Integrated Asset Analysis Framework for Model-Driven Development of SOA Based Solutions

Karthikeyan Ponnalagu, Nanjangud C. Narendra, and G.R. Gangadharan

IBM Research India, Bangalore, India
{pkarthik,narendra,gangadharan}@in.ibm.com

Abstract. In SOA based application development, a plethora of architectural constructs such as processes, services and components need to be built. This requires modeling of the application at different levels of abstraction such as business architecture, application architecture and runtime architecture. Model driven development (MDD) is hence considered the primary development approach for building SOA applications. Existing MDD methodologies and tools only support searching and discovery of assets, and do not support their analysis in order to determine their suitability for reuse. This often results in selecting potentially incompatible assets among the various layers of the solution, resulting in redundant asset customizations. In order to address this issue, we present a novel framework and methodology that enables the integrated analysis of existing assets associated across multiple abstractions of the solution from different asset repositories. This approach helps in creating a consistent asset reusability view across all the phases of SOA development with multiple reusable asset options to compare and select. We present an experimental evaluation of our methodology on real-life SOA assets distributed across multiple repositories and illustrate how our integrated mechanism can help consistently maximize reuse of assets in SOA development.

Keywords: Service-Oriented Architecture, Asset Analysis, Reuse.

1 Introduction

In SOA based solution development, the Model Driven Development (MDD) approach enables business analysts and application architects to better integrate business requirements with IT specifications [1,2,3,4]. As illustrated in Figure 1, in a typical SOA based solution development, models at higher abstraction level representing the business domain are translated into service models, which in turn are refined and realized as traditional design models (class diagrams, sequence diagrams, etc.) finally followed by implementation. The process of discovering and leveraging existing potential reusable assets from repositories will minimize the cost and effort of building SOA based solutions, rather than developing from scratch. However, improper selection of reusable assets across the various layers of the solution results in incompatibilities among the assets, thereby incurring increased customization effort. As the volume of available assets in most

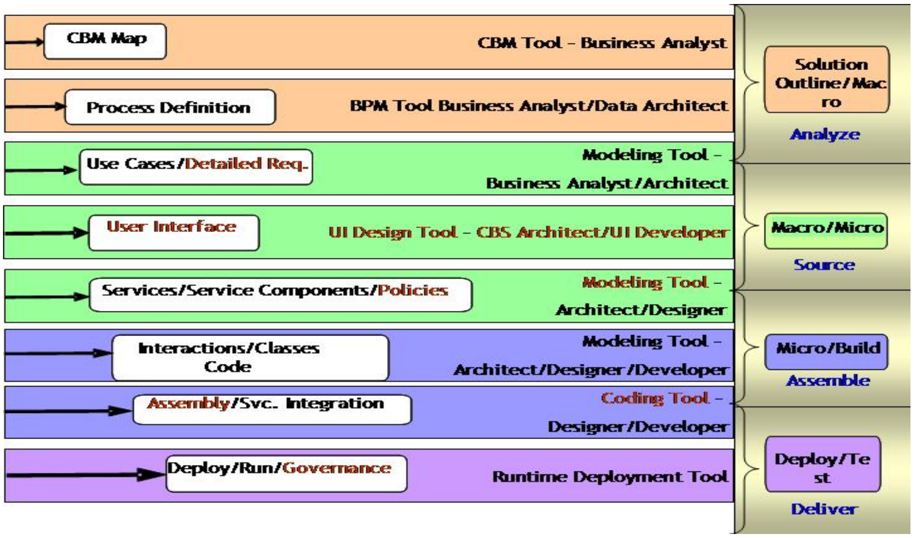
CBM Map
CBM Tool - Business Analyst
Process Definition
BPM Tool Business Analyst/Data Architect
Use Cases/Detailed Req.
Modeling Tool - Business Analyst/Architect
User Interface
UI Design Tool - CBS Architect/UI Developer
Services/Service Components/Policies
Modeling Tool - Architect/Designer
Interactions/Classes Code
Modeling Tool - Architect/Designer/Developer
Assembly/Svc. Integration
Coding Tool - Designer/Developer
Deploy/Run/Governance
Runtime Deployment Tool
Solution Outline/Macro
Analyze
Macro/Micro
Source
Micro/Build
Assemble
Deploy/Test
Deliver

Fig. 1. SOA Lifecycle Tooling View

repositories are generally large (in terms of hundreds), discovering the appropriate asset and customizing it for compatibility with the existing artifacts in the SOA based solution is quite challenging. This exercise can even take up to a week per asset for an experienced architect.

Therefore, an integrated view of assets from multiple abstractions sourced from different repositories at different stages of development is required. Such a view enables architects to identify the most appropriate set of assets to reuse with minimal customization effort. This kind of view can also be reused for similar solutions in the same domain reducing the time invested in searching for the assets in the first place. In this perspective, the key contribution of our paper is an integrated asset discovery and analysis framework to aid model-driven development of SOA based solutions. The salient features of our paper are as follows:

– A novel approach that progressively generates a reusable integrated asset analysis reference model from multiple repositories across the different abstraction layers of the solution
– An extensible mechanism that supports diverse abstractions of business and service models towards integrating newer asset specifications and repositories
– An integrated asset assessment that enables both top-down and bottom-up trade-off considerations in selecting or rejecting candidate assets
– A mechanism that enables dynamic selection of search context related filters discovered from the initial searches and employed in subsequent searches

The remainder of the paper is organized as follows. We present a Request for Quote (RFQ) process model as our running scenario in Section 2. Section 3 discusses the different architectural layers of the SOA model and corresponding support for asset analysis across different asset specifications and repositories. In

Section 4, we present our framework and methodology for analyzing existing architectural assets in SOA based solution design. Section 5 illustrates asset search and identification algorithm that analyzes existing assets across multiple repositories and helps in identification of reusable assets via a context-based filtering mechanism. In Section 6 we evaluate our framework and its constituent algorithm on our running example. Related work is discussed in Section 7, followed by concluding remarks in Section 8.

2 Running Example

Request for Quote (RFQ) business process provides a facility for automating the premium quote generation for customers who wish to obtain insurance policies in various sectors including automobiles, home, and life.

Nowadays independent search, discovery, and reuse assessment is conducted across each of the phases with differing asset requirements. Similarly, we perform independent asset searches with a random sequence for specific asset types in the RFQ application. A total of 33 independent searches are conducted through a single user interface connected with multiple repositories (see Section 6 for more details). Each search is focused on searching a specific asset type ignoring the search results from the previously executed search. Thus, a total of 182 assets are discovered. Figure 2 depicts the distribution of these assets across the different modeling abstractions.

We observe that the discovered assets have reuse compatibility issues with each other because they do not share the same characteristics such as status of their approval, supported languages, and compliance to standards or regulations. Some of these assets have high mismatch probabilities if they are considered for reuse with assets from other asset types. For example, the third search in the independent search sequence is based on case study as asset type. From the total of 8 filters associated with the assets resulted in this search, 6 filters have unique values leading to a mismatch probability of 75%. We illustrate the associated mismatch probabilities for assets belonging to all the asset types in Figure 2. Thus we have a larger scale of asset types to consider for reuse in each of the solution development phases. Such a scenario leads to redundant customizations and inconsistencies in solution development due to the lack of a methodology that integrates the asset search and discovery steps and helps in generating a consistent system-wide asset reuse view. Figure 3 illustrates the RFQ business process retrieved from one of the searches as a process map asset type belonging to business architecture layer.

3 Exploring Asset Analysis in SOA Based Solutions

Understanding the architecture and the associated models of an SOA based solution is paramount in setting the context for search, discovery and analysis of existing assets. From our experience in analyzing SOA based solutions, we have noticed that there are multiple touch points to initiate asset search. We describe

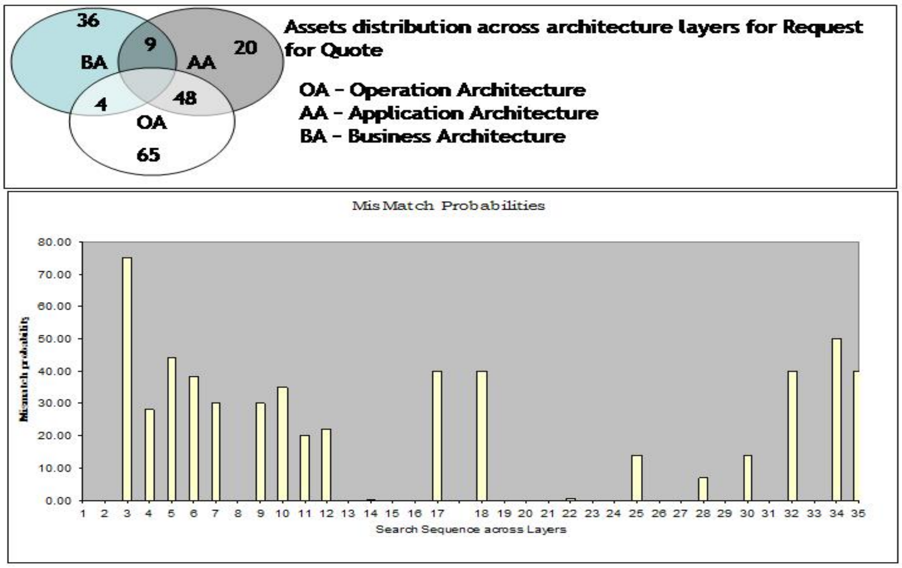

Fig. 2. Asset Distribution

them under the following three layers of abstraction: Business architecture, Application architecture, and Operation architecture.

In the context of SOA, business architecture models involve business process models, component business models, requirement models, and domain data models. The most common and widely used Business Architecture specification is the business process model as discussed in [5]. The Component Business Model (CBM) [6], a methodology for business architecture modeling defines a hierarchy of business components, helps in identifying the domains the assets can tentatively belong to, while the process model focuses on validating the functional specifications of the discovered assets. Thus the impact of discovering and analyzing existing assets in the business architecture space is much higher, because of the context of information available at the business specification level and the flexibility one can assume with the higher abstraction the associated models bring in. However there should be no conflicts in integrating the business architecture assets with subsequent selection of application and operational assets.

Application architecture methodologies provide a mature environment for architecting and designing SOA solutions. The core functionality of such methodologies includes supporting multiple business specification models and enabling architects to develop the correct SOA based design of the solution. They also provide capabilities to represent business architecture specifications such as process models in the form of UML Models. From a MDD perspective, they enable generation of multiple implementation models depending on the stipulated deployment environment.

Operational architecture models basically involve the enhancement of generated implementation artifacts for hosting in a runtime environment, focusing on

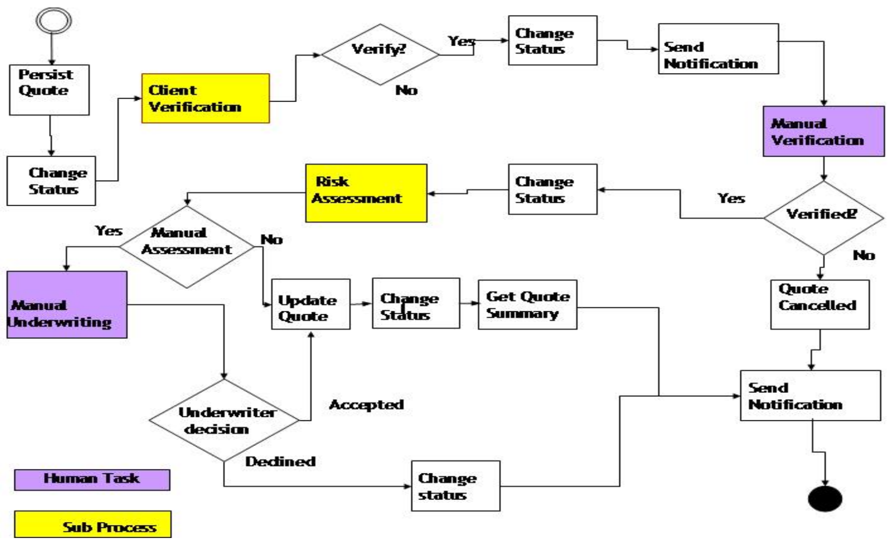

Fig. 3. Process Model Representation for the Request for Quote Solution

actual custom development and generation of exclusive set of artifacts for the runtime environment. This is supported by model driven engineering capabilities that generate starter code at the completion of the solution design exercise. Hence, operational architecture for SOA based solutions provide greater scope of code reusability especially with the MDD approach.

An integrated view of associated models related with RFQ application for these three layers is illustrated in Figure 4. Due to the lack of integrated and implicit automated asset analysis and discovery mechanisms in typical MDD approaches, there exists such a closely linked chain of capabilities that quickly model and transform a business specification model into an application architecture specification and associated set of code artifacts.

4 Asset Analysis Framework: Design and Methodology

Our framework complements the service specification and design components prevalent in MDD tools such as Rational Software Architect (RSA)[1] via discovery, selection, and adaptation of existing assets that can realize the identified service functionalities without fresh implementation at multiple SOA layers. In model driven SOA solution context, our framework incorporates a way to access assets stored in different repositories and to view existing assets from one integrated modeling tool.

Our framework basically contains three main components (see Figure 5):

1. The Contextual Asset discovery component, that enables the construction of multiple optimized asset search queries based on the business architecture

[1] http://www-01.ibm.com/software/awdtools/swarchitect/websphere/

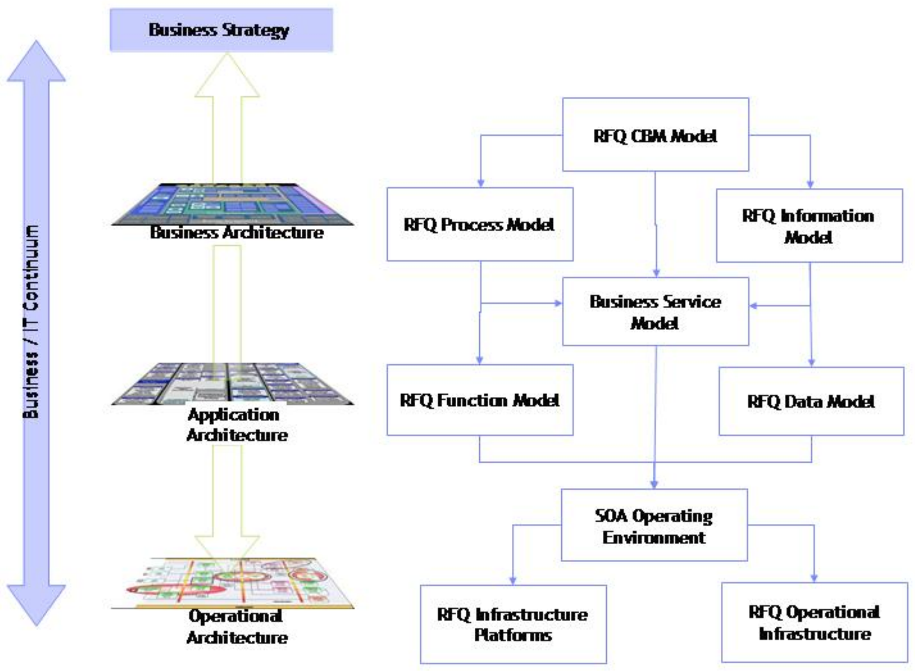

Fig. 4. An integrated view of RFQ Application models

model and to initiate asset search in iterative fashion across the multiple repositories and across multiple abstractions such as service models, designs, service component definitions and code implementations

2. The Core Asset Modeling component that retrieves the asset artifacts and represents them in the generic Asset Model.
3. The Asset Analysis Component that facilities automated or manual comparison of the selected asset models against the corresponding business architecture elements such as business process, business entity models

The requirement-centric context used for initiating search and selection of filters is formulated at two levels. The initial level is based on the type of business specification document such as process model, CBM, business use case model, etc. This basically helps in identifying the type of artifacts such as service model, detailed design, actual code implementation, deployment plan, etc., that will be required to take the solution development to its completion. The second level is based on the actual structural and behavioral content (such as the details of tasks, sub-processes, etc.) that dictates the search of functional and technical specification of the asset artifacts.

The metamodel illustrated in Figure 6 contains the UML based representation of the key asset types. Transformation of asset types and the associated structural constructs by the assets into UML metamodel helps in performing a uniform analysis with respect to the given solution requirement.

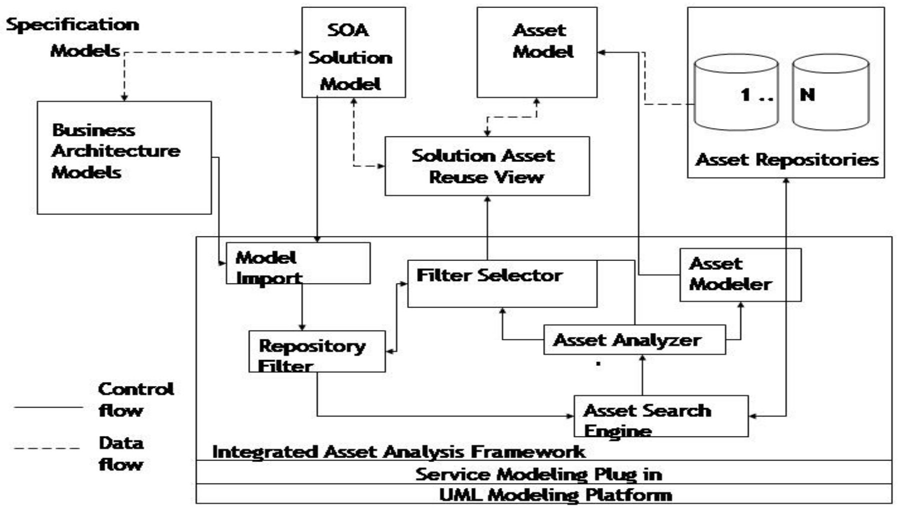

Fig. 5. Asset Analysis Framework Generic Design View

5 Contextual Asset Search and Identification Algorithm

Our novel algorithm for asset search and identification provides greater control for accurate discovery and analysis of existing assets (see Algorithm 1). At the beginning, the framework needs to be registered with the list of repositories prevalently used in the organization. The algorithm basically follows a two-phased search approach. The first phase of our algorithm iterates across all the repositories to find the suitable match for the process or the associated tasks with limited search. The search is initiated with the process related information retrieved from the UML representation and enhanced by adapting the search query based on the predesigned format supported in the repository under search. The results retrieved are consolidated with the removal of duplicate results on subsequent iterations. The assets thus retrieved are associated with their basic information appended to the corresponding elements in the process model. In each repository, a basic validation is performed to ensure whether each of the elements in the process model have at least one asset associated with them. Otherwise the search for unassociated elements are repeated in other repositories.

In most cases, such a basic search results in a huge number of assets with most of them being unrelated or obsolete. Performing a manual validation of the appropriateness of these assets for a solution context is one of the factors that make architects to go for development from scratch. In our earlier work, [7], we defined a business process consisting (a) a set of associated service models, (b) data dependencies between the services based on the execution of preceding services (produced data dependencies), (c) data dependencies between the services based on the input model of preceding services (received data dependencies), and (d) control flow dependencies that provides the choreography of the services. Extending this modeling representation for searching existing assets,

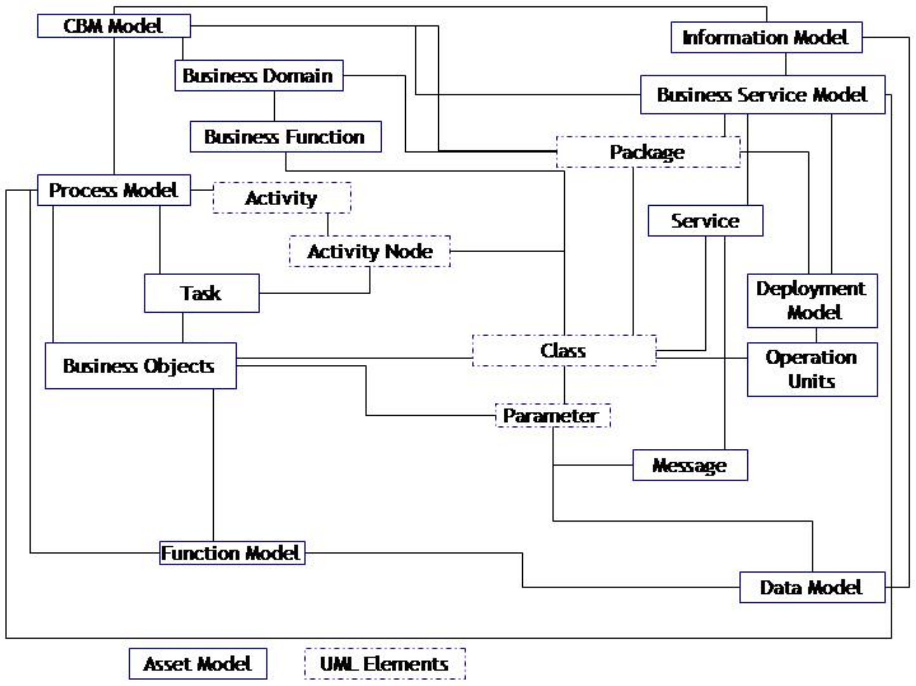

Fig. 6. Asset Analysis Metamodel

the second phase of algorithm proceeds as follows: Task models execute base search aided with the set of input and output data models. The control flow and data flow dependency models categorize search with single repositories and also extract filter matching that reduces the search space with lower number of asset artifacts. Basically filters are multi-dimensional and help us categorize results as per domain, functional, or technology perspective. This enables asset identification that can realize the functionality of a group of tasks with a single coarse-grained asset. This guided search approach will complement work such as [8], where the authors discussed SOA centric transformation of pre-identified legacy assets repositioned as services and processes. Iterating the second phase subsequently for all task groups results in a reduced single asset association for the tasks or the process itself. The actual execution of this algorithm depends on the searching capabilities and metadata support of the specific repositories.

6 Implementation and Evaluation

The prototype of the proposed framework has been developed as a set of plug-ins. The core functionalities such as representation of assets as UML models and analysis of the generated asset model from an imported asset artifact are implemented as a central plug-in. Subsequently for each supported repository, a corresponding repository specific plug-in is developed and integrated into the

Algorithm 1. Integrated Asset Search and Identification Algorithm for a Process Model

```
 1: Get $P = S, E, D, C$
 2: Get $R_1, \ldots, R_v$
 3: Populate unalloted list of elements $E[]$ with $S_1, \ldots, S_n$ for $P$
 4: $E[] = E_1 \ldots E_n + 1$
 5: for all $R_i \in R$ do
 6:    for all $E[i] \in E[]$ do
 7:       Construct base search query for $R_i$ with $E[i]$
 8:       Initiate base search with $E[i]$
 9:       Get ResultSet[] $R_p$
10:       ConsolidateSearchResult ($R_p$,$E[i]$,$E$)
11:    end for
12:    if $E[] = \emptyset$ then
13:       break
14:    end if
15: end for
16: Generate candidate asset model with final result set $F$
17: Get asset list $A[]$ from $F$
18: if $[A] > [P]$ then
19:    FilterSearch ()
20: else
21:    Build solution template with candidate assets
22:    Proceed to manual asset analysis and new process design
23: end if
```

```
procedure $ConsolidateSearchResult(ResultSet[]\ R,\ Element\ Ei,\ Vector\ E)$
 1: Identify asset validity for status and accessibility
 2: Identify asset uniqueness (whether they are not already allotted)
 3: Group related assets (through existing relationships with other allotted assets if any)
 4: Assign allocation link with the corresponding $E$
 5: Remove $E$ from unalloted list $E[]$
 6: Populate the resultant $R$ to $F$
```

```
procedure $FilterSearch()$
 1: Create task groups based on E, D, C from P
 2: for all $R_i$ do
 3:    Identify corresponding asset lists $A_E$, $A_D$, $A_C$
 4:    Identify common filters across each of $A_E$, $A_D$, $A_C$
 5:    Apply common filters across the final Asset List $A[]$
 6:    Consolidate result list and remove undiscovered from $A[]$
 7: end for
```

framework. The functionalities of the repository specific plug-ins involve: connecting to the corresponding repository through an authenticated connection as mandated by the repository; and enabling search, discovery, selection, and retrieval of assets from the repository.

We evaluated the prototype on a SOA design project for developing Request for Quote business process model (illustrated in Section 2). The objective was to identify reusable services in the business domain and to further design and implement them. We have designed the business process model using Websphere Business Modeler [2] and logical Data model in Rational Data architect [3]. Both the models can be imported into Rational Software Architect(RSA) for service identification. The Service Model is built using SOMA-ME [4], which is a service modeling application developed as a RSA plugin. RSA's UML to SOA transformation transforms UML design artifacts to create the code artifacts from the UML Service Model for those elements that are not available as an existing asset from any of the connected repositories. For evaluation, we have integrated the iRAM[4] repository with our framework to export, import, reuse, track, and analyze any type of assets including SOA service model and other related work products. We choose iRAM as the repository for conducting our experiments because it enables search and discovery of multiple asset types from a single web browser interface.

As we discussed in Section 2, the available set of assets based on the search meta-data was quite large, i.e., 182, from the first phase of our algorithm. Many of these assets have high mismatch probabilities if considered for reuse with other independently identified assets. Hence we proceed for further filtering to identify the most appropriate set of assets for building the required RFQ solution. At the completion of each search, we collected the search results and the corresponding filters associated with all of the assets available in the search space. Now we randomly selected any of the assets from the search space and identified the corresponding filters.

As we see in Figure 7, the initial search on just specific asset types is similar to the first phase of our algorithm in terms of total number of search results. Then the subsequent searches based on the retrieved filters are conducted. We observe that the subsequent searches always result in assets in the range of 2-5 (in most of the cases), irrespective of the type of assets we searched in the initial search. Now we compare the independent asset search and integrated asset search corresponding to the asset types "Architecture Documentation" and "Code/Software Components". The independent search would have resulted in 9 assets belonging to the type "Architecture Documentation" and 16 assets belonging to "Code/Software Components". With the integrated search approach, if we started our initial search with "Architecture Documentation" and selected one of the 9 assets, subsequent searches based on the filters associated with the selected asset would not have resulted in any assets. This means selecting any

[2] http://www- 01.ibm.com/software/integration/wbimodeler/
[3] http://www-01.ibm.com/software/data/integration/rda/
[4] http://www-01.ibm.com/software/awdtools/ram/

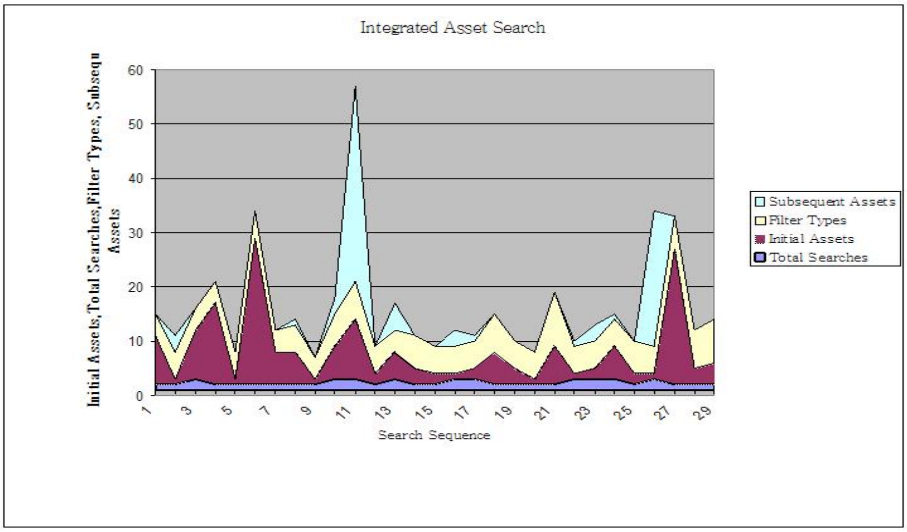

Fig. 7. Integrated Asset Search

other assets belonging to a different asset type purely based on independent searches would have resulted in reuse mismatch. For example, we can compare the redundant exercise involved in selecting and analyzing one of the 16 assets for the type "code/ Software Component" that subsequently results in reuse mismatch with a selected asset of type "Architecture documentation" in the independent search approach. Our integrated approach helps avoiding at least 15 independent searches and subsequent exercise of selecting and analyzing from at least 170 assets. This indicates a scale down from 182 assets to 9 assets to be considered if the initial search is started with the asset type "Architecture documentation". Figure 7 illustrates the integrated search in terms of number of subsequent searches, the total assets resulted from the subsequent searches , with the initial search for each of the asset type.

7 Related Work

Generally, asset analysis in model driven service design and development is seen as a challenging research problem. Rainer et. al. provide an overview of method imperatives that address the requirements for SOA projects [9]. Zimmermann et. al. [10] propose a multi-level SOA decision catalog that includes asset reuse more as part of service realization techniques. [5] presents an overview of the SOA specific repositories currently used in practice across the different phases of the SOA lifecycle. With reference to SOA design and development, a relevant work on reusing mainframe assets in SOA based solutions [11] discusses the consideration of mainframe assets. Generic guidelines for assessing mainframe assets in the context of SOA development are also described in [11].

The said approaches primarily explore the repositories space and the generic technical capabilities of interacting with the repositories. They rely on high level of manual involvement in reuse of existing assets that remain disconnected from the tool centric service modeling activities. However, in our paper, we propose an efficient and interactive framework for discovering and selecting different types of assets, and providing a modeling scope to analyze and study the assets in the context of the required solution. We also discuss how our framework with its plug-in based architecture can be extended or customized with respect to supporting new repositories or to cope with changing standards or specifications. In traditional approaches, the candidate assets both at the process and the individual task levels are identified through manual keyword centric search with limited meta data information available. Then, the assets are subjected to fitgap analysis by the architects. After passing through this crucial test, legacy transformation of such assets into actual services as discussed in [8], with or without customization, is considered. Our asset analysis algorithm performs such an optimization that captures dependencies across the business functions (tasks in business processes) that need to be realized either independently or collectively with one or more available assets, and satisfies the specified constraints both at the process as well as at the individual task (business function) levels. Furthermore, our asset analysis algorithm performs multi-level (Base search and Filtered Search) identification for facilitating optimal asset reuse across mutliple repositories.

8 Concluding Remarks

In this paper, we have presented an architectural asset analysis framework and methodology for model-driven development of SOA solutions, involving a variety of business, service, and design models. We have evaluated the asset analysis methodology using the framework during the design of an SOA solution with multiple asset repositories. We have also shown how our methodology supports asset searching based on the solution context and from multiple repositories, thereby improving asset search effectiveness. In future, we plan to work on automated comparison of assets retrieved in a given search, that leads to establishing relationships of similar candidate assets from the same or different repositories. We also plan to establish a multi-dimensional asset modeling and generation framework that helps document asset requirements and constraints as well as capabilities and analytical details of existing assets. This would enable better governance of asset publishing in repositories.

References

1. Arsanjani, A., Allam, A.: Service-oriented modeling and architecture for realization of an soa. In: IEEE SCC (2006)
2. Schmidt, D.C.: Model-driven engineering. IEEE Computer (February 2006)
3. Johnson, S.K., Brown, A.W.: A Model-Driven Development Approach to Creating Service-Oriented Solutions. In: Dan, A., Lamersdorf, W. (eds.) ICSOC 2006. LNCS, vol. 4294, pp. 624–636. Springer, Heidelberg (2006)

4. Arsanjani, A.: Soma-me: A platform for the model-driven design of soa solutions. IBM Systems Journal 47(3), 397–414 (2008)
5. Aguilar-Savén, R.S.: Business process modelling: Review and framework. International Journal of Production Economics, 129–149 (2004)
6. Cherbakov, L., Galambos, G., Harishankar, R., Kalyana, S., Rackham, G.: Impact of service orientation at the business level. IBM Systems Journal 44(4), 653–668 (2005)
7. Ponnalagu, K., Narendra, N.C.: Discovering and Deriving Service Variants from Business Process Specifications. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 691–707. Springer, Heidelberg (2008)
8. Zhou, N., Zhan, L.J., Chee, Y.M., Chenr, L.: Legacy asset analysis and integration in model-driven soa solution. In: IEEE SCC (1), pp. 554–561 (2010)
9. Gimnich, R.: Using existing software assets in soa design. In: CSMR, pp. 309–310 (2009)
10. Zimmermann, O., Koehler, J., Leymann, F.: The role of architectural decisions in model-driven soa construction. In: OOPSLA (2006)
11. Chordes, M.: Unleash the power of mainframe assets into soa (2007), `ftp://ftp.software.ibm.com/software/rational/web/whitepapers/SWW12599-USEN-00_unleashing_soa_wp_0919.pdf`