Measuring Expected Integration Effort in Web Service Composition

Tomer Sagi$^1$, Matthias Weidlich$^2$, and Avigdor Gal$^1$

$^1$ Technion – Israel Institute of Technology  
{tomers7@tx,avigal@ie}.technion.ac.il  
$^2$ Imperial College London  
m.weidlich@imperial.ac.uk

Abstract. Web services support information systems engineering by flexible composition of ready-made software components to enact business processes. A plethora of automatic and semi-automatic composition methods have been suggested. Due to duplication and overlap between services, there exists a search space of possible compositions. Evaluating alternative composition solutions is done by various properties, each requiring an associated evaluation measure. In this paper, we propose a new property, namely integration effort, which captures the expected effort by a human programmer to integrate composed services into a functioning process. We present a series of effort evaluation measures by adapting the well-known precision and recall functions from related fields of schema and ontology matching. We present an extendable framework, allowing application in different levels of refinement. Measures are empirically validated to be effective proxies of integration effort.

Keywords: Evaluation, Web service Composition, Semantic Web

1 Introduction

Service-oriented engineering of process-aware information systems (PAIS) builds on ready-made, re-usable Web services. To implement a business process (the target process) in a PAIS, Web services (the candidate services) are assembled into a composition. A typical Web service composition scenario is composed of two steps. First, candidate Web services are selected and their parameters are aligned with the tasks of the business process and linked with parameters of other services that are also part of the composition. This first step, therefore, creates a composition skeleton. In a second step, this skeleton of a composition is prepared for enactment by inserting mediating code that either transforms the established alignments and links or completes missing parameters of process tasks or candidate services.

Given a description of a business process, there is typically a large search space of possible compositions due to duplication and overlap between services. Numerous automatic and semi-automatic composition methods have been suggested to search for an optimal solution within such a (very large) search space.
To assist search and to evaluate the relative value of disparate solutions, different quality functions have been devised and grouped into two major categories: functional such as semantic quality [1] and non-functional (also named quality of service or QoS, see review by Cardoso and Sheth [2]). However, these approaches focus on the aforementioned skeleton of a composition, i.e., the quality of the candidate services that have been selected. None of the functions presented to date attempt to estimate the integration effort each solution requires to enact the skeleton of a service composition. This effort (and its associated cost) may be prohibitive, overshadowing the cost of using a Web service with slightly worse semantic quality or one that is marginally more expensive.

In this paper, we suggest measures for the integration effort induced by a service composition. Specifically, the paper makes the following contributions:

1. We provide a framework for the definition of measures for integration effort in Web service composition.
2. We present several instantiations of measures in this framework.
3. The claim that the presented measures are well correlated with the expected effort is empirically validated.

The rest of the paper is structured as follows. We position our work within the Web service composition field in Section 2 and present a formal model for service composition in Section 3. Section 4 presents a framework for the definition of measures for integration effort. Section 5 empirically validates the effectiveness of the measures as proxies to integration effort. Finally, we review related work (Section 6) and conclude the paper (Section 7).

2 Background

A multitude of Web service composition approaches has been presented over the years. The sheer breadth of work on this subject has brought Li et al. [3] to state that ‘examining all of the related work in this area becomes a mission next to impossible’. Un-deterred, they continued to provide a classification matrix for service composition, a sub-set of which is shown in Table 1.

<table>
<thead>
<tr>
<th>Technology</th>
<th>Pattern</th>
<th>Context</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Orchestration</td>
<td>Semiotics</td>
</tr>
<tr>
<td>Workflow-based</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Model-driven</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>AI-Planning</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Limiting the discussion to those dimensions pertinent to the positioning of this work, all of the mentioned technologies for service composition may benefit from our work. Measures for integration effort can be used for guiding AI-planning, as verification criteria for model-driven approaches and as fitness measures for workflow based approaches. In the pattern dimension, this work
assumes an orchestration pattern in which a central entity coordinates the activities of the different Web-services involved. In the semiotics dimension, we assume semantic annotation is available which brings this work into the category commonly referred to as Semantic Web Services (SWS). In the design-time dimension, which refers to assumptions made about the availability of a user to perform the composition, our work is placed in the semi-automatic category. The presented measures are aimed at measuring expected manual labour incurred by a user completing the skeleton of a service composition.

Given an approach to Web service composition that falls into the aforementioned classes (highlighted in Table 1), we illustrate the impact of integration effort on choosing among composition alternatives with an example.

**Example 1.** Update example with concrete refs to ontology Figure 1 presents a Web service composition scenario. The upper part specifies a business process by means of process tasks and control flow. The bottom part presents a composition of five Web services. Comparing the task and candidate service input parameters it is obvious that some integration work will be required as the ‘User Name’ parameter will be embedded in an SQL statement required by the ‘DB Lookup’ service. Also, semantic differences are apparent between the task’s Goal Dashboard input parameter and the generic Dashboard of candidate Web service. These differences in semantics and data-types will require mediation code and incur additional integration costs.

Give example that shows that syntactic and semantic similarity is not a good proxy for integration effort
3 Preliminaries

A Web service composition task is described by a process model and a repository of Web services. To solve the composition task, concrete services to perform the process tasks of the process model are automatically selected from a repository. The focus of our work is not the question of how to solve a composition task. Therefore, our model abstracts from control flow dependencies between process tasks and meta-data such as data type definitions as they can be found in common languages for the definition of process models for service composition, e.g., WS-BPEL [4]. Instead, we focus on the essential building blocks for service composition, i.e., process tasks and candidate services with their respective input and output interfaces. These interfaces are described by sets of parameters, while input parameters may be mandatory or optional.

Definition 1 (Process Task) A process task $T$ of a process model is defined as $T = (I^T_m, I^T_o, O^T)$ with $I^T_m$ as a finite set of mandatory input parameters, $I^T_o$ as a finite set of optional input parameters, and $O^T$ as a finite set of output parameters.

Definition 2 (Candidate Service) A candidate service $W$ in a service repository is defined as $W = (I^W_m, I^W_o, O^W)$ with $I^W_m$ as a finite set of mandatory input parameters, $I^W_o$ as a finite set of optional input parameters, and $O^W$ as a finite set of output parameters.

For a process task $T$ (candidate service $W$), we define $I^T = I^T_m \cup I^T_o$ ($I^W = I^W_m \cup I^W_o$) as the set of all input parameters. We now define a composition task as a set of process tasks and a set of candidate services.

Definition 3 (Composition Task) A composition task $C$ is a tuple $(T, R)$ where $T$ is a finite set of process tasks and $R$ is a finite set of candidate services.

Figure 2 summarizes the relations between the concepts in a class diagram.

Example 2. We now introduce a running example. The process model in Fig. 3(a) requires implementation of a single process task $T_1$ with two input parameters
of types Name and String, the latter is declared as optional, and a single output parameter of type Integer. Hence, we have $I^T_{m_1} = \{it_1\}$, $I^T_{o_1} = \{it_2\}$, and $O^{t_1} = \{ot_1\}$ as the input and output parameter sets. Figure 3(b), in turn, shows a set of three candidate services $\mathcal{R} = \{W_1, W_2, W_3\}$ and the input and output parameter sets for service $W_1$, for instance, are defined as $I^W_{m_1} = \{iw_{1,1}, iw_{1,2}\}$ and $O^W_{t_1} = \{ow_{1,1}\}$.

Next, we define the components of a solution to a composition task. First, the parameters of a process task may be assigned parameters of candidate services. This is typically done based on semantic annotations of the parameters, see Sycara et al. [5] for details on the ontological constructs used in semantic alignment. For our work, however, it suffices to abstract from these annotations and begin by assuming each assignment to be evaluated as either exact or imprecise, where imprecise links include plug-in, subsume and intersection match types.

**Definition 4 (Semantic Assignment)** Let $T$ be a process task and let $W$ be a candidate service and let $pt$ and $pw$ be either input parameters of $T$ and $W$, $pt \in I^T$ and $pw \in I^W$, or output parameters, $pt \in O^T$ and $pw \in O^W$. Then, $sa = (pt, pw, e)$ is a semantic assignment between the parameters with evaluation $e \in \{\text{exact, imprecise}\}$.

Note that a semantic assignment is defined over the parameters. However, it also induces a relation between a process task acting as a source and a candidate service acting as the target of the assignment.

The parameters of different candidate services, in turn, are aligned by semantic links.

**Definition 5 (Semantic Link)** Let $W_1, W_2$ be candidate services. Let $ow_{1} \in O^W_{t_1}$ and $iw_{2} \in I^W_{o_2}$ be an output and an input parameter of the two services, respectively. Then $sl = (ow_{1}, iw_{2}, e)$ is a semantic link between the parameters with evaluation $e \in \{\text{exact, imprecise}\}$.

Again, semantic links are defined between an input parameter and an output parameter, but also induce a relation between the respective candidate services, one acting as input and one acting as output.

Based on these concepts, we define a composition task solution.
Definition 6 (Composition Task Solution) Let $C$ be a composition task. Let $\mathcal{A}$ be the set of semantic assignments over the process tasks and candidate services of $C$ and let $\mathcal{L}$ be a set of semantic links defined over the services of $C$. Then $S_C = (\mathcal{A}, \mathcal{L})$ is a composition task solution.

The class diagram in Fig. 4 illustrates the relations between the presented concepts. As mentioned above, semantic assignments are defined between a parameter of a process task and a parameter of a candidate service and, thus, induce a relation between tasks and services. Similarly, semantic links defined between an input and an output parameter induce a relation between two candidate services.

Example 3. Returning to our running example, let us assume that there are two solutions to the composition task. In the first, $W_2$ is assigned to $T_1$ (Fig. 5(a)). In the second a composition of $W_3$ and $W_1$ is assigned to $T_1$ (Fig. 5(b)). For this solution, Table 2 also lists the respective assignments and links.

<table>
<thead>
<tr>
<th>Table 2. Running example: Details of solution 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>(a) Assignments</strong></td>
</tr>
<tr>
<td>Assignment</td>
</tr>
<tr>
<td>$sa_1$</td>
</tr>
<tr>
<td>$sa_2$</td>
</tr>
<tr>
<td><strong>(b) Links</strong></td>
</tr>
<tr>
<td>Link</td>
</tr>
<tr>
<td>$sl_1$</td>
</tr>
<tr>
<td>$sl_2$</td>
</tr>
</tbody>
</table>
4 Measures for Integration Effort

This section presents our measures for integration effort. To this end, we first identify the atomic operations required by the linking step of service composition (Section 4.1). Then, we turn to the definition of the actual measures (Section 4.2).

4.1 Atomic Linking Operations

The linking step is a series of coding operations designed to transform a collection of candidate services assigned to process tasks into a functioning computerized process. The need to perform these operations stems from two sources. First, there may be (partial) incompatibility of output and input parameters between pairs of candidate services and between services and tasks due to data type or semantic heterogeneity. Second, there may be unassigned input parameters of services or input/output parameters of process tasks that are mandatory.

To facilitate differentiation between effort intensive and effortless operations, we define a set of atomic operations. An atomic transformation operation \(A_T\) is a piece of mediation code written to convert a single input/output parameter from the original format to the required format. An atomic completion operation \(A_C\) entails some search for compatible services within the repository and the writing of mediation code to fit them within the existing solution in order to complete a missing mandatory parameter.

When defining measures, a standard cost (weight), reflecting the relative effort required to complete these tasks, can be assigned. Subdivision of operation sets to sub-sets with common characteristics and assignment of different weights to these sub-sets is also possible. For example, one might wish to differentiate between the transformations of basic data types and complex data types.

4.2 Definition of Measures

In related realms, such as schema matching and information retrieval, it is common practice to define two measures, precision and recall, each measuring a different aspect of the solution. The first aspect being the preciseness of the solution or to what extent the proposed solution is correct. The second refers to the completeness of the solution, or how much of the task has been successfully completed. These two measures can also serve as proxies for integration effort as imprecise solutions require an effort of identifying and cleaning out the incorrect results (similar to our \(A_T\) operation) and incomplete solutions require completion efforts (similar to our \(A_C\) operation).

We illustrate the two aspects of evaluation with the example of pair-wise matching of two data data schemas \(S_1\) and \(S_2\). The atomic unit of a solution to a schema matching task is a matched attribute pair \((a_i, a_j)\), \(a_i \in S_1, a_j \in S_2\). A solution \(M\) contains a set of matched attribute pairs that is evaluated against an exact match \(E\), which is the expected set of matched attribute pairs. Then,
two measures are defined, precision \( f_{Pr} \) and recall \( f_{Re} \):

\[
\begin{align*}
 f_{Pr} &= \frac{|M \cap E|}{|M|} \\
 f_{Re} &= \frac{|M \cap E|}{|E|}
\end{align*}
\]

Defining similar measures for the area of service composition requires definition of a minimal solution element. The smallest unit of consideration in the model presented in Section 3 is an input or output parameter of a candidate service or process task.

Specifically, we examine input parameters that are defined as mandatory. The assignment between an optional input parameter of a process task and a parameter of a candidate service is not a necessary condition for a successful implementation of the process model. The same holds true for a link between an output parameter of one candidate service and an optional input parameter of another service.

In addition, output parameters of process tasks are considered since they represent the actual results that should be produced by implementing the process model. On the level of links between candidate services, however, output parameters that are not linked to mandatory input parameters of other services are not relevant for achieving a running service composition.

Below, we capture the sets of relevant parameters with the notions of a task parameter set and a composition parameter set.

**Definition 7 (Task Parameter Set)** Let \( C = \langle T, R \rangle \) be a composition task. Then, the task parameter set \( \mathcal{P}^T \) is defined as the union of all mandatory input parameters and output parameters, \( \mathcal{P}^T = \bigcup_{T \in T} \{p \mid p \in T_m \cup O \} \).

**Definition 8 (Composition Parameter Set)** Let \( S_C = \langle A, L \rangle \) be a solution for composition task \( C = \langle T, R \rangle \). The set of services participating in the solution is denoted by \( W \subseteq R \) and contains a service \( W \), iff

- there is a semantic assignment \( \langle pt, pw, e \rangle \in A \) related to parameters of \( W \), i.e., \( pw \in T_m^W \cup O^W \), or
- there is a semantic link \( \langle ow, iw, e \rangle \in L \) related to parameters of \( W \), i.e., \( ow \in O^W \) or \( iw \in T_m^W \).

Then, \( \mathcal{P}^W = \bigcup_{W \in W} \{p \mid p \in T_m^W \} \) is the composition parameter set.

**Example 4.** Back to our running example (See Example 3), the task parameter set for the process task \( T_1 \) is \( \mathcal{P}^{T_1} = \{it_1, ot_1\} \). Composition parameter sets vary between solutions. For Solution 1, we observe \( \mathcal{P}^W = \{iw_{2,1}\} \) and for Solution 2, it holds \( \mathcal{P}^W = \{iw_{3,1}, iw_{1,1}, iw_{1,2}\} \).

We now define the structure of functions used to evaluate a solution of a composition task. Informally, evaluation functions take a collection of parameters and the semantic assignments or links between them and evaluate them to a measure in the \([0, 1]\) range. Below, we use the common notation \( 2^S \) to denote the power set of some set \( S \).
Definition 9 (Task Evaluation Function) Let $S_C = \langle A, L \rangle$ be a solution for composition task $C = \langle T, R \rangle$ and let $P^T$ be the task parameter set. A task evaluation function takes the following form:

$$f_T : P^T \times 2^A \rightarrow [0, 1]$$

Definition 10 (Composition Evaluation Function) Let $S_C = \langle A, L \rangle$ be a solution for composition task $C = \langle T, R \rangle$ and let $P^W$ be the composition parameter set. A composition evaluation function takes the following form:

$$f_W : P^W \times 2^{L \cup A} \rightarrow [0, 1]$$

In the remainder of this section, we suggest several measures for integration effort based on different instantiations of these functions.

**Binary Precision and Recall** Our first instantiation assigns a value of 1 to parameters that are exactly matched according to the evaluation value of the semantic assignment or link; and 0 to all other parameter. This approach can be applied to evaluate semantic assignments yielding a binary task evaluation function $f_b^T$ or to evaluate semantic links leading to a binary composition evaluation function $f_b^W$. Even though both functions are defined identical, they are applied to different domains as presented in Definition 9 and 10.

$$f_b^T(x, Y) = f_b^W(x, Y) = \begin{cases} 1 & \forall (p, p', e) \in Y : x \in \{p, p'\} \Rightarrow e = \text{exact} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Based on these two specific evaluation functions, we present a first versions of precision and recall adapted for service composition tasks. As outlined above, these measures share an evaluation function in the numerator and differ by the normalization factor used in the denominator.

First, binary task precision and recall proxy the alignment integration effort and are defined as follows.

**Definition 11 (Binary Task Precision and Recall)** Let $S_C = \langle A, L \rangle$ be a solution for composition task $C = \langle T, R \rangle$ and let $P^T$ be the task parameter set. For parameter $p \in P^T$, the set of related semantic assignments is defined as $A_p = \{ (pt, pw, e) \in A \mid pt = p \}$. Then, binary task precision is defined as

$$Pr^b_T = \frac{\sum_{p \in P^T} f_b^T(p, A_p)}{|A|} \quad (2)$$

and binary task recall is defined as

$$Re^b_T = \frac{\sum_{p \in P^T} f_b^T(p, A_p)}{|P^T|} \quad (3)$$
Second, we adapt the same approach to define binary composition precision and recall to proxy composition integration effort.

**Definition 12 (Binary Composition Precision and Recall)**

Let $S_C = \langle A, L \rangle$ be a solution for composition task $C = \langle T, R \rangle$ and let $P^W$ be the composition parameter set. For parameter $p \in P^W$, let $A_p$ be the set of related semantic assignments (Definition 11) and let $L_p = \{\langle ow, iw, e \rangle \in L | p \in \{ow, iw\}\}$ be the set of related semantic links. Then, binary composition precision is defined as

$$Pr^b_W = \frac{\sum_{p \in P^W} f^b_W(p, L_p)}{|L|}$$

and binary composition recall is defined as

$$Re^b_W = \frac{\sum_{p \in P^W} f^b_W(p, A_p \cup L_p)}{|P^W|}.$$

**Example 5.** Calculation of these measures for Solution 2 (Fig. 5(b) and Table 2) of our example yields the following results. Out of the two semantic assignments, one is evaluated to be exact whereas the other is imprecise. Therefore, binary task precision evaluates to $1/2 = 0.5$. Since process task $T_1$ has one mandatory input parameter and one output parameter, but only one assignment exact, binary task recall is $1/2 = 0.5$. Since both semantic links of Solution 2 are exact, binary composition precision evaluates to $2/2 = 1$. Among the solutions services, $WS_1$ has two input parameters and $WS_3$ has one. Both input parameters of $WS_1$ are covered by semantic links that are evaluated to be exact. However, input parameter $iw_{3,1}$ of $WS_3$ is covered by a semantic assignment that is imprecise. Therefore, binary composition recall evaluates to $2/3 \approx 0.66$.

**Non-Binary Precision and Recall**

In previous work [6], we have shown binary measures to be inferior compared to non-binary measures due to their lack of sensitivity and smoothness. We therefore suggest non-binary versions of precision and recall for service composition by introducing a scoring function. This function is defined over $[0, 1]$ rather than $\{0, 1\}$ as in the binary case (see Equation 1). Ideally, this function should be well correlated with the amount of transformation and completion effort required. To achieve this goal, for instance, it may be based on the semantic distance of concepts that are part of semantic annotations of input and output parameters. Another instantiation of such a scoring function may be grounded in data type disparity. That is, a value in $[0, 1]$ would estimate the transformation effort due to data-type differences between parameters, e.g. an effort estimation matrix for basic XML-Schema types or a tree-edit distance function for complex types.

**Definition 13 (Scoring Function)**

Let $S_C = \langle A, L \rangle$ be a solution for composition task $C = \langle T, R \rangle$ and let $P^T$ and $P^W$ be the task parameter set and the composition parameter set, respectively. Then, a scoring function $f_s$ takes the following form $f_s: P^T \cup P^W \times P^T \cup P^W \rightarrow [0, 1]$. 
Based on this scoring function, we are able to define non-binary variants $f_{nb}^b$ and $f_{nb}^w$ of the two evaluation functions $f_b^b$ and $f_b^w$ given in Equation 1 as follows:

$$f_{nb}^b(x, Y) = f_{nb}^w(x, Y) = f_s(x, y_1) \cdot \ldots \cdot f_s(x, y_n) \text{ for } Y = \{y_1, \ldots, y_n\}$$

Then, the non-binary versions of precision and recall are obtained by replacing the binary evaluation functions in Definitions 11 and 12 with their non-binary counterparts.

**Non-binary Parameter Importance** So far, we distinguished parameters that are declared to be mandatory from those that are optional for assigning a candidate service to a process tasks or for creating a link between two candidate services. In practice, many standards used for the description of Web service interfaces, e.g., the Web Services Description Language (WSDL), support the definition of mandatory and optional parameters. As such, these languages provide the basis for a binary interpretation of parameter importance. In some cases, however, parameter importance may not be explicitly stated, which requires white-box evaluation of Web service code to conclude on parameter optionality.

In addition, parameter importance may be evaluated with a non-binary measure to come to a more fine-grained judgment. Different approaches can be taken to instantitate such a function. For instance, the order of parameters in Web services interfaces often carries implicit hints on their importance, listing the most important parameters first, whereas less important parameters, that may be applicable only in rare cases, are listed last. We accommodate for such a fine-grained notion of importance with an explicit function.

**Definition 14 (Parameter Importance Function)** Let $S_C = \langle A, C \rangle$ be a solution for composition task $C = \langle T, R \rangle$ and let $P_T$ and $P_W$ be the task parameter set and the composition parameter set, respectively. Then, a parameter importance function $f_i$ takes the following form $f_i : P_T \cup P_W \rightarrow [0, 1]$.

Modifying the non-binary precision and recall measures to account for parameter importance entails multiplication of the respective (task or composition) evaluation function with its importance, and the replacement of a parameter / link / assignment count in the denominator with a sum over importance. Due to space limitations, we outline the adapted definition solely for task recall:

$$Re_{nb}^T = \frac{\sum_{p \in P_T} f_i(p) \cdot f_{nb}^b(p, A_p)}{\sum_{p \in P_T} f_i(p)}.$$  

**5 Empirical Evaluation**

In this section, we demonstrate the effectiveness of the proposed concepts in measuring expected integration effort.
Set-up The basis of our evaluation is the Web Service Challenge\(^3\). This challenge describes five composition tasks and 30 solutions for these tasks, that are ranked by QoS properties such as latency. A composition task is described in WSDL [7] and semantically annotated in OWL [8]. The solutions are built from a Web service repository of 4180 semantically annotated service descriptions and presented in WS-BPEL [4]. All parameters, semantic annotations and datatypes are opaque and represented by random numbers. For example, consider Task 1 of the challenge (upper right part of Fig. 6), which has 10 input parameters and four output parameters. The first place solution is a composition of seven services providing all four output parameters and requiring all 10 input parameters to execute. In the composition in Fig. 6, lines represent semantic links between the composed services and rectangular outlines mark output parameters which are semantically assigned to the process task.

For each link / assignment, a semantic path between target and candidate parameters is provided. A zero-length path represents an exact semantic match between parameters (i.e. both are a US social security number). A path of length one represents parameters which are two instances of the same semantic concept (i.e. one service outputs true and the other requires false but both belong to the truth value concept). Paths of length two or more represent subsumption of the candidate parameter by the target parameter (i.e. ISBN subsumes ISBN-10). The length of a semantic path in the ontology provides us with a measure of the

---

\(^3\) see http://www.ws-challenge.org/ [accessed 2013-09-30]
first dimension of integration effort, i.e., the number of atomic transformations needed to implement a running composition.

The second dimension of integration effort, i.e., the number of atomic completions, cannot be measured directly since the challenge required complete coverage of task parameters and does not include interim solutions. Therefore, we created artificial solutions in which some semantic links and assignments are removed. We justify this variation as follows. When a solution contains effort-intensive assignments / links, designers may consider only links whose quality is higher than some threshold. To simulate this scenario, we used an increasingly strict threshold to prune low quality assignments and links. Note that, of the 49 semantic links and assignments in the solution, only four are evaluated to be exact.

For our evaluation, we considered the non-binary measures for task/recall precision and recall since our previous work [6] highlighted the benefits of such non-binary measures already.

Results Figure 7 presents the results of our threshold experiment, averaged over all composition tasks. The value of the two non-binary composition measures is plotted on the Y axis against an increasingly lenient threshold on the X axis. We observe that both measures show the expected behaviour: As the threshold becomes more strict (from right to left), recall decreases and precision increases. To validate our hypothesis that precision will be a valid proxy for transformation effort, we plotted task precision against the number of semantic assignments requiring such an operation (Fig. 8(a)) and composition precision against the number of semantic links requiring an atomic transformation oper-
\[ R^2 = 0.8782 \]

\begin{center}
\begin{tabular}{c|c|c|c}
Task & Composition
\end{tabular}
\end{center}

(a) Task Recall
(b) Composition Recall

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig9.png}
\caption{Recall vs. atomic completion}
\end{figure}

6 Related Work

Detailed semantic quality measures are rare in Web-Service composition literature. The work by Lécué and Mehandjiev [1] identifies semantic quality of links as a partial proxy to integration effort by stating: ‘selecting links with the best functional quality will ensure easy end-to-end integration between services’. Defining two quality functions over semantic links, the work fails to address unmatched inputs, although completion of parameters represents a major source of integration effort. Further, categorical ordering of the match-quality function arbitrarily puts Also, it defines the plug in relation to be superior to subsume relations, assuming that generalization between concepts induces a containment relation. Finally, all matched output-input pairs of the same semantic match type receive the same score, thereby ignoring the transformation effort from one concrete data type to the other. This simplification may hinder use of the defined metrics as proxies for integration effort.

Related measures for data complexity, were considered by Basci and Misra [9] in the context of estimating future maintenance costs. Their work was extended by Mao [10] to include the complexity created by the composition itself under a business process specified in BPEL. Of the nine metrics considered, the data complexity metric shows most promise for our purposes as it may be conceptually correlated with the integration effort. Highly complex links should incur higher effort. However, by focusing on the data transmitted (output parameter) it fails to consider differences in complexity between the output and input parameters of the same link and the effort required to resolve these differences. Technical difficulties using this measure as a metric may arise due to the fact that it is defined over \([0, \infty)\) and therefore numerical comparison of results for different composition plans may prove difficult to rationalize. Classic Information Retrieval versions of precision and recall were considered by works such as Toch.
et. al. [11]. However, their focus is on the retrieval task within the alignment step and is limited to binary relevance judgement of a ranked list of candidate services and, therefore, is not suitable for our scenario. In earlier work, Toch et. al. [12] suggested a ranking function for services w.r.t a process task based upon parameter similarity. The proposed function resembles some of the evaluation functions presented in Section 4 by virtue of being based upon parameters and an evaluation of the extent of parameter similarity and importance. However, the function ignores the effects of composing several services to answer a single task and does not evaluate a complete process task w.r.t a proposed composition.

7 Conclusions

In this paper, we presented a framework, allowing definition of coarse and fine-grained effort based measures for the evaluation of Web service compositions. All measures are on a standard $[0,1]$ scale, allowing comparison of candidate compositions during a solution search process or during design-time evaluation of composition algorithms. Our empirical evaluation shows that the proposed measures effectively discern between effort intensive and effortless integrations. In future work, we intend to further investigate various instantiations of the framework presented.

References