined to develop new tools. We are engaging with a group of developers at Kingston University to undertake a more formal evaluation of the SRI-DM, both in terms of its representation (via a formal design review, evaluating the effectiveness of the model compared to its aspirations), and as a service design process (using the SRI-DM to guide the development of a mini-project at Kingston with a formal evaluation of the approach at the project's close).

We have already used SRI-DM ourselves to create a set of services for several core assessment scenarios (including CAA), and as a result have a number of personal reflections on the design method. We believe that one of the most difficult challenges with service design is choosing an appropriate granularity at which to define services. With SRI-DM we have chosen a top-down approach that is firmly built on a starting scenario and use-case diagram, these are typically high level views of a problem space, and translating them almost literally into service profiles, produces a high level design. But because SRI-DM does not capture business logic, or interfaces in a detailed way, it becomes easy to re-factor services in order to break down that high level design into a level with which the designers are comfortable. This approach is agile, as it requires only a little modeling overhead, and produces a stable service design before the expensive process of agreeing data models and interfaces is undertaken. It also produces design documentation as part of the design process, rather than adding a separate task of recording an external design process.

Another challenge with service design is agreeing on a service workflow. SRI-DM does not attempt to define the full rules of interaction (causal relations, points of synchronization, critical paths, etc). This is another way in which SRI-DM is an agile approach; a full model of all the ways in which services can interact is not needed to produce a working system of services, and so SRI-DM does not make designers create this. Instead it demonstrates the validity of a service design by showing *one example* of how a set of services could interact to fulfill the scenario.

Our major observation while developing services with SRI-DM is the paucity of traditional flat-file documentation for linking evidence with decision making, this inflexibility in justifying design decisions may be a real problem with SOAs due to the distributed way in which services are often created. To cope with this we have been developing the notion of a *Community Reference Model* alongside SRI-DM, this is a community web site, where the scenarios and their evidential resources can be described, linked and discussed[1]. We hope that by explicitly supporting the use-cases, service profiles and interaction diagrams of SRI-DM we can also encourage the community to start building common models of how services can interact to fulfill scenarios, leading eventually to common services themselves. We are currently developing this idea using a *Semantic Wiki*, and plan to hand this resource over to the e-assessment community through the CETIS SIG later this year.

6. Conclusions

In this paper we have presented the Service Responsibility and Interaction Design Method (SRI-DM), an agile approach to designing Web Services (agile as it is light-weight, and produces documentation as part of the design process itself). The SRI-Design Method focuses on the rapid factoring of a set of services given a well-understood scenario. We are currently evaluating SRI-DM through an independent mini-design project, and plan to take the method forward to the e-assessment development community through a web-based Community Reference Model.

[1] For an example see the FREMA web site: www.frema.ecs.soton.ac.uk

As SOAs become more reliable, and the standards underlying them more stable, it seems inevitable that they will form the basis of more distributed systems. If these systems are to be created as quickly and as flexibly as current software deployments then we must create design methodologies that are agile enough to cope with rapid turnaround, yet create designs that are fit-for-purpose, and leave a documentation trail strong enough to support software throughout its lifetime.

References

1. Baresi, L., Heckel, R., Thöne, S., and Varró, D. (2003). Modeling and validation of service-oriented architectures: application vs. style. In Proceedings of the 9th European Software Engineering Conference Held Jointly with 11th ACM SIGSOFT international Symposium on Foundations of Software Engineering (Helsinki, Finland, September 01 - 05, 2003). ESEC/FSE-11
2. Beck, K. and Cunningham, W. (1989), A laboratory for teaching object oriented thinking, ACM SIGPLAN , Notices, 24(10):1-6, October 1989
3. Benatallah B., Sheng Q., and Dumas M (2003). The Self-Serv environment for web services composition. IEEE Internet Computing, 7(1):40-48, Jan/Feb. 2003.
4. Curbera, F.; Duftler, M.; Khalaf, R.; Nagy, W.; Mukhi, N.; Weerawarana, S., "Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI," *Internet Computing, IEEE* , vol.6, no.2pp.86-93, Mar/Apr 2002
5. Dijkman, R. and Dumas, M (2004). Service-oriented Design: A Multi-viewpoint Approach. International Journal of Cooperative Information Systems 13(4), December 2004.
6. Fielding, R. T. and Taylor, R. N. 2002. Principled design of the modern Web architecture. *ACM Trans. Inter. Tech.* 2, 2 (May. 2002), 115-150.
7. Foster, I., Kesselman, C., and Tuecke, S. (2001). The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *Int. J. High Perform. Comput. Appl.* 15, 3 (Aug. 2001), 200-222.
8. Highsmith, J. and Cockburn, A. (2001), “Agile software development: the business of innovation”, Computer, Sep 2001, Volume: 34, Issue: 9, pg 120-127, ISSN: 0018-9162
9. Martin J, Arsanjani A, Tarr P, and Hailpern B, (2003), "Web Services: Promises and Compromises," Queue vol. 1, pp. 48-58, 2003.
10. Mukhi N. K. and. Plebani P (2004), "Supporting policy-driven behaviors in web services: experiences and issues.," in proceedings 2nd international Conference on Service Oriented Computing ICSOC '04, (New York, NY, USA, 2004).
11. Olivier B., Roberts T., and Blinco K., (2005) "The e-Framework for Education and Research:An Overview," DEST (Australia)
12. Quartel D.A.C., Dijkman R.M., and van Sinderen M.J.. (2004) Methodological Support for Service-oriented Design with ISDL. In: Proceedings of the 2nd ACM International Conference on Service Oriented Computing (ICSOC), New York City, NY, USA, pp. 1-10, 2004
13. Wada, H., Suzuki, J., and Oba, K. (2005). Modeling turnpike: a model-driven framework for domain-specific software development. In Companion To the 20th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (San Diego, CA, USA, October 16 - 20, 2005). OOPSLA '05. ACM Press, New York, NY, 128-129.
14. Weatherley J., (2005), "A web service framework for embedding discovery services in distributed library interfaces," in proceedings 5th ACM/IEEE-CS Joint Conference on Digital Libraries JCDL '05, Denver, CO, USA, 2005.
15. Wilson, S., Blinco, K. and Rehak, D. (2004). Service-Oriented Frameworks: Modeling the infrastructure for the next generation of e-Learning Systems. A Paper prepared on behalf of DEST (Australia), JISC-CETIS (UK), and Industry Canada.