Service-Oriented Migration and Reuse Technique (SMART)

Grace Lewis, Edwin Morris, Dennis Smith
Software Engineering Institute, Pittsburgh, PA, USA
{glewis, ejm, dbs}@sei.cmu.edu
Liam O'Brien[1]
Lero – The Irish Software Engineering Research Centre
liam.obrien@ul.ie
1. Liam O'Brien was affiliated with the Software Engineering Institute at the time this work was performed.

ABSTRACT

This report describes the Service-Oriented Migration and Reuse Technique (SMART). SMART is a technique that helps organizations analyze legacy systems to determine whether their functionality, or subsets of it, can be reasonably exposed as services in a Service-Oriented Architecture (SOA), and thus to achieve greater interoperability. Converting legacy components to services allows systems to remain largely unchanged while exposing functionality to a large number of clients through well-defined service interfaces. A number of organizations are adopting this approach by defining SOAs that include a set of infrastructure common services on which organizations can build additional domain services or applications. SMART considers the specific interactions that will be required by the target SOA and any changes that must be made to the legacy components. An early version of SMART was applied with good success to assist a DoD organization in evaluating the potential for converting components of an existing system into services that would run in a new and tightly constrained SOA environment.

1. Introduction

Many organizations have been attempting to renew their legacy systems and achieve greater interoperability by exposing all or parts of it as *services*. A service is a coarse-grained, discoverable, and self-contained software entity that interacts with applications and other services through a loosely coupled, often asynchronous, message-based communication model [3]. A collection of services with well-defined interfaces and shared communications model is called a service-oriented architecture (SOA). A system or application is designed and implemented as a set of interactions among these services.

The characteristics of SOAs (e.g., loose coupling, published interfaces, and standard communication model) offer the promise of enabling existing legacy systems to expose their functionality, presumably without making significant changes to the legacy systems. Migration to an SOA can represent a complex engineering task, particularly when the services are expected to execute within a tightly constrained environment.

SOA migration tasks can be considered from a number of perspectives including that of the end client or user of the services, the SOA architect, or the service provider.

This paper focuses on the service provider. While the paper focuses on the role of the service provider, it takes into account the needs of the ultimate user in making decisions about the relevance of migrating legacy assets to services. Section 2 discusses overall issues in the creation of services from legacy components. Section 3 outlines the SMART process for evaluating legacy components for their potential to become services in an SOA. Section 4 briefly discusses the pilot application of this process on an actual project. Section 5 provides conclusions and discusses next steps.

2. Creation of Services From Legacy Components

Enabling a legacy system to interact within a service-oriented architecture, such as a Web services architecture, is sometimes relatively straightforward—this is a primary attraction to the approach for many businesses. Web service interfaces are set up to receive SOAP messages, parse their content, invoke legacy code directly or through a custom component that invokes the legacy code, and optionally wrap the results as a SOAP message to be returned to the sender. Many modern development environments provide tools to help in this process, and commercial organizations are rapidly employing these environments to expose their business processes to the world.

Proceedings of the 13th IEEE International Workshop on Software Technology and Engineering Practice (STEP'05)
0-7695-2639-X/05 $20.00 © 2005 IEEE

However, characteristics of legacy systems, such as age, language, and architecture, as well as of the target SOA can complicate the task. This is particularly the case when migrating to highly demanding and proprietary SOAs. Such migrations will likely rely less on semi-automated migration, and more on careful analysis of the feasibility and magnitude of the effort involved. This analysis should consider:

1. Requirements from potential service users. It is important to know what applications would use the services and how they would be used. For example, what is the information expected to be exchanged? In what format?
2. Technical characteristics of the target environment. There are many technical underpinnings that need to be understood, especially in proprietary environments, such as bindings, messaging technologies, communication protocols, service description languages, and service discovery mechanisms.
3. The architecture of the legacy system. It is critical to identify architectural elements that could be problematic in the target environment or that could increase the difficulty of the effort, such as dependencies on commercial products or specific operating systems, or poor separation of concerns.
4. The effort involved in writing the service interface. Even if it is expected that the legacy system will remain intact, there needs to be code that receives the request, translates it into calls to the legacy systems, and produces a response.
5. The effort involved in the translation of data types. Service interfaces usually prescribe a set of data types that can be transmitted in messages. For newer legacy systems and basic data types this can be a small effort, especially if messages are XML documents. But, in the case of complex data types such as audio, video, and graphics, or in legacy programming languages that do not provide capabilities for building XML documents, this effort can be non-trivial.
6. The effort required to describe the services. In an SOA, services advertise their capabilities for other systems to use, and systems find the services they need by using the discovery mechanism prescribed by the target environment. The more detailed and precise the description of the service, the greater the chances it will be discovered and used appropriately. In critical situations, the description may have to include information about qualities of service, such as performance, reliability, and security; or service level agreements (SLAs).
7. The effort involved in writing service initialization code and operational procedures. Code that is deployed as services will need to initialize itself, announce its availability, and be ready to take requests. This will require the establishment of operational procedures for the deployment of services.
8. Estimates of cost, difficulty, and risk. The information gathered in the previous points should provide for more realistic estimates.

3. The Service-Oriented Migration and Reuse Technique (SMART)

The Service-Oriented Migration and Reuse Technique (SMART) was developed to assist organizations in analyzing legacy capabilities for use as services in an SOA. SMART was derived from the Options Analysis for Reengineering (OAR) method developed at the SEI that has been successfully used to support analysis of reuse potential for legacy components [1].

SMART gathers a wide range of information about legacy components, the target SOA, and potential services to produce a service migration strategy as its primary product. However, SMART also produces other outputs that are useful to an organization whether or not it decides on migration.

SMART input (from documentation and interviews) and output are depicted in Figure 1.

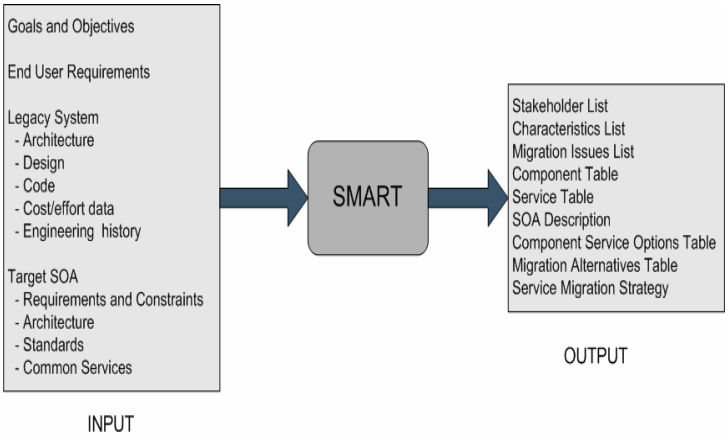

Figure 1: SMART Input and Output

SMART consists of five major activities, each

divided into several tasks. The activities and generalized process and information flows of SMART are depicted in Figure 2.

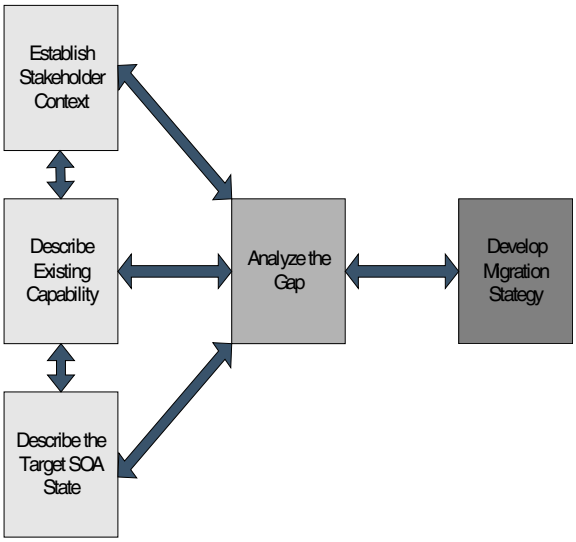

Figure 2: SMART Activities

However, the number of artifacts considered, the time required, and the specific activities of a given application of SMART depend on previous activities and expectations of the requesting organization. For example, if the requesting organization has specific legacy components in mind for migration, SMART activities will be focused on those components.

The resources and effort required for a SMART analysis will vary. SMART is most effective if a preliminary screening of assets has been made to focus on those of the highest potential value. For the pilot application described in Section 4, the organization had selected seven candidate services to analyze that included 29 classes and about 24,000 lines of code from a total application of about 800,000 lines of code. The SMART team included 3 analysts who understood the method. The team spent 6 days at the client organization interviewing stakeholders and maintenance programmers, and 4 days of additional analysis off-site. From the client organization, this application required 1 day of effort from each of 3 knowledgeable management level stakeholders, 3 days of effort from a chief architect and 2 days of effort from each of two maintenance programmers. A SMART analyst should be a person knowledgeable in software design and maintenance. It requires about 3 days to be trained in the method, plus participation in a SMART application.

Information for the first three activities (on the left) is gathered during an initial orientation meeting and through several additional meetings between the SMART team and the organization tasked with the migration activities. During these meetings, the SMART team assesses stakeholder needs, identifies the SOA vision, and elicits a high-level description of the architecture and other features of the legacy system (as listed in Figure 2). Available documentation is gathered for the legacy system in general, for legacy components that may be transitioned to services (if previously identified), and for the target SOA. In some cases, the target SOA may not be complete, so SOA documentation may describe a future state.

Information-gathering activities for the first three activities are directed by the Service Migration Interview Guide (SMIG). The SMIG contains questions that directly address the gap between the existing and target architecture, design, and code, as well as questions concerning issues that must be addressed in service migration efforts [4]. Use of the SMIG assures broad and consistent coverage of the factors that influence the cost, effort, and risk involved in migration to services.

The Service Migration Interview Guide (SMIG) is an instrument that guides the discussions with stakeholders and developers in the first three activities of the SMART process:

- Establish Stakeholder Context
- Describe Existing Capabilities
- Describe the Service-Based State

Data collected from the SMIG helps to determine the degree of difficulty and level of effort required to migrate legacy code into services. The use of this instrument assures broad coverage and consistent analysis of difficulty, risk, and cost issues.

Information gathered during the interviews includes:

- Stakeholder information
- General migration issues
- Data concerning legacy components
- Risks and issues specific to the legacy components

- Potential services
- Target SOA characteristics

It is not necessary for the team to complete all data gathering during these initial activities. Additional opportunities are provided during the Analysis activity.

The five activities and associated tasks of SMART are detailed in Sections 3.1 through 3.5.

3.1 Establish Stakeholder Context

In order to establish the context in which the migration to services will take place, the SMART team employs the SMIG to solicit information about stakeholders. Stakeholders typically include the owners and current end users of the legacy system, and the potential end users of the migrated services operating within the SOA. Other stakeholders who are sometimes important include those who are funding or controlling the migration effort, groups defining the target SOA, and Verification and Validation groups that will certify the properties of the new services.

The key to this activity is to identify who knows most about the legacy system, what it currently does, and what it should do as a service or set of services. A significant but non-obvious goal is to identify the parties who are best situated to indicate whether there is sufficient demand for the service to warrant migration efforts. Input from these parties is critical to counteract any tendency toward assuming without evidence that the legacy system is a good source for useful and appropriate services. Their input will also influence the interface design for the resulting services.

This activity also initiates the construction of a list of legacy component characteristics that will later drive the analysis process. A list of migration issues is also begun.

The SMIG contains questions that will guide the capture of information related to:

- Goal of Migration
- Expectations
- Potential Service Users
- Legacy System End Users and Owners
- Contractors
- Legacy Components and Potential Services

Selected examples of SMIG related questions for this activity include:

- Who owns the legacy system? If there is more than one owner, are these separate organizations?
- Who are the potential end users of the services? Have they provided requirements? In what form? What types of applications will be using the services?
- Have legacy components to be migrated to services been identified? What was the process? Is the list of components available?
- Have potential services been identified? What was the process? Is the list of services available?

3.2 Describe Existing Capability

The goal of the second activity of SMART is to obtain descriptive data about the components of the legacy system. The activity employs the SMIG to gather data about a specific set of topics related to the legacy system, but the SMART team has the latitude to pursue interesting leads. For example, the SMART team may ask questions about the philosophy and strategies applied for use of COTS products in the legacy system on learning that the system developers opted to use a custom (non-standard) interface to a commercial database.

Basic data solicited during this activity includes the name, function, size, language, operating platform, and age of the legacy components. Technical personnel are questioned about the architecture, design paradigms, code complexity, level of documentation, module coupling, interfaces for systems and users, and dependencies on other components and commercial products.

In addition, data about the relative quality and maturity of legacy components is gathered, including outstanding problems, change history, user satisfaction, and likelihood of meeting longer term needs. Historical cost data for development and maintenance tasks is collected to support effort and cost estimates.

The SMIG contains questions that will guide the capture of information related to

- legacy system characteristics
- legacy system architecture
- code characteristics

3.3 Describe the Target SOA State

The third activity of SMART is intended to

- gather evidence about potential services that can be created from the legacy components
- gather sufficient detail about the target SOA to support decisions about what services may be appropriate and how they will interact with the each other and the SOA

Initial information about potential services often comes via SMIG-directed conversations with legacy component owners. However, the information gathered must be tempered by data from users, corporate architects, domain groups, communities of interest, and reference models that address service definition. In some cases, these groups and models will define the entire set of services that support the organization's goals, and into which any potential services built from the legacy components must fit.

The characteristics of the target SOA will temper decisions about whether legacy components can be reused. The degree to which a legacy component is inconsistent with these characteristics will profoundly influence the overall migration costs.

Note that the target SOA can be owned by the same organization that owns the legacy components, or by another organization. It may provide a fixed or pre-existing architecture, or the architecture for the SOA may be developed simultaneously with the reengineering of legacy components. The actual placement along this spectrum will have important technical and political consequences for decisions that are made.

The SMIG contains questions that will guide the capture of information related to

- service requirements
- target SOA and legacy system adaptation
- service-oriented changes
- support

3.4 Analyze the Gap

The goal of the fourth activity is to identify the gap between the existing state and the future state and determine the level of effort and cost needed to convert the legacy components into services. This analysis may also suggest potential tradeoffs between the target architecture and the legacy components. For example if the target SOA is flexible, or if it is still in the process of being defined, a relatively minor change to its requirements may allow more legacy components to be converted to services or may simplify the conversion effort. However, substantial risks to the migration effort are introduced when the target SOA has a large number of to-be-defined areas.

SMART uses several sources of information to support the analysis activity. The issues, problems, and data gathered as the SMART team investigates the available components, required services, and SOA requirements form one source of information. A second, optional source of information involves the use of code analysis and architecture reconstruction tools [2,5] to analyze existing source code. Where documentation is insufficient or where there is uncertainty about code characteristics such as dependencies on commercial products, tool analysis is very helpful. This option can also be used with great effect to survey representative portions of the code to verify other opinions and judgments.

3.5 Develop Migration Strategy

The final activity of SMART involves recommending one or more of the options documented in the Component Service Options Table, selecting a strategy to achieve the goal, and presenting the SMART team findings. In many cases, the migration strategy may involve multiple steps, such as an initial "quick and dirty" wrapping, followed by restructuring of the application (now service) into appropriate layers, and finally by modification to use other services. Example elements of a strategy include

- the identities of specific components to migrate
- recommendations regarding the ordering of migration efforts

- specific migration paths to follow (simple wrapping vs. rewriting of code)
- identification of increments that lead to increasing capability
- suggestions regarding organization(s) best equipped to lead the migration effort
- suggested coordination with related efforts (for example, SOA infrastructure builds)

SMART provides a preliminary analysis of the viability of migrating legacy components to services, migration strategies available, and the costs and risks involved. In particular, it attempts to answer several questions:

- Which components can reasonably be used to derive services?
- What sorts of activities must be performed to accomplish the migration?
- What strategies are most appropriate for the migration effort?

The sponsoring organization receives a detailed briefing of the results of SMART, but the briefing is not intended to replace system engineering activity. It is assumed that the organization will reflect on the results and pursue further engineering analysis along the lines recommended by SMART.

4. Summary of Results From A Pilot Implementation

An early version of SMART was applied in a recent analysis of the potential for migrating a set of legacy components from a DoD command and control (C2) system to a target SOA. An overview of this application is presented below. More complete information is available in [4].

The owners of the systems recognized that a selected set of components from their C2 system, if converted to application domain services (ADS), would have broad applicability. They had targeted potential services as part of their initial analysis of ADS requirements. The SMART team's role was to perform a preliminary evaluation of the feasibility of converting a set of their components into these application domain services.

To determine the existing capabilities of the C2 system, the SMART team met with the contractor and representatives of the government to focus on a limited number of legacy components and to select characteristics for further screening. These sources provided significant detail about the legacy system, but the available architecture documentation was incomplete. In particular, logical and development views of the system architecture were not available. This represented a problem for our analysis.

The current system, written in C++ on a Windows operating system, had a total of about 800,000 lines of code and 2500 C++ classes. In addition, the system had dependencies on a commercial database and a second product for visualizing, creating, and managing maps. Both commercial products have only Windows versions.

The team focused on the 29 specific C++ classes that would presumably provide the basis for the seven potential services that the government team had previously identified, and that offered high probability of providing useful insight. The team identified characteristics that would be the focus for analyzing the components, starting with those provided by OAR and supplemented with team knowledge of the necessary characteristics of services operating within the target SOA.

In examining the potential for reuse of the existing legacy components, the team found that the current legacy code represents a set of components with significant reuse potential. However, because the current legacy system does not have sufficient architecture or other high-level documentation, it was difficult to understand the "big picture" as well as dependencies between classes.

To avoid this problem with future systems, the team recommended that the organization require the following changes from its contractors to make reuse of its legacy components more viable:

- documentation in the form of a suitable set of architectural views
- consistent use of programming standards
- documentation of code so that comments can be extracted using an automated tool
- documentation of dependencies, especially when they violate architectural patterns

A good starting point was provided by the

analysis of the legacy components, based on the characteristics identified as important during the data gathering activities. However, the team performed additional analysis of the code, as well as an architecture reconstruction to obtain additional data. The architecture reconstruction provided an "as-built" representation of the structure of the system and its dependencies. It suggested that the significant dependencies between classes will make reuse and deployment of services more difficult. If the migration effort moves forward, the results of the architecture reconstruction can be a starting point for understanding how to disentangle dependencies.

The largest risk in reusing the legacy components concerns the fact that the SOA has not been fully developed. While its overall structure has been defined, many of the specific mechanisms for interacting with it are still pending. Thus, it is not yet clear what the requirements for being a service in this environment will be in 12 or 18 months.

To address the SOA instability issue head on, the team recommended that the organization take a proactive approach in working with the developers of the target SOA to understand the implications of the evolving SOA on services.

The organization should also work closely with the developers of the applications who will be using these services. Even though the technical part of the communication will be handled by a common service, the data to be transferred during that communication must be negotiated—the contents of both the request and the response message that is communicated between the application and the service must be defined. An initial and crucial element of discussion should be the data model, given that it is used by all the potential services.

Dependencies on commercial products including mapping software and a database are a concern in the target environment. The Windows-based mapping software, for example, would need to be verified for use within the target SOA. A different mapping service might be required by the target SOA. There are also dependencies on a commercial database. These would have to be replaced by data access methods endorsed for the target SOA.

The team also noted that because there are dependencies between the primary services that were analyzed and a second forthcoming project that was being planned by the organization, there will be duplication of work if these are treated as separate projects.

5. Conclusions and Next Steps

The task of determining whether and how to expose legacy functionality as services can be complex. Disciplined analyses of existing components and the target SOA are necessary for sound migration decisions. SMART provides such disciplined analysis through a thorough and consistent process, a set of data-gathering activities that capture the scope of technical work to be accomplished, and artifacts that record critical aspects of the process.

We applied an early version of SMART to a command-and-control system and observed both significant potential for migration to services as well as shortcomings in documentation and code. In truth, the system owners will have a difficult time defining their services until the interfaces and expectations of the target SOA are better defined.

While the early version of SMART used to analyze the system proved valuable, there is significant room for improvement. SMART is being updated with the following goals in mind:

- Improve the breadth and consistency of information gathered about the engineering effort necessary to change the legacy artifact into a service. The SMIG is the first tool intended for this purpose. By incorporating significant technical "know how" into the SMIG, we also further an ultimate goal of transitioning the technique to other users.
- Incorporate decision rules on when it is most useful to include the code analysis and architecture reconstruction steps as part of the process.
- Develop machine support for capturing and analyzing data gathered during the SMART process. This will entail building templates for major artifacts, including the:
  - Stakeholder List
  - Characteristics List
  - Migration Issues List

- Component Table
- Service Table
- SOA Description
- Component Service Options Table
- Migration Alternatives Table
- Service Migration Strategy
- Final Presentation

- Develop techniques and criteria for determining when a SMART team has captured sufficient information to complete the analysis process.
- Establish a mechanism to capture the net effect of SMART on migration efforts. This information is essential for continued evolution and improvement of SMART.

As we continue to refine SMART, we plan to apply it to other projects and legacy systems. We are actively seeking organizations interested in applying the technique. We are also well on the way to establishing relationships with other organizations interested in adopting and improving SMART with us.

References

[1] Bergey, J.; O'Brien, L.; & Smith, D. "Using the Options Analysis for Reengineering (OAR) Method for Mining Components for a Product Line," 316-327. *Software Product Lines: Proceedings of the Second Software Product Line Conference (SPLC2).* San Diego, CA, August 19-22, 2002. Berlin, Germany: Springer, 2002.

[2] Kazman, R; O'Brien, L.; & Verhoef, C. *Architecture Reconstruction Guidelines,* 2nd Edition (CMU/SEI-2002-TR-034 ADA421612). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. http://www.sei.cmu.edu/publications/documents /02.reports/02tr034.html.

[3] Lewis, Grace A. and Wrage, Lutz. Approaches to Constructive Interoperability (CMU/SEI-2004-TR-020 ESC-TR-2004-020). January 2005. http://www.sei.cmu.edu/publications/documents/04.reports/04tr020.html

[4] Lewis, G., Morris, E. O'Brien, W., Smith, D. and Wrage, L. SMART: The Service-Oriented Migration and Reuse Technique. (CMU/SEI-2005-TN-029)

[5] O'Brien, L.; Stoermer, C; & Verhoef, C. *Software Architecture Reconstruction: Practice Needs and Current Approaches* (CMU/SEI-2002-TR-024, ADA407795). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002. http://www.sei.cmu.edu/publications/documents/02.reports /02tr024.html.