

# Towards a Search Engine for Functionally Appropriate, Web-enabled Models and Simulations

by

Qing Cao

B. S. Department of Automation, Tsinghua University, China 1997

M. S. Department of Automation, Tsinghua University, China 2000

Submitted to the Department of Mechanical Engineering  
in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy in Mechanical Engineering

at the

Massachusetts Institute of Technology

June 2006

© 2006 Massachusetts Institute of Technology  
All Rights Reserved

Signature of Author.....

Department of Mechanical Engineering

May 19, 2006

Certified by.....

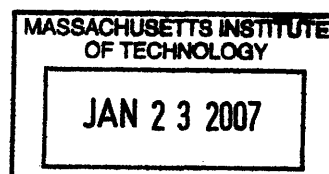
David R. Wallace  
Associate Professor of Mechanical Engineering

Thesis Supervisor

Accepted by.....

Lallit Anand

Chairman, Department Committee on Graduate Students



ARCHIVES



# **Towards a Search Engine for Functionally Appropriate, Web-enabled Models and Simulations**

by

Qing Cao

Submitted to the Department of Mechanical Engineering  
on May 19, 2006 in Partial Fulfillment of the  
Requirements for the Degree of  
Doctor of Philosophy in Mechanical Engineering

## **ABSTRACT**

New emerging modeling and simulation environments have the potential to provide easy access to design models and simulations on the Internet, much as the World Wide Web (WWW) has provided easy access to information. To support sharing, integration and reuse of web-enabled applications (design models and simulations), a search engine for functionally appropriate/similar models is needed.

There are ongoing efforts to develop ontological descriptions for web content and simulation model functionality, where semantics of available services are explicitly represented using a shared knowledge representation of concepts and rules. Simulation publishers are responsible of semantically marking up the interfaces with such ontological annotations. In contrast to such an approach, this work proposes a flexible, implicit, pattern matching solution that does not require any extra annotations to accompany the models, much as the way current web search engines operate.

A learning-through-association, similarity-based approach was developed. It uses only pre-existing low-level information in web-enabled simulation interfaces—such as model and parameters names, parameter units, parameter scale, input/output structure, causality, and documentation — to synthesize templates that become archetypes for functional concepts. Then, different interfaces are matched against templates and are classified based on how they are similar to a certain template. Newly found functionally similar interfaces can be merged into the original template, thereby both generalizing the pattern for a functional role and strengthening the most critical aspects of the pattern.

This thesis also developed algorithms based on graph theory and pre-defined heuristic attributes similarity metrics. The information from model interfaces is represented using Attributed Relational Graphs (ARG), where nodes represent parameters and arcs represent causality relationships. Templates are represented as Fuzzy Attributed Relational Graphs, which are extended ARGs whose node attributes are fuzzy sets. Then, a bipartite graph-matching algorithm is used to compare graphs and the similarity between an interface and a template. Graph merging algorithm is also designed for template generalization. A prototype implementation of proposed algorithms is developed and applied to a suite of real-life engineering models. Results validate the hypothesis and demonstrated the plausibility of the approach.

**Thesis Supervisor: David R. Wallace**  
**Title: Associate Professor of Mechanical Engineering**

**Committee Members: David Wallace, Ph.D. (Chair)**  
**Associate Professor of Mechanical Engineering,**

**David C. Gossard, Ph.D.**  
**Professor of Mechanical Engineering**

**Sanjay E. Sarma, Ph.D.**  
**Associate Professor of Mechanical Engineering**

## Acknowledgements

I deeply appreciate this institute, MIT, the great land full of wisdom and imagination, where I've trained and grown to be a researcher; where I've met and been friend with so many smart people. Many of them not only are wonderful scholars, but also shine in multi-facets. I am so lucky to have them as my professors, colleagues, and friends.

There are many people who had helped me along my long journey for Ph.D. that I want to thank for. Professor David Wallace, my advisor and mentor for 5 years, constantly guided and supported me. There has been a specific time that I was frustrated and not being productive. David didn't give up on me and shielded me with flexibility and generous courage. Professor Nicola Senin, my other advisor from University of Parma of Italy, not only kindly directed my research remotely in spite of his own very busy schedule and time differences, but also flew all the way from Italy to Boston to attend my defense. This thesis work won't be possible without them.

Professor David Gossard and Sanjay Sarma, my other committee members, thanks for providing me valuable feedbacks to keep me in the right direction. My dear previous and current CADLAB fellows: Elaine Yang, Inès Sousa, Prabhat Sinha, Sane Wu, Jaehyun Kim, Aubery Williams, Twiggy Chan, Charles Dumont, , Keith Thoresz, Renu Fondaker, Jacob Wronski, Wei Mao, Sittha Sukkasi, Amy Banzaert, Iason Chatzakis, Sangmok Han, James Penn, Barry Kudrowitz, Andrew Carvey, Monica Rush, and many other students who I've been worked with during my many years in 3-458, I am so lucky to have you as my colleagues and shared so many fun stuff with you.

This finishing point is especially meaningful to me since a very sad event had happened during my 2nd year. This event completely changed my life. I am very lucky to have a supporting network around me and they never lose their faith in me. I want to express my sincere gratefulness to Rita Fisher, my therapist and special friend for 3.5 years, thanks for listening to me and helping me to be independent, to grow, to trust and to love again; and to Shih-Chi Chen, my little brother and boyfriend, thanks for opening a new door and bringing hope and sunshine to my life. My good friends Yanxia and Wei Sun, Rong Wang and Hongyu Jiang, Tao Cheng and Yiben Lin, Wei Mao, Nuo Sheng, Minqi and Wei Wang; my suitemates in Ashdown: Sally Kwok, and Lisa Hsu: thank you all for making my life at MIT so colorful. No matter where I am, I will always cherish this memory with all of you.

Moreover, I am grateful to my family: my father Xiaolin Cao and mother Peifen Chen, who are always behind me, taking all my dreams seriously, and loving me as the apple of their eye, this thesis is my best gift to you. My sister An C. Calson and brother-in-law Randy Calson, thank you for taking care of me always.

Finally, to the beloved looking upon my shoulder, thank you and peace in heaven.

# Table of Contents

<b>List of Figures</b>	<b>vii</b>
<b>List of Tables</b>	<b>ix</b>
<b>Nomenclature</b>	<b>x</b>
<b>1. Introduction</b>	<b>1</b>
1.1 Context	1
1.1.1 Vision for a World Wide Simulation Web	1
1.1.2 Web-enabled Applications	2
1.2 Research Motivation	4
1.3 Problem Definition	5
1.4 Proposed Solution Overview	7
1.5 Thesis Structure	8
<b>2. Background</b>	<b>9</b>
2.1 The Semantic Web	9
2.2 Ontological Approach	10
2.3 Open Issues With the Ontological Approach	11
2.4 Pattern Matching Approach	13
2.5 Comparison Between Two Approaches	14
<b>3. Representing the Interface of Parametric Simulation Models</b>	<b>15</b>
3.1 Introduction	15
3.2 Identifying Function-related Information From Interfaces	16
3.3 A minimal information subset derived from interface definitions	18
3.4 Examples	20
<b>4. Graph Interface and Template Representation</b>	<b>27</b>
4.1 Introduction	27
4.2 Graph Representation of Interfaces	30
4.3 Graph Representation of Templates	32
<b>5. Similarity Graph Matching Algorithms</b>	<b>37</b>
5.1 Introduction	37
5.2 Similarity measures	40
5.2.1 Similarity Measure for Each Node Attribute	40
5.2.1.1 Similarity for Name Attributes	40
5.2.1.2 Similarity for Dimension Attributes	41
5.2.1.3 Similarity for Unit Attributes	41
5.2.1.4 Similarity for Data Type Attributes	42

5.2.1.5	Similarity for Input/Output Attributes	43
5.2.2	Similarity between a Single Attribute Value and a Fuzzy Set Attribute Variable	44
5.2.3	Similarity Between an Interface Node and a Template Node	45
5.2.4	Graph Similarity Measure: Similarity between an Interface Graph and a Template Graph	47
5.3	Graph alignment algorithm	48
5.3.1	Introduction	48
5.3.2	Brief Review of Bipartite Matching	49
5.3.3	Algorithm Overview	51
5.3.4	Generate a Bipartite Graph for Node Alignment	52
5.3.5	Node Alignment Based on Bipartite Matching	54
5.3.6	Refining Node Alignment by Aligning Arcs	56
5.4	A Template-Interface Matching Example	58
<b>6.</b>	<b>Template Generalization</b>	<b>63</b>
6.1	Introduction	63
6.2	Seeding a template	64
6.3	Generalizing a template	65
<b>7</b>	<b>Validation and Results</b>	<b>69</b>
7.1	Prototype System Design	69
7.1.1	Search Engine Framework	69
7.1.2	Modules of the Prototype System	71
7.2	Validation and Results	73
7.2.1	Goal	73
7.2.2	Metrics to Evaluate Results	73
7.2.3	Test Data Set	73
7.2.4	Test 1: Test Similarity Matching Approach	76
7.2.4.1	Goal	76
7.2.4.2	Scenario	76
7.2.4.3	Results	76
7.2.5	Test 2: Test Template Generalization Idea	79
7.2.5.1	Goal	79
7.2.5.2	Scenario	80
7.2.5.3	Results	80
7.2.6	Test 3: Recall and Precision as a Function of The Template's Degree of Generalization	82
7.2.6.1	Goal	82
7.2.6.2	Scenario	82
7.2.6.3	Results	82
7.2.7	Test 4: Compare With Keyword-based Text Search	85
7.2.7.1	Goal	85
7.2.7.2	Scenario	86
7.2.7.3	Results	86
7.3	Additional Result	89

7.3.1	Test Data Set	89
7.3.2	Goal and Scenario	91
7.3.3	Results	91
<b>8</b>	<b>Conclusion and Future Work</b>	<b>95</b>
8.1	Conclusion	95
8.2	Contributions	96
8.3	Future Work	97
	<b>Reference</b>	<b>100</b>



## List of Figures

1-1	Applications (models and simulations) interact with each other through a WWW backbone and are accessed through WWW browsers.	2
1-2	Layers of a web-enabled application. The underlying model layer is protected from direct access.	3
3-1	Attributes can provide clues to functional roles of the underlying model, help to discriminate between web-enabled applications	17
3-2	A DOME interface of quarter car suspension simulation.	21
3-3	Dependency information for example DOME interface	22
3-4	Dependency information for example DOME interface	23
3-5	A web page that contains an executable interface of a cutting power consumption calculation model.	24
3-6	A web page that contains an executable interface of process tools.	25
4-1	Interface graph of a block volume and cost interface	32
4-2	Template graph example: a template graph seeded from interface graph shown in 4-1.	35
4-3	Template graph example: a generalized template graph representing functional role “cutting power consumption evaluator”.	36
5-1	An example of topologically identical ARGs with completely different functional roles	38
5-2	A bipartite matching example (adopted from [71])	50
5-3	Flow chart of the graph matching and similarity calculation process.	51
5-4	Bipartite graph creation for matching an interface graph and a template graph	53
5-5	Using bipartite matching to obtain node alignment	56
5-6	An example scenario of the importance of “virtual” arcs	58
5-7	An interface graph (right) is compared with the cutting power template graph (left).	59
5-8	Bipartite matching process of the matching task presented in figure 5-7.	60
5-9	Aligned template and interface	62
6-1	Seeding an initial template (right) from an example interface (left).	65
6-2	Merging equivalent nodes during template generalization.	66
6-3	Template generalization example.	68
7-1	Search engine framework design ( parts marked within the thick red dotted lines are implemented in the thesis)	70
7-2	A GUI tool for similarity assessment.	72
7-3	Definition of precision and recall	73
7-4	Average precision and recall for single-interface templates.	77
7-5	Visualization of the single-interface templates search result	78
7-6	Average precision and recall for generalized templates	81
7-7	Visualization of generalized templates search result	81
7-8	Recall and Precision as a function of the template’s degree of generalization	84

7-9	(a) Comparison of recalls (b) Comparison of precisions	88
7-10	Ranked result of matching a “longitudinal turning process a type 1” template to the whole test suite	92
7-11	Column chart visualization: Ranked result of matching a “longitudinal turning process a type 1” template to the whole test suite	92
7-12	Ranked result of matching a “longitudinal turning process a type 2” template to the whole test suite	94
7-13	Column chart visualization: Ranked result of matching a “longitudinal turning process a type 2” template to the whole test suite	94

## List of Tables

2-1	Comparison between pattern matching and ontological approaches	14
3-1	The parameter attributes set	18
3-2	Basic information of example DOME interface	21
3-3	Input and output parameters of example DOME interface	21
3-4	Basic information of example cutting power consumption interface	24
3-5	Input and output parameters of example cutting power consumption interface	25
3-6	Basic information of example gas compressibility factor interface	26
3-7	Input and output parameters of example gas compressibility factor interface	26
5-1	Nomenclature	39
5-2	Data type similarities table	43
5-3	Comparison of an interface node and a template node	46
5-4	Node pair similarity	53
5-5	All non-zero similarity node pairs ranked according to the similarity scores.	60
5-6	Aligned node pairs with the similarity scores.	61
7-1	57 Interfaces and their underlying models	74
7-2	56 interfaces are manually categorized into 10 categories	75
7-3	Recall and precision for search using templates with incremental number of interfaces	82
7-4	Recall and Precision of the keyword-based search for each category	86
7-5	Comparison of recall and precision of the keyword-based search for each category	87
7-6	Cutting power consumption test suite [80]	89

## Nomenclature

$x$	A single value variable, $x=x_0$
$n$	Interface node $n = \{x_{name}, x_{unit}, x_{dim}, x_{dtype}, x_{inout}\}$
$g$	Interface graph
$X$	Fuzzy set variable $X = \{x_1^{\phi_1}, x_2^{\phi_2}, \dots, x_i^{\phi_i}, \dots, x_n^{\phi_n}\}$
$\mu()$	Membership function of a fuzzy set
$\Phi$	Frequency, number of occurrences
$N$	Template node, $N = \{X_{name}, X_{dim}, X_{unit}, X_{dtype}, X_{inout}\}$
$G$	Template graph
$s()$	Similarity function
$s(x_a, x_b)$	The similarity between two variable $x_a$ and $x_b$
$s(x, X)$	The similarity between a single-value variable $x$ and a fuzzy set variable $X$
$s(n, N)$	The similarity between an interface node and a template node
$s(g, G)$	The similarity between an interface graph and a template graph

# Chapter 1

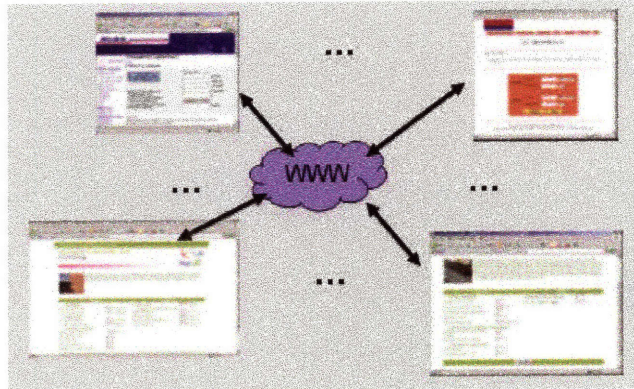
## Introduction

### 1.1 Context

#### 1.1.1 Vision for a World Wide Simulation Web

Since the last decade, new emergent modeling and simulation environments based on the Internet infrastructure have been researched and developed by academic groups and commercial companies, resulting in tools such as MIT's DOME (Distributed object-based modeling environment) [1,2], Engineous' FIPER [3], and Phoenix Integration's ModelCenter [4]. While each tool implements a different integration and solving paradigm, they all provide an easy way to access distributed models or simulations and to parametrically inter-related these models or simulations in an ad hoc fashion.

These tools provide enabling technologies for an envisioned World Wide Simulation Web (WWSW) ---- a World Wide Web (WWW) of numerous *web-enabled applications*. Web-enabled applications are parametric simulation models and simulations that are accessible through WWW browsers and can interact with each other through a WWW backbone (figure 1-1).



**Figure 1-1: Applications (models and simulations) interact with each other through a WWW backbone and are accessed through WWW browsers.**

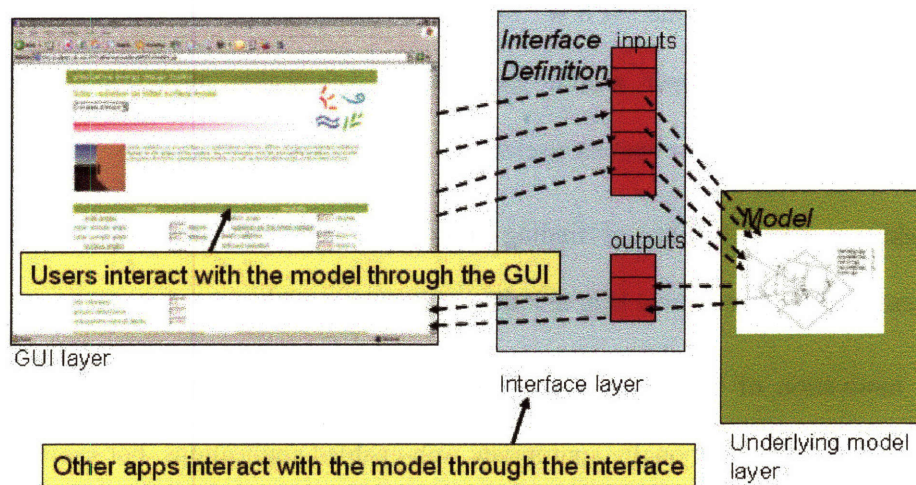
The WWSW builds a global community of individuals offering access to simulation services related to their own specialties that can then be used for the rapid exploration of design tradeoffs and global optimization [1, 5]. Engineers all over the world could contribute to the content of the WWSW, allowing it to evolve and expand, as more people use it to look for design information and test design scenarios, just as the WWW evolved and greatly benefits everyday life today.

### **1.1.2 Web-enabled Applications**

Web-enabled applications are simulation models accessible and interoperable through the WWW. Typical web-enabled application has three layers. On the top there is the GUI layer, usually web browsers, where users interact with the model. The second layer is the *interface* layer. An interface can be seen as a window or view on a model or set of models through which users can execute the model, similar to the concept of an interface in object-oriented programming. Interfaces usually define inputs and outputs of the affiliated model and are collections of various data types, ranging from simple numbers

to vectors, matrices, or any type of file (CAD, documents, images, etc.). Interfaces are parametrically operable via appropriate Internet protocols. The third layer is the underlying model layer, which the real model or simulation resides. This layer is encapsulated and invisible to public access. Accessing models or simulations via interfaces makes it possible for service providers to keep their knowledge and model definitions in-house and control user access to proprietary information.

For example, figure 1-2 shows a web-enabled application of a “solar radiation” model. Underlying simulation model is encapsulated and only interface layer is exposed.



**Figure 1-2: Layers of a web-enabled application. The underlying model layer is protected from direct access.**

Web-enabled applications are similar to common web services in two aspects: they provide certain computational capability as services; and they are operable through web interfaces, where they interact with users or other applications via proper Internet-based communications.

## 1.2 Research Motivation

New emerging modelling and simulation environments have the potential to provide easy access to models and simulations over the Internet, much as the World Wide Web (WWW) has provided easy access to information. In a long term run, as more and more participants contribute their models and simulations to the WWSW, the number of web-enabled applications on the WWSW will soon become massive. Therefore, just as it is now essential to have search engines for finding web content, building a search engine to find web-enabled applications on the WWSW becomes a pressing need. Without search engines, information on the WWW would be inaccessible and become useless. Analogically, without proper search tools, these web-enabled applications will be impossible to locate and thus become useless.

In addition, the capability of finding appropriate web applications as integration components also will be critical help to build integration models.

The motivation of this work is to design a non-traditional search engine that is capable of search for functionally appropriate web-enabled applications. For example, an automotive engineer searches for web-enabled applications regarding the kinematics and dynamics of a quarter-car five-link suspension; an energy consultant searches for web-enabled applications that can help him to evaluate the heat transfer rate of a new designed solar collector.

There are two function-related aspects this thesis is interested: *designed functional purpose* and *computational functionality*. Designed functional purpose refers to the “role” of an application in integrated systems, such as “volume calculation”, “amplifier”, etc. Computational functionality refers to the algorithms/equations of the underlying model.



For example: model “Length\*Width=Area” is different from model “ $\pi$ Radius<sup>2</sup>=Area”; on the other hand, two models based on “ $\pi$ Radius<sup>2</sup>=Area” would be considered same computational functionality even one is implemented in MATLAB and the other is in C. This thesis defines the term “*functional role*” to cover both aspects.

One may ask why a traditional keyword-type search engine is not pertinent to a “functional role search” required in this thesis. Web-enabled applications are special simulation models. Their full model definitions are encapsulated and only interfaces are public available. Unlike web pages that are mostly natural language documents, web-enabled interfaces contain rich, characteristic, structured information that reveals functional role of underlying simulation models. This requires consideration of aspects not addressed by traditional search engines, such as the exploitation of structural information not pertinent to the search of natural language documents.

### **1.3 Problem Definition**

How can we search for distributed web-applications with appropriate functional role on the WWSW? Much as a common search engine operates, the envisioned function-oriented search engine crawls the WWSW; processes each web interfaces; indexes each interface according to the functional role.

The main challenge comes from using computer to identify the functional role of a model or simulation from its low-level web interface information. The process of identifying the functional role of a model or simulation can be seen as a process of understanding the meaning of the model, or in other words, the process of inferring the semantics of the model.

In general, two broad classes of approaches have been proposed in the literature. One class adopts an ontological approach similar to the Semantic Web initiative [6], where knowledge is shared using pre-defined *ontologies*, which are formal representations of concepts and relationships pertaining to specific domains of discourse [7]. The ontological approach explicitly associate low-level information with meta-data defined in domain or upper ontologies, called “semantic mark-up”. This meta-data can be used by ontology-aware software application to make use of ontological matching, rule-based and/or first-order logic based tools to perform reasoning tasks, including classification, inference, etc., thus providing some sort of semantic processing of low-level information. Semantic inference is accomplished using rule-based logics or ontological matching [e.g. 9-17, 37, 38].

The second class of approaches attempts to extend syntax search techniques by discovering associations between high-level conceptual models (semantics) and the low-level content information (syntax). These approaches are often referred to as pattern matching based or similarity based. Typically, a query-by-example use-scenario is adopted and candidates are evaluated based upon their similarity to the query example. The semantic content of what is being searched for is implicitly captured in the syntactic pattern of the example [e.g. 29-36].

One might think that the ontological approach seems to offer the greatest potential for semantic inference. However, with consideration of the open usability issues with a typical ontological approach (to be elaborated in Chapter 2), and given the distributed infrastructure and size of the envisioned WWSW, we argue that the ontological approach

is not a good fit in this thesis context and choose to investigate towards a pattern-matching based approach.

## 1.4 Proposed Solution Overview

The proposed concept is a learning-by-example, *template* matching approach for identifying functionally appropriate web-enabled applications. A template is a synthetic data structure created from syntactic information in web interfaces that serves as pattern for the functional role of underlying simulation model. The solution method has three key steps:

Step 1: A specific interface to a web-enabled application is manually selected as representative of a functional role; the interface is elected as a template (reference pattern) for that functional role.

Step 2: Interfaces to web-enabled applications with compatible functional roles are identified through comparison to the template

Step 3: When web-enabled applications are identified as functionally compatible, their interfaces can be used to update the template, strengthening and generalizing the pattern.

The base assumption for this approach to work is that there must be a strong relationship between the structure of the low-level information at the interface level and the functional role of a web service; and that it must be possible to identify structural patterns sufficiently representative of functional roles. Moreover, since the association between concepts representing functional roles and structural patterns at the interface

level is generated through human assertion, the strength and applicability of the approach depends significantly on consistency of user behavior.

## **1.5 Thesis Structure**

Chapter 1 provides an introduction to the context and the need for a non-traditional search engine for WWSW. It also defines the problem and presents the contribution. Chapter 2 reviews related work. Chapters 3 and 4 are dedicated to introducing the graph representation of interfaces and templates, covering what syntactic information is expected to be available within the model interfaces, how this information can be represented using an ARG, and how a template pattern is created and represented using a fuzzy ARG. In Chapter 5, the graph matching concept and the similarity measures and algorithms are described, while Chapter 6 describes how templates are seeded and then generalized as new, functionally similar simulation models are identified. Chapter 7 outlines a prototype implementation of the approach and discusses experiments conducted to evaluate the proposed method. Test results are provided. The conclusion of the thesis (Chapter 8) summarizes the thesis contributions and recommends future research directions.

# Chapter 2

## Background

### 2.1 The Semantic Web

Attempting to associate a meaning (semantics) to low level information contained in web page in a form understandable by computers is a very popular research in the WWW research community, including the Semantic Web project [9] and projects stemmed from it such as Semantic Grid [10], Semantic Web Service [11,12,13,14], Semantic search engine [15,16,17], etc.

The methodology of the Semantic Web is to "marked up" web pages with semantic information of their contents; these mark-ups are either machine-readable descriptions, or just meta-tags. Software agents crawling the WWW can process these mark-ups, thereby facilitating automated information gathering and research by computers.

The Semantic Web provides several stacks of mark-up languages, from as data-centric, customizable Extensible Markup Language (XML), Resource Description Framework (RDF) to Web Ontology Language (OWL). RDF is a simple data model for referring to objects ("resources") and how they are related. Ontology adds more vocabulary for describing properties and classes: among others, relations between classes, cardinality (e.g. "exactly one"), equality, richer typing of properties and characteristics of

properties (e.g. symmetry), and enumerated classes. These languages are combined together to describe the content of Web documents [9].

Since late 1990s, World Wide Web Consortium (W3C) has been propagating the semantic web standards, markup languages and related processing tools to make machines “understand” web page content and thus support linguistic queries. The Semantic Web is said to have reached a significant size: A recent publication regarding semantic search engine has reported that about 1.3 millions of Semantic Web documents has been found on the WWW as of March 2006 [18]. Not surprisingly, most of these documents are RDF-based documents, since the more concepts and classes used, more reasoning power is required for processing software.

## **2.2 Ontological Approach**

Information systems need to combine the precision of formal semantics with the needs of cognitive transparency, as they incorporate increasingly sophisticated and heterogeneous information content. The recent explosion of interest on ontologies is an important component of this trend [19]. The ontological approach has been adopted to facilitate knowledge sharing and management, information integration, database analysis and design, information retrieval and extraction, object-oriented analysis, etc. [20]

Ontology has a long history in philosophy, in which it refers to the subject of existence. The research of using ontologies as conceptual modeling techniques exists before the Semantic Web initiative. A well cited definition of ontology in the computer science area is Gruber’s definition “specification of a conceptual classification”, which can be dated back to 1993 [21].

The ontological approach relies on explicitly representing semantics using ontologies, shared knowledge representations of concept terms and rules. Information publisher is responsible for annotating plain information with meta-data in the form of ontological semantic markups that define the meaning of the contained information. Then, by using rule-based semantic inference on semantic markups, computers can process and automatically interpret the semantic meaning of information.

The Semantic Web initiative has stimulated multi-disciplinary efforts in representation language, domain and upper ontologies, and supporting tools for editing, analysis and reasoning ontologies. The ontological approach now has expanded from pure conceptual modeling to both ontology modeling and ontology engineering. Applications can be found include enterprise integration, natural language translation, biomedical, mechanical engineering, standardization of product knowledge, electronic commerce, geographic information systems, legal information systems, biological information systems, and software engineering [20].

### **2.3 Open Issues with the Ontological Approach**

Although ontologies and the ontological approach have been very popular in academic and commercial projects, there are a number of open usability challenges that may have implications for their breath and size of application:

First, it can be difficult to translate a domain of knowledge into a formal knowledge representation—expertise in both formal knowledge representation and the given domain is required. Once defined, users of the ontology must have sufficient understanding of the formal representation to correctly define the additional meta-information that

accompanies their models. Usually the annotation process needs much manual work. Even with the use of helpful knowledge acquisition tools, this step adds overhead to the process of publishing simulation models [22].

Second, simulation models providers and requesters are heterogeneous participants. It can be difficult to establish a common shared ontology standard. This is the same issue the Semantic Web project is facing. Many projects has made a hypothesis that communication would be easier if everyone describe things in the same way, and a particular brand of unifying description will therefore be broadly and swiftly adopted. However, since meta-data describes a worldview, incompatibility is an inevitable by-product of vigorous argument [23].

Often ontologies are developed locally and additional work is needed to alignment them. This ontology alignment problem requires that mappings are made between the different representations, but automating the process of specifying such correspondences remains a key challenge [24, 25].

Third, it is not clear how frameworks based on ontologies will react to subsequent changes or extensions with time and usage. In some cases, modifications might take as much effort as rebuilding a whole new knowledge base.

The ontological approach is expected to be work well in the following context: the domain to be organized needs to be small corpus, formal categories, stable entities, restricted entities, and clear edges; the participants need to be authoritative source of judgment, coordinated users or expert users [26]. Given the problem in this thesis, the WWSW is going to be large corpus, no global authority, and unstable entities; and



participants could be uncoordinated and amateur users. This is our primary consideration while seeking alternative directions.

## **2.4 Pattern Matching Approach**

Pattern matching is the act of checking for the presence of the constituents of a given pattern. In contrast to pattern recognition, the pattern is specified. Such a pattern concerns conventionally either sequences or structures. Pattern matching is used to check that things have the desired structure, to find relevant structure, to retrieve the aligning parts, and to substitute the matching part with something else [27]. For example, regular expression method is a kind of pattern matching method that used in textual matching (string matching). Another example application area is the content-based image retrieval area. Content-based image retrieval uses the visual contents of an image such as color, shape, texture, and spatial layout to represent and index the image [28, 29, 30]. These visual contents together form certain patterns, Then, by using similarity matching algorithms to match syntactic patterns, computers can infer whether different items are semantically related, therefore, classifying image components and objects with similar content patterns; Other applications include character recognition [31,32], geographic information identification [33], 3D object recognition [34,35,36], etc.

The pattern matching approach utilizes the much less structured syntactic information that is readily available, thereby avoiding the challenges associated with fixed knowledge frameworks, but at the cost of giving up an explicit representation of the content's meaning. Therefore, pattern matching approach could face the so called "semantic gap" challenge [39]. Semantic gap is defined as a mismatch between higher-level abstraction (what is the real knowledge) to low-level information (what is really being implemented)

[40]. When adopting this approach, one should assure that patterns in the syntactic information are sufficient correlated to the high level content being searched.

## 2.5 Comparison Between Two Approaches

Table 2-1 concludes this chapter with a comparison between the two approaches briefly reviewed above.

**Table 2-1: Comparison between pattern matching and ontological approaches**

Approach	Pattern matching approach	Ontological approach
Goal	Content-based search and retrieval	Semantic search and retrieval
Methodology	Infer the content semantics from syntax/structural pattern similarities	Annotate syntax with explicit, formal knowledge description
Search on	Low level information	Meta-data mark-ups
Inference based on	Pattern matching	Rule-based and/or first logic, Ontological matching.
Pros	<ul style="list-style-type: none"> <li>• No extra manual annotation of meta-data</li> <li>• No need to worry about creating/modifying an ontology</li> </ul>	<ul style="list-style-type: none"> <li>• Explicit, crisp definition of concepts and their relationships</li> <li>• Rule-based reasoning (classification, inference, etc.)</li> </ul>
Cons	<ul style="list-style-type: none"> <li>• Semantics is not explicitly known</li> <li>• Relies on sufficient discriminating power of the patterns</li> <li>• Relies on strength of the relationship between patterns and concepts</li> </ul>	<ul style="list-style-type: none"> <li>• Difficult to create</li> <li>• Difficult to modify</li> <li>• Difficult annotation</li> <li>• Difficult ontology alignment</li> <li>• Challenges of logic-based reasoning</li> </ul>

## **Chapter 3**

# **Representing Interfaces of Parametric Simulation models**

### **3.1 Introduction**

In order to build a computer data model representing functional related pattern, we need to decide what information in interface definitions are potentially important associated with functional roles. Several researchers have worked on how to represent behavior and function in engineering design [41-45]. For example, Szykman [41,42] uses function and flow model to represent an artifact based on domain-specific taxonomies. Mocko [45] characterizes a simulation as an executable behavioral model with assumptions, context, inaccuracy and interface description such as parameters, variables, and constants. This work requires a full definition of the model.

However, unlike design databases or design repositories that are usually shared within a company, emerging distributed simulation environments envisioned in this work are for heterogeneous service providers. It is assumed that service providers may in fact wish to keep their knowledge/model definition locally and only expose executable interfaces to users. Therefore, instead of developing generic formal functional and behavioral representation for computational services, the goal was to identify a minimum subset of information that needs to be available within a computational service interface for it to be

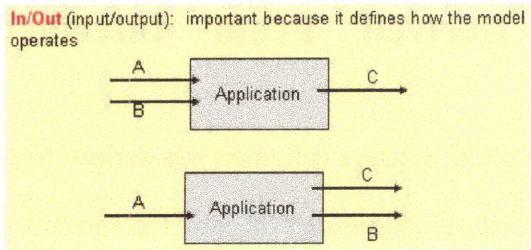
useable, independent of how the simulation environment enabling the service is implemented.

## **3.2 Identifying Function-related Information From Interfaces**

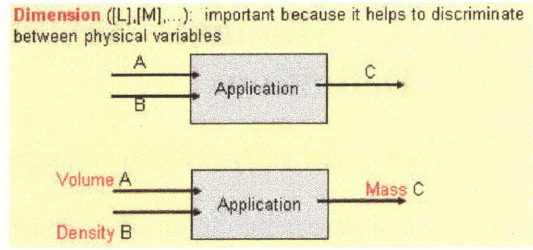
What information is available in interfaces? Could it provide clues about functional roles? A study was completed to identify what syntactic information would likely be available within simulation models interfaces that could be relevant to functional roles. Several existing systems, including traditional web service pages that provide executable web interfaces to models and simulations were surveyed.

Interfaces of web-enabled applications studied in thesis typically include contain operable parameters. Minimal information to make interface useable includes: a list of input and output parameters by name; which parameters are inputs or outputs; the data type of each parameter; and the units of the parameters. In some cases, causal dependency information between inputs and outputs, configuration information, and documentation were also available. These types of information are relatively ad-hoc and vary from service to service.

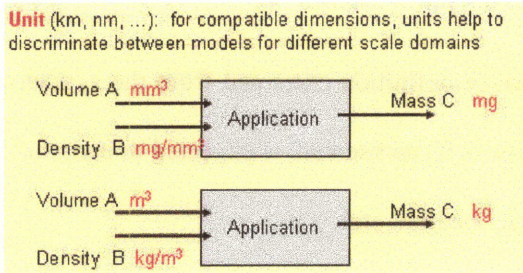
Some attributes are considered important information since they may provide clues to the functional role of the underlying application and help to discriminate between web-enabled applications. Figure 3-1 shows how input/output, dimension, unit, data type, name, and causality are important function-related information that can help to discriminate applications.



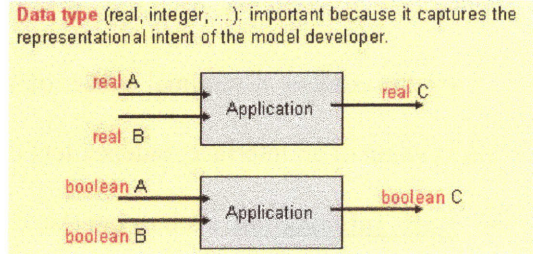
(a)



(b)



(c)



(d)

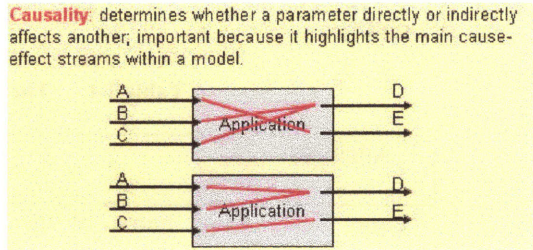
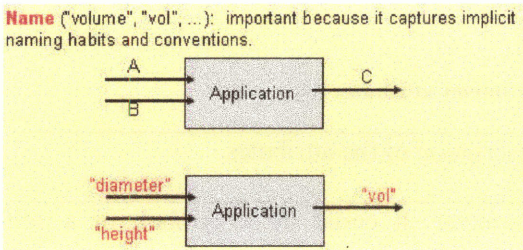


Figure 3-1 Attributes can provide clues to functional roles of the underlying model; help to discriminate between web-enabled applications

### 3.3 A Minimal Subset Derived From Interface Definitions

Based on the analysis before, a minimal set of interface definition was defined from common syntactic definition of a model interface, independent of how the simulation environment is implemented.

**Definition 3-1:** A minimal subset of interface definition is a set of definition that contains minimal information with which an interface is usable.

In this context, a minimal subset of interface definition extracted from the syntactic information of an interface, independent of how is it represented, is expected to contain:

- **Basic information**: such as interface name, location etc.
- **Input and output parameters**: Each parameter in an interface has associated attributes such as a name, units, dimensions, data type, and whether they are an input or output.

Table 3-1 The parameter attributes set

Attribute	Description of the attributes
Name:	The name of the parameter. It could be a symbol, or a self-explaining term, such as “ $\pi$ ”, “g”, or “fiber Poisson ratio”, etc. This attribute is consider a string variable.
Dimension:	The functional dimension of a parameter, complies with the definition used in dimensional analysis [46]. E.g. basic dimensions include length “L”, mass “M”, time “t”, current “I”, etc.
Unit:	The unit in Metric or English systems. For example, the metric

	<p>unit of “g” is “m/s<sup>2</sup>”.</p> <p>There are some special cases when a parameter is <u>unitless</u>:</p> <p>1) A parameter may be dimensionless. A dimensionless parameter is has no physical unit associated with it. It has a dimension of “1”.</p> <p>Such a number is typically defined as a product or ratio of quantities which have units of identical dimension [47], such as Reynolds Number or radians.</p> <p>2) A parameter is not a quantitative variable, therefore it is unitless, e.g. a Boolean parameter, true or false;</p> <p>3) A parameter contains no unit because model builders did not bother specify the units of the model parameters.</p>
Data Type:	<p>The data representation used for a parameter, e.g. real, integer, Boolean, String, Matrix, Vector, etc.</p> <p>Engineers learn how to use data type to fulfill their modeling purpose, e.g. they might want to use a “real” parameter when involving accurate calculation; might want to use a “integer” to model the total day number of one year, etc. Therefore, data type is chosen in the parameter attribute set.</p>
Input/Output:	<p>Whether this parameter is an input parameter or an output parameter. Independent variables that are changed by users are considered as inputs. Driven variables that are changed by the simulation and exposed in an interface are considered outputs.</p>

- **Causal relationship**: dependency information between inputs and outputs. This information is available in the DOME interface standard. Since the prototype system uses mainly DOME interfaces for testing and doing experiments, we include this information in the minimal subset of interface definition. However, it is not guaranteed to be commonly available for other format of interfaces and thus the approach may exploit this information when available but should not require it.

In some cases, detailed interface description, configuration and documentation (in natural language) are also available. These types of information are relatively ad-hoc and vary from service to service. Since this additional information was not reliably present, it was not included in the minimum subset of information expected in interfaces definitions.

## 3.4 Examples

This section uses some examples to show the minimal subset defined applies to real world web-enable applications, including one DOME interfaces and two web-service interfaces.

### **Example 1: A DOME Interface**

Figure 3-2 is another example DOME interface. The simulation model supporting this interface is a MATLAB model for calculating the dynamics of a quarter-car suspension model. In DOME WWSW, the MATLAB model is encapsulated and wrapped as a DOME service provider. Geometry, mass, stiffness, and damping variables are inputs to the model. The model output is vertical acceleration.



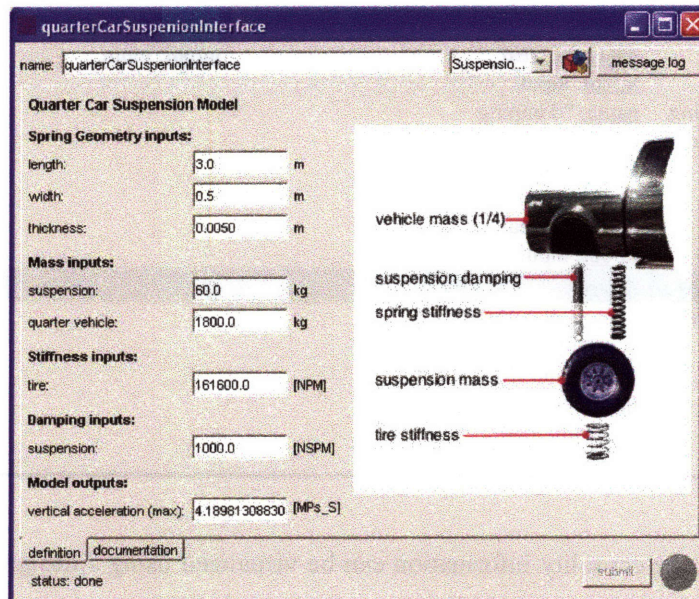


Figure 3-2 A DOME interface of quarter car suspension simulation

The minimal information subset extracted from this interface definition is listed below: Basic information is shown in Table 3-2. Input and output parameters are listed in Table 3-3, and causality information is shown in Figure 3-3 and 3-4.

Table 3-2 Basic information of example DOME interface

Name:	“quarter car suspension interface”
Location	cadlab20:8080/Public/Quarter car suspension/quarterCarSuspensionInterface
Description	NULL

Table 3-3 Input and output parameters of example DOME interface

<b>List of inputs:</b>			
length	<u>name:</u> “leaf spring length” <u>dimension:</u> [L] <u>unit:</u> m <u>data type:</u> real <u>in/out:</u> input	width	<u>name:</u> “leaf spring width” <u>dimension:</u> [L] <u>unit:</u> m <u>data type:</u> real <u>in/out:</u> input
thickness	<u>name:</u> “leaf spring thickness” <u>dimension:</u> [L] <u>unit:</u> m <u>data type:</u> real <u>in/out:</u> input	suspension	<u>name:</u> “suspension mass” <u>dimension:</u> [M] <u>unit:</u> kg <u>data type:</u> real <u>in/out:</u> input
quarter vehicle	<u>name:</u> “quarter vehicle mass” <u>dimension:</u> [M]	tire	<u>name:</u> “tire stiffness” <u>dimension:</u> [MT <sup>-2</sup> ] <u>unit:</u> N/m

	<u>unit:</u> kg	<u>data type:</u> real
	<u>data type:</u> real	<u>in/out:</u> input
	<u>in/out:</u> input	
damping coefficient	<u>name:</u> "damping coefficient"	
	<u>dimension:</u> [MT <sup>-1</sup> ]	
	<u>unit:</u> N.s/m	
	<u>data type:</u> real	
	<u>in/out:</u> input	
<b><u>List of outputs:</u></b>		
acceleration	<u>name:</u> "maximum acceleration"	
	<u>dimension:</u> [MT <sup>-2</sup> ]	
	<u>unit:</u> m/s <sup>2</sup>	
	<u>data type:</u> real	
	<u>in/out:</u> output	

Similarly, the causality information can be visualized using a design structure matrix (Figure 3-3), or a directed graph (Figure 3-4). All inputs parameters affect the single output parameter "maximum acceleration".

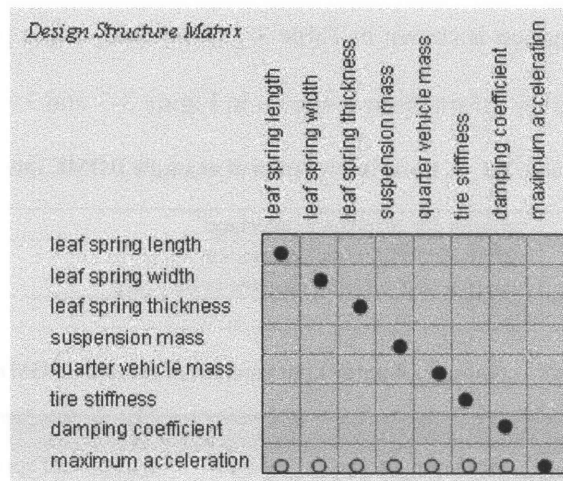
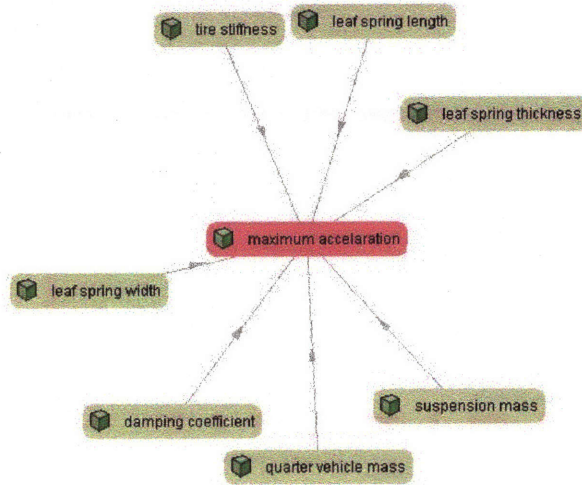


Figure 3-3 Dependency information for example DOME interface



**Figure 3-4** Dependency information for example DOME interface

**Example 2: Common Web Service Interfaces**

The minimum subset of information is generic and not limited to being mined from DOME interfaces only. It can also be extracted from traditional web service pages that provide executable interfaces to parametric models.

The following example is a web service page of a cutting power consumption evaluation model [48]. If the interface definition is the region marked in red within Figure 3-5, one can follow extract similar minimum subset of information to represent this model interface, is shown in table 3-4 and 3-5.

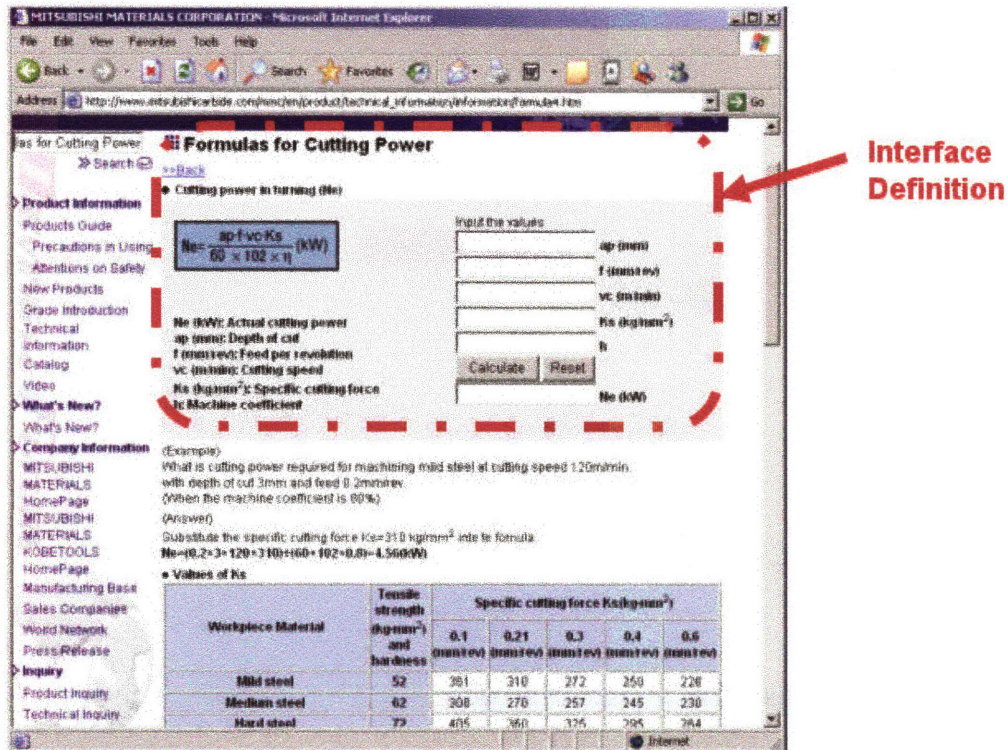


Figure 3-5 A web page that contains an executable interface of a cutting power consumption calculation model

Table 3-4 Basic information of example cutting power consumption interface

Name:	“cutting power in turning (Ne)”
Location:	<a href="http://www.mitsubishicarbide.com/mmc/en/product/technical_information/information/formula4.htm">http://www.mitsubishicarbide.com/mmc/en/product/technical_information/information/formula4.htm</a>
Description	<p>Formula for cutting power in turning:</p> $N_e = \frac{a_p \cdot f \cdot v_c \cdot K_s}{60 \times 102 \times \eta} \text{ (kW)}$ <p>Where:</p> <ul style="list-style-type: none"> <li>Ne (kW): Actual cutting power</li> <li>ap (mm): Depth of cut</li> <li>f (mm/rev): Feed per revolution</li> <li>vc (m/min): Cutting speed</li> <li>Ks (kg/mm<sup>2</sup>): Specific cutting force</li> <li>h: Machine coefficient</li> </ul>

**Table 3-5 Input and output parameters of example cutting power consumption interface**

<b>List of inputs:</b>	
Depth of cut	<p><u>name:</u> "ap"  <u>dimension:</u> [L]  <u>unit:</u> mm  <u>data type:</u> real  <u>in/out:</u> input</p>
Cutting speed	<p><u>name:</u> "vc"  <u>dimension:</u> [LT<sup>-1</sup>]  <u>unit:</u> m/min  <u>data type:</u> real  <u>in/out:</u> input</p>
Machine coefficient	<p><u>name:</u> "h"  <u>dimension:</u> []  <u>unit:</u> no unit  <u>data type:</u> real  <u>in/out:</u> input</p>
Feed per revolution	<p><u>name:</u> "P"  <u>dimension:</u> [LR<sup>-1</sup>]  <u>unit:</u> mm/rev  <u>data type:</u> real  <u>in/out:</u> input</p>
Specific cutting force	<p><u>name:</u> "Ks"  <u>dimension:</u> [ML<sup>-2</sup>]  <u>unit:</u> kg/mm<sup>2</sup>  <u>data type:</u> real  <u>in/out:</u> input</p>
<b>List of outputs:</b>	
Actual cutting power	<p><u>name:</u> "Ne"  <u>dimension:</u> [ML<sup>2</sup>T<sup>-3</sup>]  <u>unit:</u> kW  <u>data type:</u> real  <u>in/out:</u> output</p>

Figure 3-6 is another example web service page taken from an online process simulation toolkit [49]. The procedure calculates a gas compressibility factor. The minimal information set for this interface is in table 3-6 and 3-7.



**Figure 3-6 A web page that contains an executable interface of process tools**

**Table 3-6 Basic information of example gas compressibility factor interface**

Name:	“Gas Compressibility Factor”
Location:	<a href="http://www.processassociates.com/process/property/z_factor.htm">http://www.processassociates.com/process/property/z_factor.htm</a>
Description	This procedure calculates the compressibility factor of a real gas using the Redlich Kwong equation of state.

**Table 3-7 Input and output parameters of example gas compressibility factor interface**

<b>List of inputs:</b>			
Operating pressure	<u>name:</u> “Operating pressure” <u>dimension:</u> [ML <sup>2</sup> T <sup>-1</sup> ] <u>unit:</u> psi <u>data type:</u> real <u>in/out:</u> input	Operating temperature	<u>name:</u> “Operating temperature ” <u>dimension:</u> [K] <u>unit:</u> °F <u>data type:</u> real <u>in/out:</u> input
<b>List of outputs:</b>			
Critical pressure	<u>name:</u> “Critical pressure” <u>dimension:</u> [ML <sup>2</sup> T <sup>-1</sup> ] <u>unit:</u> psi <u>data type:</u> real <u>in/out:</u> output	Critical temperature	<u>name:</u> “Critical temperature” <u>dimension:</u> [K] <u>unit:</u> °F <u>data type:</u> real <u>in/out:</u> output

It should be mentioned that, unlike DOME interfaces, causality information is not available in the web service pages definition. For example in figure 3-5, since the full equation is given, we can manually figure out the causality information. In a more general case like example in figure 3-6, where the full definition of the model is not available causality information cannot be used.

As a summary, this section has proposed a minimum subset of syntactic information that one can expect to extract from model interfaces available as services in a WWSW. This information will be used as a basis for pattern matching in attempt to infer functional roles.

## **Chapter 4**

# **Graph-based Interface and Template Representation**

### **4.1 Introduction**

In chapter 3, a minimal subset of information that can be derived from model interfaces was proposed. In this work, we use the concept of graph similarity, graph distance, and graph matching as a basis for the novel approaches we've developed for classification tasks instead of using restrictive vector models.

Graphs are important and effective mathematical tools to represent relationship and structural information. It's been widely adopted in many different problems, including sorting, compression, traffic/flow analysis, resource allocation, pattern recognition, etc. Modeling data as graphs is extremely desirable in many applications since these graphs can retain more information than sets or vectors of simple feature primitives [51]. In addition, the well-studies mathematic framework of graph theory provides many punished algorithm to draw upon, such as graph similarity techniques have been applied to pattern recognition fields, such as image recognition [29,30,50], map recognition [33], handwriting recognition [31,32], 3D shape recognition [34,35,36], etc.

In this chapter, an *attributed relational graph* (ARG) is proposed as a data representation for representing interface and template. Attributed relational graphs<sup>1</sup> are introduced in work by Tsai and Fu in 1979 [32]. It is a relational data structure that consists of nodes and a set of arcs that represent the relation between the nodes. Attributed relational graphs have been broadly used as a data representation for pattern recognition applications. Usually, nodes represent objects or parts of objects and arcs are used to represent relationships between corresponding nodes.

With an interface ARG representation defined, the representation for a template, which represents a set of similar interfaces, can be considered. A template is a synthetic data structure that stores patterns found in an interface initially and generalized by associating with more similar interfaces. Therefore, a template can be designed an aggregated graph model that represents a set of similar ARGs.

There are two broad classes of graph representations for a set of ARGs. One broad class is based on probability. Random graph is a graph generated by some random process [52]. Wong et.al. [53] first adopted Random ARG model for pattern recognition. A random graph is an ARG where the vertex/arc attributes have random variables as their values. Due to the computational complexity associated with high-order joint probability analysis, random graphs typically assume that all random variables are mutually independent and are thus simplified to first order probability distributions.

The other class of graph models for representing a set of similar ARGs is based on fuzzy theory. Fuzzy set theory is introduced by Dr. Lotfi Zadeh of UC/Berkeley in

---

<sup>1</sup> In broad literature, the term “Attributed Relational Graph” is interchangeable with the term “Attribute Graph”.



1965[54]. A fuzzy set is a set whose elements are characterized by a membership function  $\mu$  that is associated with each element indicating the degree of membership value of this member to the set. The value normally is a real value between the interval [0, 1].

Chan and Cheung [31] first proposed a fuzzy attributed relational graph (fuzzy ARG) model to represent objects in Chinese characters recognition. Krishnapuram et.al. used fuzzy ARG model to represent image in image retrieval [29,30]. In a fuzzy ARG, node/edge attributes use fuzzy sets as their values. Fuzzy attributed graphs incorporate vagueness by associating node and edge attributes with fuzzy sets, whose value measures degrees to which objects satisfy imprecisely defined properties. By generating a degree of class membership for an object instead of dichotomous classification, fuzzy methods provide a way to estimate missing or incomplete knowledge [29].

Although the mathematical operations based on fuzzy set theory may look similar to those for probabilistic ARGs, the two approaches differ in terms of their purpose. Sometimes, information in the form of frequency histograms or other probability curves is even used as the basis to construct a membership function for attributes in fuzzy ARGs. However, it should be remembered that membership functions are NOT necessarily probabilities [55].

A fuzzy AGR is the representation chosen in the work because:

- In this context, templates represent functional concept is learned (or generalized) from several example interfaces. The functional concept may have vagueness due to incomplete knowledge.

- Random graphs rely on large training sets to determine probability distributions for each attributes value. As fuzzy sets are usually intended to model human's cognitive states so they can be determined from either simple small-sample or sophisticated elicitation procedures. Given the limited training set available in our case a fuzzy ARG is more appropriate.

This chapter will first introduce the design of an ARG model for interface, because once interface ARG representation is defined, it serves towards a basis for representing a template. Then the design of an ARG model for template is present. Examples for interface graph and template graph are provided.

## 4.2 Graph Representation of Interfaces

Based on the minimal subset defined in Chapter 3, the graph represent of an interface is formally defined below:

**Definition 4-1(a):** an **interface node** can be formalized as:

$n = \{x_{name}, x_{unit}, x_{dim}, x_{dtype}, x_{inout}\}$ , where  $x_{name}$  represents the state of the name attribute for that particular node, and so forth.

**Definition 4-1(b):** An **interface arc**  $a_i$  connects the node  $n_j$  to the node  $n_k$  and it is

denoted by  $a_i = (n_j, n_k)$ . Arcs do not contain any attribute.

**Definition 4-1(c):** An **interface graph** is a directed graph:

$g = \{n_1, n_2, \dots, n_i, \dots, n_n; a_1, a_2, \dots, a_i, \dots, a_m\}$ , where  $\{n_1, n_2, \dots, n_i, \dots, n_n\}$  is a set of attributed nodes (vertices) and  $\{a_1, a_2, \dots, a_i, \dots, a_m\}$  is a set of arcs (edges).

In definition 4-1, the input and output parameters are represent as graph nodes, and parameter attribute set (name, unit, dimension, data type and whether they are inputs or outputs<sup>2</sup>) are presented as the node attribute set. The causality information is represented as graph arcs (when this information is available). The direction of an arc between two nodes shows which node drives which. Arcs can be an empty set  $\emptyset$ . When there is no arc, the graph is simply a set of disjointed attributed nodes. The basic information such as interface name, location is left out graph representation as auxiliary information of the graph model.

Figure 4-1 provides an example of the graph representation. This is an interface with underlying model calculating a block volume and cost interface from box's dimensions and material unit cost. The light grey nodes are inputs and dark grey nodes are outputs. Inputs are connected to outputs via directed arcs. Arcs are only present when causal information is available.

---

<sup>2</sup> Please refer to table 3-1

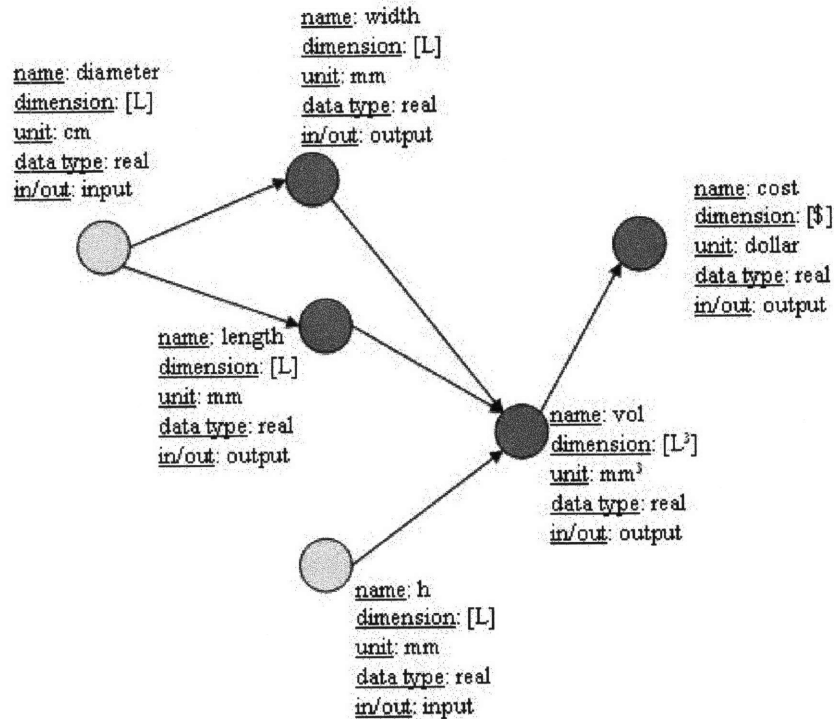


Figure 4-1 Interface graph of a block volume and cost interface

### 4.3 Graph Representation of Templates

A template is a fuzzy ARG represents a set of similar interfaces. To make the graph matching process less complicated, the template fuzzy ARG is designed so that the basic ARG structure stays the same with interface ARG:

- Template ARG contains attributed nodes with same fixed attributes of name, unit, dimension, data type and whether they are inputs or outputs; same as interface ARG's fixed node attribute set.
- Template ARG contains un-attributed arcs, used to represent causal relationships between nodes (when this information is available). The direction of an arc between two nodes shows which node drives which.

Meanwhile, a template needs to be able to continuously aggregate new information from example interfaces. The aggregating mechanism used in this work is based on merging equivalent nodes and arcs, which is explained in Chapter 6. Based on the aggregating mechanism, a template ARG extends from the basis of interface ARG:

- Template ARG's nodes and arcs are weighted: the weight of a node/arc is determined by how frequently the given node or arc is seen in exemplar ARGs. Frequently observed nodes or arcs are considered more significant.
- The value of each template ARG's node attribute is a fuzzy set variable, whose members are possible values learnt from examples and associated with a frequency number. The fuzzy set provides a thesaurus of example attributes values.

Based on the above description, a template graph can be formally defined.

**Definition 4-2:** A **fuzzy set variable** (of node attribute) is defined as:

$$X = \{x_1^{\phi_1}, x_2^{\phi_2}, \dots, x_i^{\phi_i}, \dots, x_n^{\phi_n}\}$$
, where  $x_i$  indicates the  $i^{\text{th}}$  state of the variable  $x$  and  $\phi_i$  indicates the number of times such state was asserted (*number of occurrences*).

**Definition 4-3(a):** A **template node** is defined as:

$$N = \{X_{name}, X_{dim}, X_{unit}, X_{dtype}, X_{inout}\}$$
, where  $X_{name}$  represents the fuzzy set variable of the name attribute for that particular node, and so forth.

**Definition 4-3(b):** A **template arc**  $a_i$  connects a template node  $N_j$  to the node  $N_k$

and it is denoted by  $a_i = (N_j, N_k)$ , Arcs do not contain any attribute.

Definition 4-3(c): A **template graph** is defined as:

$G = \{N_1^{\phi_1}, N_2^{\phi_2}, \dots, N_i^{\phi_i}, \dots, N_n^{\phi_n}; a_1^{\phi_1}, a_2^{\phi_2}, \dots, a_i^{\phi_i}, \dots, a_m^{\phi_m}\}$ , where  $N_i$  is a template node defined in Definition 4-3(a);  $a_i$  is a template arc defined in Definition 4-3(b).

Figure 4-2 provides an example of a fuzzy ARG template graph. This template graph is an initial template seeded from the single example interface shown in Figure 4-1. All the frequency numbers are marked in red color in the graph. The frequency number in the bottom left of a node designates the weight of that node; and the frequency number on an arc designates the weight of that arc. In the initial template graph, every node and arc has an initial frequency of 1. Each node attribute is a fuzzy set variable. The initial member of the fuzzy set variable is learned from the interface. For example, the unit attribute value of the template node “hole diameter” is the fuzzy set  $\{cm^1\}$ , the member “cm” in that fuzzy set is learned from the unit of original interface node “hole diameter”, and the frequency number associate with that member is 1.

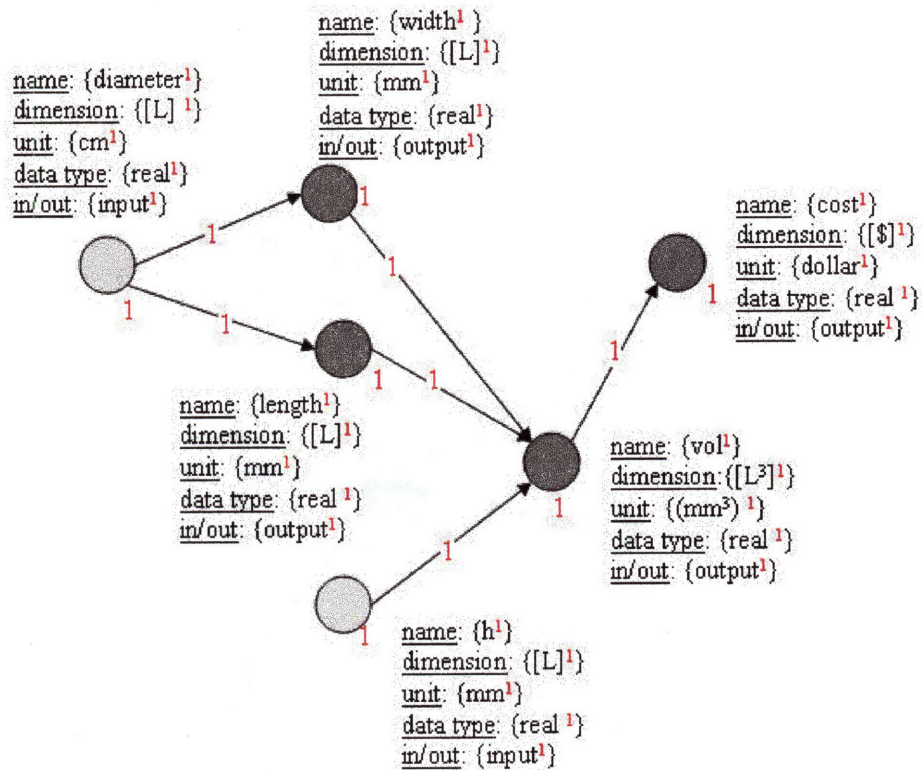


Figure 4-2 Template graph example: a template graph seeded from interface graph shown in 4-1

Figure 4-3 is another example of template graph. This template graph represents a functional role as “cutting power consumption evaluator”. It is learned from two similar example interfaces. As we can see, some nodes/arcs have a weight of 2, e.g. “D<sub>1</sub>”, “F<sub>s</sub>”, “P<sub>s</sub>”, and etc, indicating they are common elements both examples have and are merged during aggregation process. They may be more significant information in the pattern. Some nodes/arcs have a weight of 1, such as “k”, indicating they didn’t find equivalent elements during aggregation and are consider less important. During the aggregation process, frequency numbers associated with fuzzy set variables are changed as well. The

method for updating frequency information as part of the template adaptation process will be described in Chapter 6.

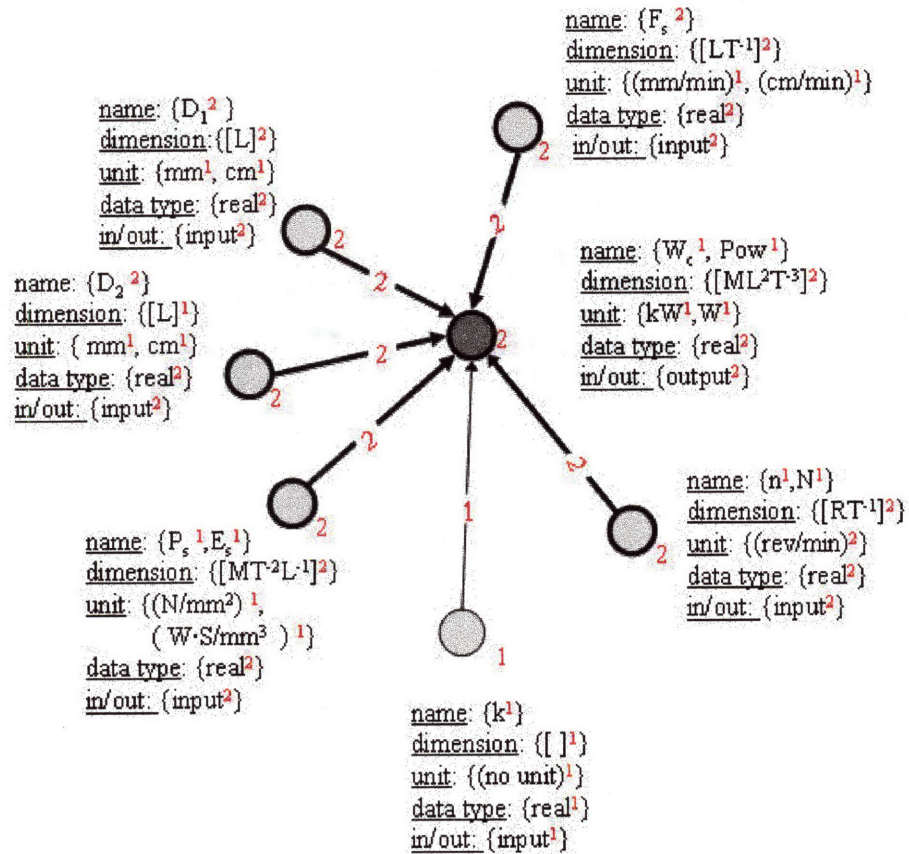


Figure 4-3 Template graph example: a generalized template graph representing functional role “cutting power consumption evaluator”



# Chapter 5

## Similarity Graph Matching Algorithms

### 5.1 Introduction

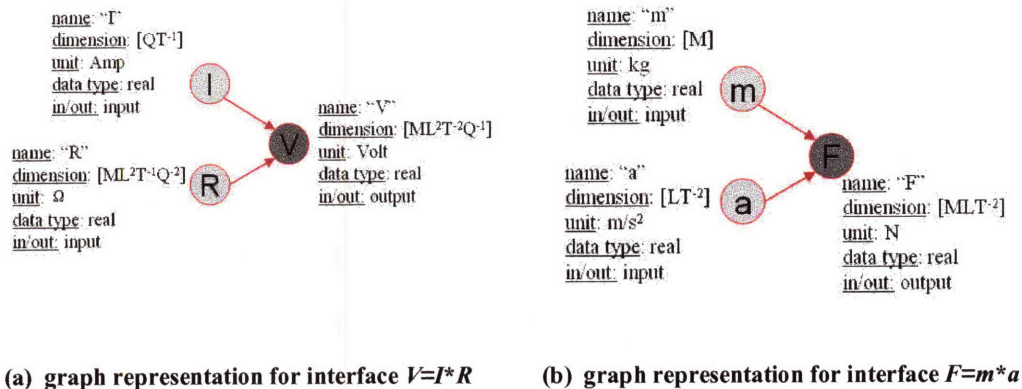
Chapter 4 defined an Attributed relational graph (ARG) model for representing interfaces and templates. By adopting a graph-based representation for use in pattern matching between templates and interface, determining functional role similarity becomes a graph matching task between an interface graph (ARG) and a template graph (fuzzy ARG).

Standard problems of graph matching include graph distance (a numeric measure of dissimilarity between graphs, edit distance [32, 60, 61], maximum common graph/minimum common graph [58], relaxation model [33], etc); exact graph matching (graph/sub-graph isomorphism [56]) and inexact matching (error-correcting [57], common sub-graph [58], fuzzy graph matching [59], etc.). A detailed review can be found in [51, 62].

A majority of published works, including the first paper on graph “isomorphism” by Ullman in 1976[56], are algorithms for un-attributed graphs. The graph matching problems for un-attributed graph focus on matching graphs based on their topological structure. Matching ARG graphs, on the other hand, is different from topological

matching. The focus of ARG graph matching is an inexact matching that gauges attributes consistency and overcomes structural errors. The isomorphism between two ARGs means the corresponding matching pairs of vertices and arcs must have consistent attribute values.

In this work, the topological information in this work (arcs) specify how inputs and outputs are connected to each other based on causality information, which is often not well-suited for determining alignment. For example, causality information will not always be available and, therefore, the graph would become several disjointed nodes. Further, even if causality information is available, many different functional roles might have identical causal structure, and the same functional role can have many different causal structures.



**Figure 5-1** An example of topologically identical ARGs with completely different functional roles

For example, a simple example of topologically identical ARGs with completely different functional roles is shown in Figure 5-1. The left side interface calculates electric potential difference (V) from the current (I) and the resistance(R) by the Ohm's law  $V=I \cdot R$ . The right side interface calculates force (F) from mass (m) and acceleration (a)

by the Newton's law of motion  $F=m*a$ . The interface graph is topologically the same, but their functional roles are completely different.

Thus, graph matching algorithms based on topological structure are not suited to this work. Instead, a graph matching method emphasizing node alignment is proposed since the ARG model defined in this work only contains attributed nodes and un-attributed arcs.

For the sake of explanation, we introduce the nomenclature used in this chapter first.

**Table 5-1 Nomenclature**

$x$	A single value variable, $x=x_0$
$n$	Interface node $n = \{x_{name}, x_{unit}, x_{dim}, x_{dtype}, x_{inout}\}$
$g$	Interface graph
$X$	Fuzzy set variable $X = \{x_1^{\phi}, x_2^{\phi}, \dots, x_i^{\phi}, \dots, x_n^{\phi}\}$
$\mu()$	Membership function of a fuzzy set
$\Phi$	Frequency, number of occurrences
$N$	Template node, $N = \{X_{name}, X_{dim}, X_{unit}, X_{dtype}, X_{inout}\}$
$G$	Template graph
$s()$	Similarity function
$s(x_a, x_b)$	The similarity between two variable $x_a$ and $x_b$
$s(x, X)$	The similarity between a single-value variable $x$ and a fuzzy set variable $X$
$s(n, N)$	The similarity between an interface node and a template node
$s(g, G)$	The similarity between an interface graph and a template graph

## 5.2 Similarity measures

Graph similarity is a numeric measure of similarity between graphs, usually between [0, 1]. It provides a mapping from a qualitative “alike-ness” concept to numeric values, the higher the value, the more similar the graphs are. This thesis developed a graph similarity measure that is an aggregation of node attributes similarity. Both interface ARGs and template ARGs contain fixed node attributes set. Similarity measures for each node attribute are presented first.

## **5.2.1 Similarity Measure for Each Node Attribute**

### **5.2.1.1 Similarity for Name Attributes**

Names are treated as strings and thus existing string similarity metrics can be adopted. There are many published string similarity metrics, such as Levenshtein distance (defined as the minimum number of operations needed to transform one string into the other [63]); cosine similarity measure; Jaro metrics proposed by Jaro[65] and later refined by Winkler[66](defined based on the number and order of the common characters between two strings), etc. A comparison of common string metrics can be found in [67]. Among the surveyed algorithms, the Jaro-Winkler algorithm is reported to yield best matching result.

In this work, a public domain algorithm library [68] is applied to match names. The algorithm tokenizes name into a vector of terms and use Jaro-Winkler similarity metric [66] (which is based on the number and order of common characters) to compare similarity between terms; then the token-by-token similarity score is aggregated using TFIDF(Term Frequency Inverse Document Frequency) algorithm[64]. TFIDF is typically considered a weighing mechanism. The importance of a word to a document

increase proportionally to the number of appears in the document but is offset by how common the word is in all of the documents in the collection or corpus.

The hybrid Jaro-Winkler with TFIDF similarity function replace the exact token matches used in TFIDF with approximate token matches based on the Jaro-Winkler scheme. For further reading on Jaro-Winkler with TFIDF similarity metric please refer to [67].

### 5.2.1.2 Similarity for Dimension Attributes

The functional dimension of a parameter, as defined in Table 3-1, complies with the definition used in dimensional analysis [46]. Basic dimensions include length “L”, mass “M”, time “T”, etc. Fundamental dimensions can be derived from units. For example, length unit “m” has a dimension of [L]; power unit “kW” has a dimension of  $[ML^2T^{-3}]$ ; and force unit “N” has a dimension of  $[MLT^{-2}]$ .

Similarity function for dimension is binary. Similarity is 0 for incompatible dimensions or 1 for compatible dimensions. The rationale behind a crisp dimensional similarity metric is that if two dimensions are not the same, they are incompatible.

### 5.2.1.3 Similarity for Unit Attributes

Unit similarity is determined after verifying fundamental dimensional compatibility (e.g. length). With dimensional compatibility is ensured, unit similarity is determined by comparing differences in the scale of the units. The rationale behind considering the scale difference in units is that when a parameter has a unit of “km” (kilometer), it is not likely to have the same functional role as another parameter with a unit of “nm” (nanometer).

A unit is first converted to corresponding base unit (SI base unit and derived unit) of the same dimension. Then scale factor can be calculated as the scale difference of a unit to the base unit based on a unit conversion measure [69]. For example, the base unit for length is m. Thus a unit attribute of km would be converted to 1000 m. Therefore, the scale factor of km is 1000.

**Equation 5-1:** Given two units, unit a and unit b, we defined a heuristic similarity function to calculate their similarity:

$$s(x_{unit\ a}, x_{unit\ b}) = \begin{cases} 0 & \text{different dim.} \\ 0.5^{|\log(SF(x_{unit\ a}) - SF(x_{unit\ b}))|} & \text{compatible dim.} \end{cases} \quad (\text{Equation 5-1})$$

In equation 5-1,  $SF( )$  denotes the function calculates the scale factor of a given unit. The heuristic similarity function map the logarithmic scale factor difference of two units into a  $[0, 1]$  range.

For example: given the two units “kW” and “hp”,

- Step 1: Check dimensions, both units have a dimension of  $[ML^2T^{-3}]$ .
- Step 2: Since the base unit here is “W”, scale factors are calculated as:  
SF(kW)=1000 ; SF(hp)=735.4988
- Step 3: and finally using equation 5-1, the similarity between “kW” and “hp” is calculated as:  $s("kW", "hp") = (0.5)^{|\log(1000) - \log(735.4988)|} = 0.911$

Unit similarity is conditional upon dimensional compatibility. It is likely that a parameter does not have a unit, either because it is dimensionless, or because the model creator didn't bother to define the unit. In such a case, the unit similarity is zero.

#### 5.2.1.4 Similarity for Data Type Attributes

The data types considered in this work are: real, integer, boolean, string, matrix, and vector. To determine the similarity between those data types, a survey was conducted by interviewing 8 modelers, who are research assistants and faculty at MIT. The resulting data type similarity table is shown in Table 5-2. This heuristic similarity table provides a elementary reference for this work to estimate the similarity between different data types.

**Table 5-2 Data type similarities table**

	Real	Integer	Boolean	String	Matrix	Vector
Real	1	0.666667	0.233333	0.125	0.275	0.283333
Integer	0.666667	1	0.433333	0.125	0.275	0.283333
Boolean	0.233333	0.433333	1	0.208333	0.125	0.133333
String	0.125	0.125	0.208333	1	0.041667	0.058333
Matrix	0.275	0.275	0.125	0.041667	1	0.75
Vector	0.283333	0.283333	0.133333	0.058333	0.75	1

Given two data type  $x_{dtype a}$ , and  $x_{dtype b}$ , the similarity  $s(x_{dtype a}, x_{dtype b})$  can be found in table 5-2. For example: the similarity between a “real” data type and a “real” data type is 1. The similarity between a “real” data type and a “integer” data type is 0.67.

### 5.2.1.5 Similarity for Input/Output Attributes

The in/out attribute defines the role of the parameter in an interface. The rationale behind an in/out similarity metric is that if two parameters under comparison are both inputs and both outputs, they may be compatible. If not, they are considered to be incompatible. Thus, the Similarity function for the in/out attribute is binary. A score of 0 is assigned to incompatible input/output attributes while 1 is assigned to compatible input/output attributes.

**Equation 5-2:** Given two data type  $x_{inout a}$ , and  $x_{inout b}$ , the similarity  $s(x_{inout a}, x_{inout b})$  is calculated as:

$$s(x_{inout a}, x_{inout b}) = \begin{cases} 1 & \text{if } x_{inout a} = x_{inout b} \in \text{inputs} \\ 1 & \text{if } x_{inout a} = x_{inout b} \in \text{outputs} \\ 0 & \text{otherwise} \end{cases} \quad (\text{Equation 5-2})$$

For example:  $s(\text{"input"}, \text{"input"}) = 1$ , and  $s(\text{"input"}, \text{"output"}) = 0$ .

## 5.2.2 Similarity between a single attribute value and a fuzzy set attribute variable

When a parameter's node in an interface graph is compared to a node in a template graph, one must compare the interface's single attribute values to the fuzzy set of possible attribute values associated with a template parameter's node.

Let an interface single attribute variable be  $x$ , where  $x = x_0$ .  $x_0$  is a state<sup>3</sup> of variable  $x$ ; let a template fuzzy set attribute variable as  $X$ , where  $X = \{x_1^{\phi_1}, x_2^{\phi_2}, \dots, x_i^{\phi_i}, \dots, x_n^{\phi_n}\}$ , and  $x_i$  indicates the  $i^{\text{th}}$  state of the variable  $X$  and  $\phi_i$  is the number of occurrence associated with each state.

**Equation 5-3:** The similarity of a single state  $x_0$  of a variable  $x$ , and the template variable  $X$  (using number of occurrences) is calculated as:

$$s(x_0, X) = \frac{\sum_{i=1}^n \phi_i \cdot s(x_0, x_i)}{\sum_{i=1}^n \phi_i} \quad x_i \in X \quad (\text{Equation 5-3})$$

For example, we are evaluating the similarity between an interface unit attribute value "hp", and a template unit attribute value fuzzy set  $\{kW^1, W^1\}$ . Using the unit attribute

---

<sup>3</sup> states may be referred to as "values" if the variable is numeric.



similarity function defined in 5.2.3, the similarity between “hp” and “kW” and that of “hp” and “W” are calculated first.

Given:  $x = \text{hp}$ ;  $X = \{kW^1, W^1\}$ , using equation 5-1:

$$s(\text{hp}, kW) = (0.5)^{|3-2.866|} = 0.911; \quad s(\text{hp}, W) = (0.5)^{|2.866|} = 0.1371$$

Then, the overall interface parameter-template parameter unit similarity score is the weighted sum of these values.

Therefore, the overall similarity between the interface value “hp” and the template value  $\{kW^1, W^1\}$  can be calculated using Equation 5-3:

$$s(\text{hp}, \{kW^1, W^1\}) = \frac{s(\text{hp}, kW) * 1 + s(\text{hp}, W) * 1}{2} = \frac{0.91 * 1 + 0.137 * 1}{2} = 0.52$$

### 5.2.3 Similarity between an interface node and a template node

Similarity between an attributed interface parameter (node) and a fuzzy attributed template parameter (node) is calculated based on attributes similarities.

**Equation 5-4:** Let an interface node be  $n$ , where  $n = \{x_{name}, x_{dim}, x_{unit}, x_{dtype}, x_{inout}\}$ ; let a template node by  $N$ , where  $N = \{X_{name}, X_{dim}, X_{unit}, X_{dtype}, X_{inout}\}$ , the similarity between  $n$  and  $N$  is calculated as:



$$s(n, N) = \begin{cases} \frac{w_{name} s(x_{name}, X_{name}) + w_{unit} s(x_{unit}, X_{unit}) + w_{dtype} s(x_{dtype}, X_{dtype})}{w_{name} + w_{unit} + w_{dtype}}, & s(x_{dim}, X_{dim}) = 1, s(x_{inout}, X_{inout}) = 1; \\ 0 & \text{otherwise.} \end{cases}$$

(Equation 5-4)

In Equation 5-4,  $w_{name}$ ,  $w_{unit}$ , and  $w_{dtype}$  are pre-defined weights for name, unit, and data attributes. Initially all the weights are set to unity. Methods for automatically tuning these weights are discussed in future work in chapter 8.

For example, table 5-3 provides an example of comparison of an interface node “w” side-by-side with a template node “Wc”.

**Table 5-3 Comparison of an interface node and a template node**

An interface node “w”	A template node “Wc”
 <pre> name: {w} dimension: {[ML<sup>2</sup>T<sup>-3</sup>]} unit: {hp} data type: {real} in/out: {output} </pre>	 <pre> name: {W<sub>c</sub><sup>1</sup>,Pow<sup>1</sup>} dimension: {[ML<sup>2</sup>T<sup>-3</sup>]<sup>2</sup>} unit: {kW<sup>1</sup>,W<sup>1</sup>} data type: {real<sup>2</sup>} in/out: {output<sup>2</sup>} </pre>

Step 1: Using Equation 5-3 and attribute similarity measures defined in 5.2.1, we calculate the similarity for each attribute:

- Name:  $s("w", \{ "W_c^1", "Pow^1" \}) = s("w", "W_c") \cdot \frac{1}{2} + s("w", "Pow") \cdot \frac{1}{2} = 0.43$
- Dimension:  $s\left([ML^2T^{-3}], \{ [ML^2T^{-3}]^2 \}\right) = s([ML^2T^{-3}], [ML^2T^{-3}]) \cdot \frac{2}{2} = 1$
- Unit:  $s(\text{hp}, \{ \text{kW}^1, \text{W}^1 \}) = \frac{s(\text{hp}, \text{kW}) + s(\text{hp}, \text{W})}{2} = 0.91 \cdot \frac{1}{2} + 0.137 \cdot \frac{1}{2} = 0.52$
- Data type:  $s(\text{real}, \{ \text{real}^2 \}) = s(\text{real}, \text{real}) \cdot \frac{2}{2} = 1$

- Input/output:  $s(\text{output}, \{\text{output}^2\}) = s(\text{output}, \text{output}) * \frac{2}{2} = 1$

**Step 2:** After obtaining all attribute similarities, node pair similarity is then evaluated by combining the attribute pair similarities using Equation 5-4.

Given all the weights are equal, The similarity between the two nodes can be calculated as:  $s(n, N) = \frac{0.43 + 0.52 + 1}{3} = 0.65$ .

## 5.2.4 Graph Similarity Measure: Similarity between an Interface Graph and a Template Graph

The similarity measure that determines the degree of overall match between an interface graph and a template graph is calculated by the weighted sum of aligned node similarity.

Given an interface graph  $g$ ,  $g = \{n_1, n_2, \dots, n_j; a_1, a_2, \dots, a_k\}$ . And a template graph  $G$ ,  $G = \{N_1^{\phi}, N_2^{\phi}, \dots, N_i^{\phi}, \dots, N_n^{\phi}; a_1^{\phi}, a_2^{\phi}, \dots, a_i^{\phi}, \dots, a_m^{\phi}\}$ . Notice the template nodes size  $n$  and arcs size  $m$  could be different from interface nodes size  $j$  and arcs size  $k$ .

Let  $p$  be a pair of all aligned node pairs; and  $p$  denotes an aligned node pair of  $n_p$  and  $N_p$ , where  $n_p$  is a node from  $g$ , and  $N_p$  is a node from  $G$ . The total number of aligned node pairs is designated as  $\#Pairs$  in equation 5-5, with  $\#Pairs \leq \min(j, n)$ .

**Equation 5-5:** The similarity between  $g$  and  $G$  is calculated as:

$$s(g, G) = \frac{\sum_{i=1}^{\#Pairs} \phi_{N_{p_i}} s(n_{p_i}, N_{p_i})}{\sum_{i=1}^n \phi_{N_{p_i}}} \quad (\text{Equation 5-5})$$

where  $p_i$  is the  $i^{\text{th}}$  pair of aligned node pair.  $\phi_{N_{p_i}}$  is the number of occurrences of the template node  $N_{p_i}$ , within the template graph.

The denominator in equation 5-5 is a normalization factor. The total aligned node pairs similarity is normalized to the total weighted sum of number of occurrences of template nodes. This normalization factor is introduced so that the similarity score will be in the range of  $[0, 1]$  and solve the problem of comparing similarity measures computed on graph pairs with different number of nodes.

Example of calculating graph similarity will be given in Section 5.4.

## 5.3 Graph Alignment Algorithm

### 5.3.1 Introduction

The desired graph matching algorithm in this thesis context should be flexible enough to handle different size and topology graphs, which falls into the in-exact graph matching category. Traditional in-exact graph matching algorithms such as graph editing distance methods have been adopted to matching ARGs graphs, e.g. Chan & Cheung [31], Eshera & Fu [60, 61]. However, since topological graph matching is a NP-problem, graph editing based graph matching algorithm can be computational expensive and scales poorly to larger graphs.

In this thesis, the graph matching problem is simplified as node alignment problem. This assumption is made based on that the ARG model used to represent interface and template is designed to encode the majority information in nodes, not in arcs, for a good reason. Arcs are optional when causality information is not given. Therefore, the topology information in these ARGs is not a primary concern as opposed to [31, 60, 61]

where the arcs are important since they encode rich relationship information. The simplification relieves the potential computational cost, especially for matching graphs large in size. However, since the simplification only provides approximate solution to the graph matching, post-processing is needed to refine or validate the matching.

Based on the above simplification, a *maximal, bipartite matching* algorithm based on similarity of node attributes using predefined fuzzy similarity measures is developed.

### 5.3.2 Brief Review of Bipartite Matching

In the mathematical field of graph theory, a *bipartite graph* is a special graph where a set of vertices can be divided into two disjoint sets such that no two vertices of the same set share an edge [74]. Bipartite matching is a process that utilizes the bipartite graph model to modeling matching problem, especially assignment problems (e.g. job allocation problem, stable marriage problem). Bipartite matching method for ARG graphs has been seen in published work on the 3-D image recognition [34, 50].

Mathematically, given a bipartite graph  $G = (V, E)$ , a bipartite matching  $M$  of  $G$  is defined as subset of  $E$ , such that no vertex in  $V$  is incident to more than one edge [73].

Typical solutions to bipartite matching are *maximum cardinality matching* (where the solution gives a maximum sized matching), and *maximal matching* (where no additional edges can be added into the solution given the already mapped nodes and arcs). Maximum cardinality matching is well used in matching unweighted graphs. Maximal matching is more suitable for weighted graph matching.

Figure 5-2 gives a bipartite matching example. The edges of the graph is ordered, therefore it is a weighted bipartite matching problem. A primitive greedy algorithm [71] can yield fast solution of a maximal matching.

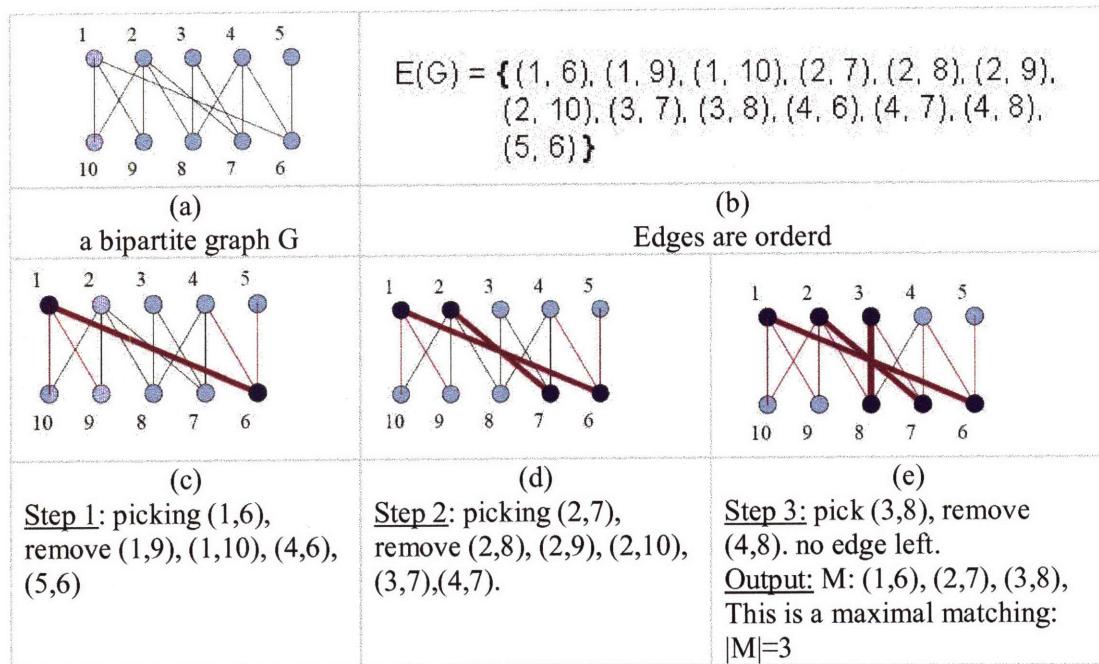


Figure 5-2: A bipartite matching example (adopted from [71])

The matching obtained by the greedy algorithm is a maximal matching, as shown in Figure 5-4(e). The matching  $M \{ (1,6), (2,7), (3,8) \}$  has a cardinality of 3.

Polynomial algorithms exist for bipartite matching since cycles are impossible in bipartite graphs. Standard algorithms for bipartite matching are based on network flow, using a simple transformation to convert a bipartite graph into an equivalent flow graph, such as Ford-Fulkerson method [73], Hopcroft and Karp method [73], etc. More effective algorithms such as Hungarian method [70] adopt the *augmenting path* idea to construct matching.

### 5.3.3 Algorithm Overview

The entire process of calculating graph similarity via graph alignment is depicted in Figure 5-2.

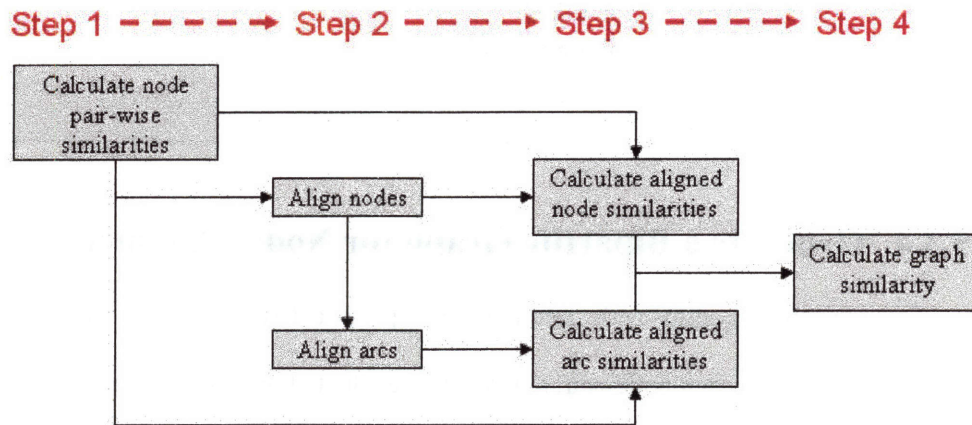


Figure 5-3 Flow chart of the graph matching and similarity calculation process.

A maximal bipartite matching algorithm is used to align compatible nodes based upon both node attributes similarity and arc information. The aligned template and interface nodes are used to compute an overall similarity score between the interface graph and the template graph.

Pseudo code is provided in Algorithm 5-1:

**Algorithm 5-1: Overall graph matching algorithm**

Input: an interface graph (ARG), a template graph (Fuzzy ARG)

- 1 Calculate node pair-wise similarities
- 2 Create a bipartite graph from non-zero-similarity node pairs, and use maximal bipartite matching algorithm to align nodes
- 3 Check arc compatibility
- 4 Calculate overall graph similarity

Output: a matching M, a similarity score

### 5.3.4 Generate a Bipartite Graph for Node Alignment

The algorithm begins with creating node pairs between the interface graph and template graph. For each node pair, similarity is evaluated based on the method presented in Section 5.3.2. Node pairs with similarity scores of 0—indicating they are not compatible—are eliminated. This step obtains a ranked list of all possible matching node pairs.

Then a weighted bipartite graph can be created from all map-able node pairs. Naturally, this bipartite graph is partitioned into two independent set of nodes: interface graph nodes and template nodes.

Figure 5-4 shows an example: an interface graph shown in Figure 5-4(b) is compared with a template graph shown in Figure 5-4(a). Assume we've calculated the node pair wise similarity as defined in Table 5-4.



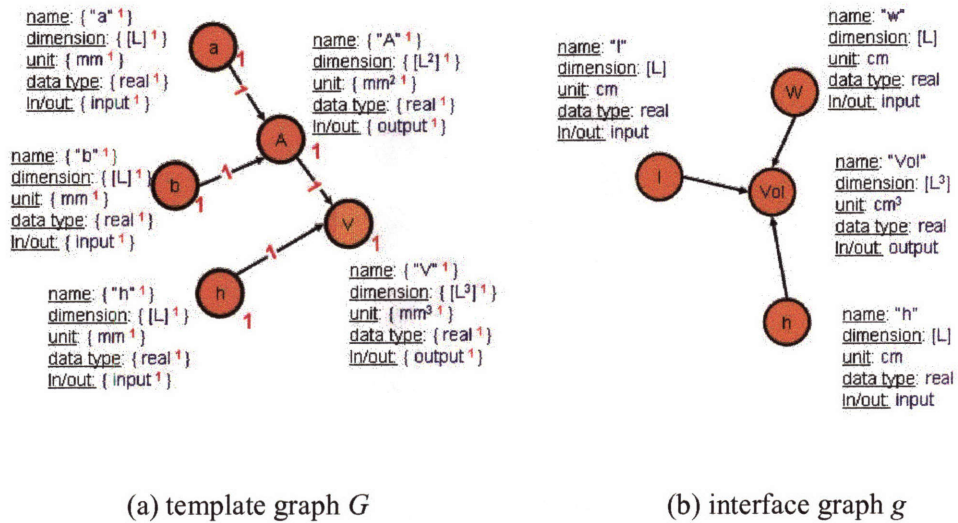
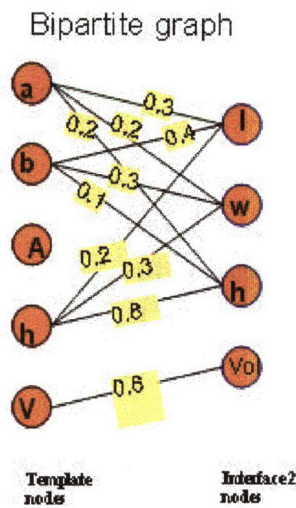


Figure 5-4 (a) (b) Matching a template graph and an interface graph

Table 5-4: node pair similarity

Template Node	Interface Node	Similarity
a	l	0.3
a	w	0.2
a	h	0.2
b	l	0.4
b	w	0.3
b	h	0.1
h	l	0.2
V	Vol	0.9

Based on the result of Table 5-4, corresponding bipartite graph for align the two graph can be generated. The template nodes go to the left list of the bipartite graph and interface nodes go the right list of the bipartite graph. For each row in the node pair similarity table, create a link (edge) on the bipartite graph between corresponding template node and interface node. The bipartite graph created for the purpose of node alignment is shown in figure 5-4(c).



**Figure 5-4 (c) Bipartite graph creation for matching an interface graph and a template graph**

Generated bipartite graph is shown in Figure 5-4(c). The nodes in the interface graph are brown and are the right side set of nodes in the bipartite graph. The nodes in the template graph are blue and are the left side set of nodes in the corresponding bipartite graph. The edges in the bipartite graph designate the consistency between interface nodes and template nodes. For example: edge (a, l) connecting left side node “l” with right side node “a”, indicate that interface node “l” is compatible with template node “a”, with a similarity score of 0.3.

### 5.3.5 Node Alignment Based on Bipartite Matching

Once the weighted bipartite graph is constructed, finding an optimal node alignment between original interface graph and template graph is a maximal bipartite matching task. In this work, a greedy bipartite matching method [71] similar to the example presented in presented in figure 5-2 was adopted to yield quick solution. Since the edges of the

bipartite graph represent node pair-wise similarities, they can be ordered from highest similarity score to lowest similarity scores. The greedy bipartite matching method will always try to pick an edge with highest similarity score possible.

A detailed algorithm description of the method is listed below as Algorithm 5-2.

**Algorithm 5-2: Greedy bipartite matching**

Input: an ordered list of node pairs, (edges  $E$  of the bipartite graph)

$M = \emptyset$ ;

While  $E(G) \neq \emptyset$ :

    Pick the first  $e \in E(G)$

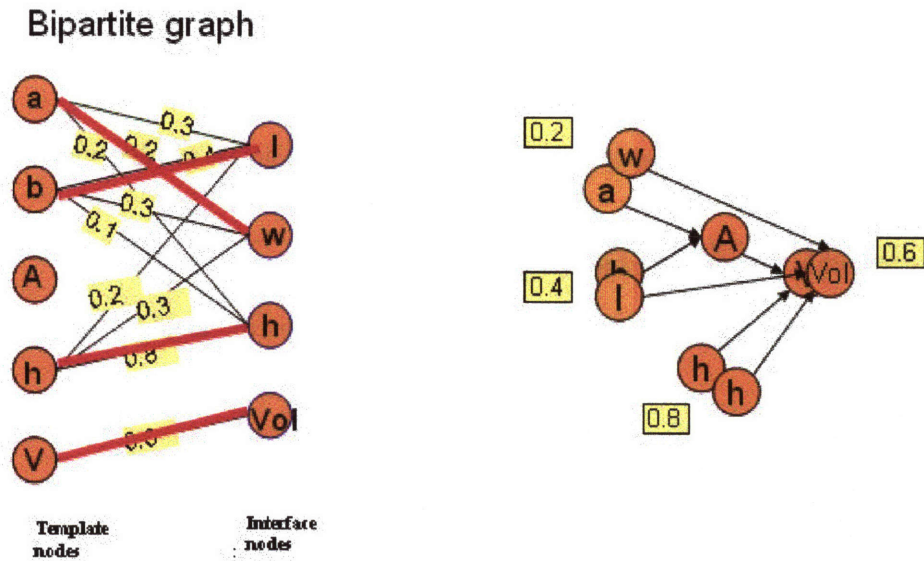
$M = M \cup \{e\}$

    Remove  $e$  and all other edges in  $E(G)$  that are adjacent to  $e$

Repeat;

Output: a matching  $M$

For example, Figure 5-5 shows a maximal matching solution obtained by greedy bipartite matching algorithm 5-2 on the bipartite graph created in Figure 5-4. Given an order for the edges, a maximal matching can be found by always picking the highest ranked edge available. In our case, the edges are node pair-wise similarities; therefore, they can be ordered from highest similarity score to lowest similarity scores.



(a) A maximal matching of the weighted bipartite graph in figure 5-4(c). (b) Node alignment for the graph matching task in figure 5-4 (a) and (b)

**Figure 5-5 Using bipartite matching to obtain node alignment**

### 5.3.6 Refining Node Alignment by Aligning Arcs

The result of bipartite matching algorithm provides a node alignment based on node pair-wise similarities. Since the designed algorithm only uses nodes information to create the bipartite graph, the structural information (arcs) are completely ignored during matching, the solution could be error-prone if the structural information plays a significant role in graph matching.

A possible improvement to address this problem could be to embed arc information as additional node attributes. This approach has been proposed by Kim in 1991 [34],

where discrete relaxation technique incorporated into bipartite matching algorithms to “absorb” arc information during node alignment.

In this work, as a primary refinement, we proposed to check the arc consistency of initial node alignment, and dump those node pairs without matching neighboring arcs. Given a node alignment obtained from bipartite matching, algorithm 5-3 further checks if there are any matching arcs based on the node alignment. This procedure is important since it will reinforce any structural alignment that might exist.

**Algorithm 5-3: Refine node alignment**

Input: a node-wise alignment M between template graph T and interface graph G,

For each node pair with connecting arcs

- Check if there is any real arc can be aligned;
- Check if there is any “virtual” arc can be aligned.

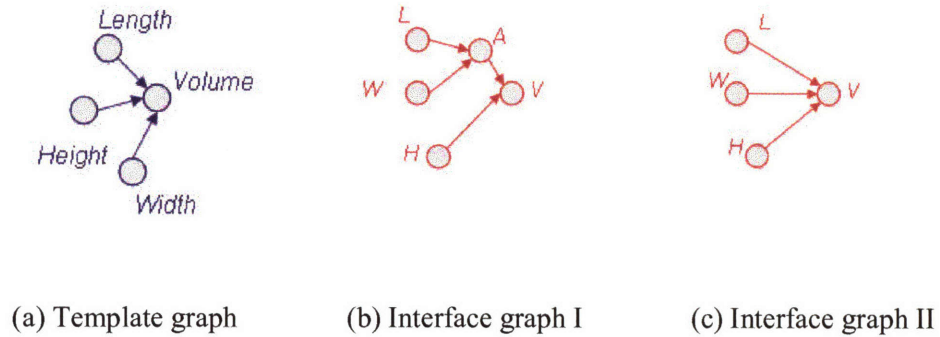
Removing those node pairs that don’t belong to any aligned arc or virtual arc from M

Output: refined node-wise alignment M’

A “virtual” arc is an indirect linkage between two nodes. It indicates an indirect causal relationship between the two nodes. The reason why indirect linkage between nodes is considered is because two interfaces with same functional role might be implemented differently or exposes intermediate parameters.

Figure 5-6 gives an example scenario of the importance of “virtual” arcs. The template shown in figure 5-5 (a) is a box volume calculator template, with length, width, and height as inputs and volume as the output. Interfaces shown in figure 5-5(b) and (c) are both volume calculators. In figure 5-6 (b), we can see 4 the real arcs connecting node

$L$  and  $A$ , node  $W$  and  $A$ , node  $A$  and  $V$ , and node  $H$  and  $V$ . Moreover, there are also 2 “virtual” arcs connecting node  $L$  and  $V$ , node  $W$  and  $V$ .



**Figure 5-6** An example scenario of the importance of “virtual” arcs

The interface graph in (c) can be easily matched with the template graph in (a) since they have the same topology. Interface graph in (b) contains an intermediate parameter  $A$  (the area). Without considering the hidden linkage, the only direct arc matched is the arc  $H$  to  $V$ . Although there is no direct arc between  $L$  to  $V$  and  $W$  to  $V$  in interface graph in (b), there exist “virtual” arcs connecting them. Thus, nodes  $L$  and  $W$  will also be included in the graph alignment, contributing to the overall similarity.

## 5.4 A Template-Interface Matching Example

This section provides a complete example, matching an interface to an existing template (Figure 5-7).

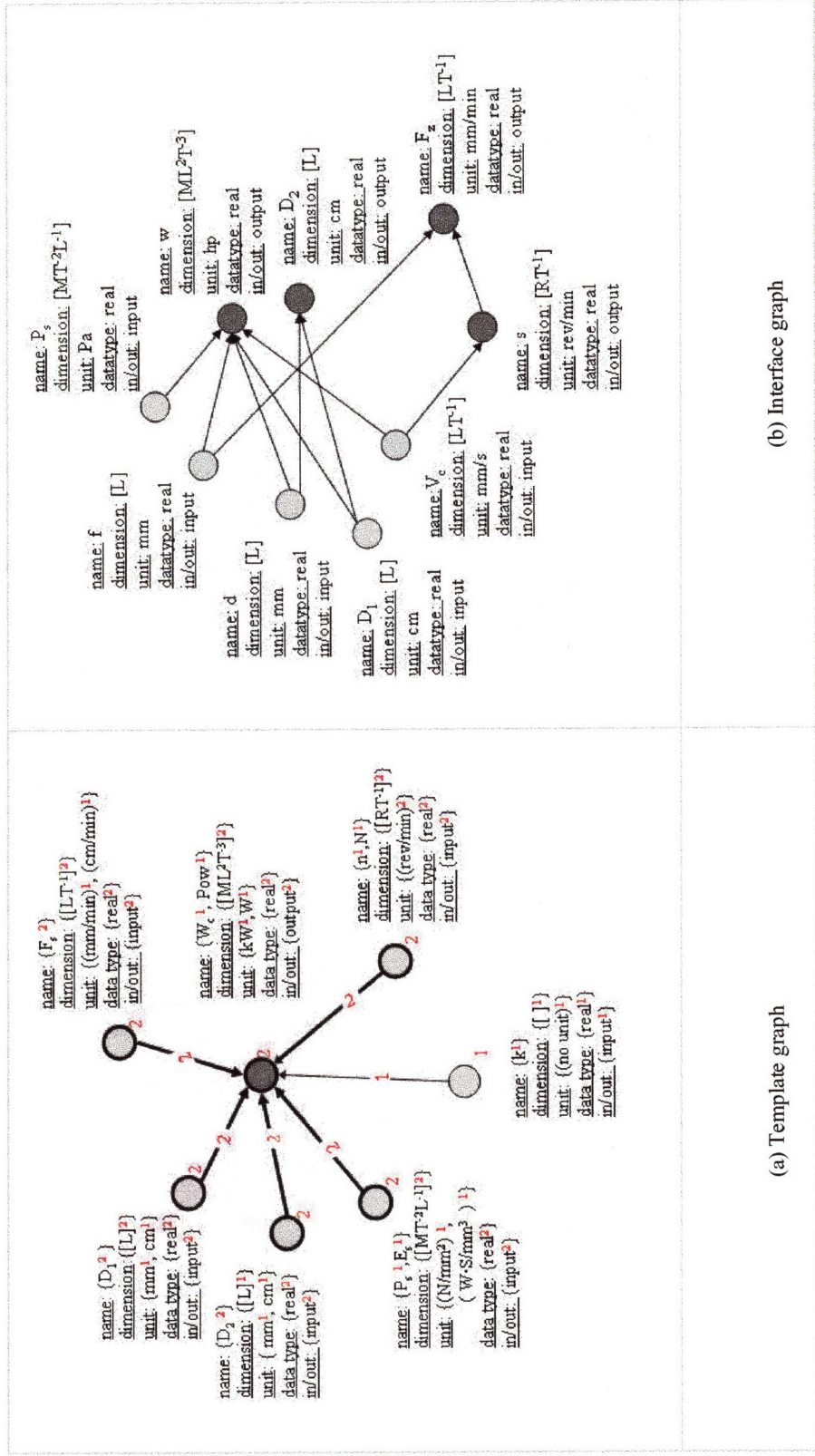


Figure 5-7 An interface graph (right) is compared with the cutting power template graph (left).

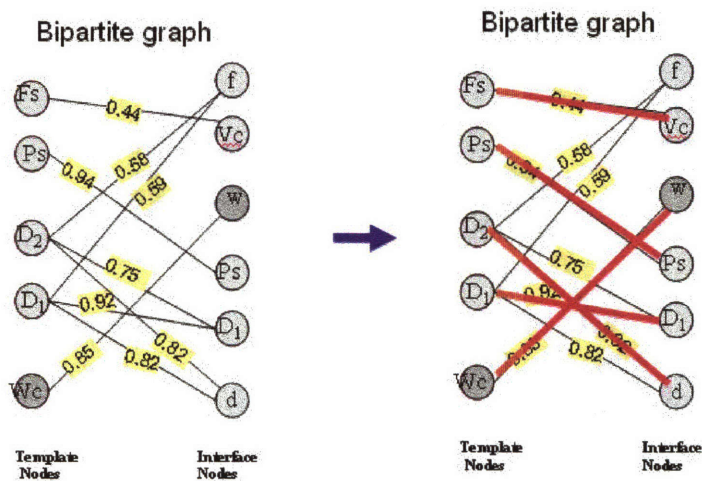
The template is the “cutting power consumption evaluator” template discussed in Chapter 4. The template graph is shown in Figure 4-3. The interface is a cutting power model that estimates power using cutting force and cutting velocity ( $W_c = F \times V_c$ ).

The graph matching algorithm begins with creating node pairs between the two graphs and evaluates the similarities for each node pair. Table 5-5 shows the ranked matching node pair list.

**Table 5-5 All non-zero similarity node pairs ranked according to the similarity scores.**

Template Nodes	Interface Nodes	Similarity Score
[P <sub>s</sub> , E <sub>s</sub> ]	P <sub>s</sub>	0.94
[D <sub>1</sub> ]	D <sub>1</sub>	0.92
[W <sub>c</sub> , Pow]	w	0.85
[D <sub>1</sub> ]	d	0.82
[D <sub>2</sub> ]	d	0.82
[D <sub>2</sub> ]	D <sub>1</sub>	0.75
[D <sub>1</sub> ]	f	0.59
[D <sub>2</sub> ]	f	0.58
[F <sub>s</sub> ]	V <sub>c</sub>	0.44

Then, we can apply the bipartite matching algorithm 5-2 to found out an initial node alignment (figure 5-8).



**Figure 5-8 Bipartite matching process of the matching task presented in figure 5-7.**



Table 5-6 shows the aligned node pairs list. Notice that the interface node s is not aligned with any template node and the template node k is not aligned with any interface node. They do not appear in Table 5-6 for this reason.

**Table 5-6 Aligned node pairs with the similarity scores.**

Template Node	Interface Node	Similarity Score	Template Node Weight
[D <sub>1</sub> ]	D <sub>1</sub>	0.92	2
[P <sub>s</sub> , E <sub>s</sub> ]	P <sub>s</sub>	0.94	2
[D <sub>2</sub> ]	d	0.82	2
[W <sub>c</sub> , Pow]	w	0.85	2
[F <sub>s</sub> ]	V <sub>c</sub>	0.44	2

After the aligned node pairs are determined, algorithm 5-3 refines the node alignment via finding real and virtual arc alignment. 4 aligns arcs that have been found are shown in Table 5-7. No virtual arcs are found in this case. Therefore, all five aligned nodes will contribute to the overall graph alignment.

**Table 5-7 Aligned arcs**

Template Arc	Interface Arc
[P <sub>s</sub> , E <sub>s</sub> ]-[W <sub>c</sub> , Pow]	P <sub>s</sub> --w
[D <sub>1</sub> ]-[W <sub>c</sub> , Pow]	D <sub>1</sub> --w
[D <sub>2</sub> ]-[W <sub>c</sub> , Pow]	d--w
[F <sub>s</sub> ]-[W <sub>c</sub> , Pow]	V <sub>c</sub> --w

Figure 5-9 shows the matching result of the template graph and the interface graph in figure 5-7. Overall, five aligned nodes and four aligned arcs are found in the graph matching process.

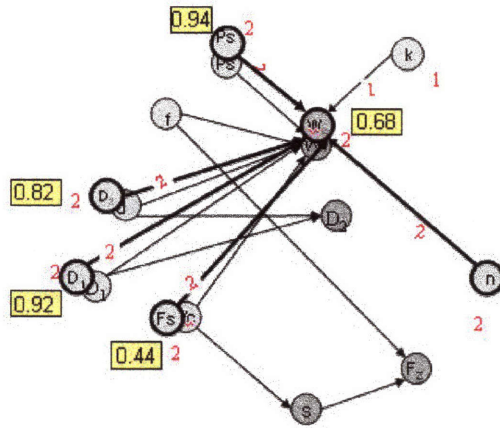


Figure 5-9 Aligned template and interface

Finally, the graph similarity score can be computed using Equation 5-5:

$$S = \frac{(0.92 \times 2 + 0.94 \times 2 + 0.85 \times 2 + 0.82 \times 2 + 0.44 \times 2)}{2 + 2 + 2 + 2 + 2 + 1} = 0.72$$

The similarity score by itself does not have meaning since it is a relative measure. However, when comparing a list of interface graphs to an existing template, the similarity scores will provide a ranking for how those interfaces are similar to the functional template.

# Chapter 6

## Template Creation and Generalization

### 6.1 Introduction

A template is a synthetic data structure that used to represent patterns associated with a particular functional role. When a functional role is known for a given web-enabled application, or simulation model, its interface may be asserted as representative for such functional role. Assertion, which is always by a person, gives origin to a template — a reference pattern for the known functional role.

This thesis proposed a fuzzy template concept. Initially, the template is initialized with a single example interface. By aggregating similar model interfaces into one template, we can obtain a more generalized template that emphasizes desired patterns while still including less frequent but relevant patterns, and de-emphasizing less relevant artifacts that may have been present in an exemplar but did not pertain to the functional role of interest. Since a template represents certain functional role that is not explicitly defined and associated with a set of similar example interfaces. It can be seen as a fuzzy functional concept. Each example interfaces can be seen as members of the fuzzy set, and the degree of membership is the similarity score between each example interface and the template.

The template graph representation is formally presented in Chapter 4 section 3. The template graph is a weighted fuzzy ARG, where nodes and arcs themselves are weighted with frequency numbers, and node attribute values are fuzzy sets. The advantage of the representation is that allows a template to generalize patterns capturing information found in several instances of functionally similar models.

This chapter will explain the aggregating mechanism: how to create an initial template seeding and how to subsequently aggregate new information (or learning). Currently, the decision of when to create and generalize templates for a given functional role will be made by a user. The possibility of using incremental clustering approaches to provide some feedback to assist the user in such decisions is discussed future work section of chapter 8.

## **6.2 Seeding a Template**

Initially, when a user asserts a model interface to be an example of a certain functional role, information from that interface is used to initiate a template. Figure 6-1 is an example of seeding an initial template from an example interface. The template graph is a fuzzy ARG that looks much like the original interface ARG, except that nodes, node attributes, and arcs are associated with frequency information. Since the seeded template is based on only one example, all frequency information in the template has the value of 1.

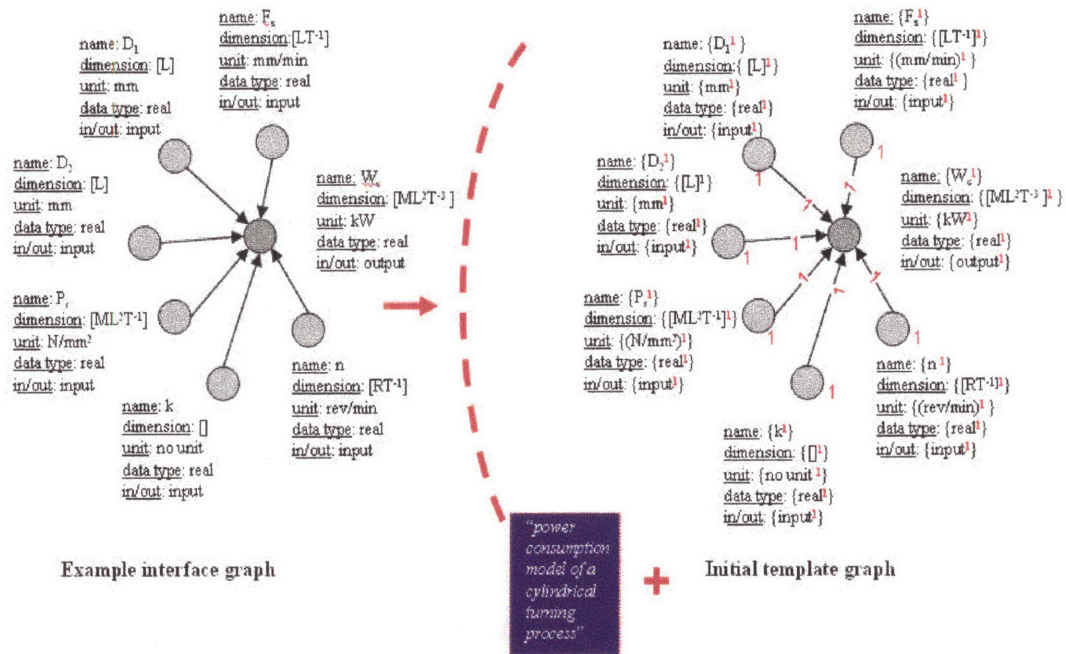


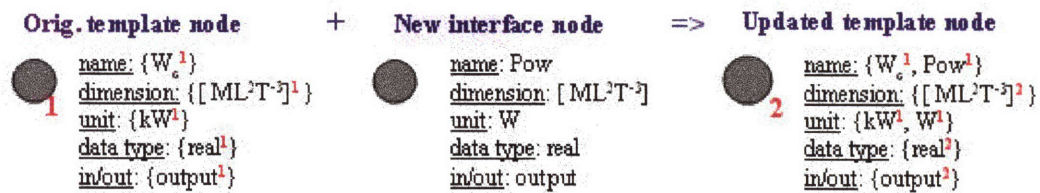
Figure 6-1 Seeding an initial template (right) from an example interface (left).

A template also contains auxiliary information such as descriptions of the functional role it represents, such as “power consumption model of a cylindrical turning process” as shown in figure 6-1. Also, version history is kept for each template about the exemplars that have been used to create the template.

### 6.3 Generalizing a Template

After a template is seeded, it can be adapted or generalized using information from other interfaces with a similar functional role. In order to aggregate new information into a template, a graph merging algorithm for Fuzzy ARGs based on node alignment was developed.

Graph merging begins by matching the template graph with the graph of the functionally similar interface that is to be added to the template. Details of the similarity-based matching process were presented in Chapter 5. The graph matching algorithm aligns equivalent nodes between the two graphs and determines which node-arc-node pairs are equivalent. Based on the alignment, equivalent nodes and arcs are merged together. Then the values from new interface attribute are added into attribute fuzzy value sets in the template and all frequency numbers are updated as appropriate. Figure 6-2 illustrates how the information from a node in the interface is merged into a corresponding template node



**Figure 6-2 Merging equivalent nodes during template generalization.**

In Fig. 6-2, an interface node named “Pow” (middle) is merged into the original template node “W<sub>c</sub>” (left), yielding updated template node “W<sub>c</sub>” (right). During the merging, for each node attribute, the interface value is added into corresponding template fuzzy set. If the fuzzy set already contains a member equal to the interface attribute’s value, the frequency of that member is increased by 1. Otherwise, the interface value is added into the template fuzzy set as a new member with frequency of 1. For example, when updating the unit attribute, the new interface unit is “W”, which is not contained in the template unit fuzzy set  $\{kW^1\}$ . Therefore, “W” is added as a new member with a frequency number of 1. The updated unit fuzzy set is  $\{W^1, kW^1\}$ . For the data type

attributes, since the new interface value “real” is already contained in the template data type fuzzy set  $\{real^1\}$ , one simply increases its frequency by 1. The updated fuzzy set then is  $\{real^2\}$ . Finally, the frequency number for the node is increased to 2.

After merging equivalent nodes, equivalent arcs can be merged based on the alignment of node-arc-node pairs. Arc merging is simpler since arcs do not have attributes. Thus, if an arc in the new interface matches an arc in the template, the template arc’s frequency number is increased by 1.

Equivalent nodes and arcs are merged together as indicated by increased frequency numbers, thereby strengthening their presence in the template. Non-aligned interface nodes and arcs are also added into the template graph with a frequency number of 1.

Figure 6-3 illustrates the template generalization process. The original template is the cutting power template from Fig. 6-1. Now a second, functionally equivalent but differently implemented service with the same functional role. This second interface is used to calculate cutting power consumption during turning process, but is implemented differently using the mathematical model  $W_c = E_s \times MMR$ , shown in Fig. 6-3 (a). The original template is shown in Fig. 6-3 (b), and the updated aggregated template with new information is shown in Fig. 6-3(c).

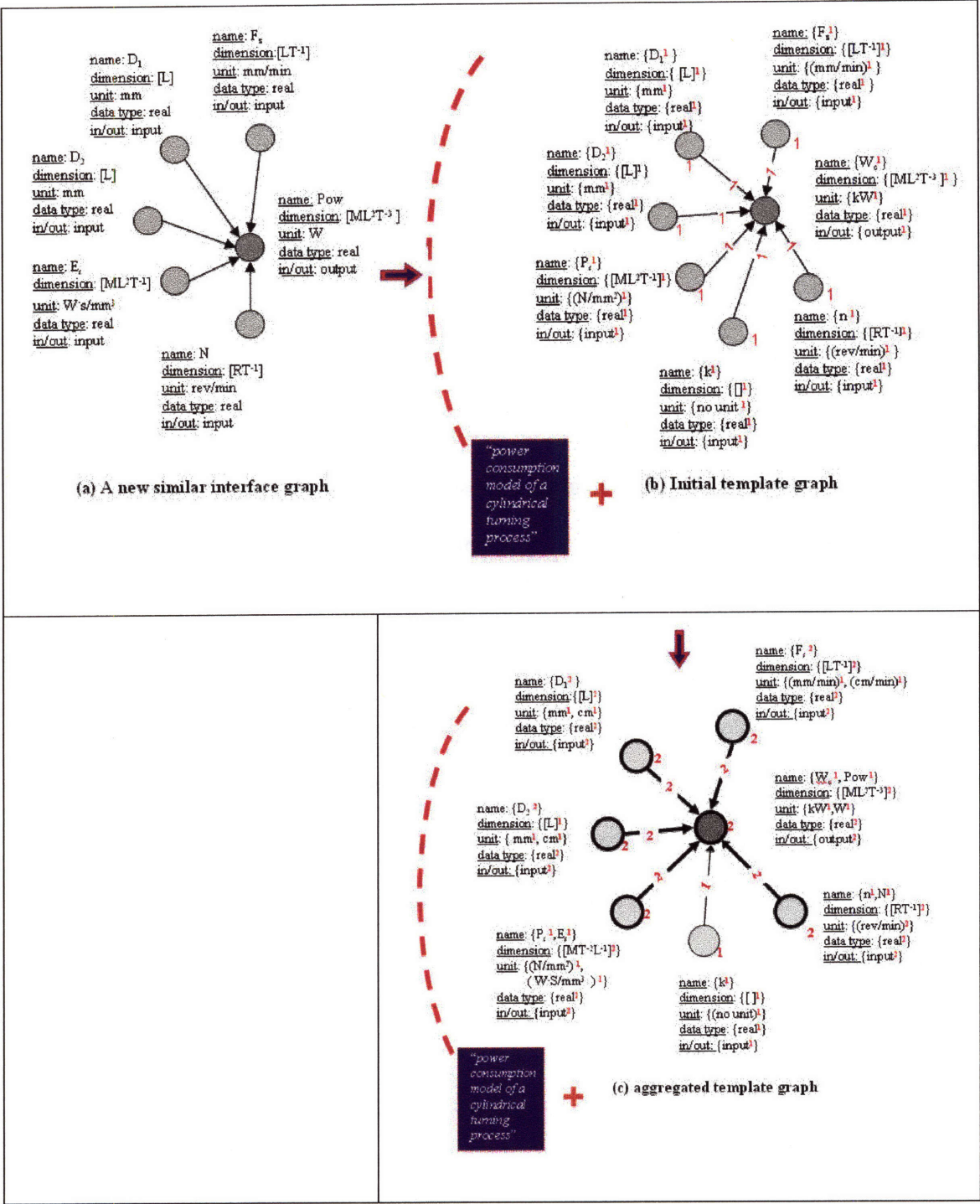


Figure 6-3 Template generalization example.



# Chapter 7

## Validation and Results

### 7.1 Prototype System Design

#### 7.1.1 Search Engine Framework

A traditional search engine usually contains software agent (“crawler”) that can crawls the WWW and indexes web pages, storing the indexed information in databases. Users usually go to the homepage of a search engine web site, type in a keyword, and the search engine will apply string-matching algorithms to indexed documents and computing a ranked list of results and display them in a webpage. The ranking mechanism would involve complicate link analysis, frequency counting, and elementary structural analysis such as weighted title words, such as Google search engine [75].

A function-based search engine would work similar ways. But the similarity analysis and ranking will be performed in the graph theoretical framework developed in this thesis. Figure 8.1 shows the framework design of the expected search engine, combining both indexing and searching (left) and pattern management part. The search engine contains several key modules:

- **Crawler:** responsible of crawling numerous simulation interfaces on the WWSW to collect information;

- **Indexer:** based on the result of pattern management module, index interfaces and store them in the collections;
- **Searcher:** understand the query and search in the collections; display search results;
- **Pattern matching module:** converting interface into graph models; conducting similarity matching between interface graphs and pattern graphs.
- **Pattern management module:** interact with human experts to create or manage templates.

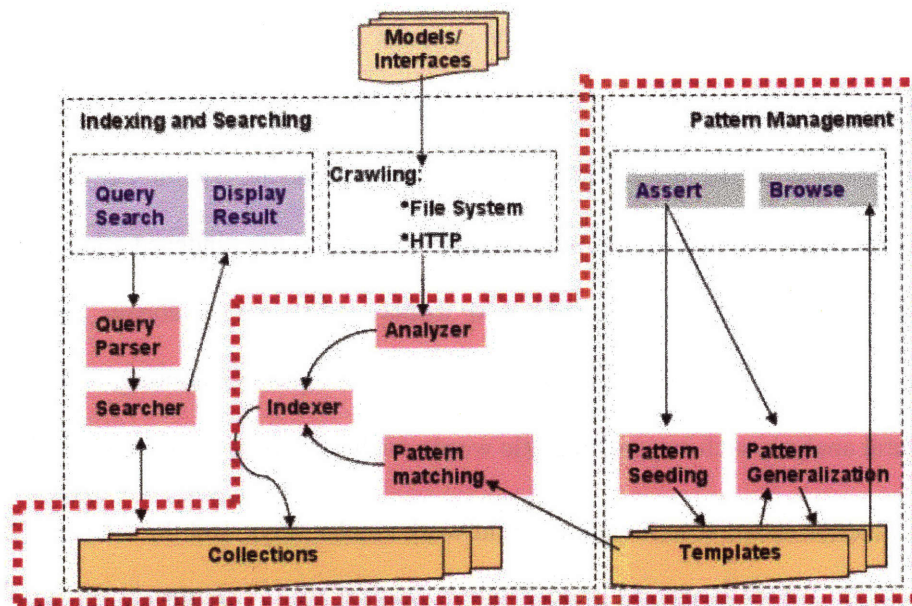


Figure 7-1 Search engine framework design ( parts marked within the thick red dotted lines are implemented in the thesis)

In figure 7-1, parts marked within the thick red dotted lines are implemented in thesis as a prototype system to validate the concept of similarity-based template-matching method.

## 7.1.2 Modules of the Prototype System

A prototype system was developed in pure JAVA. There are four core modules and one testing module. The core modules include:

- **Data structure:** classes for graph representation, such as directed graph, attributed node, ARG, fuzzy ARG, and other basic data structure classes.
- **Graph matching:** algorithms for creating bipartite graphs, bipartite matching using the greedy bipartite matching algorithm and the Hungarian algorithm, and ranking algorithm for result,
- **Similarity analysis:** similarity measure defined for attributes, nodes, arcs and graphs. Several open source Java APIs have been embedded in this module. An open-source Java-based package of approximate string-matching techniques ( SecondString [68] ) is used for name similarity analysis. UCUM [69], a unit conversion Java toolkit whose syntax for units of measures is based on the standard ISO 2955, is used for unit conversion during the unit similarity analysis. Besides attribute similarities, algorithms for calculating node pair-wise similarity and overall graph similarity are also implemented in this module.
- **Framework:** this module contains 1) utility classes be called by search agent that processes interfaces files, load and save to disks, indexing, searching and etc; 2) template management classes for saving, loading templates.3) GUI classes for visualization graphs.

Based on prototype system, a GUI tool for similarity assessment between a template and an interface is also developed (figure 7-2). The GUI tool loads a template and an



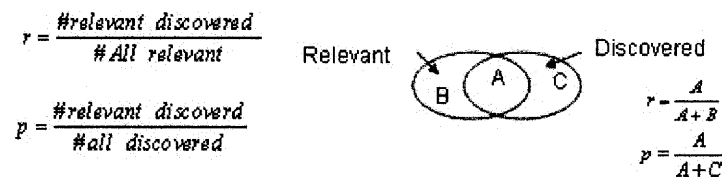
## 7.2 Validation and Results

### 7.2.1 Goal

A series of tests on real-world models and applications have been conducted in the thesis. The goal is to validate the similarity matching approach and the template generalization idea.

### 7.2.2 Metrics to Evaluate Results

To obtain quantitative evaluation of performance, the quantities precision  $p$  and recall  $r$  are introduced (figure 7-3). Precision and recall are basic measures in the field of information retrieval field. Precision indicates the proportion of relevant document to all the documents retrieved; Recall is the proportion of relevant documents that are retrieved, out of all relevant documents available [76].



**Figure 7-3 Definition of precision and recall**

Precision can be seen as an indicator of matching accuracy, while recall indicates the matching completeness.

### 7.2.3 Test Data Set

The test dataset consists of 57 different web-enabled interfaces belonging to models from an online energy-related modeling toolkit “Alternative Energy Toolkit” [77] (<http://cadlab.mit.edu/altenergy>).

**Table 7-1 57 Interfaces and their underlying models**

Index	Interface Name	Model Name
1	load profile Interface	Electricity load profile
2	extraterrestrial radiation estimates Interface	Extraterrestrial radiation
3	monthly boiler fuel cost - complete Interface	Fuel cost
4	solar time Interface	sun-earth geometric relationship
5	boiler sizing - complete Interface	Boiler sizing
6	annual boiler fuel cost - complete Interface	Fuel cost
7	Sunrise-sunset local standard times Interface	solar angle
8	daily roof-wall cooling energy - complete Interface	Roof & wall cooling energy
9	fenestration heat gains - complete Interface	24-hr profiles of fenestration heat gain
10	daily transmission heating energy - complete Interface	Transmission heating energy
11	PV operation - simple Interface	PV array operation
12	cooling-heating degree-hours - complete Interface	Degree-hours
13	flat panel collector - complete Interface	Efficiency of flat panel collector
14	inverter operation with input losses Interface	Inverter operation
15	inverter operation with loss coefficients Interface	Inverter operation
16	ventilation heat loss - complete Interface	Ventilation heat loss
17	PV module LCEA - complete Interface	PV module life-cycle energy analysis
18	Operation parameters of PV with MPP tracker Interface	PV module operational characteristics
19	Operation parameters of PV without MPP tracker Interface	PV module operational characteristics
20	lead-acid battery - complete Interface	PV system lead-acid battery
21	daily infiltration heating energy - complete Interface	Infiltration heating energy
22	simplified diesel generator energy analysis ?complete Interface	Diesel generator energy analysis
23	air-conditioner electricity - complete Interface	AC electricity load
24	fenestration cooling load - complete Interface	24-hr profiles of fenestration cooling loads
25	PV controller - complete Interface	PV system controller operation
26	engine-generator life-cycle costing - complete Interface	Engine-generator system life-cycle costing
27	complete solar-time-based Interface	sun-earth geometric relationship
28	local standard time Interface	sun-earth geometric relationship
29	profile of internal cooling loads Interface	24-hr profiles of internal cooling loads
30	global irradiance on inclined surface Interface	Global irradiance on inclined surface
31	global irradiance on inclined surface - with anisotropic reflections Interface	Global irradiance on inclined surface
32	outdoor air temperature - complete Interface	sol-air temperature-Matlab
33	solar altitude and azimuth Interface	sun-earth geometric relationship
34	outdoor air temperature - complete Interface	sol-air temperature
35	profile of solar radiation on tilted surface - complete Interface	Solar radiation on inclined surface
36	incident angle profile Interface	Solar radiation on inclined surface
37	PV load breakdown - complete Interface	PV system load breakdown
38	NPV saving - complete Interface	Net-present-value of saving
39	simplified PV module energy analysis ?complete Interface	PV module energy analysis
40	roof-wall conduction cooling load - complete Interface	cooling load
41	electricity and energy emission - complete Interface	NOx, SO2, CO2 emissions from fuel consumption
42	transmission heat losses - complete Interface	Transmission heat loss
43	Complete solar angles Interface	solar angle
44	daily fenestration cooling energy - complete Interface	Fenestration cooling energy
45	PV Life-cycle costing - complete Interface	PV system life-cycle costing
46	daily internal cooling energy Interface	daily internal cooling energy
47	CO2 emission - complete Interface	CO2 emission from electricity generating systems
48	daily ventilation heating energy - complete Interface	Ventilation heating energy
49	top heat loss - complete Interface	Top heat loss of flat panel collector
50	PV operation - complete Interface	PV array operation
51	Solar Declination Interface	solar angle
52	sunrise-sunset solar times Interface	sun-earth geometric relationship
53	service water heating - complete Interface	server water heating
54	infiltration heat losses - complete Interface	Infiltration heat loss
55	sol-air cooling-heating degree-hours - complete Interface	sol-air cooling-heating degree-hours
56	solar Declination Interface	sun-earth geometric relationship
57	sunrise-sunset local standard times Interface	sun-earth geometric relationship

The underlying models (totally 39) cover common concepts in energy transfer, power generation/consumption, energy cost, HVAC, heat transfer, and environmental impact of energy technologies. These models and interfaces are implemented in DOME.

The interfaces and its underlying models are listed in table 7-1. On average, these interfaces contain about 10~15 input/output parameters; while the larger interfaces involve over 60 parameters.

In order to evaluate the search result, these 57 interfaces were manually categorized into 10 categories by a human expert, shown in Table 7-2. The manually categorization is performed by the creator (models builder) of the “Alternative energy toolkit”. The classification is assumed to reflect crisper and vague concepts. For better visualization, each category is marked with a unique color.

**Table 7-2: 57 interfaces are manually categorized into 10 categories.**

Functional Roles		Interfaces
# 1	Environment	41, 47
# 2	Economic	3,6,23,26, 38,45
# 3	Energy	8,10,17, 21, 22,39, 44, 46,48, 53
# 4	Load demand	1, 37
# 5	Solar radiation	2,4,7,27,28,30,31,33,35,36,43,46,52,51,56,57
# 6	Surrounding condition	12, 32,34,55
# 7	System component	5,14,15,20, 25
# 8	Photovoltaic	18,19, 11,50
# 9	Solar thermal	13,49
#10	Building heat transfer	9, 16, 24, 29, 40, 42, 54

## 7.2.4 Test 1: Test Similarity Matching Approach

### 7.2.4.1 Goal

The first experiment was designed to validate the similarity matching approach. The experiment checks whether similar interfaces found by the algorithm are useful and functionally similar (or relevant) according to a human experts' opinion.

### 7.2.4.2 Scenario

The testing scenario is described below:

1. Make templates from single interfaces. For each individual interface, create an initial template. This initial template can be seen as a replication of the original interface. Therefore, searching using an initial template is equivalent to searching with a single exemplar interface. All templates are indexed the same number as its exemplar interface. For example, template created from interface # 4 is indexed as template # 4.
2. Compute similarity for all 57 interfaces.
3. Keep matches with similarity score  $\geq 0.1$ , this is to add a cut-off threshold to the result.
4. Calculate  $r$  and  $p$  for each retrieved set.
5. Average  $r$  and  $p$  for all templates belonging to the same functional role.

### 7.2.4.3 Results

The search results are given in figure 7-4: categories with higher recalls tend to have lower precisions.



	Functional Roles		Average Recall	Average Precision
#1	Environment		100.00%	100.00%
#2	Economic		74.34%	69.44%
#3	Energy		31.00%	71.81%
#4	Load demand		100.00%	27.78%
#5	Solar radiation		36.89%	99.17%
#6	Surrounding condition		50.00%	70.00%
#7	System component		60.00%	45.40%
#8	Photovoltaic		93.75%	54.22%
#9	Solar thermal		50.00%	18.06%
#10	Building heat transfer		75.51%	60.36%

**Figure 7-4 Average precision and recall of single-interface templates**

The detailed search result is visualized in figure 7-5. Templates and interfaces belonging to a same category have the same category color. For example, the first two rows are search results of templates 41 and 47 belonging to category #1 (Environment), marked in brown color. Similar interfaces found for that template are listed row-wise in the result columns based on relative similarity score from high to low. For example, in the first row, similar interfaces found for template 41 are interface 41 and interface 47, with interface 41 being more similar to the template than interface 47.

Figure 7-5 shows that for most templates, similar interfaces found by the algorithm are in the same category — meaning the algorithm search result emulates human expert’s categorization. Also the top ranked interface in search result of each template is the original interface generated that template— as a validation of trivial case of graph matching algorithm.

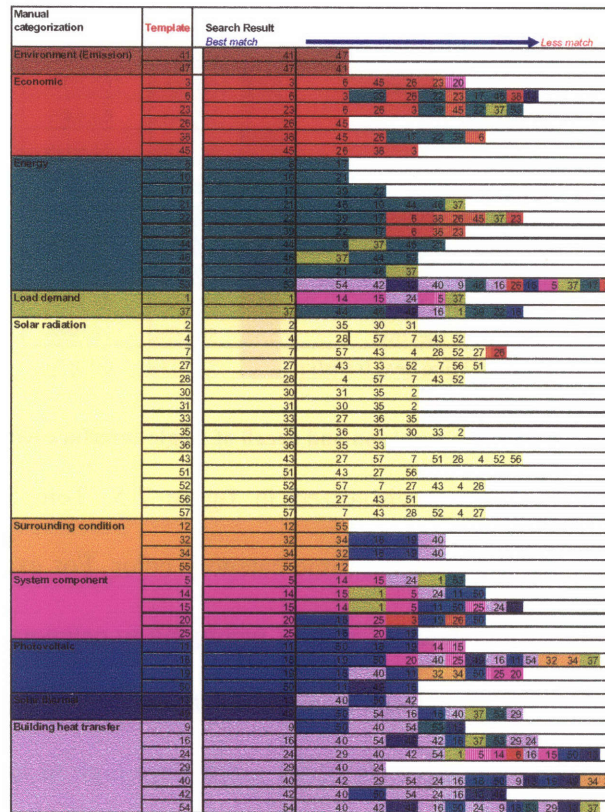


Figure 7-5 Visualization of single-interface templates search result

The algorithm execution speed was also noted. The experiment involved a total of 3249<sup>4</sup> graph comparisons and required 8.9 seconds on a 1.6GHz Dell 600M Inspiron with 1G RAM.

This test demonstrates the efficacy of heuristic fuzzy similarity functions and matching algorithms. However, there are some categories that have more discrepant than others categories. For example, the two templates belonging to category “solar thermal” didn’t find same-category interfaces except the interfaces that were used to create the template. We studied these interfaces and found that they were functionally equivalent in terms of input and output parameters characteristics:

<sup>4</sup> 3249=57\*57

- Interface 13 belongs to a model that calculates the fin efficiency, collector efficiency and useful heat transfer of flat-panel collector, based on the geometries of the collector, operation condition, and the working fluid properties.
- Interface 49 belongs to a model that calculates a flat-panel collector's top heat loss by takes into account both convective and radiative losses, and requires a number of input parameters, including the absorbing plate's temperature, area and emissivity, the number of covers and their emissivity, the collector's tilt angle, and the ambient temperature [78].

After consulting with the expert who categorized the interfaces his opinion was that, although the two models are not functionally similar, they were both relevant interfaces to dealing with two different solar thermal problems involving flat-panel collectors.

This highlights an interesting issue — the ambiguity of the meaning of “similar”. In this work, this ambiguity is addressed by fuzziness incorporated in the template design. A template represents a fuzzy concept of functional role, reflected by a discrete set of exemplar interfaces with similar functional purposes. The observation leads to the next experiment, which can be seen as a validation of the template generalization idea.

## **7.2.5 Test 2: Test Template Generalization Idea**

### **7.2.5.1 Goal**

The second test is to validate the template generalization scenario: whether a generalized template generated by graph merging method is capable of better retrieving a wider set of functionally relevant interfaces.

A generalized template represents a fuzzy functional concept that is learned from a set of functionally equivalent/similar interface exemplars. In this experiment, we assert that multiple interfaces from each manually defined category together represent a fuzzy functional concept, such as “emission”, “solar radiation”, “building heat transfer”, etc. Then, a template is generated by merging these interfaces.

### 7.2.5.2 Scenario

The testing scenario is described below:

1. Make generalized templates with all the interfaces for a given functional role. The maximum generalized template is generated using graph merging algorithm explained in Chapter 6. The first interface of each category is used to initialize the template. Then, the rest interfaces of this category are merged into the initial template. For example, a template of category #1 “Emission” is initialized from interface 41; and generalized by merging interface 47.
2. Compute similarity for all 57 interfaces.
3. Keep matches with similarity score  $\geq 0.1$ , this is to add a cut-off threshold to the result.
4. Calculate  $r$  and  $p$  for each retrieved set.
5. Average  $r$  and  $p$  for all templates belonging to the same functional role.

### 7.2.5.3 Results

The maximum generalized templates search results are given in figure 7-6, where we can see as recall reaches 100%, precision tends to go down (but not in all cases).

Functional Roles		Recall	Precision
#1	Environment	100.00%	100.00%
#2	Economic	100.00%	66.67%
#3	Energy	100.00%	76.92%
#4	Load demand	100.00%	13.33%
#5	Solar radiation	100.00%	100.00%
#6	Surrounding condition	100.00%	57.14%
#7	System component	100.00%	38.46%
#8	Photovoltaic	100.00%	36.36%
#9	Solar thermal	100.00%	33.33%
#10	Building heat transfer	100.00%	58.33%

Figure 7-6 Average precision and recall for generalized templates

A visualization of the detailed search result is shown in figure 7-7. We again use color to distinguish between different manual categories. Each row listed interface from “best match” to “less match”.

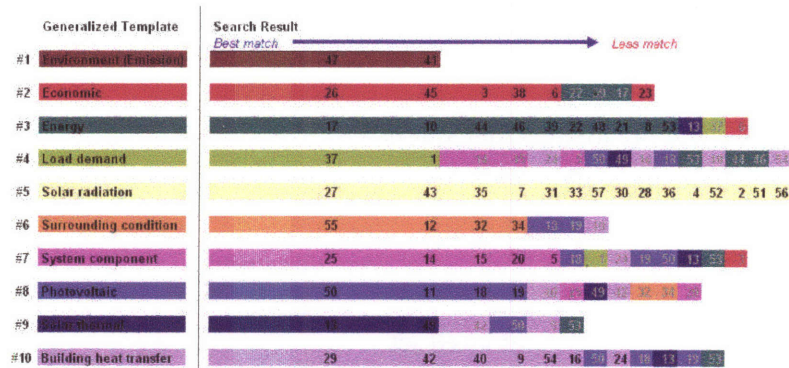


Figure 7-7 Visualization of generalized templates search result

One would expect all in each category to be found because each template contains all the information about the category. The result in Figure 7-3 demonstrates that generalized template of each category can retrieve all the interfaces belonging to that same category. For a category # 5 (Solar radiation), the result has 0 miss-categorized results. The result in Figure 7-3 validates that a generalized template can retrieve more functionally similar

interfaces. Future more, the model interfaces from the same category are top hits; it demonstrates that the right patterns are emphasized.

## 7.2.6 Test 3: Recall and Precision as a Function of The Template's Degree of Generalization

### 7.2.6.1 Goal

The goal of the third test is to explore the recall and precision changes as a function of the template's degree of generalization.

### 7.2.6.2 Scenario

The testing scenario is described below:

1. Make templates from increasing numbers of interfaces belonging to the same functional role.
2. Compute similarity for all 57 interfaces.
3. Keep matches with similarity score  $\geq 0.1$ , this is to add a cut-off threshold to the result.
4. Calculate  $r$  and  $p$  for each retrieved set.

### 7.2.6.3 Results

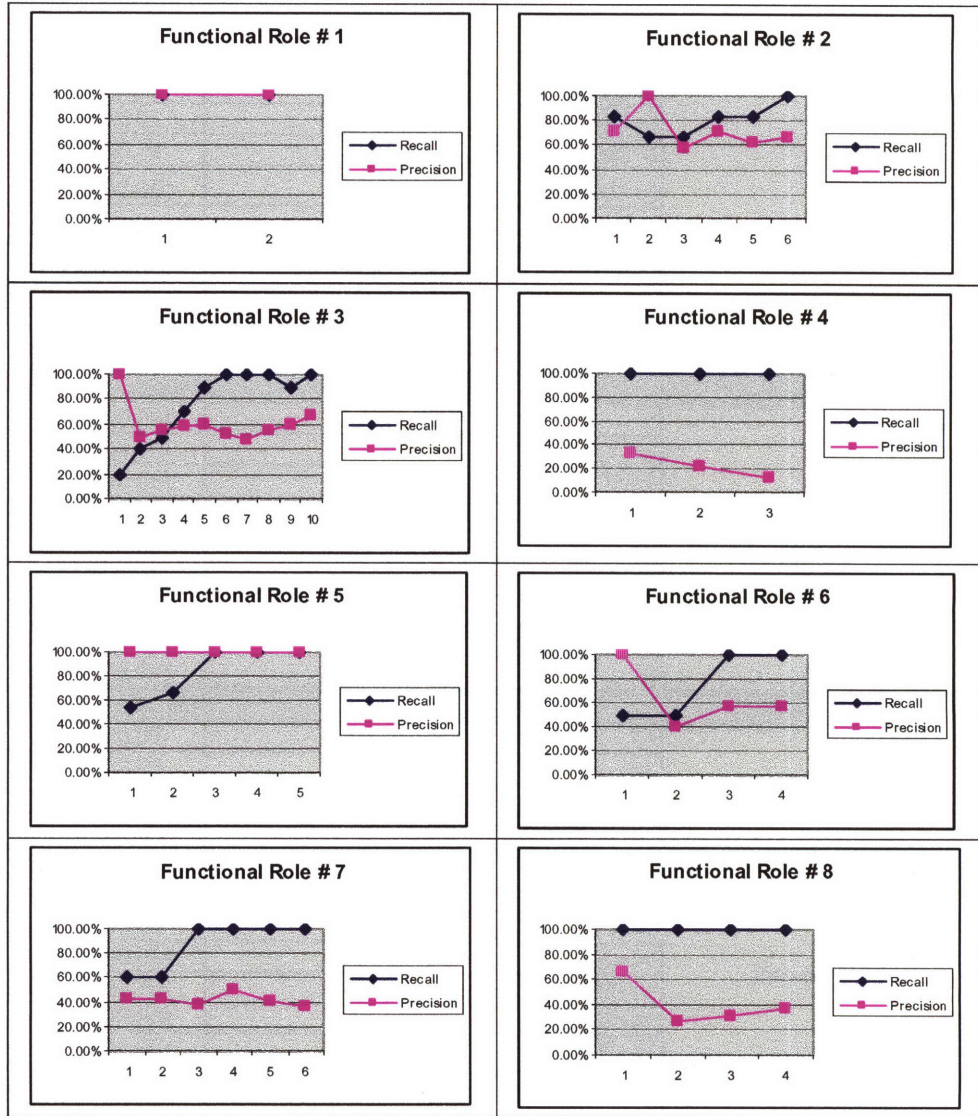
The recall and precision for searching using templates with incremental number of interfaces is shown in table 7-3 and figure 7-8.

**Table 7-3 Recall and precision for search using templates with incremental number of interfaces**

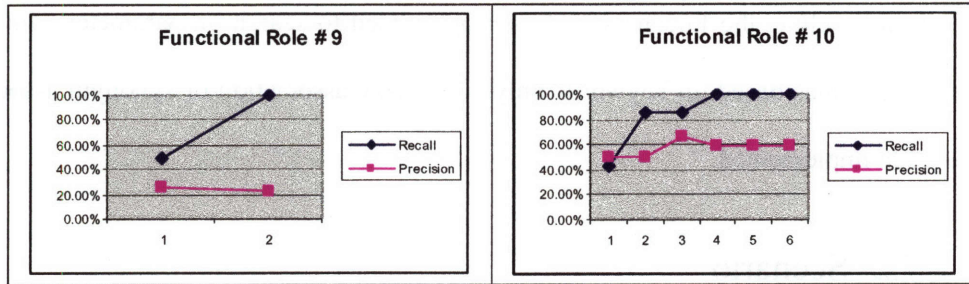
Functional Role #1		Recall	Precision
Environment	47	100.00%	100.00%
	47+41	100.00%	100.00%

Functional Role #2		Recall	Precision
Cost	3	83.33%	71.43%
	3+26	66.67%	100.00%
	3+26+6	66.67%	57.14%
	3+26+6+45	83.33%	71.43%
	3+26+6+45+38	83.33%	62.50%
	3+26+6+45+38+23 (all)	100.00%	66.67%
Functional Role # 3		Recall	Precision
Energy	8	20.00%	100.00%
	8+39	40.00%	50.00%
	8+39+22	50.00%	55.56%
	8+39+22+44	70.00%	58.33%
	8+39+22+44+21	90.00%	60.00%
	8+39+22+44+21+53	100.00%	52.63%
	8+39+22+44+21+53+48	100.00%	47.62%
	8+39+22+44+21+53+48+46	100.00%	55.56%
	8+39+22+44+21+53+48+46+10	90.00%	60.00%
	8+39+22+44+21+53+48+46+10+17	100.00%	66.67%
Functional Role #4		Recall	Precision
Load Demand	1	100.00%	33.33%
	37	100.00%	22.22%
	1+37	100.00%	11.76%
Functional Role #5		Recall	Precision
Solar radition	27	53.33%	100.00%
	27+43	66.70%	100.00%
	27+43+35	100.00%	100.00%
	27+43+35+30	100.00%	100.00%
	all	100.00%	100.00%
Functional Role #6		Recall	Precision
surrounding condition	12	50.00%	100.00%
	32	50.00%	40.00%
	12+32	100.00%	57.14%
	12+32+55+34	100.00%	57.14%
Functional Role # 7		Recall	Precision
system component	14	60.00%	42.86%
	14+5	60.00%	42.86%
	14+20	100.00%	38.46%
	14+20+25	100.00%	50.00%
	14+20+25+5	100.00%	41.67%
	14+20+25+5+15	100.00%	35.71%
Functional Role # 8		Recall	Precision
Photovoltaic	11	100.00%	66.67%
	11+18	100.00%	26.67%
	11+18+19	100.00%	30.77%
	11+18+19+50	100.00%	36.36%
Functional Role # 9		Recall	Precision
Solar thermal	13	50.00%	25.00%
	13+49	100.00%	22.22%

Functional Role #10		Recall	Precision
Building heat transfer	9	42.86%	50.00%
	9+16	85.71%	50.00%
	9+16+29	85.71%	66.67%
	9+16+29+42	100.00%	58.33%
	9+16+29+42+40	100.00%	58.33%
	all	100.00%	58.33%







**Figure 7-8 Recall and Precision as a function of the template's degree of generalization**

By analyzing the result, we can see that: recall has a general trend to reach 100%, but the precision doesn't have a general trend; however in many cases the precision tends to increase up to a level and then fall.

## 7.2.7 Test 4: Compare With Keyword-based Text Search

### 7.2.7.1 Goal

The goal of the fourth test is to compare the performance of the proposed similarity-matching based search with keyword-based text search. The primary focus is to compare the accuracy (precision) and completeness (recall) of the search.

The text search tool used in the comparison is an open source package called Apache Lucene [79]. Lucene is a text search engine API, high-performance, and cross-platform. It is the guts of a search engine; we have the flexibility of writing our own code for UI and the processing of selecting and parsing input data files to pump them into a search engine.

As a baseline, the Lucene search is programmed to search on parameter names and interface names based on keywords only. No fuzzy association or synonym association has been implemented.

### 7.2.7.2 Scenario

The testing scenario is described below:

- Index the interface names and parameter names of all 57 interfaces by processing interface definitions files.
- Use selected keywords for each functional role to pull result out for the Lucene search.
- Calculate  $r$  and  $p$  for each retrieved set.

It's tricky to find a reasonable keyword to pull result out for the Lucene search. The keywords used are shown in the table 7-4. Sometimes the name of the functional role won't get any result, such as # 6 ("surrounding conditions"); "air" is used instead.

### 7.2.7.3 Results

**Table 7-4 Recall and precision of the keyword-based search for each category**

Functional role		Search results	# Retrieved	# Correct	Recall	Precision
# 1	Emission	"emission"	2	2	100%	100.00%
# 2	Economic	"cost"	6	6	100%	100.00%
# 3	Energy	"energy"	15	9	100%	60.00%
# 4	Load demand	"load"	10	2	100%	20.00%
# 5	Solar radiation	"solar"	23	13	86.70%	56.52%
		"solar radiation"	4	1	67%	25.00%
# 6	Surrounding condition	"air"	10	3	75%	30.00%
# 7	System component	"system"	2	0	0%	0.00%
		"component"	0			
# 8	Photovoltaic	"PV"	11	4	100%	36.36%
# 9	Solar thermal	"thermal"	2	1	50%	50.00%
		"solar thermal"	0			
# 10	Building heat transfer	"building"	4	1	12.50%	25.00%
		"heat transfer"	3	0	0.00%	0.00%

We compare the recalls and precisions in above result with the result from searching with generalized template (second test) in table 7-5. In addition, figure 7-9 (a) and (b) plot the recall and precision comparison as category bar charts.

In general, graph-based search has a better overall recall and precision than keyword based search. Keyword based search can't obtain 100% recall in all categories. And the precision is lower than graph-based search in many categories. In some categories (where the functional role name is rather abstract, e.g. system component), the keyword based search didn't get any result.

**Table 7-5 Comparison of recall and precision of the keyword-based search for each category**

Functional Roles	Keyword search		Generalized template search	
	Recall	Precision	Recall	Precision
Emission	100.00%	100.00%	100.00%	100.00%
Economic	100.00%	100.00%	100.00%	66.67%
Energy	100.00%	60.00%	100.00%	76.92%
Load demand	100.00%	20.00%	100.00%	13.33%
Solar radiation	86.70%	56.52%	100.00%	100.00%
Surrounding condition	75.00%	30.00%	100.00%	57.14%
System component	0.00%	0.00%	100.00%	38.46%
Photovoltaic	100.00%	36.36%	100.00%	36.36%
Solar thermal	50.00%	50.00%	100.00%	33.33%
Building heat transfer	12.50%	25.00%	100.00%	58.33%

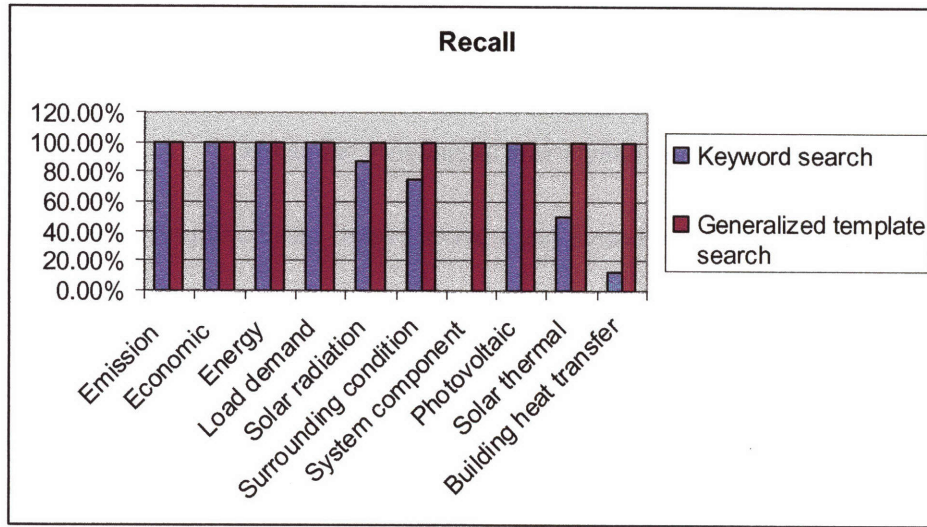


Figure 7-9 (a) Comparison of recalls

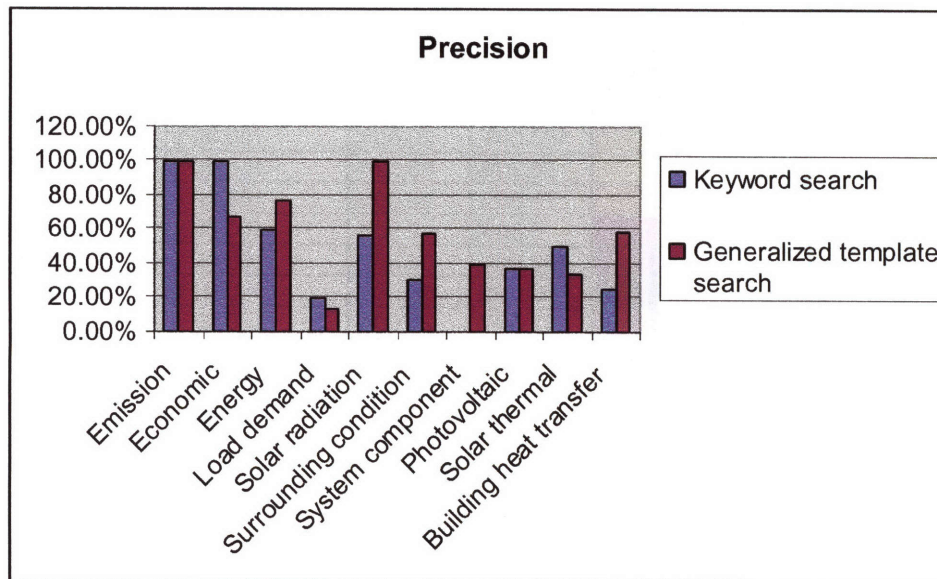


Figure 7-9 (b) Comparison of precisions

## 7.3 Additional test on a synthesized dataset: a cutting power consumption test suite

### 7.3.1 Test Data Set

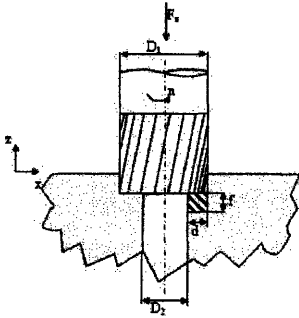
A small cutting model test suite (shown in Table 7-5) of 15 interfaces is synthesized to represent a number of challenging identification and discrimination cases. The test suite is made from three different types of model interfaces for cutting power consumption models for different machining processes. Interfaces in the test suite are sub-categorized by their types: “Type 1”, “Type 2” and “Type 3”, as shown in Table 7-6 column-wise.

The set is designed so that Type 1 and 2 interfaces have the same core modeling approach but, due to causality differences, have slightly different functional roles in terms of inputs and outputs. Type 3 interfaces have similar functional inputs and output to the Type 1 interfaces, but they were implemented using different core models. Thus, type 1 and type 3 are functionally almost equivalent types; while type 2 are different from type 1 and 3 from a functional standpoint, even though the similarity between type 1 and type 2 is a little higher than the similarity between type 1 and type 3, as type 1 and type 3 share a similar core modeling approach.

**Table 7-6 Cutting power consumption test suite [80]**

Cutting processes	Type 1	Type 2	Type 3
	$W_c = F \times V_c$ $V_c = n\pi D$	$W_c = F \times V_c$ $V_c$ is given	$W_c = E_s \times MMR$

	Cylindrical turning	<b>Inputs:</b> $P_s$ Pa $F_z$ mm/min $n$ rev/min $D_1$ mm $D_2$ mm  <b>Outputs:</b> $W_c$ kW	<b>Inputs:</b> $P_s$ Pa $a$ mm/rev $d$ mm $D_1$ mm $V_c$ m/s  <b>Outputs:</b> $P_{ow}$ W $F_z$ mm/min $n$ rev/min $D_2$ mm	<b>Inputs:</b> $E_s$ hp*s/in <sup>3</sup> $F_z$ in/min $N$ rev/min $D_1$ in $D_2$ in  <b>Outputs:</b> $w$ hp
	Radial turning	<b>Inputs:</b> $P_s$ Pa $F_x$ mm/min $n$ rev/min $d$ mm $D_1$ mm  <b>Outputs:</b> $W_c$ kW	<b>Inputs:</b> $P_s$ Pa $a$ mm/rev $d$ mm $D_1$ mm $V_c$ m/s  <b>Outputs:</b> $P_{ow}$ W $F_x$ mm/min $n$ rev/min	<b>Inputs:</b> $E_s$ hp*s/in <sup>3</sup> $F_x$ in/min $N$ rev/min $d$ in $D_2$ in  <b>Outputs:</b> $w$ hp
	Face milling	<b>Inputs:</b> $P_s$ Pa $F_x$ mm/min $n$ rev/min $d$ mm $D$ mm $Z$ no unit  <b>Outputs:</b> $W_c$ kW	<b>Inputs:</b> $P_s$ Pa $a$ mm/rev $d$ mm $D$ mm $V_c$ m/s $Z$ no unit  <b>Outputs:</b> $P_{ow}$ W $F_x$ mm/min $n$ rev/min	<b>Inputs:</b> $E_s$ hp*s/in <sup>3</sup> $F_x$ in/min $N$ rev/min $d$ in $D$ in $n$ no unit  <b>Outputs:</b> $w$ hp
	Drilling (drill a hole)	<b>Inputs:</b> $P_s$ Pa $F_z$ mm/min $n$ rev/min $Z$ no unit $D$ mm  <b>Outputs:</b> $W_c$ kW	<b>Inputs:</b> $P_s$ Pa $a$ mm/rev $D$ mm $Z$ no unit $V_c$ m/s  <b>Outputs:</b> $P_{ow}$ W $F_z$ mm/min $n$ rev/min	<b>Inputs:</b> $E_s$ hp*s/in <sup>3</sup> $F_z$ in/min $N$ rev/min $D$ in $n$ no unit  <b>Outputs:</b> $w$ hp

	<b>Drilling (enlarge a hole)</b>	<b>Inputs:</b> $P_s$ Pa $F_z$ mm/min $n$ rev/min $Z$ no unit $D_1$ mm $D_2$ mm  <b>Outputs:</b> $W_c$ kW	<b>Inputs:</b> $P_s$ Pa $a$ mm/rev $V_c$ m/s $Z$ no unit $D_1$ mm $D_2$ mm  <b>Outputs:</b> $Pow$ W $F_x$ mm/min $n$ rev/min	<b>Inputs:</b> $Es$ hp*s/in <sup>3</sup> $F_z$ in/min $N$ rev/min $D_1$ in $D_2$ in  <b>Outputs:</b> $w$ hp
---	--	---	---	---

### 7.3.2 Goal and Scenarios

The goal of this experiment is to test how well the similarity matching algorithm was able to:

- Discriminate among similar groups of interfaces.
- Show that type 1 and 3 are more similar than type 1 and 2 – since type 1 and 3 are functionally equivalent.
- Show that type 2 and 1 are more similar than type 2 and 3—since type 1 and 2 have the same core modeling approach

Similar to the single-interface template experiment scenario described in section 2.1, a single-interface template is created from each interface and compared to all interfaces in the suite. In this case study, we utilize a different visualization of result from previous experiments, because we want to visualize the relative similarity scores of each interface as well.

### 7.3.3 Results

As shown in figure 7-10, when a longitudinal turning power consumption interface of Type 1 was used as a template to match against the entire test suite, the algorithm retrieves all the interfaces back and rank Type 1 interfaces the top group, followed by Type 3 interfaces, and then Type 2 interfaces were ranked as the third group. Figure 7-11 is a column chart visualization of the result in figure 7-10, where one can visualize the differences among relative similarity scores. The chart clearly shows that Type 1 interfaces more similar to Type 3 than to Type 2.

Find similar model interfaces for template < longitudinal turning power consumption type1> :

Rank	Interface Name	Relative Similarity Score
#1	longitudinal turning power consumption type1	100.00%
#2	plunge milling (enlarge a hole) power consumption type1	97.50%
#3	radial turning power consumption type1	95.06%
#4	face milling power consumption type1	92.62%
#5	plunge milling (drill a hole) power consumption type1	80.89%
#6	longitudinal turning power consumption type3	79.60%
#7	plunge milling (enlarge a hole) power consumption type3	77.10%
#8	radial turning power consumption type3	74.66%
#9	face milling power consumption type3	72.22%
#10	plunge milling (drill a hole) power consumption type3	61.45%
#11	plunge milling (enlarge a hole) power consumption type2	56.45%
#12	longitudinal turning power consumption type2	37.35%
#13	radial turning power consumption type2	37.35%
#14	face milling power consumption type2	37.35%
#15	plunge milling (drill a hole) power consumption type2	23.12%

**Figure 7-10** Ranked result of matching a “longitudinal turning process a type 1” template to the whole test suite



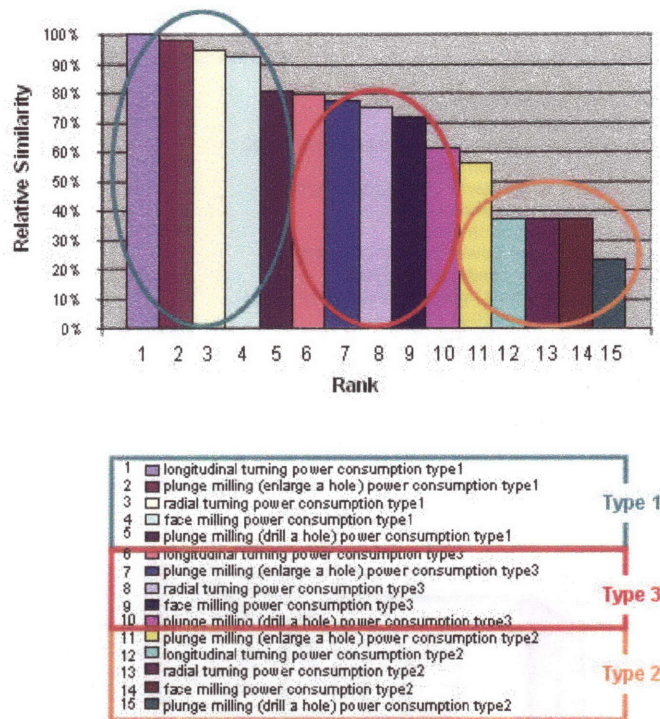


Figure 7-11 Column chart visualization: Ranked result of matching a “longitudinal turning process a type 1” template to the whole test suite

Similarly, the test is repeated using a longitudinal turning power consumption interface of Type 2 to match against the entire test suite. The result is given in Figure 7-12 and 7-13. The algorithm retrieves all the interfaces back and ranks Type 2 interfaces as the top group, followed by Type 1 interfaces, and then Type 3 interfaces are ranked the third group. The chart in Figure 7-13 shows that Type 2 interfaces are more similar to Type 1 than to Type 3.

Find similar model interfaces for template < longitudinal turning power consumption type 2 > :

Rank	Interface Name	Relative Similarity Score
#1	longitudinal turning power consumption type2	100.00%
#2	radial turning power consumption type2	88.89%
#3	plunge milling (enlarge a hole) power consumption type2	74.48%
#4	face milling power consumption type2	53.89%
#5	plunge milling (drill a hole) power consumption type2	53.89%
#6	radial turning power consumption type1	26.52%
#7	face milling power consumption type1	26.52%
#8	plunge milling (drill a hole) power consumption type1	26.52%
#9	longitudinal turning power consumption type1	24.90%
#10	plunge milling (enlarge a hole) power consumption type1	24.90%
#11	radial turning power consumption type3	18.95%
#12	face milling power consumption type3	18.95%
#13	plunge milling (drill a hole) power consumption type3	18.95%
#14	longitudinal turning power consumption type3	17.32%
#15	plunge milling (enlarge a hole) power consumption type3	17.32%

Figure 7-12 Ranked result of matching a “longitudinal turning process a type 2” template to the whole test suite

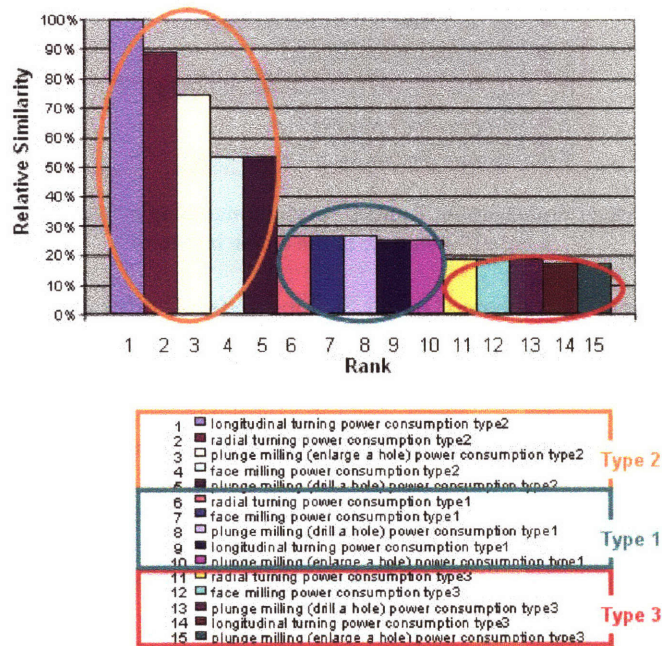


Figure 7-13 Column chart visualization: Ranked result of matching a “longitudinal turning process a type 2” template to the whole test suite

The experiment results show that the algorithm is able to discriminate between different sub-categories; and correctly determines that similarities between Type 1 and 3 are greater than those between Type 1 and 2; and similarities between Type 2 and 1 are greater than those between Type 2 and 3.

# Chapter 8

## Conclusions and Future work

### 8.1 Conclusions

New emerging Internet-based simulation environments including the DOME have provided enabling technologies for a World-Wide-Simulation Web (WWSW) — a World-Wide-Web (WWW) of web-enabled applications. The motivation of this work is to design a non-traditional search engine that is capable of searching for functionally appropriate web-enabled applications. As prior art, we studied ontological approach and pattern matching approach and discussed their strengths and limitations. We believe that there are a number of usability issues that will prevent the widespread, practical adoption of the ontological approach.

Alternatively, this work proposes a flexible, implicit, pattern matching solution that does not require extra annotations of the models, much as the way current web search engines operate. In this work, a hypothesis is made that interface structural pattern is sufficiently representative of functional roles of the underlying model. This work, for the first time, studied pre-existing low-level information in interfaces and proposed a minimal subset of information that are relevant to the functional role of underlying simulation models. Then, this minimal subset is used to synthesize templates. Templates

are synthetic data structures representing functional roles. When searching for web applications with a desired functional role, the template representing that functional role is used as an exemplar query and functionally similar interfaces are identified based on result of similarity matching. Additionally, newly found functionally equivalent interfaces can be merged into the original template, thereby both generalizing the pattern for a functional role and strengthening the most critical aspects of the pattern.

A prototype system was implemented in JAVA and applied to a suite of real-life engineering models to validate the approach.

The results demonstrated the plausibility of the approach:

- The hypothesis that functional roles of web-enabled applications can be related to structural interface information holds well.
- The proposed combination of pattern-based similarity measurement and template generalization works well in retrieving functionally comparable web-applications.
- The proposed approach outperforms text-based search tools in functional search.

## **8.2 Contributions**

The contribution of this thesis is a structural pattern recognition approach that recognizes functionally similar web-enabled applications based on available interface definitions; and, through implementation and testing on real-life engineering models, demonstrated the plausibility of the approach.

This thesis also developed algorithms based on graph theory and pre-defined heuristic attribute similarity metrics. The traditional method used in pattern matching applications is the vector-space model with similarity measure defined by Euclidean space. This thesis use the concept of graph similarity, graph distance, and graph matching as a basis for the novel approach we've developed for classification tasks instead of using restrictive vector models. The graph representation and algorithms provide flexibility and improve performance of the machine learning process.

Graph similarity has been largely studied in the graph matching field. A number of applications can be found in pattern recognition literature. However, to our best knowledge, there is no graph matching applications that deal with content-based categorization and classification of web-enabled applications, web services or parametric simulation models.

This thesis work provides an alternative solution for automatic interpretation of functional roles of simulation models by discover associations between content semantics and structural/syntactic patterns of available information. Although the work has been primarily tested on the DOME WWSW, the approach can be applied to other emergent environments. It can also be extended to facilitate function-oriented search capability for common web services on WWW, such as integrate with the UDDI (Universal Description, Discovery and Integration) [8].

### **8.3 Future Work**

The experiments conducted in this thesis show promising results. However, there are some open issues that require further investigation. Recommended future work has two

primary directions. The first direction is to improve performance and accuracy of the graph matching method. One possible area of research is investigating performance sensitivity to changes in similarity metrics. This can be done by adding user feedbacks [81] to dynamically tune and improve the heuristic similarity functions. Users might classify the systems matching results into relevant and irrelevant groups, and the system to adjust node attributes weights in the heuristic similarity functions to improve correspondence.

Another recommendation is to refine the bipartite matching algorithm. The greedy algorithm used in this work has limitations. The solution (local optimal) is not unique. There is no mechanism to find out all local optimal matching and compare them. One way to improve this is to adopt the Hungarian method [70] for finding optimal solution to weighted bipartite matching. The Hungarian method is to solve bipartite matching by finding augmenting path. The computational cost can be higher:  $O(n^3)$  in the worst case as opposed to  $O(n^2)$  of the greedy algorithm.

Meanwhile, another drawback of current bipartite matching is that by only considering node attribute similarity, structural information (arcs) is ignored during the initial node alignment. Although subsequent checks are made to validate structural compatibility, the current approach still could be error-prone. A possible improvement to address this problem could be to embed arc information as additional node attributes [34].

The second major direction for further research is additional study of template learning, generalization, and adaptation. One recommended future work is investigating performance with different template growth mechanisms, and how to control the learning

process. These include studying when a template should stop the generalization and adaptation process, and how to prune irrelevant information from well-adapted templates.

Conceptually, a template represents a fuzzy functional role of which we may not have complete knowledge. Currently, the decision of when to create and adapt a template for a functional role is done manually. Manual assertions might be subjective and introduce unfavorable noise to the template. Future research may be conducted on the potential adoption of incremental clustering approach [82] to provide some feedback to assist the user assertion. Incremental clustering method calculates the entropy of interfaces belonging to a template's cluster, to see if the updated template yield better similarity distribution in the cluster or not.

## Reference

- [1] Senin, N., D.R. Wallace, and N. Borland, "Distributed Object-Based Modeling in Design Simulation Marketplace". *Journal of Mechanical Design*, 2003. 125(2): p. 2-13.
- [2] Wallace, D.R., E. Yang, and N. Senin, "Integrated Simulation and Design Synthesis". MIT CADLAB technical report. 2002.
- [3] Engineous, The Federated Intelligent Product EnviRonment (FIPER) project 1999~2003.
- [4] Phoenix Integration, ModelCenter software, 1995~2005.
- [5] Wallace, D.R., et al., "Integrated Design in a Service Marketplace". *Computer-aided Design*, 2000. 32(2): p. 97-107.
- [6] Wikipedia contributors, "Semantic Web," Wikipedia, The Free Encyclopedia, [http://en.wikipedia.org/w/index.php?title=Semantic\\_Web&oldid=56205886](http://en.wikipedia.org/w/index.php?title=Semantic_Web&oldid=56205886).
- [7] Wikipedia contributors, "Ontology (computer science)," Wikipedia, The Free Encyclopedia, [http://en.wikipedia.org/w/index.php?title=Ontology\\_%28computer\\_science%29&oldid=56457748](http://en.wikipedia.org/w/index.php?title=Ontology_%28computer_science%29&oldid=56457748).
- [8] UDDI (Universal Description, Discovery and Integration project) homepage, <http://www.uddi.org/>.
- [9] W3C Semantic Web homepage, <http://www.w3.org/2001/sw/>.
- [10] Semantic Grid Project community portal, <http://www.semanticgrid.org/>.
- [11] McIlraith, S., T. Son, and H. Ze, "Semantic Web Services". *IEEE Intelligence*, 2001. 16(2): p. 46-53.
- [12] SWWS (Semantic Web Enabled Web Services) Project homepage, [http://swws.semanticweb.org/swws?cmd=show\\_entity&entity=Home+English](http://swws.semanticweb.org/swws?cmd=show_entity&entity=Home+English).
- [13] Bussler, C., A. Maedche, D. Fensel. "A Conceptual Architecture for Semantic Web Enabled Web Services". *ACM Special Interest Group on Management of Data*. Volume 31, Number 4, Dec 2002
- [14] W3C OWL-S standard, "OWL-S: Semantic Markup for Web Services". <http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/>
- [15] Heflin, J., "Towards the Semantic Web: Knowledge Representation in a Dynamic, Distributed Environment". Ph.D. thesis, University of Maryland at College Park, 2001.
- [16] Semantic Search - The SHOE Search Engine, <http://www.cs.umd.edu/projects/plus/SHOE/search/>.
- [17] Swoogle semantic search engine homepage, <http://swoogle.umbc.edu/>.
- [18] Finin T, and L. Ding, "Search Engines for Semantic Web Knowledge". *Proceedings of XTech 2006: Building Web 2.0*, May 2006.
- [19] Applied Ontology Overview, Applied Ontology Journal homepage, <http://www.applied-ontology.org/policies.php#focus>
- [20] N. Guarino, "Formal Ontology and Information Systems". In *Proceedings of Formal Ontology and Information Systems*, Trento, Italy, June 1998. IOS Press.
- [21] Gruber, T.R., "A translation approach to portable ontologies". *Knowledge Acquisition*, 1993. 5(2): p. 199-220.



- [22] Kim, J., P. Will, R. Ling, and B. Neches “Knowledge Rich Catalog Services for Engineering Design”, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*. Volume 17 , Issue 4 (September 2003) Pages: 349 - 366, 2003
- [23] Shirkey, C, “The Semantic Web, Syllogism, and Worldview”, first published November 7, 2003. [http://www.shirky.com/writings/semantic\\_syllogism.html](http://www.shirky.com/writings/semantic_syllogism.html).
- [24] Doan, A., J. Madhavan, P. Domingos, and A. Halevy. “Learning to Map between Ontologies on the Semantic Web”. In *Proceedings of the eleventh international conference on World Wide Web*, Honolulu, Hawaii, USA, May 7-11, 2002
- [25] Rahm, E., and P. Bernstein. “A survey of approaches to automatic schema matching”, *The VLDB Journal* 10, 334-350, 2001
- [26] Shirkey, C., “Ontology is Overrated: Categories, Links, and Tags”, first published at spring 2005. [http://www.shirky.com/writings/ontology\\_overrated.html](http://www.shirky.com/writings/ontology_overrated.html)
- [27] Wikipedia contributors, "Pattern matching," *Wikipedia, The Free Encyclopedia*, [http://en.wikipedia.org/w/index.php?title=Pattern\\_matching&oldid=46146795](http://en.wikipedia.org/w/index.php?title=Pattern_matching&oldid=46146795)
- [28] Long, F., H Zhang and D. Feng, “Fundamentals of Content-based Image Retrieval”, [http://research.microsoft.com/asia/dload\\_files/group/mcomputing/2003P/ch01\\_Long\\_v40-proof.pdf](http://research.microsoft.com/asia/dload_files/group/mcomputing/2003P/ch01_Long_v40-proof.pdf)
- [29] Krishnapuram, R., et al., Content-based Image Retrieval based on a fuzzy approach. *IEEE transaction on knowledge and data engineering*, 2004. 16(10): p. 1185-1199.
- [30] Medasani, S. and R. Krishnapuram. A Fuzzy Approach to Content-based Image Retrieval. in *IEEE International Fuzzy Systems Conference Proceedings*. 1999. Seoul, Korea.
- [31] Chan, K.P. and Y.S. Cheung, Fuzzy-attributed graph with application to Chinese character recognition. *IEEE Transaction on Systems, Man, and Cybernetics*, 1992. 22(1): p. 153-160.
- [32] Tsai, W.H. and K.S. Fu, Error-Correcting Isomorphism of attributed relational graphs for pattern recognition. *IEEE Transaction on Systems, Man, and Cybernetics*, 1979. 9(12): p. 757-769.
- [33] R. C. Wilson and E. R. Hancock, "Structural Matching by Discrete Relaxation," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, vol. 19, pp. 634 - 648, 1997.
- [34] Kim, W.Y. and C. Kak, 3-D Object Recognition Using Bipartite Matching Embedded in Discrete Relaxation. *IEEE transaction on pattern analysis and machine intelligence*, 1991. 13(3): p. 224-251.
- [35] Cyr, C. and B. Kimia. A Similarity-based Aspect-Graph Approach to 3D Object Recognition in *ICCV*. 2001.
- [36] Iyer, N., et al. A Reconfigurable, Intelligent 3D Engineering Shape Search System Part I: Shape Representation. in *ASME DETC'03, 23rd Computers and Information in engineering (CIE) Conference*. 2003. Chicago, Illinois.
- [37] Kraines, S. B., Batres, R., Koyama, M., Wallace, D. R., Komiyama, H. Internet-based collaboration for Integrated Environmental Assessment in Industrial Ecology. submitted to *J. Industrial Ecology*
- [38] Wang, Y., E. Stroulia. “Flexible Interface Matching for Web-Service Discovery”, in *Proceedings of the Fourth International Conference on Web Information Systems Engineering, WISE'03*

- [39] Santini, S. and R. Jain (1998). Beyond Query by Example. ACM Multimedia'98. Bristol, UK, ACM: 345-350.
- [40] Wikipedia contributors, "Semantic gap," Wikipedia, The Free Encyclopedia, [http://en.wikipedia.org/w/index.php?title=Semantic\\_gap&oldid=36019261](http://en.wikipedia.org/w/index.php?title=Semantic_gap&oldid=36019261).
- [41] Szykman, S., R. D. Sriram, C. Bochenek and J. Racz, "The NIST Design Repository Project". Advances in Soft Computing - Engineering Design and Manufacturing, Roy, R., T. Furuhashi and P. K. Chawdhry (eds.), Springer-Verlag, London, 1999., July, 1998, pp.5-19.
- [42] Szykman, S., R. D. Sriram, C. Bochenek, J. Racz, and J. Senfaute, "Design Repositories: Engineering Design's New Knowledge Base", IEEE Intelligent Systems, IEEE Educational Activities Department Piscataway, NJ, USA, Volume 15, Issue 3 (May 2000) pp: 48 – 55, 2000
- [43] Kopena, J., W. Regli, "Extensible Semantic For representing electromechanical assemblies", in Proceeding of DETC'03, September 2-6, 2003, Chicago, IL.
- [44] Kopena, J., W. Regli "Design Repositories on the semantic web with description-logic enabled services", in Proceeding of SWDB 2003: pp349-356
- [45] Mocko, G., R. Malak, C. Paredis, and R. Peak, "A knowledge repository for behavioral models in engineering design", in Proceeding of DETC'04, September 28-Oct 3, 2004, Salt Lake City, Utah.
- [46] A. A. Sonin, "The physical basis of dimensional analysis," Department of Mechanical Engineering, MIT, 1997.
- [47] Wikipedia contributors, Dimensionless number," *Wikipedia, The Free Encyclopedia*, [http://en.wikipedia.org/w/index.php?title=Dimensionless\\_number&oldid=39256489](http://en.wikipedia.org/w/index.php?title=Dimensionless_number&oldid=39256489),
- [48] Mitsubishi Materials Corporation, Formulas for Cutting Power. 2003, [http://www.mitsubishicarbide.com/mmc/en/product/technical\\_information/information/formula4.htm](http://www.mitsubishicarbide.com/mmc/en/product/technical_information/information/formula4.htm)
- [49] Process Associates of America: Process Tools, Gas compressibility factor, 1995-2003, [http://www.processassociates.com/process/property/z\\_factor.htm](http://www.processassociates.com/process/property/z_factor.htm).
- [50] S. Lee and W. Kim, "Robust Character Image Retrieval Method Using Bipartite Matching and Pseudo-bipartite Matching," Lecture Notes In Computer Science, vol. 2402, pp. 295-306, 2002.
- [51] A. Schenker, et al. "Graph-Theoretic Techniques for Web Content Mining", Series in Machine Perception and Artificial Intelligence, Vol 62. World Scientific Publishing Co., Ed. H. Bunke, P.S.P. Wang, 2005.
- [52] Wikipedia contributors, "Random graph," *Wikipedia, The Free Encyclopedia*, [http://en.wikipedia.org/w/index.php?title=Random\\_graph&oldid=54276837](http://en.wikipedia.org/w/index.php?title=Random_graph&oldid=54276837)
- [53] A. K. C. Wong, "Structural Pattern Recognition: A Random Graph Approach," in *NATO ASI Series: Pattern Recognition Theory and Applications* vol. F30, P. A. Devijver and J. Kittler, Eds.: Springer-Verlag, Berlin Heidelberg, 1987, pp. 323-345.
- [54] L.A. Zadeh, "Fuzzy sets", *Information and Control*, vol 8. pp. 338-353, 1965
- [55] M. Kantrowitz, E. Horstkotte, and C. Joslyn, "Answers to Questions about Fuzzy Logic and Fuzzy Expert Systems ", 1993~1996. <http://www.cs.cmu.edu/Groups/AI/html/faqs/ai/fuzzy/part1/faq.html>.
- [56] J. R. Ullman, "An algorithm for subgraph isomorphism," *Journal of Assoc. Computing Machinery*, vol. 23, pp. 31-42, 1976.

- [57] B. Messmer and H. Bunke, "A new algorithm for error-tolerant subgraph isomorphism detection," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, vol. 20, pp. 493-504, 1998.
- [58] H. Bunke and K. Shearer, "A graph distance metric based on the maximal common subgraph," *Pattern Recognition Letters*, vol. 19, pp. 255-259, 1998.
- [59] A. Perchant and I. Bloch, "Fuzzy morphisms between graphs," *Fuzzy Sets and Systems*, vol. 128, pp. 149-168.
- [60] M. A. Eshera and K.S. Fu, "A Graph Distance Measure for Image Analysis," *IEEE Transaction on Systems, Man, and Cybernetics*, vol. SMC-14, pp. 398-408, 1984.
- [61] M. A. Eshera and K.S. Fu, "An image understanding system using attributed symbolic representation and inexact graph-matching," *IEEE transaction on pattern analysis and machine intelligence*, vol. PAMI-8, pp. 604-618, 1986.
- [62] H. Bunke, *Graph matching: Theoretical foundations, algorithms, and applications*. in Proc. Vision Interface 2000. Montreal, Canada.
- [63] Wikipedia contributors, "Levenshtein distance," *Wikipedia, The Free Encyclopedia*, [http://en.wikipedia.org/w/index.php?title=Levenshtein\\_distance&oldid=39136064](http://en.wikipedia.org/w/index.php?title=Levenshtein_distance&oldid=39136064)
- [64] Wikipedia contributors, "Tf-idf," *Wikipedia, The Free Encyclopedia*, <http://en.wikipedia.org/w/index.php?title=Tf%E2%80%93idf&oldid=33794930>
- [65] Jaro, M. A. *Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida*. *Journal of the American Statistical Association* 84: 414-420. 1989.
- [66] Winkler, W. E. 1999. *The state of record linkage and current research problems*. Statistics of Income Division, Internal Revenue Service Publication R99/04. Available from <http://www.census.gov/srd/www/byname.htm>
- [67] Cohen, W.W., Pradeep Ravikumar, Stephen E. Fienberg, "A Comparison of String Distance Metrics for Name-Matching Tasks", 18<sup>th</sup> International Joint Conference on Artificial Intelligence, Workshop on Information Integration on the Web, 2003
- [68] W. W. Cohen and P. Ravikumar. *Secondstring: An open-source Java toolkit of approximate string-matching techniques*. Project web page, <http://secondstring.sourceforge.net>, 2003.
- [69] G. Schadow and C. J. McDonald, "Units of Measure in Clinical Information Systems," *JAMIA*, vol. 6, pp. 151-162, 1999.
- [70] H. W. Kuhn, *The Hungarian method for the assignment problem*, *Naval Res. Logist. Quart.* 2:83-97, 1955
- [71] M. Zito, "Bipartite matching: Hungarian algorithm ": Course Material, Comp309, University of Liverpool, UK, 2005. <http://www.csc.liv.ac.uk/~michele/TEACHING/COMP309/2005/Lec8.2.4.pdf>
- [72] R.Sedgewick, "Algorithm in C++", Addison-Wesley Publishing Company, 1992.
- [73] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and C. Stein, "Introduction to Algorithms", 2nd Ed., MIT Press, 2001.
- [74] Wikipedia contributors, "Bipartite graph," *Wikipedia, The Free Encyclopedia*, [http://en.wikipedia.org/w/index.php?title=Bipartite\\_graph&oldid=55677856](http://en.wikipedia.org/w/index.php?title=Bipartite_graph&oldid=55677856)
- [75] Google, <http://www.google.com>
- [76] Wikipedia contributors, "Information retrieval," *Wikipedia, The Free Encyclopedia*, [http://en.wikipedia.org/w/index.php?title=Information\\_retrieval&oldid=57102093](http://en.wikipedia.org/w/index.php?title=Information_retrieval&oldid=57102093)
- [77] *Alternative Energy Toolkit*, <http://cadlab.mit.edu/altenergy>, 2004

- [78] Sukkasi, S., Master Thesis: The Alternative Energy Design Toolkit, in Mechanical Engineering Department. 2004, MIT.
- [79] Apache Lucene project homepage, <http://lucene.apache.org/java/docs/index.html>
- [80] Qing Cao, Nicola Senin, David R. Wallace, "Functional classification of computational services in an internet-based distributed modeling environment", Proceedings of the ASME IDETC/CIE Conferences, Long Beach, CA, September, 2005.
- [81] Yao, Y. Y. "Measuring Retrieval Effectiveness Based on User Preference of Documents", Journal of the American Society for Information Science; Mar 1995; 46, 2; ABI/INFORM Global pg.133.
- [82] Dong Su Seong, Ho Sung Kim, and Kyu Ho Park, Incremental Clustering of Attributed Graphs, IEEE TRANSACTIONS ON SYSTEMS, MAN. AND CYBERNETICS. VOL. 23, NO. 5, 1993, pp.1399-1410.