

The SOA Magazine

Feature Article



Working with SOA and RUP

by Solmaz Boroumand

Published: March 13, 2008 (SOA Magazine Issue XVI: March 2008)

[Download this article as a PDF document.](#)

Abstract: The Rational Unified Process (RUP) has been a successful methodology used to support object-oriented project delivery because it provides a series of proven disciplines and practices, several of which can still be leveraged in support of SOA, particularly in the areas of business modeling and service-oriented analysis. This article explores combining specific parts of RUP with the Mainstream SOA Methodology in support of improving the required business modeling processes that support the definition of service inventory blueprints and individual service candidates.

Introduction

Methodologies are a lot like collections of best practices organized into a formal framework. Since service-orientation and service-oriented architecture have become the focal point in the IT community, several approaches and platforms for the delivery of SOA projects have been suggested by companies and research centers, but few constitute an actual methodology.

The Rational Unified Process (RUP) developed by Rational Software in the 1990's emerged in support of object-oriented solution delivery to provide a collection of customizable techniques and practices that help realize the goals of object-orientation. Although service-orientation is considered by many to have succeeded object-oriented ways, this is not necessarily true.

As much as service-orientation owes most of its existence to object-oriented methods, there is a similarly significant link between established object-oriented methodologies like RUP and the now maturing SOA methodologies. Therefore, it is to our benefit to reap the proven practices of RUP in order to apply them to SOA delivery processes. Of course, there are parts of RUP that are less appropriate than others, and that's where a customization effort needs to come in. This article is about the exploration of such a customization.

To support this study we will be focusing on "MSOAM", the Mainstream SOA Methodology [REF-3] popularized by Erl's books. This methodology provides a generic set of approaches and processes for delivering modern-day SOA projects. It is the least proprietary SOA methodology in the industry, in that it is intentionally kept generic so that it can be customized and extended for specific enterprises and requirements. In that regard, its goals are very similar to those of RUP. MSOAM's processes range from analysis and modeling to the design of service endpoints. As part of a mainstream methodology, these processes simply raise common considerations and concerns, but do not dictate a rigid sequence.

Brief Overview of RUP

RUP is an iterative methodology that organizes project cycles into often overlapping streams that are applied and expressed in active and passive phases.

The active phases are:

- Inception
- Elaboration
- Construction
- Transition

The passive phases are explained by:

- Disciplines
- Tasks
- Workflows
- Outputs
- Roles
- Process Elements

The common RUP stages (referred to as "disciplines") are:

- Business Modeling
- Requirement Analysis
- Design
- Implementation
- Deployment
- Configuration
- Change Management
- Project Management
- Environment

RUP is well established in the IT community and we will therefore not explain it in detail in this article. If you are new to RUP, then I suggest reading Kruchten's classic book [REF-1]. For the purpose of this article, we will be focusing primarily on the RUP Business Modeling discipline.

Brief Overview of MSOAM

MSOAM is focused on both the definition and delivery of collections of services called "service inventories" and the definition and delivery of individual services as they progress through the following individual lifecycle stages:

- Business Modeling/Define Enterprise Business Models
- Service-Oriented Analysis/Inventory Analysis
- Service Contract Design (Service-Oriented Design)
- Service Logic Design (Service-Oriented Design)
- Service Development
- Service Testing
- Service Deployment
- Service Governance

A key philosophy of MSOAM is that the more up-front analysis that is carried out in support of modeling a service inventory and its services, the greater the likelihood that those services will have prolonged lifespans and therefore will impose less TCO (total cost of ownership) compared to services that are built using bottom-up approaches.

MSOAM rationalizes (forgive the pun) this by focusing on governance burden, as shown in Figure 2.

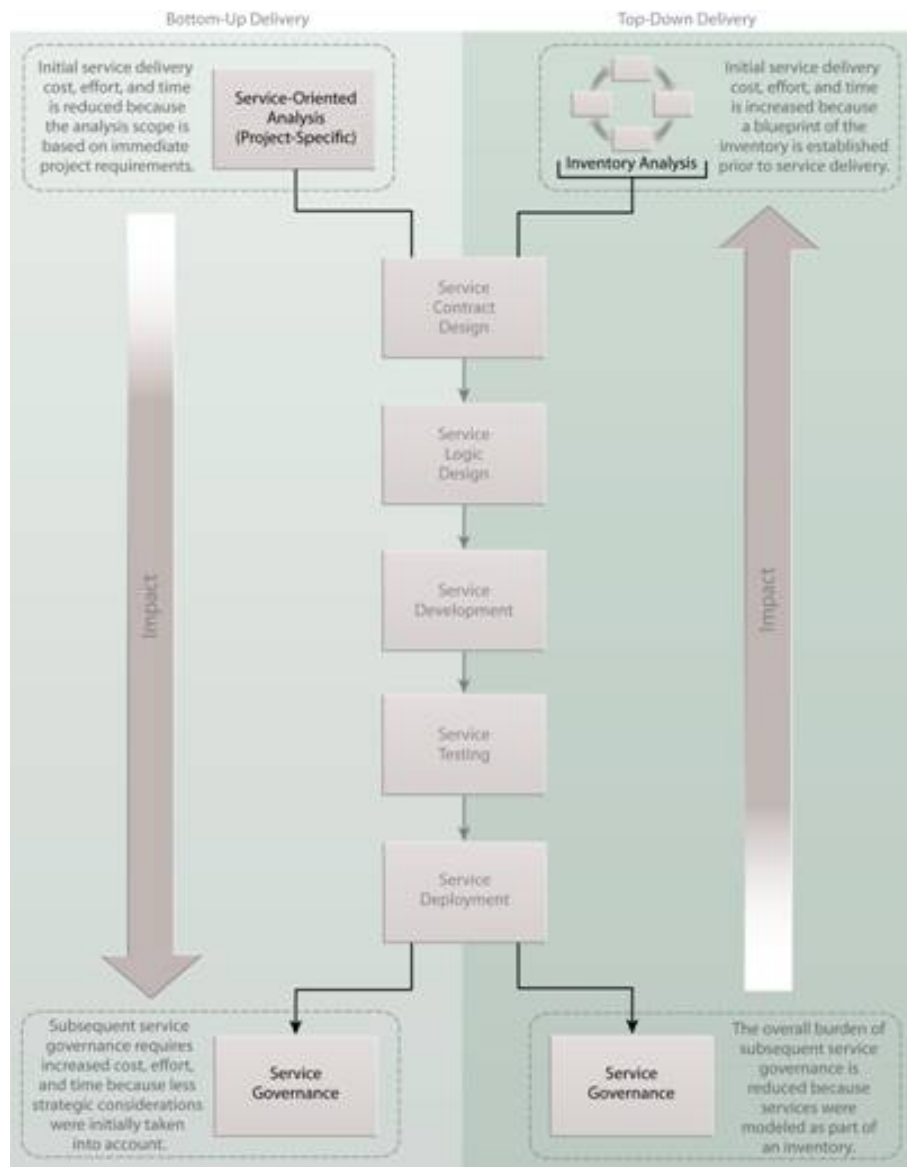


Figure 1: Bottom-up delivery approaches require less up-front impact and allow for the faster delivery of services; however, the result is that the impact is deferred to the governance stage, and the burden of owning and evolving bottom-up delivered services is increased. Top-down delivery demands more up-front analysis, but results in less per-service governance burden.

In the center of Figure 1 you will see the lifecycle stages 3 through 7 from the previous list. The focus is on what happens during the initial analysis stage and its impact on how the service turns out after the development and deployment stages, when service governance comes into the forefront.

Essentially, as indicated by the big vertical arrows in Figure 1, less up-front analysis (or analysis with a narrower, project-centric scope) results in more long-term governance burden. More up-front analysis (or analysis with a broader, inventory-centric scope) results in less long-term governance burden.

Following a top-down approach involves the stages documented in the Inventory Analysis Cycle, shown in Figure 2.

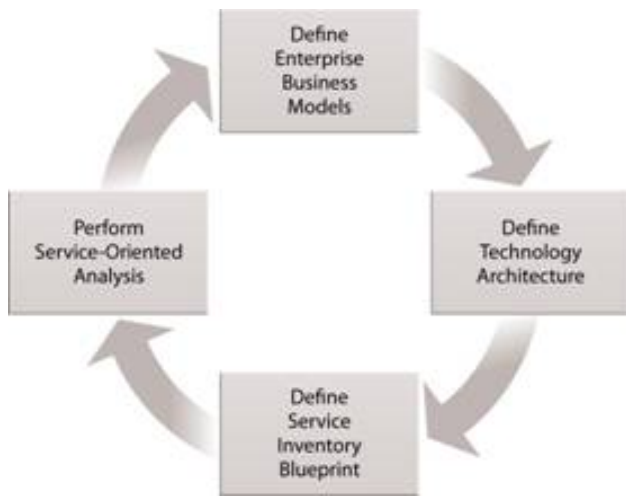


Figure 2: The four primary stages that comprise the Inventory Analysis Cycle. The ultimate results of repeated iterations through this cycle is the delivery of a well-defined service inventory blueprint.

We won't describe the steps in Figure 2 in detail, but it is worthwhile to just explain that the focus of this cycle is on the conceptual definition of the service inventory blueprint through repeated iterations of the Service-Oriented

Analysis process as well as the other indicated stages. At some point, it is considered the right time to actually begin building services, in which case the Service-Oriented Design process starts, as shown in Figure 3.

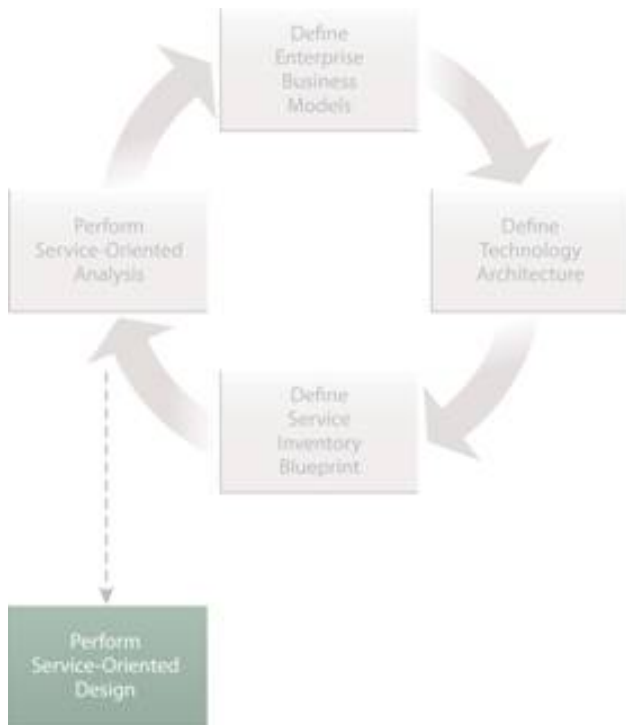


Figure 3: When sufficient iterations of the Service Inventory Analysis process have been completed, a decision is made to begin designing actual service endpoints that use the analysis results as a starting point. Subsequent to Service-Oriented Design, other phases are completed, as per the previously displayed list.

Note: This article is only focused on the analysis aspects of MSOAM. For additional information, see the resources [REF 2, 3] listed in at the end of this page.

Before we even get to the Service-Oriented Design phase, we need to ensure that the Service-Oriented Analysis processes and sub-processes are carried out adequately. In particular, we need to back up right to the beginning to focus on business modeling.

MSOAM is a practical approach to SOA, but it expects that business modeling tasks be completed prior to proceeding with service-specific phases. Therefore, it provides no real guidance in this area. This gap can be filled by RUP Business Modeling. Even though this RUP discipline was originally created in support of object-orientation, based on research and practice, many of its techniques have been found to be equally applicable to SOA projects, especially in support of creating comprehensive input for top-down analysis processes.

There are two entry points for the marriage of RUP and MSOAM, each of which represents a different level of

business modeling:

- preparatory and on-going business modeling for the Define Enterprise Business Models stage
- iterative and system model-centric business modeling for the Service-Oriented Analysis stage

The following sections explore each of these stages.

RUP Business Modeling Steps for Inventory Analysis

A service inventory represents a collection of related services that are independently standardized and governed. The scope of a service inventory can vary, but it must cross multiple silos to represent a meaningful domain (as per the Domain Inventory and Enterprise Inventory design patterns [REF-4]). The scope of a planned service inventory determines the scope of the Service Inventory Analysis. A key step in this cycle is the Define Enterprise Business Models stage (which is represented by the top block in Figure 2). This step is described as follows:

"Many of the services that will eventually be modeled and designed will be business services responsible for accurately encapsulating and expressing business logic. Therefore, a key input for this process is a comprehensive, up-to-date set of business models and specifications (such as business process definitions, business entity models, logical data models, etc.). The amount of business documentation required is determined by the scope of the planned service inventory." [REF-3]

RUP Business Modeling techniques can be applied to establish and extend these enterprise business models through each iteration of the Inventory Analysis Cycle. To do so, the emphasis is on defining and refining use-cases that capture the current status of whatever business domain the service inventory represents.

This means that during RUP Business Modeling steps, all of the existing processes are recognized and analyzed and every single element of the business (including rules, stakeholders, sub-processes, use-cases, actors, and goals) is identified and documented. These tasks are carried out in two primary steps, as shown in Figure 4.

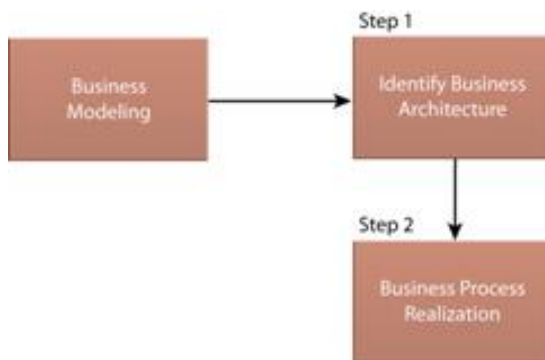


Figure 4: The two primary steps of the preparatory Business Modeling stage.

As you can tell by the description text, Step 1 is about identifying required business models and related business intelligence, and Step 2 is about using this information to define the processes in a more concrete manner.

Each step represents a further sub-process. The additional steps that are part of Identify Business Architecture are displayed in Figure 5.

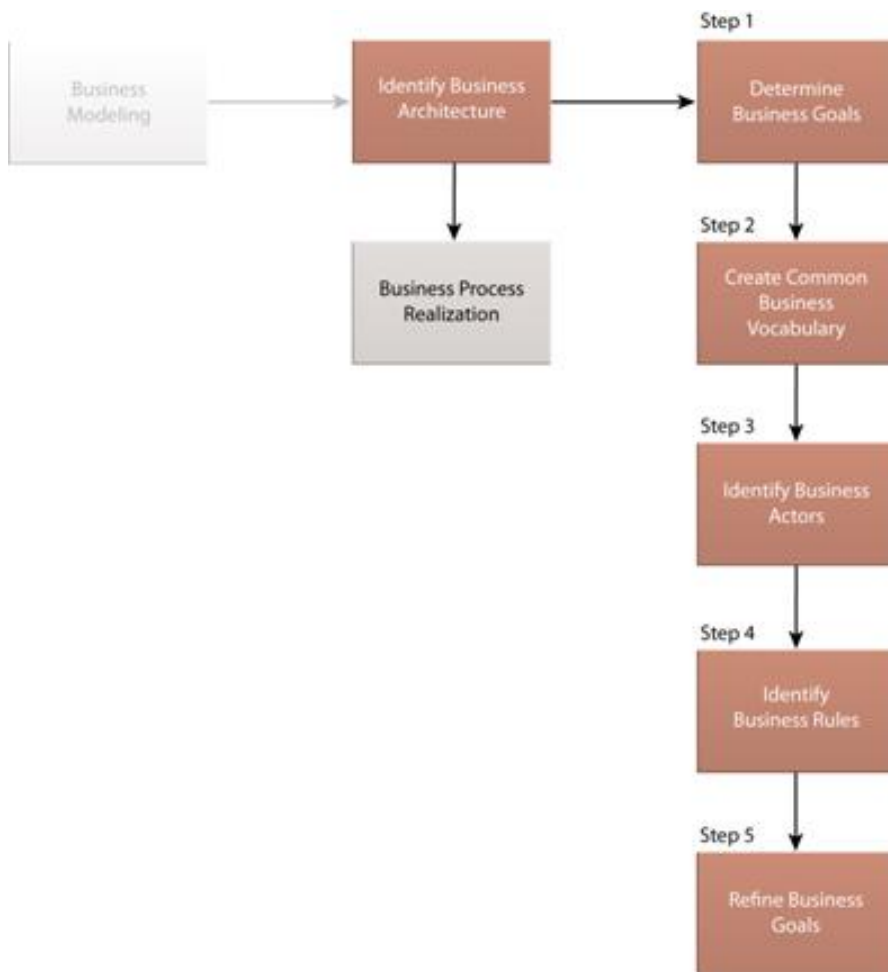


Figure 5: The five steps that comprise the Identify Business Architecture stage.

The Identify Business Architecture sub-process is focused on identifying the parts of the overall enterprise business architecture that pertain to the current analysis scope, as determined by the planned service inventory. Within this business domain, business goals are defined (1), and common business terms are standardized (2) because, as you might recall, standards are applied within the scope of an inventory. The actors within this domain are identified (3), as are the key business rules that will eventually determine the logic and constraints of business processes. Finally, business goals are revisited before finalizing this documentation to ensure that, given all of the information that's been collected and consolidated, these goals are still valid and appropriate.

Note: Shared terms and statements added to a common, centralized business glossary (that is perhaps also made available in a central repository) ensures consistency across all iterations of the RUP Business Modeling process and the Service-Oriented Analysis process. This is another means by which RUP-based documentation can infiltrate MSOAM processes to become a natural part of the methodology used to model service inventories.

The subsequent Business Process Realization step has its own sub-process, as shown in Figure 6.

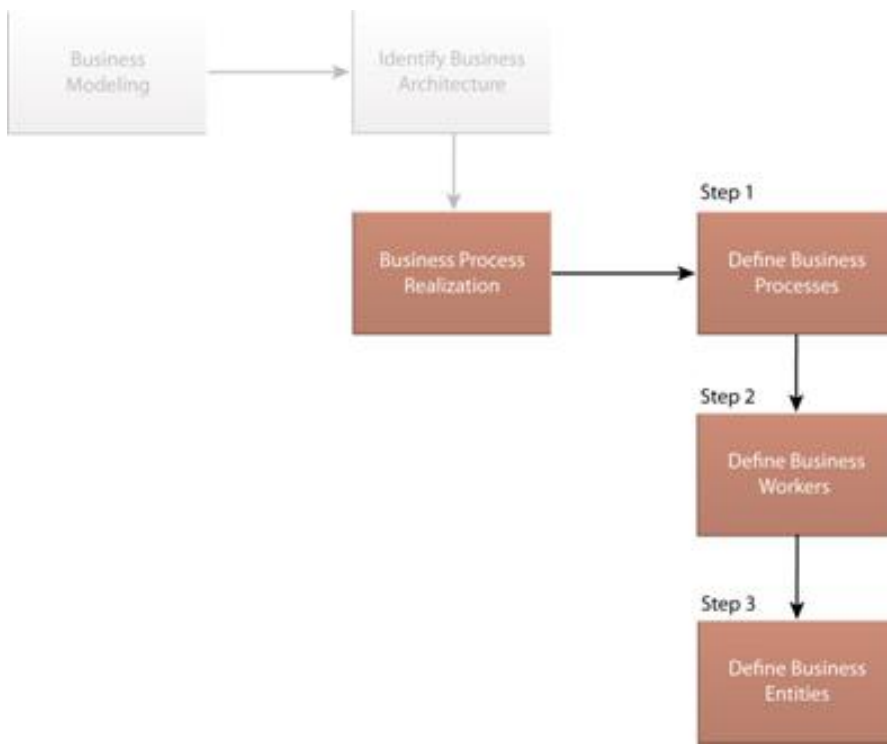


Figure 6: The steps required to carry out the Business Process Realization stage.

In this phase, we draw from all of the information collected during the Identify Business Architecture step to pinpoint three specific types of specifications: business process definitions (1), worker and role definitions (2), and business entity definitions (3). All three of these elements will make their way into the eventual Service-Oriented Analysis process and will provide extremely accurate and comprehensive input that will help form the basis for service candidates and the service inventory blueprint.

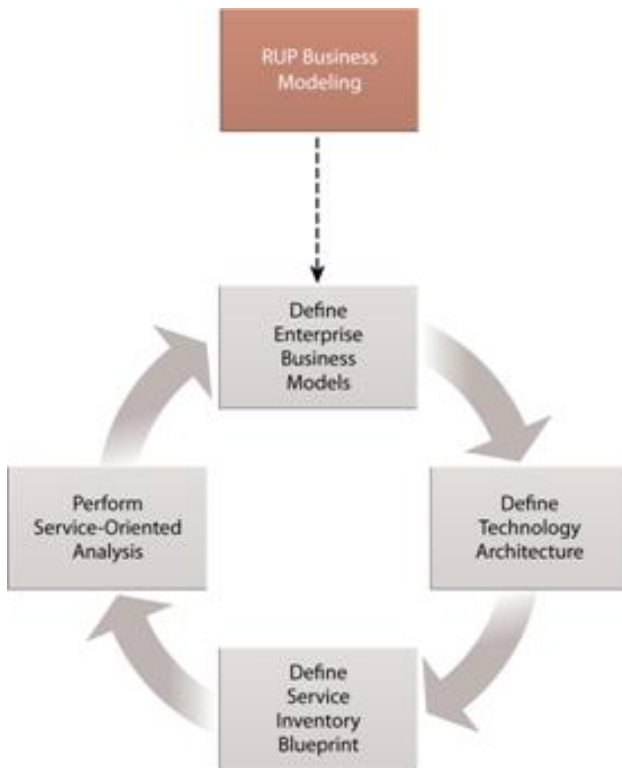


Figure 7: All of the previously explained RUP steps provide preparatory and on-going business modeling in support of the MSOAM Define Enterprise Business Models stage.

RUP Business Modeling Steps for Service-Oriented Analysis

The preparatory RUP Business Modeling steps that were just described have been basically collecting business

information that becomes available as input for each iteration of the Service-Oriented Analysis stage. This part of the overall service inventory lifecycle is focused on the application of service-orientation to conceptual services called service candidates.

Here's the description for the Service-Oriented Analysis phase:

"To effectively deliver standardized services in support of building a service inventory, it is recommended that organizations adopt a methodology specific to SOA and consisting of structured analysis and design processes. Within SOA projects, these processes are centered around the accurate expression of business logic through technology, which requires that business analysts play a more active role in defining the conceptual design of solution logic. This guarantees a higher degree of alignment between the documented business models and their implementation as services. Agnostic business services especially benefit from hands-on involvement of business subject matter experts, as the improved accuracy of their business representation increases their overall longevity once deployed.

Service-oriented analysis establishes a formal analysis process completed jointly by business analysts and technology architects. Service modeling, a sub-process of service-oriented analysis, produces conceptual service definitions called service candidates. Iterations through the service-oriented analysis and service modeling processes result in the gradual creation of a service inventory blueprint." [REF-3]

Figure 8 shows us what the traditional Service-Oriented Analysis process steps look like.

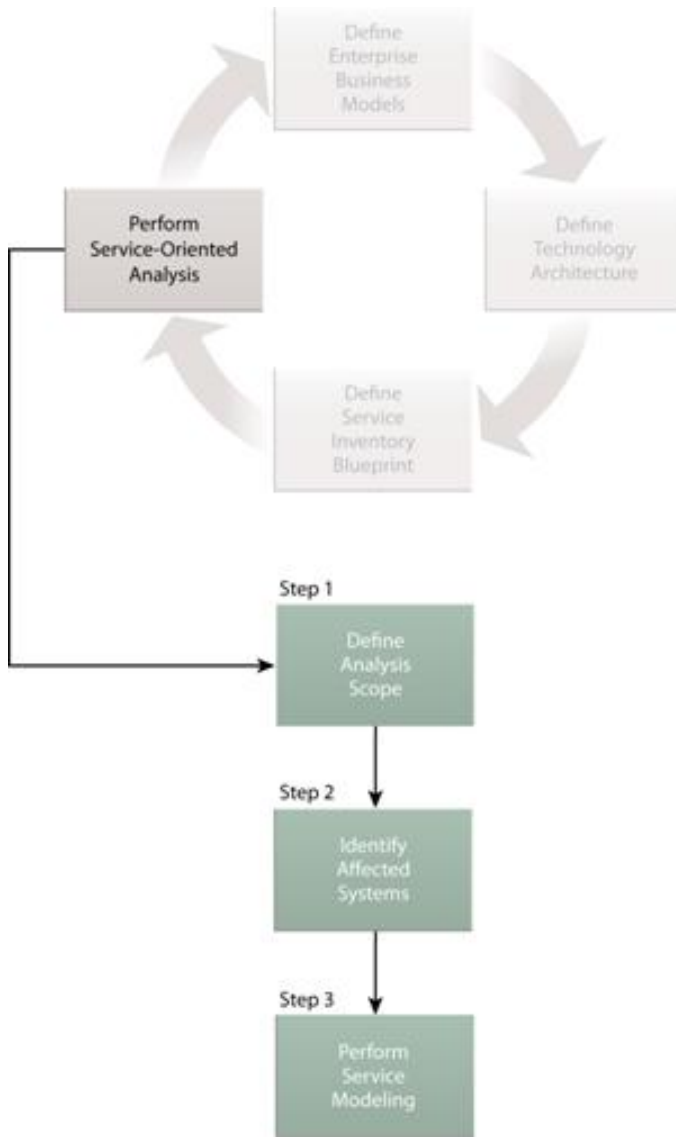


Figure 8: The vanilla Service-Oriented Analysis process. These steps are almost always customized when implemented in specific enterprises.

Now study Figure 9 and you can see how, when RUP Business Modeling is incorporated, additional steps are added to the default Service-Oriented Analysis process.

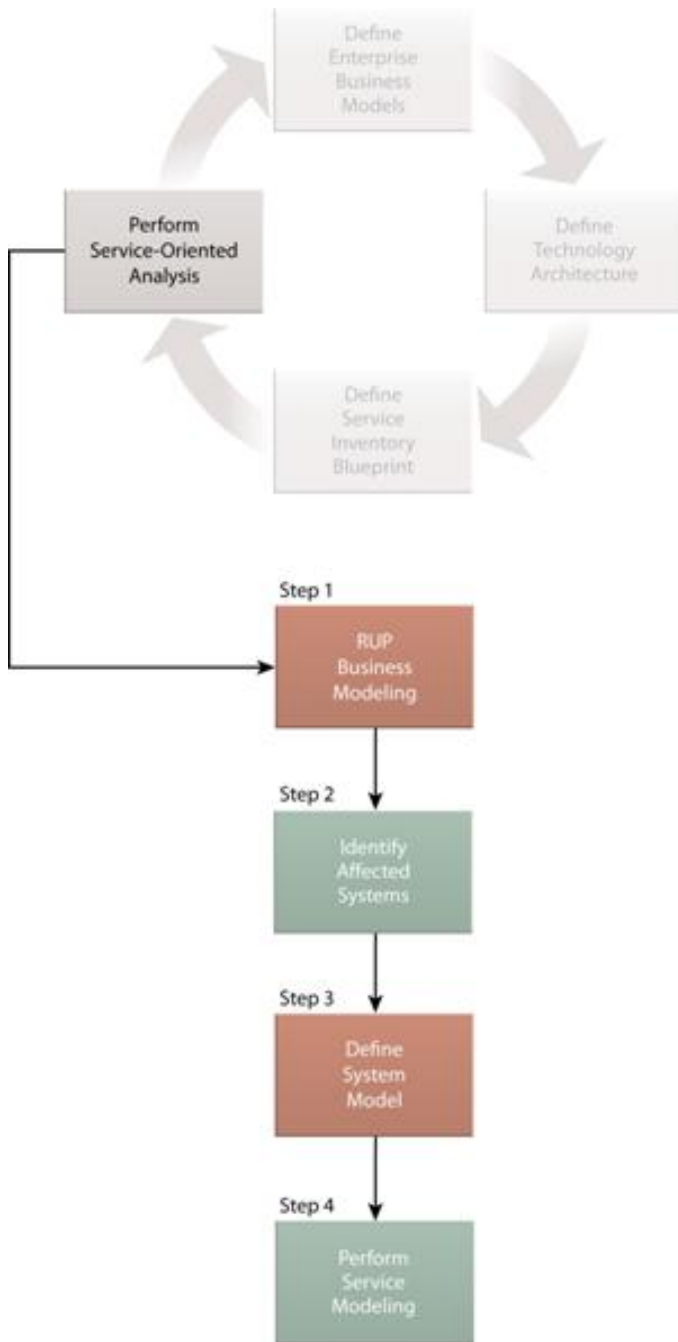


Figure 9: The extended, RUP-enabled Service-Oriented Analysis process. The red process steps are new.

1. Defining Analysis Scope/RUP Business Modeling

Traditionally, MSOAM required that this step be completed so that the scope of the analysis moving forward was well-defined. However, as a result of the work performed by the previously explained preparatory RUP Business Modeling steps, the analysis scope will already have been determined. Therefore, in this augmented MSOAM Service-Oriented Analysis process, we can replace this step with the RUP Business Modeling process step. Alternatively, you can remove this step altogether.

2. Identify Affected Systems

This traditional Service-Oriented Analysis step requires that before proceeding to the service modeling stage where service candidates are defined, architects take a good look at the systems that relate to the analysis scope (as previously established). If legacy, COTS, or ERP applications reside within this scope, then any constraints or features they provide that may tie into the modeling and design of services need to be documented as input for the service modeling sub-process.

3. Define System Model

Normally, when following the regular MSOAM process, you would now be proceeding to the service modeling sub-process. However, as a result of combining RUP with MSOAM, this new step is inserted here in order to make a transition from a business process to an "automation process".

In the preceding RUP Business Modeling phases the focus was on the business use-cases and related business elements in relatively abstract terms. Now, the focus is on how the business process needs to be automated. This is documented as part of a system model specification.

The Define System Model step can encompass a variety of additional steps, such as those suggested in Figure 10.

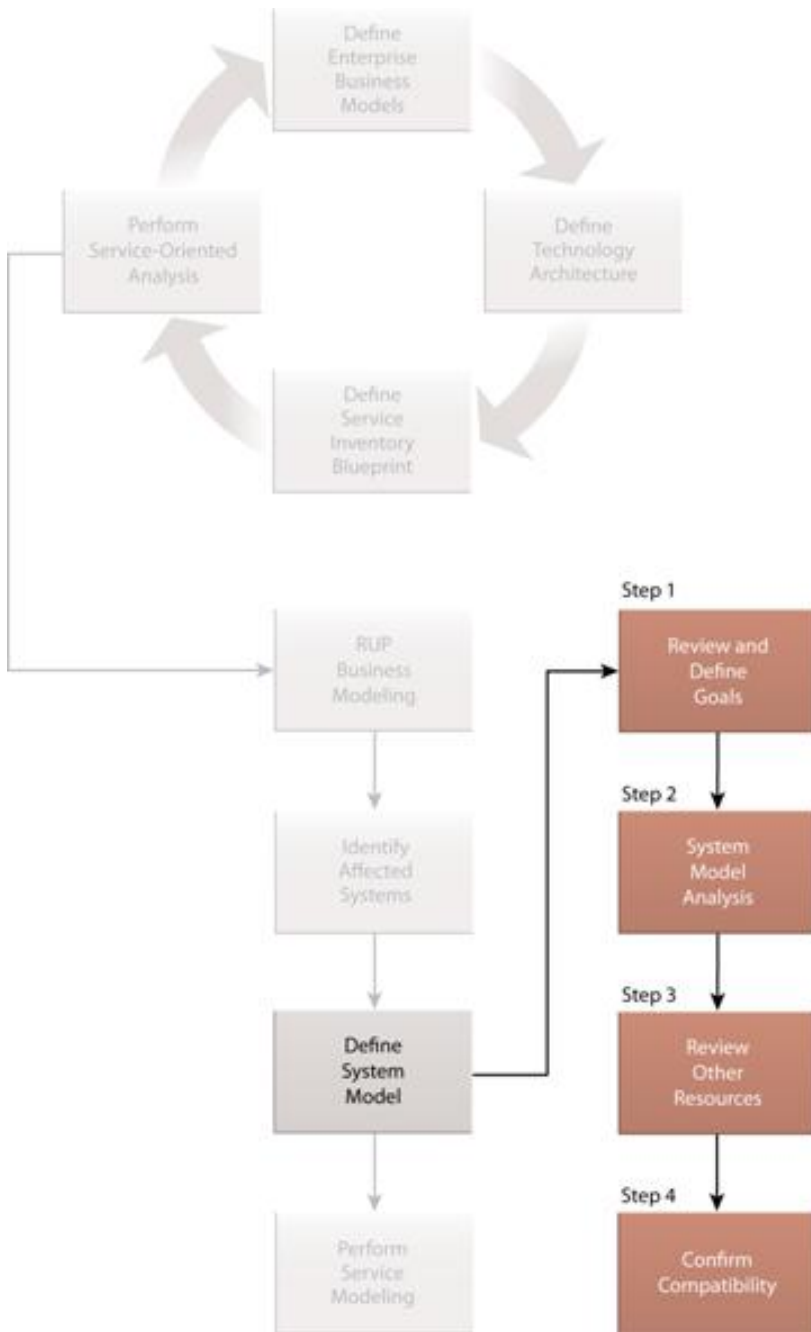


Figure 10: Four recommended steps to defining a system model. Step 2 introduces a separate sub-process as explained shortly.

1. Review and Refine Goals

Because there is a greater understanding of the technology environment in which the process will need to be automated, the first step is to revisit the original goals. Taking technology constraints and environmental factors into account, it is often necessary to adjust or even scale back the original goals and expectations associated with the

business process. However, it may even turn out that new goals can be achieved because new features or infrastructure capabilities were discovered (although this is a less common occurrence).

2. System Model Analysis

This important step essentially establishes an abstract system model from the original abstract business model. The details and considerations defined by this step provide information that will be useful to both Service Modeling and Service-Oriented Design stages, especially in relation to the application of Service Autonomy and Service Loose Coupling principles [REF-2].

The System Model Analysis step is carried out by completing a mini-analysis comprised of the sub-process depicted in Figure 11.

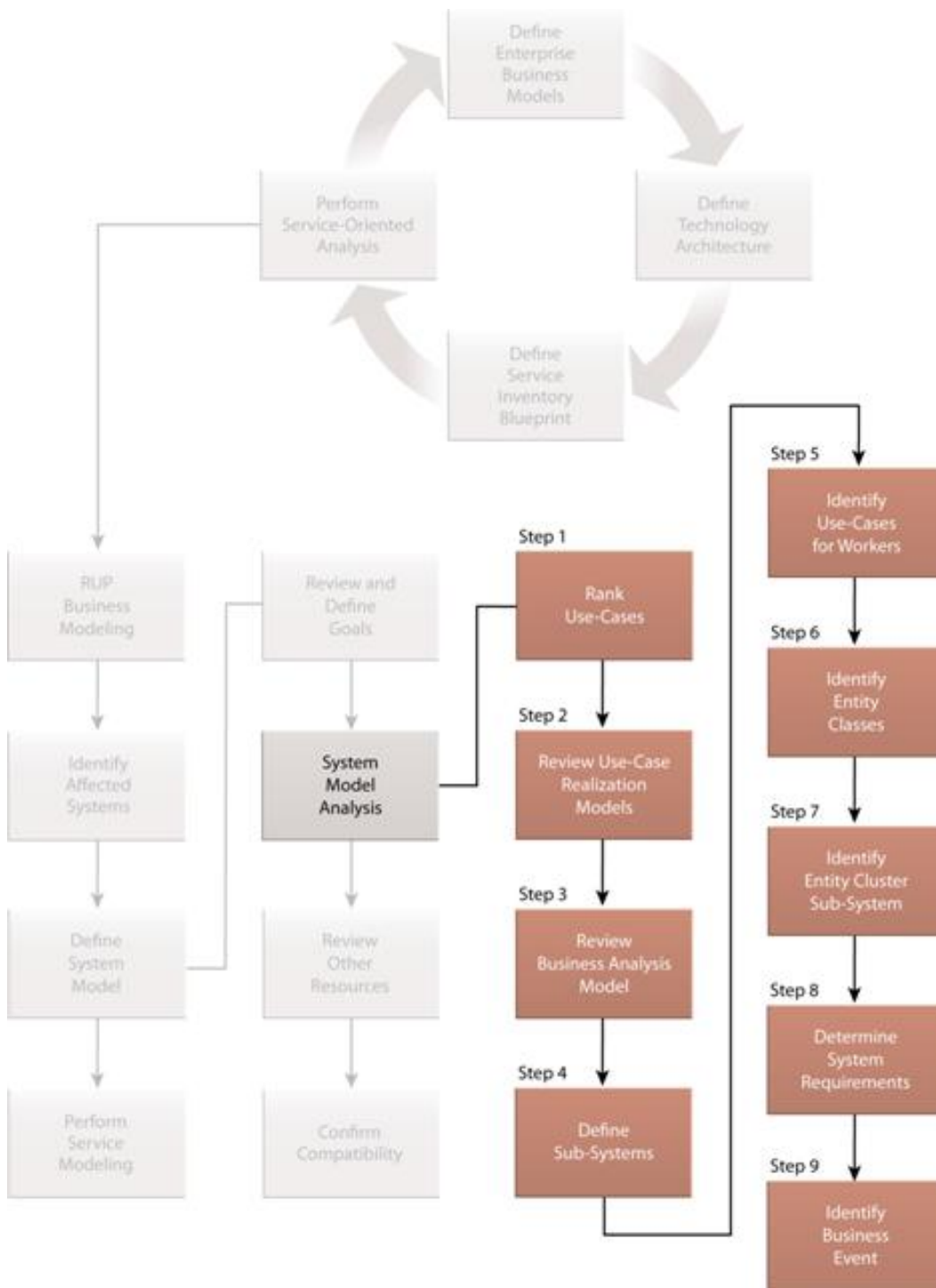


Figure 11: Recommended steps that can be part of a system model analysis.

If the scope of this iteration through the analysis phases encompasses multiple use-cases, then they need to be

prioritized by order of importance (1). This ranking may become relevant when certain trade-offs need to be made at a later stage (where use-cases less important to the business compromise their automation requirements before the more important ones do). Business use-case realization documents are then reviewed (2) to obtain the collaboration diagrams between actors, entities, and events. At this level of review, every automation requirement of the planned system should be detected and documented, resulting in a skeleton system model. The next step (3) asks that we take the time to align it to whatever extent possible with the original business process model.

When we need to work with multiple inter-related use-cases, it may be necessary to structure the system model into a primary system with one or more sub-systems (4). More often than not, a sub-system will correspond to a secondary use-case (as per the ranking performed earlier). Now that use-cases have been individually defined, ranked, and separated into sub-systems, we can determine which sub-system(s) relate to each potential worker (5), as per the previously defined actors. It is at this stage that the original business actors can be considered "system actors".

Every business entity that was identified in the original business process model should be supported by a corresponding system entity class in the system model (6). This may be as simple as mapping business entities from a logical data model to the physical database tables that implement them as information sets. Often, business entities need to be grouped (clustered) because they are closely related to each other. For example, an invoice entity will be tied to related invoice history, invoice line item, and invoice total business entities (if that's the level of detail the logical data model has been created with). In this case, sub-systems can be established (7) to support or be associated with these entity clusters. This type of mapping can form the basis for entity service candidates that are later defined as part of the service modeling process.

To whatever extent possible, the performance and reliability goals of each system and sub-system should be defined and documented (8). This also ties into how workers (system actors) relate to the sub-systems and the nature of the overall business tasks that need to be automated. In fact, this step should really be repeated for each system actor.

Finally, depending on the type of business logic and requirements that need to be fulfilled by the automated process, specific events can be identified and also documented (9). If it turns out that a particular business process is more event-driven, that might make a case for asynchronous communication (and perhaps the use of publish-and-subscribe and queuing mechanisms down the road), when service candidates enter the design and development stages. For now, this type of detail can tie into how service capability candidates (the conceptualized service operations or API) will be defined within the service profile document.

3. Review Other Resources

This step is here as a precaution. After performing such a thorough analysis of the system model, all kinds of new details and requirement may have emerged. It is therefore recommended that you consider looking outside of the current scope (as originally established in the Define Analysis Scope step) in order to determine if other enterprise resources affect the planned services or if other parts of the infrastructure will support or hinder some of the recently discovered requirements (such as when new security requirements are discovered, for example).

4. Confirm Compatibility

As a final step, before all of this information is carried over into the service modeling phase, it is suggested that you ensure that the system model and all of its parts have been documented using terminology, notation, and other conventions that are in alignment with any other business and system modeling that was performed prior to (or is being performed in parallel with) this iteration of the Service-Oriented Analysis process.

With some knowledge of the affected systems, this step allows for a process automation definition that results in systematic requirements at a high-level. The result is a revision to the original business process documentation with numerous references to required technology resources, manual and automated steps (and how to transition between them), and also more concrete details as to required response times and even exception conditions.

This last step basically improves the overall quality of business process documentation before this information becomes input into the Service Modeling process [REF-3].

Conclusion

Service-orientation and object-orientation are not the same. Each is distinct with its own goals and approaches.

However, by understanding that one has roots in the other, we can leverage established practices and techniques and incorporate them into how we plan to attain SOA today. The purpose of this article was only to explore one of many possible ways of combining proven parts of an object-oriented methodology with a mainstream SOA methodology. In particular, the intention was to maintain the "mainstream" philosophy of MSOAM and supplement it with equally generic RUP practices. In the end, the proposed combined methodology is still expected to act as a starting point for any given SOA project, thereby allowing enterprise teams to further customize and extend these processes however they want.

References

[REF-1] The Rational Unified Process-An Introduction, Philippe Kruchten, Addison-Wesley, www.pearsonhighered.com/educator/academic/product/0,3110,0321197704,00.html

[REF-2] "SOA: Principles of Service Design", Thomas Erl, www.soabooks.com

[REF-3] SOAMethodology.com, www.soamethodology.com

[REF-4] "SOA Design Patterns", Thomas Erl, Prentice Hall/PearsonPTR, pre-release manuscript available at www.soapatterns.org

THE PRENTICE HALL SERVICE-ORIENTED COMPUTING SERIES FROM THOMAS ERL



[Home](#) [Past Issues](#) [Contributors](#) [What is SOA?](#) [SOA Glossary](#) [SOA School](#) [SOA Books](#) [About](#) [Legal](#)

Copyright © 2006-2008
SOA Systems Inc.