

Matching methods for semantic interoperability in Product Lifecycle Management

by
Il Yeo

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in The University of Michigan
2009

Doctoral Committee:

Adjunct Professor Debasish Dutta, Co-chair
Lalit M. Patil, Co-chair
Associate Professor Kazuhiro Saitou
Assistant Professor Clayton D. Scott
Lakshmi Y. Srinivas, Ford Motor Company

© Il Yeo
All Rights Reserved
2009

For my late father, Bongseok Yeo, whose love is always with me,
and for my mother, Heesun Lee, whose prayer never cease.

ACKNOWLEDGEMENTS

This thesis would not have been possible without the support, encouragement and contributions of many individuals. First, I would like to thank my co-adviser, Prof. Debasish Dutta, for his support and guidance. He was an living example to me on how to speak, write and act as an engineer.

I am extremely grateful to my co-adviser, Dr. Lalit Patil. It is just impossible to enumerate what I have received from him. He has been a great teacher, mentor and friend to me. I hope that I have gained some of his strengths and can pass it on to others.

I would also like to thank the other members of my thesis committee: Prof. Saitou, Prof. Scott and Dr. Srinivas, for their insights, ideas and encouragement during the course of this work.

I would like to acknowledge the financial support provided by National Science Foundation, the National Institute of Standard and Technology, the Korean Organization of Science and Engineering Foundation, the Product Lifecycle Management Alliance and the Department of Mechanical Engineering at the University of Michigan.

In addition, I greatly appreciate the lively exchange of ideas and the supportive environment provided by the current and former members of the PLM Alliance, Farhad Ameri, Nikhil Joshi, Chandresh Mehta and Seungcheol Yang. They shared much of their time to discuss my research and to help my presentations.

Furthermore, I would like to thank my friends, in particular, the wonderful members of BESEL (Believers who Speak English as a Second Language) at Ann Arbor Christian Reformed Church. I could regain energies whenever I need from many pleasant activities with them.

I am indebted to my parents and family, for their constant encouragement, patience and support.

Finally, I have been and am happy that I have my wife, Yeonsoo, whom I can share ideas and emotions with and understand each other.

TABLE OF CONTENTS

DEDICATION	ii
ACKNOWLEDGEMENTS	iii
LIST OF FIGURES	viii
CHAPTER	
I. Introduction	1
1.1 Background	1
1.2 Semantics-based product data interoperability	2
1.3 Issues faced during product concept mapping/translation	5
1.3.1 Heterogeneities in product development systems	5
1.3.2 Uncertainties in semantic maps	7
1.4 Dissertation goals	8
1.5 Outline of the dissertation	10
II. Literature review	12
2.1 Terminologies	12
2.2 Related work: Matching methods	15
2.3 Related work: Combination methods	17
2.4 Related work: Translating the concepts	20
2.5 Summary	21
III. Instance-Based Concept Matching (IBCM)	23
3.1 Motivation	23
3.2 Objective	24
3.3 Overview of IBCM	25
3.4 Details of IBCM	27
3.4.1 Preprocessing	27
3.4.2 Attribute matching	29
3.4.3 Concept partition matching	32
3.5 Case study	38

3.5.1	Preprocessing	39
3.5.2	Attribute similarity	40
3.5.3	Concept partition similarity	42
3.5.4	Discussion	44
3.6	Summary	46
IV. FEedback Matching Framework with Implicit Training (FEMFIT)		47
4.1	Motivation	47
4.2	Objective	48
4.3	Product concept matching problem	49
4.4	Nonlinearity in combination	52
4.5	Overview of FEMFIT	55
4.6	Details of FEMFIT	58
4.6.1	Concept ordering	58
4.6.2	Matchers - Calculating individual similarities	60
4.6.3	Ranking SVM - Combining multiple measure result	61
4.6.4	Expert decision - Finalizing the match	65
4.7	Case study	66
4.7.1	Ontological formulation	66
4.7.2	Concept ordering	68
4.7.3	Individual measures	69
4.7.4	Result	71
4.8	Summary	78
V. Translating the concepts		80
5.1	Motivation	80
5.2	Objective	81
5.3	Product concept translation and related work	81
5.4	Problem formulation	84
5.5	Modeling concept translation with a directed graph	85
5.5.1	Nodes	86
5.5.2	Arcs: basis functions	87
5.6	Graph search to find the translation rule	89
5.6.1	Node evaluation	90
5.6.2	Termination criteria	92
5.6.3	Translation rule	93
5.7	Case study	94
5.7.1	Element-level translation	95
5.7.2	Structure-level translation	96
5.8	Summary	98

VI. Conclusion	101
6.1 Research contributions	101
6.2 Application to other areas	102
6.3 Future work	103
Bibliography	115

LIST OF FIGURES

Figure

1.1	Multiple CAD/CAM Systems Used in the Automobile Supply Chain (Adapted from [1])	3
2.1	(a) An example of product concept hierarchy, (b) Example product data (instance) of <i>Pad</i> . Instances have attributes and data values, (c) the visual representation of the product data described in (b)	14
3.1	Overall procedure of the proposed method. From the instances of sending and receiving system, it calculates concepts similarity through 3 step procedure	26
3.2	Preprocessing of input into the attributes and its data values. First, instances of a concept are partitioned so that instances in the same partition have the same set of attributes (partitioning). Now, each partition can be represented as a set of attributes of which each attribute has a set of data values (attribute collection).	28
3.3	Attribute pairing as bipartite graph matching: (a) (weighted) similarity values are assigned to the edges in a bipartite graph connecting two disconnected sets of vertices, i.e., attributes of concept partition <i>A</i> and <i>B</i> , (b) dummy attribute b_3 has been added with all new similarity with 0, and (c) pairs have been found such that the similarity total is maximized, i.e., 0.8 in this case.	36
3.4	Pin model created in NX6 is shown in the left. The models can be represented differently in different CAD software, and the representations - concept names and attribute names - are shown for NX6, CATIA and Pro Engineer. The enclosed area with dotted line, i.e., matching <i>Revolve</i> (NX6) and <i>Shaft</i> (CATIA), shows the specific example dealt with in this case study.	39

3.5 Partitioning of *Revolve* instances. From NX model files, *Revolve* features are extracted, and then the features are divided into the partitions in a way that each partition has a set of instances with same set of attributes. In the tables, some attributes are bold faced to show the difference from other partitions. In the value fields, quotation is used to denote the enumeration attributes, and '-' is used to denote the reference to another instance. 41

3.6 (a) Signature tuple for attributes of the partitions from *Revolve* and *Shaft* concept, (b) similarity matrix between the attributes. Note that the upper left values are similarity between numeric attributes and were calculated from the signature tuples shown in (a). Lower right values are similarity between reference attributes and from the recursive evaluation of concept (partition) similarity. Similarity between numeric and reference attributes are all set to zero. 42

3.7 (a) Entropy H and normalized entropy H' for the attributes of compared partitions. Attributes of 0 entropy are not shown in the table for brevity. (b) Weighted attribute similarity calculated from attribute similarity and normalized entropy. From the similarity matrix, identified pairs are shown as shaded cells. 43

3.8 Translation between *Revolve* instance (NX) and *Shaft* instance (CATIA). In translating *Revolve* into *Shaft* attribute information is lost - 6 attributes shown as either boldfaced or italicized. In the reverse direction, 4 attributes (boldfaced) are created automatically with correct values, but values for 2 attributes (italicized) cannot be determined. Note that further information for reference attributes are omitted and shown as '-'. Those are additional instances and can be found through further translation of the instances. 45

4.1 Relation between an individual name similarity measure and the overall similarity measure on the constraint concepts in CATIA and NX. A nonlinear function is required to capture this relation correctly. 53

4.2 Conditional dependency of Parallelism constraint in CATIA and Parallel constraint in NX. The similarity measure that considers the dimension of the space is necessary before considering the name-based similarity measure 54

4.3 Overview of FEMIT. Solid lines indicate the sequence of execution with necessary information flow, and the dotted lines indicate the flow of the training data 56

4.4	(a) an example of partially ordered set of dependency relation, (b) A directed acyclic graph (DAG) generated from the set, (c) possible linearizations of the DAG through topological sorting	59
4.5	A ranked set of data points p_1, p_2, p_3 and p_4 are plotted on the plane. Different vectors can be set up but W_1 can order the points correctly when projected onto whereas W_2 generates a wrong rank. Ranking SVM seeks a weight vector like W_1 (Adapted from [2])	62
4.6	A ranking procedure with Ranking SVM. After the points from previous history are transformed into a feature space and a vector W is calculated projected on which the similarity ordering is maintained maximizing the minimum margin between ranked pairs. Once the vector is found, new data set is plotted on the space and is projected onto the vector W . Based on the value, a user finalizes the matching.	65
4.7	Example of a cell phone assembly model with matching concepts to capture constraints in the software NX, the former Unigraphics, and CATIA. The translation is through the standard representation ISO 10303-108. The case study considers only the part from NX to ISO 10303-108 for brevity (enclosed with dotted line).	67
4.8	Partial concept hierarchies of the ontologies modeling assembly constraints in (a) NX and (b) ISO 10303-108 assembly constraints	68
4.9	Concept ordering example: (a) dependency network for NX assembly constraints is shown in Directed Acyclic Graph (DAG), (b) a linearization is generated through topological sorting. In the order set, concepts with larger indices are independent from the concepts with smaller indices.	69
4.10	Concept flattening example: (a) <i>Angle</i> constraint has three ancestors and they have their own logical descriptions, (b) flattened <i>Angle</i> inherit the logical descriptions from its ancestor concepts. Note that the inherited description can be overridden by its own description. In the example, <i>hasGeometry min 1</i> is overridden by <i>hasGeometry exactly 2</i>	71

4.11	Table shows how FEMFIT confirms matching over iteration: for each iteration, that is, for each source concept, the recommendation is given as ranked list. When the user select the correct match, it trains them further and gives the improved recommendation for the ensuing iterations. Note that there is no recommendation in the first iteration; learning starts from the input of the user after the end of first iteration.	72
4.12	Ranks of the real corresponding concept in the ordered set by each combining method for 9 matching processes.	73
4.13	Averages and standard deviations of the ranks for different combination algorithms with different combinations of individual measures: Name (N), PathName (PN), and Property (PR) matchers	75
4.14	Comparison between the result of FEMFIT and SVM regression: (a) Rank of the corresponding concepts for each iteration. Note that there is no result for the first iteration because there is no input before the first expert decision, (b) Averages and standard deviations of the ranks for different combinations of individual measures: Name (N), PathName (PN), and Property (PR) matchers	77
5.1	Extrude instance representations in SolidWorks and NX, CAD software applications. Concept (attribute) matching is to find correspondence between the concept/attribute names, and the translation is to actually transform the instance data for each attribute.	83
5.2	Translation problems are modeled as a graph search. Functions in the path from the initial nodes to the goal node are combined to construct the translation rule for the given target attribute (b_1 in this case).	86
5.3	The figure shows a process to expand a node to create new nodes. In this example, there is two single input basis functions - function that converts <i>mm</i> into <i>inch</i> and the one that converts <i>Celsius</i> into <i>Fahrenheit</i> , and each produces a new node. The created nodes has a tuple of the same length with the original node - in this case, it is 3-tuple.	89

5.4	The flowchart shows a procedure for the proposed method to find a translation rule with an approach based on graph search algorithm. Specifically, best-first search is used - after evaluating the evaluation function $f_{eval}(n)$ for each node n , expand the node with the smallest value, i.e., approximately the closest to the goal node. The iteration goes on until it reaches termination criteria - either successful finding of the goal node or failure. Note that the figure shows only success cases for brevity.	90
5.5	The figure shows a process to calculate the distance heuristic for an example node given a goal node. In this example, relative distance measure is used as distance heuristic. Relative distance is obtained for each pair of values, and averaged into one distance value.	92
5.6	A schematic showing the translation of a part in <i>NX</i> into <i>SolidWorks</i> . To translate, the attribute values marked as “?” should be identified from the source instances. In the case study, the process to find only one attribute value - <i>Name</i> - enclosed with a thick box is demonstrated.	95
5.7	A translation rule is constructed with a graph search. There are 3 initial nodes, and the nodes with the smallest distance are evaluated. When the goal node is reached, the path information (arcs and the start node) is used to construct the translation rule. Note that the nodes are shown with only 1-tuple for brevity.	97
5.8	Instances of color information used in (a) PLM XML by <i>Siemens</i> and (b) 3D XML by <i>Dassault Systemes</i> . In PLM XML, RGBA information is represented as a string of 4 values, while 3D XML shows them as separate attributes. This type of heterogeneity cannot be captured with the 1-to-1 conversion functions explained in the previous section and we need an extended function. Note that the nodes are shown with only 1-tuple for brevity.	98
5.9	A translation rule for one target attribute is constructed from 4 attributes. Only one basis function is shown to demonstrate the process; more functions can be included but will not change the result.	99

CHAPTER I

Introduction

1.1 Background

Globalization is leading to new paradigms for product development and manufacturing enterprises. Manufacturers are facing increasing pressures from diverse global markets to deliver higher-quality, lower-cost products with shorter turnaround. In addition, corporations, both federal and private, recognize the need and advantages of harnessing global talents through geographically distributed, cross-functional teams.

At the same time, significant advances in computing and database capabilities have led to improved performance in every phase of the product's lifecycle. Manufacturers use the advances in Computer-Aided Design (CAD), Computer-Aided Manufacturing (CAM), Product Data Management (PDM) and other tools, along with available knowledge to develop the physical form, logic, specifications and other information that defines a product.

There is an increasing emphasis on unifying the multiple activities within the knowledge-intensive, distributed, multi-functional enterprise. This has led to the vision of Product Lifecycle Management (PLM) as a business strategy that views the

above-mentioned value chain as one product-centric enterprise, and not a set of silo-ed processes. Ameri and Dutta [3] discuss several aspects of PLM and synthesize that it is “a computational framework which effectively enables capture, representation, retrieval and reuse of product knowledge.”

The overall idea is to utilize emerging software technologies in areas, such as knowledge management, data translation and web-based collaboration to facilitate innovation through integration, i.e., by allowing faster and effective product data interoperability and seamless collaboration between various functions of the enterprise.

1.2 Semantics-based product data interoperability

An important feature of product information is its meaning (semantics) in the context in which it is generated and used. Product knowledge (product description, configuration, attributes and requirements), in contrast to text data, is not self-descriptive. This knowledge should ideally be available to the stakeholders (e.g., Original Equipment Manufacturers (OEMs), suppliers) in a manner that will enable consistent interpretations of the data.

The growth of the Internet has ensured connectivity among the various stakeholders, i.e., right information could be made available at the right time. Yet, conservative estimates in a study by Brunnermeier and Martin [1] for the National Institute of Standards and Technology (NIST) suggest that imperfect interoperability costs at least 1 billion dollars per year to the United States automotive supply chain. The majority of these costs are due to resources spent in correcting and recreating data that is not usable by the receiving applications. This inability to use translated

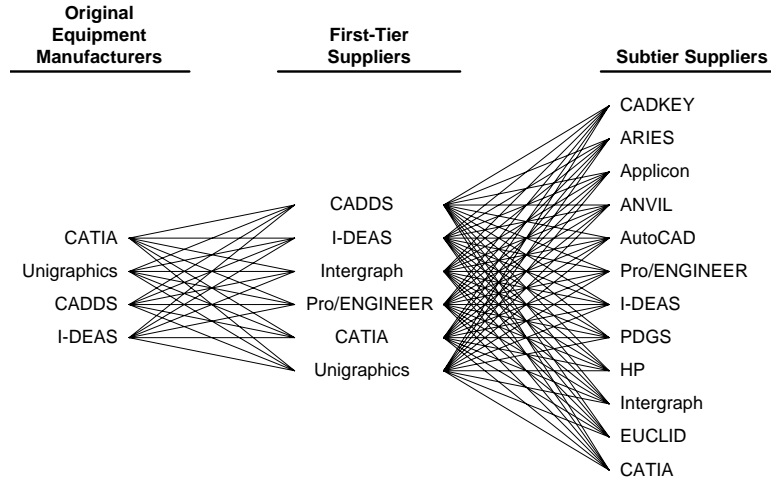


Figure 1.1: Multiple CAD/CAM Systems Used in the Automobile Supply Chain (Adapted from [1])

product information is mainly because its semantics is lost during the translation to a new domain. In a follow-up study, Gallaher et al. [4] present a similar impact of the lack of interoperability for the aerospace and shipbuilding industries.

It is necessary for the success of OEMs, federal departments such as the U.S. Army [5], their suppliers and software vendors that interoperability is correct and efficient, i.e., it is not merely syntactic data exchange. This becomes even more important in the context of increasing and changing capabilities (evolving semantics) of existing systems in distributed product development as shown in Figure 1.1.

Therefore, there is a need to promote drastically increased levels of interoperability of product data across a broad spectrum of stakeholders while ensuring that the semantics of product knowledge are preserved, and when necessary, translated.

Semantic technologies include standards and methodologies aimed at providing and using these semantics. A semantics-based integration can be achieved through the following steps:

1. Representation of the semantics of the participating domains

Currently, semantics of product data is rarely accessible by the software other than the one which created the data. For example, the semantics of CATIA part files is only accessible in CATIA. Although partial semantics are sometimes accessible through the Application Programming Interface (API) provided by the software, there is no standard for APIs of different software, and it is impossible for a generic method to access the semantics of product data created with different software tools. Therefore, an explicit standard representation of semantics is an essential part for any generic method to interpret the data.

2. Automated mapping of concept semantics across representations

Explicit semantic representations do not guarantee the compatibility of different representations, and the next step toward interoperability is to identify the semantic map between the concepts across different representations. Determining semantic maps requires determining similarities across several thousands of pairs of concepts (and attributes). Furthermore, maps need to be determined across several different systems that are used in the product development area. This process, if done manually, is labor intensive and error-prone. Therefore, there is a need for an automated approach to calculate the semantic similarities and thereby determine the semantic maps across different representations.

3. Automated translation of data based on the pre-determined semantic maps

Once the semantic maps across the different representations are obtained, the

next phase uses the maps for actual translation of the syntax from the sending system to that of the receiving system. This can be facilitated by determination of translation rules as some combination of possible functions to convert an instance form to another, e.g., converting date from mm/dd/yyyy form to dd/mm/yyyy form. However, several thousands of such rules would need to be determined corresponding to each pair of concepts and attributes in the map across each system. Therefore, there is a need for an automated system to determine such translation rules.

Research has focused on developing ontologies/representations [6, 6, 7, 8], and this research focuses on the second and third step: automated mapping and translation. Next section describes the issues faced in those steps.

1.3 Issues faced during product concept mapping/translation

Mapping (or matching) and translation of concepts across different representations in product development is challenging because of the heterogeneities in the representations, and the uncertainties in the semantic maps.

1.3.1 Heterogeneities in product development systems

Heterogeneities are mainly structural differences in the two systems that need to be integrated. These are of the following types:

Naming Conflicts The multiple information resources used in product development are disparate. This leads to heterogeneities such as the following:

- Different ways of representing same information, e.g., a *Pad* in CATIA is called as an *Extrude* in NX, the former Unigraphics.
- Similar names for different information, e.g., *Fillet* in NX is called as a *Blend* in Solidworks, and *Blend* in NX is called as a *Fillet* in Solidworks.

Composite concept relationships One concept from a representation might match to a combination of a set of concepts from the other representation. This may present itself in the following forms:

- **Concept Generalization** A concept in one ontology (e.g., Base_Extrude in Solidworks) can match only to its generalized counterpart in the other ontology (e.g., Extrusion in NX). This requirement could have been neglected as translation to a system that does not support some concepts; yet, the product is still valid in both systems.
- **Disparate information across representations** Different representations capture different amounts and types of information. For example, NX supports more concepts, e.g., primitives, such as block, cylinder, as compared to Solidworks, although both are commercial CAD software. In addition, NX can store both design and manufacturing information, whereas ADAMS can handle only product-motion knowledge. Solidworks, on the other hand, does not have a manufacturing module.
- **Concept aggregation** In some cases, maps should be from a concept that represents the higher level definition in one representation to its lower level composition in another representation. Concept aggregation indi-

cates the relationship between a part and a whole, in this case, across different representations. For example, the concept, *Orientation* in ADAMS can only be equated to its composition in the form of angle constraints on each of the 3 axes.

- **Concept enumeration** In a simplest form, an instance belonging to concept a in ontology A may belong to either concept b or c in ontology B , where a is said to be the aggregation of b and c . For example, a *coincidence* constraint in CATIA can be *Coincident*, *PointOnCurve* or *PointOnString* in NX depending on the associated geometry objects.

1.3.2 Uncertainties in semantic maps

Semantics-based integration also needs to overcome uncertainties in the semantic maps due to the following:

- **Incompletely developed representations** These could be due to limited expressiveness of the representation language or implicit meanings associated with some terms. This adds to uncertainty in the semantic maps, because they will be based on partial semantics.
- **Maps with multiple options** Several cases with multiple options may exist across different representations resulting in multiple correct solutions. For example, consider the concept *cylinder* created in NX. CATIA has at least two options for this solid: *pad*, and *shaft*. It is not clear how one of the two options should be selected, because while cylinder is a shape-defining element, *pad* (*shaft*) captures the method of creating the solid.

- **Lack of formal basis to capture designer’s intent** There is a lack of a formal meaning of “information” or “feature vectors” attached to a product concept. In addition, there is no single accepted approach to formally capture or represent correct maps. For example, emerging exchange standards include numeric shape-based invariants to determine the accuracy of translation of product data from one system to another. However, there is no formal basis to justify their use for this purpose. Kuzminykh and Hoffmann [9] discuss how these quantities alone are insufficient to exclude many translation errors caused due to differing semantics.

This section described the issues faced during the mapping and translation of the product concepts. Although some issues have been considered by existing methods, many remain unsolved. Next section presents the goals of the dissertation.

1.4 Dissertation goals

The goal of this research is to design and develop techniques to determine maps across product representations that will enable semantically correct translation/interoperability of product data.

We focus on the following specific tasks:

1. Methods to identify maps between the different representations to identify semantically aligned concepts. For this, we consider:
 - a) A method that uses implicit semantics captured in the instances of product data to determine the maps. This is necessary to address problems that pre-

vious research in this area have been unable to address (See Section 1.3.2) because of focus on using explicitly defined schema or data definition exclusively. The proposed method should address 1:n maps across the representations and should work without complete and explicit representations.

b) A method that unifies the multiple individual views used to determine maps. This is necessary because any single matching method is not enough to determine the semantic maps across the different systems, since each method presents only one view. The proposed method must handle the complex interrelationships among the individual matchers.

2. A method to identify translation rules to enable physical translation of concepts from one system to another. This is necessary, because even after the semantic maps are obtained, the syntax in the sending system should properly transform to the syntax in the receiving system.

Evaluation

We are not aware of other research/literature in this area that could be used to evaluate our approaches objectively. Therefore, for this thesis, we will develop case studies, and evaluate the resulting semantic maps manually, at times comparing with possibly competitive techniques.

Next section presents the outline of the dissertation to address the tasks.

1.5 Outline of the dissertation

This chapter discussed product data interoperability as the requirement for the modern product development systems, and the specific tasks to enable the interoperability. Especially, we emphasized the role of semantic mapping and the translation based on the map, and explained the challenges in finding the map and the translation rules between heterogeneous product development systems. Next chapters will further discuss the details.

Chapter II first explains concepts and terminologies used throughout the dissertation, and documents related work for the 3 research tasks identified in the previous section.

Chapter III describes a method - Instance-Based Concept Matching (IBCM) that uses instances of product models to capture implicit semantics, and ultimately to find the semantic map. It elaborates how to collect necessary information, and to process them to obtain the map.

Chapter IV explains a procedure to combine multiple views / matching methods to find more accurate semantic maps between the concepts; we propose FEedback Matching Framework with Implicit Training (FEMFIT), a method incorporates expert knowledge to find the underlying relation, thereby the combination of the individual methods which enables more reliable approximation on the semantic map.

Chapter V defines a translation problem for product data, and explains a method toward the translation. The proposed method automatically finds a translation rule for a concept to translate based on a graph search algorithm.

Chapter VI summarizes the research tasks and discusses the expected academic and industrial contributions of this work. It also includes a discussion about the limitations of the present work and future research.

CHAPTER II

Literature review

This chapter discusses related work in the field of concept matching - the focus is on the work in the product domain, but we also include the work in more generic domain when necessary. Next section presents the terminologies that will be used throughout the dissertation, and following sections will discuss specific literature for 3 research topics explained in the previous chapter.

2.1 Terminologies

Product developments systems have their own representations and terminologies are not always shared. In this research, we adopt entity-relationship model [10] to represent the product data throughout the dissertation. Some important terms are:

- *Concept* is an abstraction of the things with common characteristics in a specific domain. It can be something physically recognizable, e.g., *Cylinder* or *Line*, or something conceptual like *Pad* or *Shaft* (operation). Concept has both intensional and extensional aspects; 1) a set of attributes defines a concept (intensional), and 2) the concepts can also be explained with its instances

(extensional).

- *Attribute* is a field associated with a concept that describes a relation to other instances. For example, in CATIA, a CAD system, a feature *Pad* can be defined with a set of attributes *Name*, *Profile* and *Length*. In product domain, we classify attributes into 3 types - character, numeric and reference. In the example, *Name* is a character field that defines a specific name for the *Pad* instance, and *Profile* is a reference field linking to a specific instance of *Sketch* concept, the sketch on which *Pad* is created. *Length* defines the height of the *Pad* instance in numeric value, such as *inch* or *mm*.
- *Instance* is a specific entity which can be characterized with a specific concept; instance of a certain concept can be created by assigning specific values to the attributes of that concept. Thus, when the concept has n attributes, the instance of the concept is of n -tuple. For example, an instance of *Pad* can be given as (“Pad.1”, Sketch.1, 20).

Figure 2.1 shows an example where one product geometry is represented with the explained terms: (a) a product concept hierarchy, (b) product data which instantiates the concepts, and (c) a visual representation of the example product data.

Typically, concept matching problem is defined as to find a *matching* or *corresponding* concept in the receiving system, given a concept in the sending system. Because we discuss matching in the context of translating information across systems, we say concept *A* *matches* or is *correspondent* to concept *B* when *all* instances of *A* can be represented as an instance of *B*.

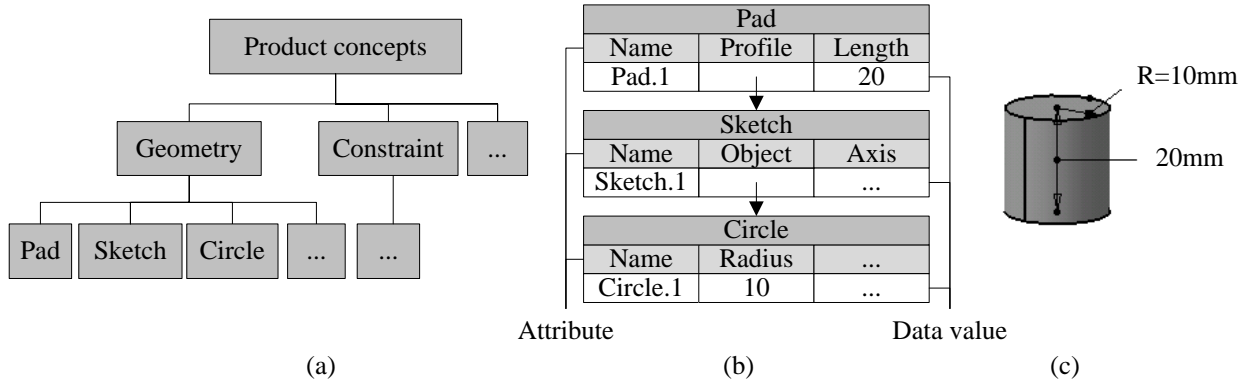


Figure 2.1: (a) An example of product concept hierarchy, (b) Example product data (instance) of *Pad*. Instances have attributes and data values, (c) the visual representation of the product data described in (b)

However, such comparison of instances are virtually impossible; it is not possible to collect *all* instances of a specific concept, and furthermore, there is no formal basis to determine whether two instances in different domains represent the same. Therefore, matchers or matching algorithms find the matching concept based on *similarity* (or *similarity value*) calculation; given a concept in the sending system, the most similar concept in the receiving system is recommended as the matching concept. For this reason, matchers are sometimes called *similarity measure*.

Rahm and Bernstein [11] present a detailed survey of various matchers, and according to the classification, matchers are classified either into *individual* or *combining* matchers; *individual* matchers use one criteria, e.g., concept name or data types for the concept, to determine the similarity, while *combining* matchers combine multiple individual matchers to obtain the similarity based on multiple views or aspects. In this dissertation, for simplicity, we use *matcher* or *matching method* to indicate individual matcher.

Next section reviews the existing work on matching methods.

2.2 Related work: Matching methods

This section discusses related work for the problem 1.a discussed in Section 1.4. More details on the problem are presented in Chapter III.

Rahm and Bernstein [11] classify any (individual) matching methods into either schema-based and instance-based methods. So far, in product domain, all existing work rely only on schema information, e.g., concept names [12, 13], structural (hierarchical) information [7, 14] and concept definition [7, 14, 12, 13] are used for matching.

Schema, however, is not always available detailed enough to find correct matches. Most of the existing work in product domain [7, 14, 12, 13] assume the existence of ontological/complete definitions, which has several problems as discussed in Section 1.3.2. To overcome those issues, therefore, we will develop an instance-based method in this research.

In the Computer Science domain, instance-based methods have been proposed to capture the semantics which schema does not explicitly define. It can be used alone or can be combined with schema-based methods for better accuracy. Although no approach has been found in product domain, there are some work in generic database matching.

Instance-based matching approaches can be classified into two groups depending on the availability of doubly annotated instances, which means the instances in different systems can be identified to be whether equal or not. For example, *Ya-*

hoo!(www.yahoo.com) and *dmoz*(www.dmoz.org), two large Internet directory service pages, have different directory structures. But the specific instances, which is a link to web pages, can be compared across since instances always have a unique *address*.

When the information is available, set theoretic measure is used to calculate the similarity between the concepts, e.g., schema for books [15], genes [16] and the Internet directories [17, 18]. Set theoretic measures, such as Jaccard coefficient [15, 18] and κ -statistics [17], are used to get the similarity from the number of common/uncommon instances. In some cases, the instance matching is not given directly, but machine learning techniques are used to determine the instance memberships and use information theoretic measure to calculate similarity [19, 20].

In this research, however, we cannot compare the instances of product concepts from different product development systems directly, therefore, it is impossible to determine if two instances are common or uncommon. When the annotation is not available, the (mostly statistical) patterns of dataset associated with the concepts are compared. Larson et al. [21] compares concepts, specifically, database attributes, based on their domain relations - {EQUAL, CONTAINS, OVERLAP, CONTAINED-IN and DISJOINT}.

Li and Clifton [22] argue that the evaluation based on domain information is susceptible to noise - small errors could lead to a wrong conclusion, and propose using statistical signatures, e.g., min/max, average, coefficient of variation (CV) and standard deviation (SD). With the signatures, attributes can be projected onto a vector space where dissimilarity can be obtained from the distance between vectors.

However, both techniques ([21], [22]) have been applied to the concepts whose instances are single numeric values. Concepts in product development systems typically have multiple attributes, and the instances of the concepts cannot be compared directly - it is impossible to extract numeric patterns from the instances directly. In this research, therefore, attributes will be compared with the statistical signatures first, and the attribute similarities will be used to measure similarity between the concepts to which the attributes belong. More details are provided in Chapter III.

2.3 Related work: Combination methods

This section discusses related work for the problem 1.b discussed in Section 1.4. More details on the problem are presented in Chapter IV.

Semantic maps can be better approximated by considering more views or aspects, that is, by combining multiple matching methods, and the existing approaches mostly propose multiple matching methods and combine those into a combined similarity measure.

Patil [14] employs a two-level approach: set-based comparison of concept descriptions and semantic contexts from the logical product data ontologies, and then linearly combines the individual metrics to determine the overall semantic similarity.

Lee et al. [12] obtain three individual similarities by comparing ontological concepts using: character strings, ontological definitions, and the Bayesian similarities obtained from the ontological structure. Then, they use a linear combination of these concepts to obtain the overall similarities between concepts across CAD and PDM document data.

Zhang and Gu [23] propose the sequential application of methods based on concept names, class hierarchy and class description. At a given stage, only those concept pairs that have similarities above a certain threshold are selected for further analysis by the next similarity metric.

However, there is no formal procedure to decide weights or thresholds for the linear combination. Moreover, especially in product domain where concepts have relatively complex relations, the relations between the individual matching methods and the overall or combined similarity is not necessarily linear, and the existing combining schemes cannot properly estimate the model to represent the overall similarity. There is a lack of research in the area of product development that has attempted to formalize and determine the requirements on the combination of matching methods.

In the Computer Science domain, there are some approaches using learning method - the relation can be extracted from the user's input on the match result in the training dataset. For example, Doan et al. [19] applied machine learning techniques to decide the weight values for two individual matchers. They used linear regression to fit the training data set where matching pairs are given value 1, and 0 for non-matching pair - the maximum and minimum of the similarity value range. Wang et al. [24] use the same training procedure - positive label for matching pair, and negative label for non-matching pair, but use Support Vector Machine (SVM) rather than linear regression to combine multiple instance based matchers and schema-based matchers.

For those learning methods the labels should be converted to a numeric value

before actual calculation, for example, the similarity with match label is considered to be similarity 1, and 0 for the one with non-match label. In the area of product development, however, such training input leads to *biased* training, that is, all the pairs are biased to the extreme value regardless of their actual similarity. For example, *Extrusion* in NX matches to *Pad* in CATIA. However, *Extrusion* also matches to *Pocket* in CATIA and is not identical to neither *Pad* nor *Pocket*. That is, setting similarity value 1 for the pair *Extrusion* - *Pad* will overestimate the actual similarity. Likewise, *Edge Blend* in NX does not match to *Chamfer* in CATIA, but they share many properties, e.g., based on edge object and requires one numeric value (radius or angle).

Therefore, we believe that assigning similarity value of 0 to such non-matching but similar pair underestimates the actual similarity and such estimation can add up to a large error in a domain where relation among concepts are relatively complicated like product development domain.

In our previous effort [13], to reduce such estimation error, we proposed a matching method based on Support Vector Regression (SVR) where we actually assign continuous values of $[0, 1]$ for training data, instead of 0 and 1. The result has shown that SVR outperforms linear regression in terms of matching accuracy.

However, the new approach creates another issue - it requires domain experts to further provide the actual similarity value, for which no formal basis is available, and it is much harder than just saying match or non-match. In this research, therefore, we propose a method which reduces the estimation error, and at the same time, without

numeric similarity values as user input. More details are provided in Chapter IV.

2.4 Related work: Translating the concepts

This section discusses related work for the problem 2 discussed in Section 1.4. More details on the problem are presented in Chapter V.

In product domain, research focus has been on the concept matching - either individual matching methods or combination methods, but there have been little research efforts toward automating the development of any translators.

In the Computer Science domain, literature documents not many but some research efforts toward ontology translation, e.g., OntoMorph [25], Web-PDDL [26], ODEDialect [27], MBOTL [28]. The common approach is first to categorize the types of heterogeneities and necessary translations. The basic division is between syntactic and semantic heterogeneities, and some with more categories like lexical and pragmatic. For further explanation of the categorizations, refer to Corcho and Gómez-Pérez [27].

The next step is to provide representations for the translation. For example, Corcho and Gómez-Pérez [27] propose a set of declarative languages (ODEDialect) and a list of translation rules in the proposed languages, and Silva Parreiras et al. [28] present another language called MBOTL for a similar purpose.

However, to the best of our knowledge, finding a specific translation rule automatically given a source and target concept is not considered in any approaches, and is left to the manual identification. Therefore, this research will present a method to automate the process to find translation rules given source and the target concepts.

More details are provided in Chapter V.

2.5 Summary

This chapter started with important terminologies used throughout this dissertation - we adopted entity-relation model [10] to represent product data. With the terminologies clarified, we presented related work for each of 3 research tasks discussed in this dissertation:

- **Matching methods:** In the field of product development, research only considered schema-based matching methods with an assumption that concept definitions are available. However, such definitions are only partially available, and furthermore, schema cannot capture the implicit semantics. There are some instance-based matching methods in the Computer Science domain, but those approaches work with rather simple concepts whose instances are single numeric values. Therefore, we need a matching method which uses instance information to find similarity between the product concepts of which instances are represented as a set of attribute values. More details are provided in Chapter III.
- **Combination methods:** In the field of product development, research used weighted average to combine multiple matching methods. But there has been no formal procedure to determine the weight values for each method/measure, and furthermore, there is an intrinsic estimation error due to the linear approximation of nonlinear relation. There have been some approaches in the

Computer Science domain with machine learning approaches, but using the learning methods typically involves an explicit training which further requires extra effort from the user. In this research, therefore, we need a method which can handle the nonlinearity, and at the same time, without such explicit training data set. More details are provided in Chapter IV.

- **Translating the concepts:** In the field of product development, research has focused on the first part of translation - finding semantic maps, but there has been little effort to find the translation rules given the map. In the Computer Science domain, there have been several efforts, but limited to the representation of translation rules. In this research, therefore, we focus on the procedure to find the translation rules automatically given the semantic maps. More details are provided in Chapter V.

The following chapter presents our approach - Instance-Based Concept Matching (IBCM) to address the problems in finding semantic maps.

CHAPTER III

Instance-Based Concept Matching (IBCM)

As discussed in chapter I, there is a need for a matching method that accurately identifies the semantic map between the concepts from the sending and the receiving product development systems. This chapter explains the approach developed in this research to enable the same.

3.1 Motivation

To enable interoperability, standard for data representation is a key requirement [29]. ISO 10303, informally known as STandard for the Exchange of Product model data (STEP), is an international effort toward standardizing the computer-interpretable representation and exchange of product data for engineering purposes. In spite of wide acceptance and success, its coverage is rather limited and important data semantics is lost during translation [14, 30]; there have been efforts toward new standards, such as Core Product Model (CPM) [31, 32, 6] and Product Semantic Representation Language (PSRL) [7].

Given the standard, translation between individual format and standard is also required, and first step toward translation is matching between the concepts from

disparate systems. Finding the mapping manually is a long and error-prone task, and automating the procedure is an area of research in product domain [14, 12, 33, 13], as well as in generic database matching [11, 34, 35, 36].

There are, however, some important challenges remain unsolved with the existing approaches:

- Some concepts have multiple definitions and such concepts can map to different concepts in the receiving system depending on the specific instances. In other words, the exact definition is decided when it is instantiated. Therefore, schema information alone which only deals with explicit semantic cannot remove such ambiguity.
- Attributes explain much about the concepts, and it has been a useful source for the concept matching. However, at the attribute level, there is no such information to use for attribute matching, and explicit semantics are very limited - in most cases, name itself with data types. Most existing methods assumed the matching at the attribute level, but it is not practical in real cases.

Next section presents the objective of this research.

3.2 Objective

The objective of this research phase is to develop a semantic matching method that addresses the challenges discussed in the previous section. We will use instance information, rather than schema information on which existing method heavily depend, to capture implicit semantics of the given representation for accurate identifi-

cation of semantic matching between the concepts. The proposed method should:

- identify the multiple definitions for a given concepts, and find a map for specific definitions.
- find a map at the attribute level using the implicit semantics.

Next section explains the overview of our method: Instance-Based Concept Matching (IBCM).

3.3 Overview of IBCM

This section presents the overview of IBCM, and explains the difference of IBCM from existing work. Figure 3.1 shows the procedure of the proposed method. Overall, the procedure takes instances of sending and receiving systems as input, and calculates the similarities between concept partitions of sending and receiving system. The procedure is divided into 3 steps:

Step 1: Preprocessing

The input to IBCM is a set of instances from sending and receiving system, each tagged with a concept from which it is instantiated. The first step, preprocessing, is further divided into *Concept partitioning* and *Attribute collection*.

Not all concepts have unique definitions; some have multiple definitions and a specific definition is decided when it is instantiated. In *Concept partitioning* process, the concepts with multiple definitions are divided into partitions where each partition has a unique definition. This process removes ambiguities in concepts before matching.

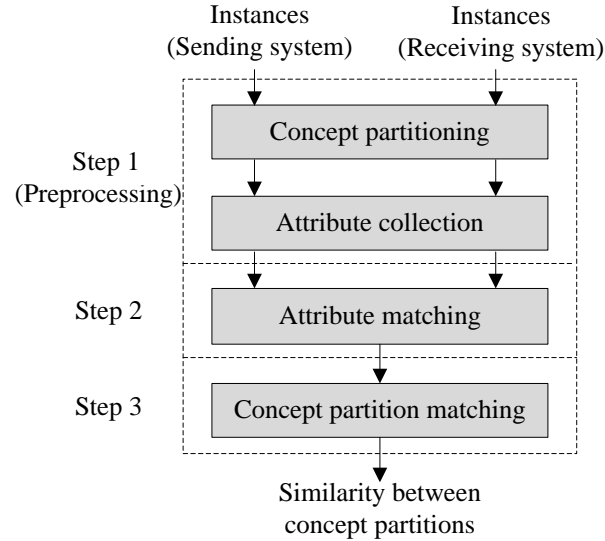


Figure 3.1: Overall procedure of the proposed method. From the instances of sending and receiving system, it calculates concepts similarity through 3 step procedure

Now, each partition has a set of instances belonging to the partition. The instances in the same partition has the same set of attributes (because they have unique definition), and *Attribute collection* process collects all the attributes as a set of values; this prepares the next step - attribute matching.

Step 2: Attribute matching

From the previous step, partitions are identified with a set of attributes, and each attribute can be characterized as a set of data values. In this step, *Attribute matching*, attribute similarities between the attributes from the partitions of sending and receiving systems are calculated.

Each attributes are described as a set of values extracted from the instances, and the calculation is based on the implicit semantics captured from the instance information, rather than using explicit semantics which is very limited - mostly,

attribute names or data types.

The output of the process is the similarities of all the pairs of attributes.

Step 3: Concept partition matching

From the previous step, attribute similarities are obtained for all the pairs of attributes from sending and receiving systems. In this step, *Concept partition matching*, similarities between the (concept) partitions are calculated.

In the process, attribute similarities are adjusted as *weighted* attribute similarities reflecting the relative importance of the attributes, and based on those, attribute map will be identified between two partitions. The map is the outcome of the process in addition to the similarity value between the partitions.

3.4 Details of IBCM

This section describes the 3 components of the proposed method.

3.4.1 Preprocessing

In preprocessing stage, attributes and its data values are collected from the instances after clustering the instances into the homogeneous groups through 1) concept partitioning, and 2) attribute collection.

Concept partitioning

Each concept in the sending and receiving system has a set of instances associated with it. Instances of the same concept does not always have the same set of

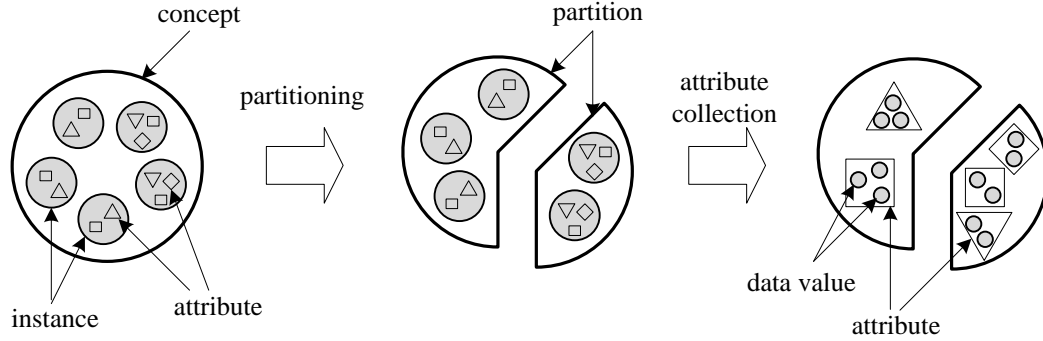


Figure 3.2: Preprocessing of input into the attributes and its data values. First, instances of a concept are partitioned so that instances in the same partition have the same set of attributes (partitioning). Now, each partition can be represented as a set of attributes of which each attribute has a set of data values (attribute collection).

attributes because concepts may have more than one definitions. For example, a concept *Pad* can be defined as $\langle Name, Profile, Length \rangle$, or it can also be defined as $\langle Name, Profile, Profile \rangle$, *Profiles* being upper and bottom one respectively. Therefore, instances of *Pad* have either set of attributes. Figure 3.2 includes schematic of the partitioning process.

In concept partitioning, instances of a concept with more than one definitions are partitioned so that within same partition, all instances have the same set of attributes. Now, each concept partition can be characterized with a unique definition - a fixed set of attributes.

In existing work, either in product domain [7, 14, 12, 13] or in general case matching [36, 34, 35, 11], a concept is assumed to have a unique definition, and thereby, do not require such partitioning process, and thus cannot properly handle the cases described above.

Attribute collection

After the partitioning, each partition has one or more instances, say m instances. When the partition has a fixed definition, say a set of n attributes, each instance is n -tuple, which can be represented as a column vector of n -dimension. Now, a data value matrix V can be created by concatenating those m vectors. Then, the matrix V is given as:

$$V = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1m} \\ v_{21} & v_{22} & \cdots & v_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ v_{n1} & v_{n2} & \cdots & v_{nm} \end{bmatrix},$$

where v_{ij} is data value for i^{th} attribute of j^{th} instance. Then, data values for i^{th} attribute is a *multiset*, a set with possible repetition in members, of all the elements in i^{th} row in the matrix V . Figure 3.2 includes schematic of the collecting process. Next section explains the procedure to calculate attribute similarity from the attributes and data values.

3.4.2 Attribute matching

From the preprocessing, a set of attributes are identified for each partition, and one or more data values are associated with each attribute. In this step, attribute similarity is calculated with the data values of the attributes. The problem can be

stated as:

For: attribute a with a set of data values from sending system and,

attribute b with a set of data values from receiving system

Given: two multisets of data values associated with a and b respectively

Find: attribute similarity, $T(a, b) \in [0, 1]$

The overall idea of calculating attribute similarity is that two attributes are similar, i.e., higher similarity value, when they have *similar* multisets. In product domain, we classify attributes into 3 categories: 1) numeric, 2) character and 3) reference attributes. We will first explain how the similarity of the attributes of the same types can be calculated for each of 3 types, and then discuss the similarity between the attributes of different types.

Numeric attribute

There are attributes with numeric values as its data, such as *length* or *angle* attributes. IBCM constructs signature vector for each attribute, and the similarity between attributes are approximated by comparing the vectors. Each element in the vector should capture a certain aspect of the attributes.

Specific signatures should be decided based on specific domain problem, but there are some general statistic signatures for this purpose such as min/max, average, coefficient of variation (CV) and standard deviation (SD) as explained in [22]. Then, from each attribute, which we represent as multiset, signature tuple or vector can be constructed, and the distance between two signature vectors can be used to approx-

imate the *dissimilarity* between two attributes.

There are different measures for similarity or dissimilarity between vectors such as Euclidean distance and cosine similarity. In this research, we use standardized Euclidean distance (SED) [37] as dissimilarity measure. The measure is different from Euclidean distance in that square of each signature is inverse weighted by its variance before added to a sum. This prevents the measure from too sensitive to one or a few coordinate with larger scale.

The range for standardized Euclidean distance (SED) is $[0, \infty]$, and it is normalized into $[0, 1]$ as:

$$\text{Normalized SED}(a, b) = \frac{SED(a, b) - SED_{min}}{SED_{max} - SED_{min}}$$

where SED_{max} and SED_{min} is the maximum and minimum distance for all possible pairs of attributes. Now, similarity T of range $[0, 1]$ is obtained as:

$$T(a, b) = 1 - \text{Normalized SED}(a, b)$$

Specific procedure with a case example is shown in Section 3.5.

Character attribute

There are attributes with character values as its data, such as *name* or *description*. Although there are many methods to compare textual information, e.g., string matching [38, 39, 40] or synonym matching [39], the objective here is to find a similarity between two (multi) sets of character data, rather than comparing two individual strings.

To calculate the character attribute similarity, we use the same procedure used for numeric attribute similarity. Initially, statistic signatures cannot be found for textual information, but Li and Clifton [22] proposed to use “number of bytes actually used to store data” as a numeric equivalent, and the method can be adapted here.

Reference attribute

Reference attributes takes an instance of another concept as its value. For example, *Pad* concept has an attribute that takes a *Sketch* instance. For the reference attribute similarity, we use recursive evaluation of concept similarity to which the attributes are referring. For example, an attribute a in *Pad* takes *Sketch* and an attribute b in *Extrude* takes *Section*, the attribute similarity between a and b is drawn from the concept similarity between *Sketch* and *Section*.

In the proposed method, the similarity between two attributes of different types, e.g., similarity between numeric and reference attributes, is considered to be zero. Although there might be exceptions, e.g., an attribute “reason to change” in engineering change management can be in the form of either free form text, or pre-defined enumeration which is numeric depending on specific systems, we believe that such cases are negligible in number and would not affect the overall performance.

3.4.3 Concept partition matching

In this step, similarity between concept partitions is calculated from attribute similarity. Existing work [7, 14, 13] assume that sending and receiving systems share

the same attributes and that attribute matching is explicitly available, e.g., only the attributes of same name are considered to be equivalent. When the matching is available, set theoretic measure, such as Jaccard coefficient or a measure by Tversky [41], can be used to get the concept similarity.

In the proposed method, we do not have the assumption, and the partition similarity will be calculated with attribute similarity obtained from the previous step. In this step, we first calculate information content of each attribute and update the similarity such a way that the similarity between the attributes of more information content is weighted more. Then, based on the *weighted* attribute similarity, we find the attribute pairing by using the solutions for assignment problem, such as Hungarian algorithm [42, 43] and Gale-Shapley algorithm [44].

Based on the identified pairs, we propose a new measure to calculate the partition similarity, which obtains overall similarity from a set of similarity pairs. The matching problem is stated as:

For: concept partition A with attribute a_1, a_2, \dots, a_p from sending system and,
concept partition B with attribute b_1, b_2, \dots, b_q from receiving system

Given: attribute similarity, $T(a_i, b_j) \in [0, 1]$ for all $i = 1, 2, \dots, p$ and $j = 1, 2, \dots, q$

Find: concept partition similarity, $S(A, B) \in [0, 1]$, and
attribute map between the partitions

The overall idea of calculating the partition similarity is that two concept partitions are more similar when they have more similar attributes. To obtain a method implementing the idea, we first define the boundary conditions, i.e., conditions for

similarity value at 1 and 0:

$$S(A, B) = \begin{cases} 1 & \text{if } p = q \text{ and there exists a permutation } s \text{ on } b_i\text{s} \\ & \text{such that } T(a_i, s(b_i)) = 1 \text{ for all } i = 1, 2, \dots, p \\ 0 & \text{if } T(a_i, b_j) = 0 \text{ for all } i = 1, 2, \dots, p \text{ and } j = 1, 2, \dots, q \end{cases}$$

Now, we propose a method that works for the ranges in between without violating those boundary conditions. The method consists of three phases: 1) it first calculates weight for each attribute and obtains weighted attribute similarity, 2) finds the pairing (or map) between attributes a_i s and b_j s based on the weighted similarity, and then, 3) calculates partition similarity from the pairing.

Weighted attribute similarity

Each partition, in general, has multiple attributes, but not all of them are equally important in describing the concept. For example, “Radius” and “Edges” attributes are indispensable to define a concept *EdgeBlend* in NX, while the feature can be further controlled with optional attributes like “Roll Over Smooth Edges (ROSM),” which does not exist in the corresponding concept in other systems, such as *EdgeFillet* in CATIA.

In this procedure, attributes are weighted according to its relative importance so that important attributes are more emphasized when finding the attribute pairing. We approximate the importance of the attributes with Shannon’s entropy function H [45], a measure for the information content of a given variable - more weights are assigned to the attributes with more information content. The rationale is that less important attributes tend to remain unchanged over different models set to the

default value for most of the cases. For example, when 95 features have the option “Roll Over Smooth Edges (ROSM)” out of 100 “EdgeBlend,” the information content for the attribute is given as:

$$H(ROSM) = - \sum_{i=1}^n p(x_i) \log p(x_i) = -(0.95 \times \log 0.95) - (0.05 \times \log 0.05) = 0.086$$

So far the value itself, 0.086 in the example above, is not informative in itself; the entropy values will be normalized with the sum of the entropies for all attributes. We define a normalized entropy as $H'(a_i) = H(a_i) / \sum_{i=1}^p H(a_i)$.

Then, the weighted attribute similarity $T'(a_i, b_j)$ is the attribute similarities $T(a_i, b_j)$ multiplied by the average of the normalized entropy $H'(a_i)$ and $H'(b_j)$ for the involved attributes. That is,

$$T'(a_i, b_j) = \frac{H'(a_i) + H'(b_j)}{2} \times T(a_i, b_j)$$

where $H'(a_i) = H(a_i) / \sum_{i=1}^p H(a_i)$ and $H'(b_j) = H(b_j) / \sum_{j=1}^q H(b_j)$.

Next section explains attribute pairing based on the weighted similarities.

Attribute pairing

The objective of attribute pairing is to find a matching between two sets of attributes that maximizes the total similarity of chosen pairs. This is assignment problem which is to find a maximum weight matching in a weighted bipartite graph - attributes becomes vertices with edges between vertices have the weighted similarity value between the attributes.

In this research problem, number of attributes of compared concepts are different in general ($p \neq q$). Because this is a *unbalanced* case of assignment problem, dummy

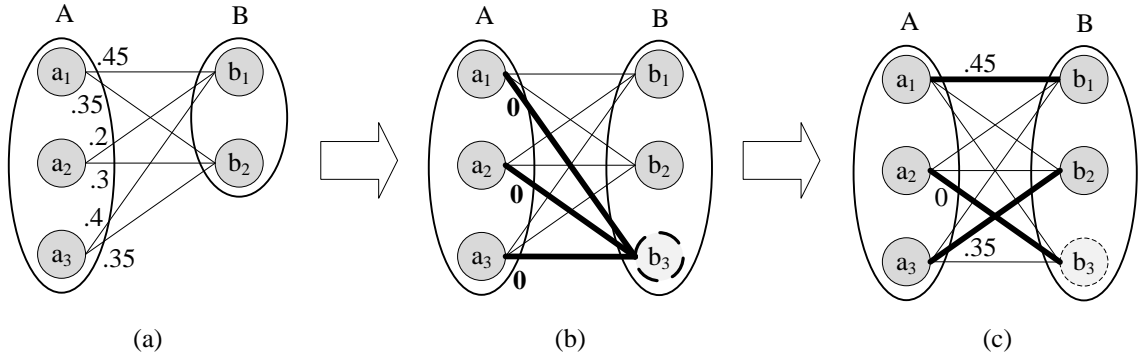


Figure 3.3: Attribute pairing as bipartite graph matching: (a) (weighted) similarity values are assigned to the edges in a bipartite graph connecting two disconnected sets of vertices, i.e., attributes of concept partition A and B , (b) dummy attribute b_3 has been added with all new similarity with 0, and (c) pairs have been found such that the similarity total is maximized, i.e., 0.8 in this case.

attributes can be added to make it a balanced case [46]: when $p > q$, dummy attributes of $b_{q+1}, b_{q+2}, \dots, b_p$ are added to the set of attributes of receiving system. Since the attributes are dummy, information content for the attributes are set to 0, and also assign 0 to the weighted similarity connected to the attributes, that is, $H'(b_j) = 0$ and $T'(a_i, b_j) = 0$ for all $i = 1, 2, \dots, p$ and $j = q + 1, q + 2, \dots, p$. It can be handled in the same way when $p < q$. Figure 3.3 demonstrates the process of adding dummy attribute when $p = 3$ and $q = 2$.

Although the whole solution space for the problem is $n!$ where $n = \max(p, q)$, there are known polynomial time solutions to the problem; Hungarian algorithm finds the maximum weight matching, i.e., an optimal solution, in $O(n^3)$, of which first version was developed by Kuhn [42], known as Hungarian method, and was later revised by Munkres [43]. Figure 3.3 shows the final pairing after using the algorithm with the example. Detailed explanation for the algorithm is found in [43].

When problem space is large and time becomes an issue in solving the problem, following alternatives can be considered. Although they do not guarantee the optimal solution, local solutions can be returned in faster time:

- Greedy matching: In the bipartite graph, the attribute pair with the highest similarity value is selected for matching. After storing the pair, both vertices are removed from the graph. It iterates the process until there is no more vertices and edges in the graph. It returns matching in linear time ($O(n)$).
- Gale-Shapley algorithm [44]: The algorithm is known for the solution for stable marriage problem (SMP). When the algorithm is executed, all the attribute pairs become “stable,” which means, in the resultant one-to-one alignment M , any pairs $\langle a_i, b_j \rangle \in M$ and $\langle a_l, b_m \rangle \in M$ satisfy the condition that $T'(a_i, b_j) + T'(a_l, b_m) \geq T'(a_i, b_m) + T'(a_l, b_j)$. It returns stable matching in quadratic time ($O(n^2)$) [47].

The solution (matching) of this step can be represented as a permutation s on $\{b_1, b_2, \dots, b_r\}$ where a_i matches to $s(b_i)$ for all $i = 1, 2, \dots, r$, where $r = \max(p, q)$.

Similarity calculation

Given the bipartite graph and matching obtained from the attribute pairing, we define the similarity as a ratio of the total attribute similarity to a maximum similarity for the given size of bipartite graph:

$$S(A, B) = \frac{(\text{total attribute similarity})}{(\text{maximum attribute similarity})} = \frac{\sum_{i=1}^r T'(a_i, s(b_i))}{\sum_{i=1}^r \left(\frac{H'(a_i) + H'(s(b_i))}{2} \times 1 \right)} = \sum_{i=1}^r T'(a_i, s(b_i))$$

where $r = \max(p, q)$.

Through the procedures described so far, similarity between two partitions can be obtained. For a partition in a sending system, the procedures can be iterated with all the partitions in the receiving system to find a corresponding partition with which the highest similarity is achieved.

Going back to the original research problem, given a concept in a sending system, we can find a set of partitions for the concept, and for each of them, we can retrieve all corresponding partitions from the receiving system. Therefore, we find not only the corresponding concept(s) in the receiving system, but also the matching between the attributes with which translation rule can be further identified. Brief example to find initial translation will be presented at the end of case study in next section.

3.5 Case study

We are not aware of other research/literature in this area that could be used to evaluate our approach objectively. Therefore, for this chapter, we develop a case study, and evaluate the resulting semantic maps manually.

Product development involves various types of information such as requirement, bill of material and geometry information. In this case study, we demonstrate our method in matching product concepts for geometry information. Geometry information is created and managed with computer software, i.e., CAD software, for example, CATIA, NX or Pro Engineer. However, all those software use different data format, and the data created in one software cannot be directly used in other software. Fig-

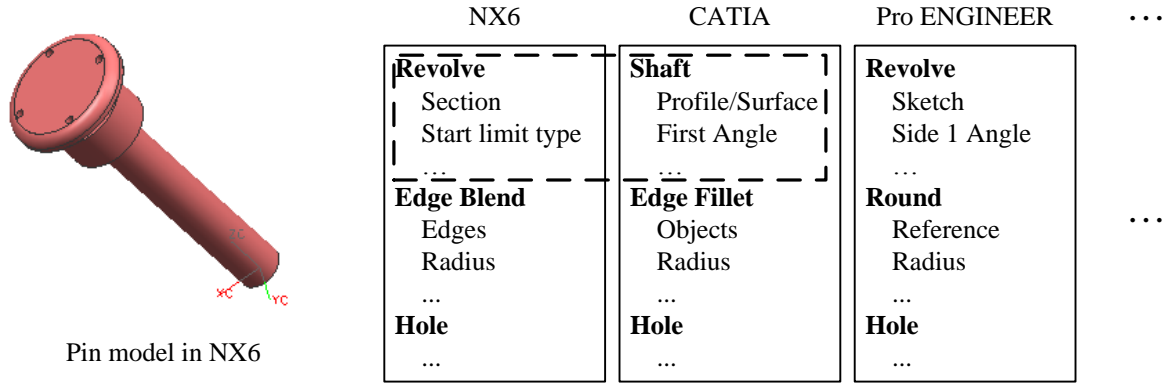


Figure 3.4: Pin model created in NX6 is shown in the left. The models can be represented differently in different CAD software, and the representations - concept names and attribute names - are shown for NX6, CATIA and Pro Engineer. The enclosed area with dotted line, i.e., matching *Revolve* (NX6) and *Shaft* (CATIA), shows the specific example dealt with in this case study.

Figure 3.4 shows a pin model created in NX6, and the required features and attributes for the model in different software to represent the same model.

ISO 10303, informally known as STEP, provides neutral format which enables translation of product information between different CAD software. However, it only translates the boundary representation of geometry, and the feature and attribute information is lost during the translation. In this section, we will demonstrate the procedure to find the matching between feature and attributes concepts across different CAD software. Specifically, we will use the example case of *Revolve* (NX6) and *Shaft* (CATIA V5).

3.5.1 Preprocessing

To collect the instance information of *Revolve* (NX) and *Shaft* (CATIA), programs have been developed with NX6 and CATIA V5 automation API to extract specific

feature and attribute information from the CAD files. In this case:

- To collect *Revolve* instances, 3437 NX models have been checked. Among them, 92 models contain *Revolve*, and were used in the case study.
- To collect *Shaft* instances, 4036 CATIA models have been checked. Among them, 1539 models contain *Shaft*, and were used in the case study.

The CAD files are the sample files come with each CAD software during installation.

After separating the files with specific features, i.e., *Revolve* (NX) and *Shaft* (CATIA), the feature information is extracted and is partitioned into groups in a way that each group has a set of instances with same set of attributes. Figure 3.5 shows the result of the partitioning of *Revolve* instances. From 92 models, 122 instances are extracted, and the instances are divided into 3 partitions. Same procedure is used for *Shaft* instances but not shown here for space reason.

Once the instances are partitioned and represented as tabular format as in Figure 3.5, attribute collection is just to collect the values of each column in the table. In the example, there are 3 partitions, and each has a set of attributes, which is not mutually exclusive.

3.5.2 Attribute similarity

This section shows attribute similarity between the partition defined by the first group of *Revolve* concept (first out of 3 tables in Figure 3.5), and the attributes of *Shaft* partition in CATIA which is not shown for brevity.

The compared attributes have two types: numeric and reference attributes. To get similarity between numeric attributes, standardized Euclidean distance is used

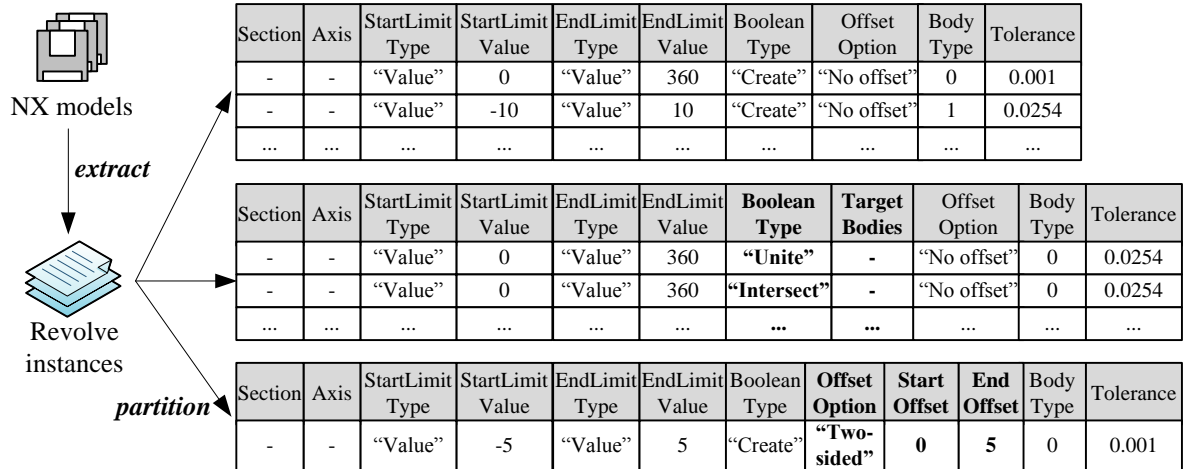


Figure 3.5: Partitioning of *Revolve* instances. From NX model files, *Revolve* features are extracted, and then the features are divided into the partitions in a way that each partition has a set of instances with same set of attributes. In the tables, some attributes are bold faced to show the difference from other partitions. In the value fields, quotation is used to denote the enumeration attributes, and '-' is used to denote the reference to another instance.

between 5-tuple signature for each attribute, i.e., $\langle \text{Min, Max, Average, Standard deviation, Number of distinct values} \rangle$. Figure 3.6 shows (a) the tuples for each attribute of *Revolve* and *Shaft* partition, and (b) similarity matrix between the attributes. Note that there are attributes with zero entropy, e.g., *Thickness 1* and *Thickness 2* for *Shaft*. Those attributes either have same value or are not used at all for all the instances considered. They are not shown in the figure for space reason, and does not affect the ensuing calculation because it makes the weighted attribute similarity zero all the time. Entropy calculation for each attribute is given in the next section. Similarity between reference attributes are calculated recursively with the concepts the attributes refer.

Next section calculates partition similarity based on the attribute similarities.

		Min	Max	Average	Std. deviation	# distinct values
Revolve	StartLimitValue	-30	0	-1.26	4.63	4
	EndLimitValue	5	360	323.70	104.17	5
	BodyType	0	1	0.12	0.33	2
	Tolerance	0.001	0.03	0.02	0.01	2
Shaft	FirstAngle	0	445	359.77	10.39	4
	SecondAngle	-95	355	0.10	9.39	3

(a)

		<i>Shaft</i>			
		First Angle	Second Angle	Sketch	Axis
Revolve	StartLimitValue	0.70	0.99	0	
	EndLimitValue	1	0		
	BodyType	0.51	0.67		
	Tolerance	0.51	0.67		
	Section	0		0.8	0.2
	Axis			0.2	1

(b)

Figure 3.6: (a) Signature tuple for attributes of the partitions from *Revolve* and *Shaft* concept, (b) similarity matrix between the attributes. Note that the upper left values are similarity between numeric attributes and were calculated from the signature tuples shown in (a). Lower right values are similarity between reference attributes and from the recursive evaluation of concept (partition) similarity. Similarity between numeric and reference attributes are all set to zero.

3.5.3 Concept partition similarity

From the attribute similarity obtained in the previous step, partition similarity is calculated. We first calculate entropy and normalized entropy for each attribute and get weighted similarity between attributes. The result are shown in Figure 3.7 (a); for each attribute, entropy H and normalized entropy H' is shown. Note that the attributes with 0 entropy are not shown for brevity. With the normalized entropy, we calculate the weighted similarity between attributes (shown in Figure 3.7 (b)). This table can be represented as a bipartite graph, and Hungarian algorithm [43] is applied to find a optimum pairs, which is to maximize the sum of the weighted similarity. The identified pairs are shown in the figure as shaded cells. Note that Gale-Shapley algorithm [44] and greedy matching we have briefly explained in Section 3.4.3 also find the same result in this specific case. The concept similarity from this pairing is

		H	H'
Revolve	StartLimitValue	.388	.150
	EndLimitValue	.474	.184
	BodyType	.370	.143
	Tolerance	.403	.156
	Section	.474	.184
	Axis	.474	.184

		H	H'
Shaft	FirstAngle	.020	.267
	SecondAngle	.015	.200
	Sketch	.020	.267
	Axis	.020	.267

(a)

		Shaft					
		First Angle	Second Angle	Sketch	Axis	Dummy	Dummy
Revolve	StartLimitValue	.146	.174	0	0	0	0
	EndLimitValue	.225	0	0	0	0	0
	BodyType	.105	.115	0	0	0	0
	Tolerance	.108	.119	0	0	0	0
	Section	0	0	.180	.045	0	0
	Axis	0	0	.045	.225	0	0

(b)

Figure 3.7: (a) Entropy H and normalized entropy H' for the attributes of compared partitions. Attributes of 0 entropy are not shown in the table for brevity. (b) Weighted attribute similarity calculated from attribute similarity and normalized entropy. From the similarity matrix, identified pairs are shown as shaded cells.

the sum of the similarities of the identified pairs, which is:

$$\begin{aligned}
 S(\text{Revolve}, \text{Shaft}) &= T'(\text{StartLimitValue}, \text{SecondAngle}) \\
 &\quad + T'(\text{EndLimitValue}, \text{FirstAngle}) \\
 &\quad + T'(\text{BodyType}, \text{Dummy}) + T'(\text{Tolerance}, \text{Dummy}) \\
 &\quad + T'(\text{Section}, \text{Sketch}) + T'(\text{Axis}, \text{Axis}) \\
 &= .174 + .225 + 0 + 0 + .180 + .225 = \mathbf{0.804}
 \end{aligned}$$

The similarity between the first partition of *Revolve* and a partition from *Shaft* is determined as 0.804 through the procedure. Although this value itself does not finalize the matching, same procedure can be applied for the *Revolve* partition with

other concept partitions from CATIA to get the pair with the highest similarity, which will be the corresponding partition for the given *Revolve* partition.

3.5.4 Discussion

In this case study, we have presented a procedure to calculate a similarity between two partitions - *Revolve* and *Shaft*. Once the procedure is applied to all other pairs of partitions, complete similarity matrix between partitions of sending and receiving systems can be constructed and the matching map is obtained. At this point, however, such complete evaluation cannot come easily; accessibility to the CAD data in both software is only partially available through APIs, and converting them into a standard format, entity relationship model, in this specific case, for further processing takes much manual effort. However, there is effort toward more open CAD format, such as PLM XML by *Siemens* and 3D XML by *Dassault Systemes*, and it would enable further automation of the proposed approach for complete matching procedure.

Once the correspondence between the partitions are identified, we have useful information for the translation. Figure 3.8 shows an example where *Revolve* instance in NX is translated into an instance of *Shaft* instance in CATIA, and vice versa. When *Revolve* is translated into *Shaft*, some attributes are lost during the process, but necessary information for *Shaft* is conserved. In the reverse direction, 4 attributes are created automatically during translation - *StartLimitType*, *EndLimitType*, *BooleanType* and *OffsetOption*. The information is not explicit in the original instance, but all the instances in the corresponding partition have the same data val-

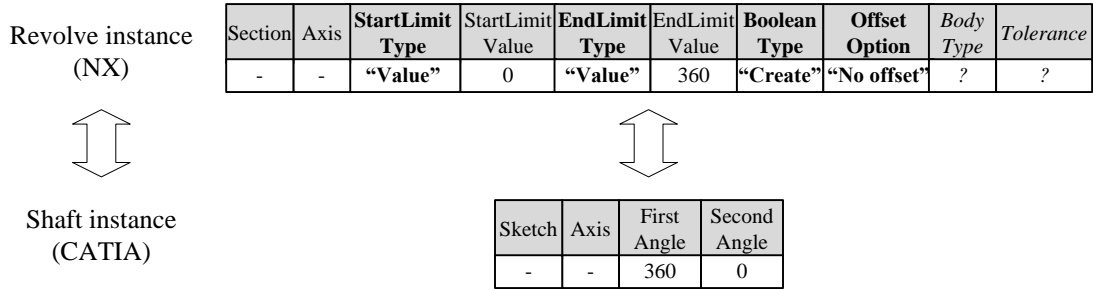


Figure 3.8: Translation between *Revolve* instance (NX) and *Shaft* instance (CATIA). In translating *Revolve* into *Shaft* attribute information is lost - 6 attributes shown as either boldfaced or italicized. In the reverse direction, 4 attributes (boldfaced) are created automatically with correct values, but values for 2 attributes (italicized) cannot be determined. Note that further information for reference attributes are omitted and shown as '-'. Those are additional instances and can be found through further translation of the instances.

ues for those attributes and they can be assigned accordingly. Of course, not all the attributes can be filled up; in the figure, the values are undecided for two attributes, *BodyType* and *Tolerance*. Once the attribute maps are identified, we evaluate their correctness using our domain knowledge. Such an approach is necessitated, because, to be best of our knowledge, there is no standard data repository to enable a comparative evaluation nor documented results from efforts that might have addressed this problem in this domain.

In Section 3.4.2, we explained similarity measures for numeric, character and reference attributes. In the case example, however, there is no character attributes used (enumeration type is considered to be non-negative integer, thus numeric attribute). In CAD domain, character attributes are very rare and, if any, it is created automatically, e.g., feature names like *Revolve.1*. We presented some guidelines for the character attributes, and specific measures can be determined with respect to the

specific domain problem. The attribute similarity measure integrates into the overall method as a module, thus, there will not be any issues in incorporating different measures if needed.

3.6 Summary

Identifying a semantic map between the concepts is the first step toward the integration of heterogeneous systems. So far, existing matching methods have used schema or definitional information to find the map.

Schema information, however, does not contain implicit semantics of the data and certain aspects cannot be captured with schema only information. In this research, we have developed an instance-based matching method that can detect 1-to-n concept maps. Use of instance information enables capturing the implicit semantics; concepts with multiple definitions can be divided into partitions of unique definition. This clarifies the map for the concept with respect to the specific instance that the matching and ultimately translation is aimed at. Furthermore, use of instance information also provides resources for the attribute matching while existing methods typically assume the availability of the map at the attribute map.

The proposed method, however, does not override the existing work; rather, it extends the problem so that it can approach the matching problem as a whole with more realistic conditions. Also, the performance of the proposed instance-based matching techniques would be improved when combined with existing schema-based techniques. Next chapter discusses the method to combine multiple matching methods for more accurate identification of the semantic map.

CHAPTER IV

FEedback Matching Framework with Implicit Training (FEMFIT)

As discussed in chapter I, there is a need for a method that accurately combines the result of the multiple individual matchers to better approximate the semantic similarity between the concepts from the sending and receiving product development systems. This chapter explains the approach developed in this research to enable the same.

4.1 Motivation

Literature documents the role of ontologies and standards in semantic interoperability for product data interoperability [29]. Ontologies are typically used to explicitly capture and represent semantics of information, thus enabling seamless connectivity across different applications [48]. The first step toward semantic interoperability with ontologies is to find a semantic maps between the concepts from different ontologies [35].

In the domain of product development, approximate semantic similarities are determined using different attributes or features derived from the disparate components

of product data. In addition, different methods or metrics can be applied to calculate the similarity values. At the same time, there is no formal basis to define a correct approach and the results may not be reliable. Therefore, there is a need to consider a variety of features along with the variety of methods to approximate similarity.

There have been approaches to combine the multiple individual measures but not without limitations, including:

- Linear combination has been used extensively for the combination, but the determination of the weight values for each measures is left completely to the expertise of the user without any formal basis. Furthermore, the relation between the multiple matching methods are not necessarily linear, and nonlinear relations cannot be captured properly.
- Existing approaches with learning methods can handle the nonlinearity issue, but they require massive training data to be generated from domain experts manually.

Next section presents the objective of this research.

4.2 Objective

The objective of this research phase is to develop a method to combine the result of the individual matchers to better determine the similarity between the concepts which leads to a better mapping between the product concepts. The requirements for the intended method include:

- Capturing nonlinearity: The method should be able to capture the possible

nonlinearity between the individual measures for accurate approximation of the true similarity between the concepts.

- Minimizing user input: The method should work with only minimal input from the domain expert, and the procedure should be formally defined.

Next section defines the specific research problem.

4.3 Product concept matching problem

The product concept matching problem is described as:

Given: a product concept $c_s \in C_s$, a source ontology

Find: a concept, $c_t^* \in C_t : S^*(c_s, c_t^*) \geq S^*(c_s, c_t), \forall c_t \in C_t$, a target ontology

where S^* is a function that calculates the “true” similarity between two product concepts, source concept c_s and target concept c_t , based on which matching should be decided. However, such function, in general, does not exist as an analytical formula, and the problem is to find a function S that approximates the function S^* .

Literature documents various methods that calculates the similarity between concepts [11, 34, 35, 36]. Rahm and Bernstein [11] divide the methods into two groups: individual matchers and combined matchers. Individual matchers are similarity functions that use a specific measure for a specific part of concept information. In most cases, however, one individual matcher cannot successfully approximate the function S^* , and combined matcher is used for better approximation. Combined matcher is a combination of $k \in \mathbb{N}$ individual matchers, and when $S_i(c_s, c_t)$ denote the i^{th}

individual matcher where $1 \leq i \leq k, i \in \mathbb{N}$, the combined matcher S is described as:

$$S(c_s, c_t) : S_1(c_s, c_t) \times S_2(c_s, c_t) \times \dots \times S_i(c_s, c_t) \dots \times S_k(c_s, c_t) \rightarrow [0, 1]$$

However, even the combined matcher cannot approximate the true similarity for all cases, and so far, no matching method has claimed complete automation. Therefore, based on the result of the combined matcher, human, a domain expert, should finalize the match for any given product concept.

At the same time, although there are many individual matchers and combination methods are available, it should be an domain expert to select the methods because the procedure requires specific domain knowledge. Therefore, the overall procedure to match product concepts can be divided into 3 sub-parts:

Step 1: Selecting individual matchers There are many matching algorithms are available, and the user, a domain expert, has to choose a set of individual matchers that can capture the specific type of similarity based on which the correspondence will be decided later by the user. For example, sometimes string matching algorithm is enough for the problem, but sometimes synonym matcher or more complicated ones are also required.

Step 2: Selecting combination method The user also has to choose a combination method which combines multiple similarity measures into one similarity value. Again, there are multiple methods are available, which we explain further in the next section, and the user has to choose the most appropriate method for their specific domain problem. The combined similarity is expected to approximate the true similarity based on which the correspondence should be decided.

Step 3: Finalizing the match Once the combined similarities for all pairs are calculated, the user has to finalize the match. Because the matching algorithm cannot always calculate the true similarity correctly, users are not necessarily to choose the pair with the highest combined similarity. However, any “good” matching algorithm is capable of good approximation, and it is expected for the user to find the corresponding concepts by checking only high-ranked concepts.

The focus of this research is on the second step - combination method. We believe that a good combination method makes easy not only Step 3, but also reduces load for Step 1.

Chapter II presented related work regarding combination methods. The proposed method will be distinguished from the existing work in following aspects:

- Compared to the work in product development domain, it provides formal basis to determine the relation among the measures. Also, it is able to handle both linear and nonlinear relations.
- Compared to the learning methods used for concept matching in generic domain, it does not under or overestimate the input from the domain expert, and this leads to more accurate estimation.
- Compared to our previous work, it does not require any explicit training. That is, the domain expert does not have to provide the similarity value for massive training data.

After explaining nonlinearity issue in the next section, we will propose a match-

ing framework which has better accuracy with no additional cost to the users in Section 4.5.

4.4 Nonlinearity in combination

As explained earlier, multiple similarity measures are combined for better approximation of the semantic similarity; we call this combined measure as combined or overall similarity. In this section, we claim that the overall similarity represents a nonlinear combination of the individual similarities due to the following reasons:

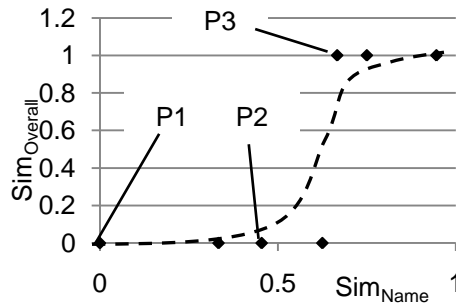
- **Nonlinear relation between individual and combined similarity** Various contributors to the similarity may not be necessarily linearly-related to the overall similarity.

For example, Figure 4.1 demonstrates the issue in comparing concepts that capture 2-dimensional (2D) constraints across the CAD software CATIA and NX. Relevant concepts in CATIA include *Radius*, *Tangency*, while concepts in NX include *Perpendicular*, *EqualRadius*, *RadiusDim*, *Tangent* and others that are not shown in Figure 4.1(a) for readability. The individual similarity (Sim_{Name}) is calculated using the edit distance proposed by Wagner and Fischer [49]. The overall similarity ($Sim_{Overall}$) is obtained from expert knowledge. Different pairs of concepts are denoted by P_1, P_2, \dots as shown in Figure 4.1(b). Figure 4.1(c) shows that the individual name-based similarity is not necessarily linear to the overall similarity. Therefore, the overall perception of similarity is such that a linear combination that uses Sim_{Name} will not give an accurate



(a) Some concepts representing 2D constraints in CATIA and NX

Point	CATIA Concept	NX Concept	Sim_{Name}	$Sim_{Overall}$
P1	Radius	Perpendicular	0	0
P2	Radius	EqualRadius	0.4545	0
P3	Radius	RadiusDum	0.6667	1
...

(b) Similarity values obtained by individual similarity, Sim_{Name} using edit distance, and overall similarity $Sim_{Overall}$ given by the expert

(c) Nonlinearity of overall similarity in the context of the individual similarity

Figure 4.1: Relation between an individual name similarity measure and the overall similarity measure on the constraint concepts in CATIA and NX. A nonlinear function is required to capture this relation correctly.

representation of the semantic similarity, since we see that there is a certain threshold below (or above) which the perceived difference between of the similarity values is not really the mathematical distance obtained by using the metric.

- **Dependent measures** Several times, some individual similarities are conditionally dependent on other individual similarities. These conditional depen-

CATIA	NX	Sim_{Name}	Sim_{Dim}	$Sim_{Overall}$
Parallelism (in 2D)	Parallel (in 2D)	0.73	1	1
Parallelism (in 2D)	Parallel (in 3D)	0.73	0	0

Figure 4.2: Conditional dependency of Parallelism constraint in CATIA and Parallel constraint in NX. The similarity measure that considers the dimension of the space is necessary before considering the name-based similarity measure

dencies must be captured because they represent significant semantic interrelationships. However, linear combinations cannot capture these relationships.

For example, Figure 4.2 shows two basic concepts, *Parallelism* in CATIA and *Parallel* in NX that restrict the relation between different geometric elements. Both apply to 2D and 3-dimensional (3D) objects. So, a name-based matcher indicates that they are very similar in every single instance (with a value of $Sim_{Name} = 0.73$). However, this value is insignificant in the absence of the similarity measure, Sim_{Dim} that captures the dimensionality of the space, viz., 2D or 3D. If $Sim_{Dim} = 0$, then the value of Sim_{Name} is irrelevant. In other words, if the concepts are in different spaces, one in 2D and the other in 3D, then the overall similarity should be zero regardless of values from other matchers.

- **Similarity overlap** Different individual matchers might use overlapping criteria to evaluate the individual similarities. Therefore, their results can have an overlap. Unless the overlap is clearly identified and adjusted, the overall similarity can implicitly give more weight to the overlapped criteria, and can reduce the importance of other criteria.

Weighted average, which is used in most matching problems, cannot address the challenges posed due to inherent nonlinearity as discussed above.

In [13], we proposed a matching method based on Support Vector Regression (SVR) to match assembly information from two different CAD applications, and has shown that the ability to handle nonlinearity actually improves the matching accuracy. However, it requires users to assign continuous value in the range of [0, 1] for the pairs in training data, instead of just 1 (match) and 0 (non-match), and even domain experts cannot objectively pin point the correct similarity value in the range.

In this research, we propose FEedback Matching Framework with Implicit Training (FEMFIT) to address the challenges.

4.5 Overview of FEMFIT

Figure 4.3 shows a schematic of FEMFIT. The input to the framework is source and target ontology, the ontology of sending and receiving system respectively. The framework works in *iteration* - in each iteration, a matching concept for one source concept is identified. Concept ordering is executed once before the iterations to decide the sequence of the source concepts to proceed.

In the iteration, the machine learning algorithm and the domain expert work in turn for each iteration to find the matching concept in the receiving system; when the learning algorithm suggests a ranked list of concepts for a given source concept based on the information accumulated so far, a domain expert chooses the “true” matching concept from the list. If the expert chooses a concept which is not in the first rank,

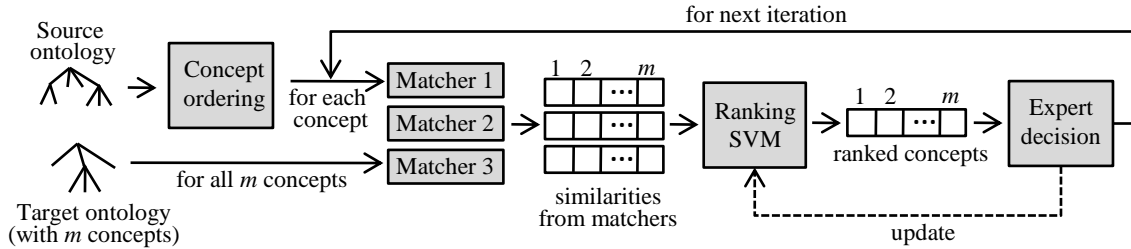


Figure 4.3: Overview of FEMIT. Solid lines indicate the sequence of execution with necessary information flow, and the dotted lines indicate the flow of the training data

the algorithm modifies itself in a way that it reflects the expert’s decision. In that way, the learning algorithm trains itself with the expert’s input over the iterations. As the iteration goes on, the algorithm is expected to make decisions as the expert does, and it is likely that the “true” matching concept is ranked high in the list. That way, the expert can focus on only small number of high-ranked concepts to make a decision without looking all the way down to the lower ranked concepts.

The overall framework comprises the following four components (shown as shaded boxes in Figure 4.3). This section presents a brief explanation of the function of each of those components.

- Concept ordering** The first step is concept ordering which is executed once on the source ontology. The input to this step is all the concepts from the source ontology, and the output is an ordered list of the concepts. The actual matching process which goes in iteration is affected by the sequence of the source concepts to iterate because of the *dependencies* among the concepts; some matching decision cannot be made without the result of other matchings. In this step, source concepts are ordered so that all the dependent matching

come after the matching result that they depend on. Detailed description will be given in the next section.

- **Matchers - Calculating individual similarities** Once the sequence of concepts are obtained, the process iterates for each source concept with each of all concepts from the target ontology. Multiple individual matchers are used to calculate the (individual) similarities between the source concept and the concepts from the target ontology. For k matchers, the output will be k arrays of size m ; i^{th} array's j^{th} element is a similarity value between the source concept and j^{th} concept in the target ontology using i^{th} matcher.
- **Ranking SVM - Combining multiple measure result** Ranking SVM, trained with the previous decisions from domain expert, combines individual matcher results (a set of arrays) into one array of overall similarity values; j^{th} element in the array means the overall similarity between the given source concept and j^{th} target concept. Detailed description will be given in the next section.
- **Expert decision - Finalizing the match** Finally, the domain expert finalizes the decision on the matching (or equivalently, corresponding) concept based on the similarities calculated by Ranking SVM. The concept the expert chooses, which may or may not be the concept of the first rank in the list, is confirmed as the *matching* concept for a given source concept, and the result is fed back to the learning algorithm so that it is trained further.

Next section further explains each component in detail.

4.6 Details of FEMFIT

In this section, the four components of FEMFIT are explained in further detail.

4.6.1 Concept ordering

Concept ordering is stated as:

Given: a set of (unordered) source concepts $C_{src} = \{c_1, c_2, \dots, c_n\}$,

Find: an ordered set $C'_{src} = \{c'_1, c'_2, \dots, c'_n\}$ which is a permutation of C_{src} ,

such that: for all i and j such that $i > j$, similarity calculation for c'_i is *independent* of that of c'_j

When we say similarity calculation for a source concept c_a is *independent* of that of c_b , we mean that we can calculate similarity between c_a and any of target concepts without knowing any similarity values between c_b and any of target concepts. For example, for a string matching algorithm on *name* of the concepts, a NX constraint concept *Angle* is independent of *Bond* because the string comparison between *Angle* and any other target concepts is not affected by the similarity of *Bond* with others.

Dependency is defined in the opposite way. For example, *Concentric* constraint in NX has *Circle* as its attribute. When we use a property matcher, which we further explain in the case study later, similarity calculation between *Concentric* with other target concepts is dependent on the similarity values between *Circle* and other target concepts.

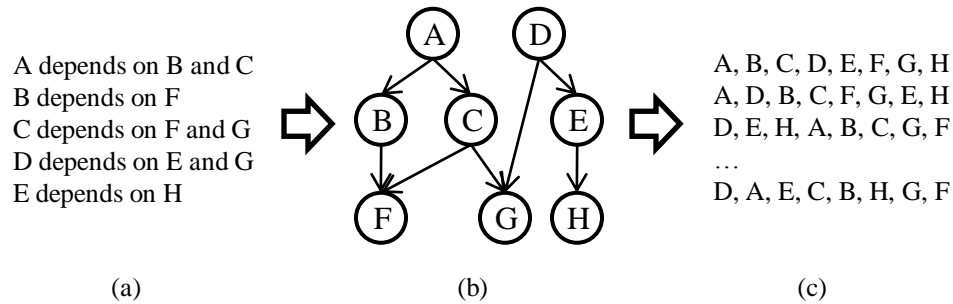


Figure 4.4: (a) an example of partially ordered set of dependency relation, (b) A directed acyclic graph (DAG) generated from the set, (c) possible linearizations of the DAG through topological sorting

With the dependency as partial order binary relation, a whole set of concepts in an ontology can be represented as a partially ordered set (or poset), and the poset can be converted into one or more directed acyclic graph (DAG). Figure 4.4 shows the example of creating the ordered set from an partially ordered set: (a) shows the partially ordered set with dependency as a binary relation, and (b) shows the DAG generated from the set.

From the DAG of concepts, a totally ordered set (or equivalently, linearly ordered set) is obtained by topological sorting [50]. Figure 4.4 (c) shows possible total orders (or linearizations) of the graph. As can be seen, in general, there are more than one linearizations, but any solution is fine; in implementing the procedure, we just stop after getting the first solution. When there are more than one DAG, each DAG produces a ordered set, and the sets can be concatenated in any order. Finally, one ordered set is obtained and in the set, an concept with a large index is always independent from the concept with a smaller index.

Note that there is a case where creating DAG is impossible. For example, Melnik et al. [51] modify similarities incrementally based on the similarity of each other, and

in this case, the dependency relation turns into a cyclic graph. However, it is one of small number of cases compared to the matching algorithms proposed so far, and is out of our research scope.

Once the source concepts are ordered, next step is to calculate individual similarities between each source concepts and all the target concepts. Next sections has the discussion.

4.6.2 Matchers - Calculating individual similarities

Individual matchers calculate the similarity between two concepts with respect to one specific aspect, e.g., name or hierarchical information. In this research, we do not propose specific individual matchers. Literature documents various matchers in product domain [14, 12, 33, 13] as well as in generic domain [11, 34, 35, 36].

The domain expert is to choose a set of individual matchers to use. However, the selection process itself requires an expertise in matching algorithms which the domain experts in product development systems may not have. With conventional techniques like linear combination, adding more matchers requires more work - to make sure that the matcher does not at least degrade the overall performance, and to decide the weight value for the matcher.

With FEMFIT, however, adding more matchers does not increase any effort toward the domain expert because it trains itself with the exactly same input. In addition, even the poor matchers do not degrade the overall performance as compared to other combination methods. The experiment and discussion will be given in the case study.

Next section discusses how individual measures are combined with Ranking SVM.

4.6.3 Ranking SVM - Combining multiple measure result

To combine the multiple measures from individual matchers, we use Ranking SVM. Given the multiple source and target concept pairs, the algorithm will output a ranked list with similarity values for all the pairs (note that for each iteration, which is for each source concept, we have the same number of pairs as the number of all the target concepts). In this section, we will first review the basic idea of Ranking SVM, and then explain how it is used to combine the individual matchers to determine maps across concepts in product ontologies.

Review: Ranking Support Vector Machine (Ranking SVM)

Ranking SVM was introduced by Joachims [2] to optimize search engines with clickthrough data. The idea is to attain the ranking of pages given a query based on the clickthrough data provided by users. With the ranking, a vector W is calculated such that the collected pages are ranked properly when projected on the vector. Then, the ranking of the new set of pages can be attained by projecting them onto the vector.

In more general sense, Ranking SVM is trained with a dataset, each data point has n independent variable and 1 dependent variable. Training is to find a vector in a transformed nonlinear space, onto which all the dependent variables of data points are correctly ordered.

Figure 4.5 shows the schematic to find a weight vector geometrically. In the figure, a data points are plotted on a plane. Suppose, according to the dependent variables

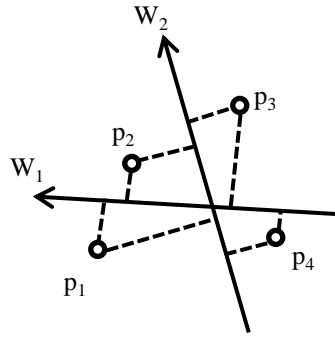


Figure 4.5: A ranked set of data points p_1 , p_2 , p_3 and p_4 are plotted on the plane. Different vectors can be set up but W_1 can order the points correctly when projected onto whereas W_2 generates a wrong rank. Ranking SVM seeks a weight vector like W_1 (Adapted from [2])

of dataset, the correct ordering is p_1 , p_2 , p_3 and p_4 . There can be a different vectors onto which points can be projected. In this example, W_1 can order them correctly, whereas W_2 results in a wrong order - p_3 , p_2 , p_1 and p_4 .

Note that there can be multiple vectors that can order them correctly. Ranking SVM chooses the one separating the points most when projected onto itself. Ranking SVM is equipped with the features of conventional SVM: the points can be transformed into a nonlinear feature space for better ordering, and still violating points can be accommodated with with some penalty over the maximizing the separation.

In addition, the algorithm works without the full ranking of all the points, in other words, with partial feedback [2]. Then, only available ranking information will be used as constraint equation in the overall optimization problem. Next section describes how it is applied to the concept matching problem.

Ranking SVM for product concept matching

In the proposed framework, the learning algorithm and a domain expert work in turn: when the learning algorithm generates a ranked list, the domain expert selects the matching concept in the list. The decision confirms a map for that specific pair, and also, the decision is used to further train the learning algorithm. The matching is processed sequentially according the result of the concept ordering.

Suppose, we are now dealing with 3rd source concept c'_{s_3} after the ordering. Then, we would already know the matching concepts for c'_{s_1} and c'_{s_2} , say c'_{t_1} and c'_{t_2} respectively, and this information can be directed into the algorithm as:

$$S(c'_{s_1}, c'_{t_1}) \geq S(c'_{s_1}, c_t), \quad \forall c_t \in O_B, \quad s.t. \quad c_t \neq c'_{t_1}$$

$$S(c'_{s_2}, c'_{t_2}) \geq S(c'_{s_2}, c_t), \quad \forall c_t \in O_B, \quad s.t. \quad c_t \neq c'_{t_2}$$

where S is a unknown function which returns the overall similarity given two concepts. Now, we define points p in the k -dimensional Euclidean space, k^{th} element being the result from k^{th} individual matcher, as:

$$p_{i^*} = (Sim_1(c'_{s_i}, c'_{t_i}), Sim_2(c'_{s_i}, c'_{t_i}), \dots, Sim_k(c'_{s_i}, c'_{t_i}))$$

$$p_{ij} = (Sim_1(c'_{s_i}, c_{t_j}), Sim_2(c'_{s_i}, c_{t_j}), \dots, Sim_k(c'_{s_i}, c_{t_j}))$$

Then, the the ranking problem for n th concept is given as:

$$\min_{\vec{w}, \vec{\xi}} \frac{1}{2} \vec{w} \cdot \vec{w} + C \sum \xi_{i,j}$$

$$\text{subject to } \vec{w} \cdot \Phi(p_{i*}) \geq \vec{w} \cdot \Phi(p_{ij}) + 1 - \xi_{i,j}$$

$$\xi_{i,j} \geq 0$$

$$\text{for } i = 1, 2, \dots, n - 1, \text{ and } j = 1, 2, \dots, m$$

where m is a number of concepts in O_B . Because a vector satisfying all the constraints may not exist, term $\xi_{i,j}$ is used for the the training error of the point from the pair of i^{th} source concept and j^{th} target concept, and C is a trade-off between training error and margin.

Figure 4.6 shows the matching procedure with Ranking SVM. First, a nonlinear transformation $\Phi(\cdot)$ is applied to all the points in Euclidean space from the history and plot them on a feature space. In the feature space, find a vector W projected on which the similarity ordering is maintained maximizing the minimum margin between ranked pairs. For a new data set, in this example, pairs with 3rd concept from the concept ordering, plot them in the feature space and check the projected values onto the vector W . These values are provided to the user, and the user finalizes the matching. Note that in the implementation of the algorithm, the nonlinear transformation is not actually applied and relatively simple calculation handles the nonlinearity, which is called kernel trick [52].

When the user finalizes the decision on the matching for this concept, this information is also stored as history data and used by the algorithm to calculate similarities for next concepts and the matching procedure iterates.

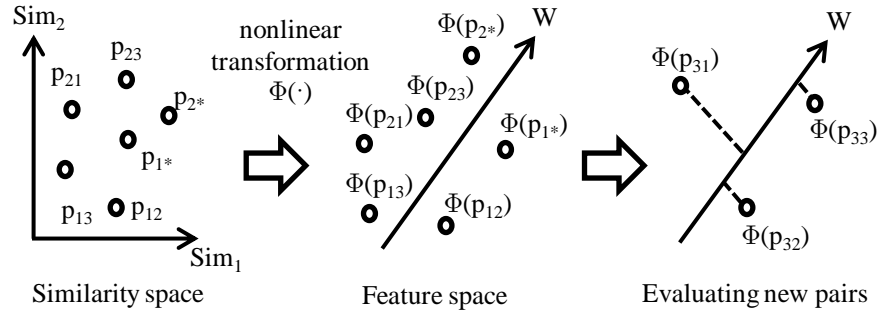


Figure 4.6: A ranking procedure with Ranking SVM. After the points from previous history are transformed into a feature space and a vector W is calculated projected on which the similarity ordering is maintained maximizing the minimum margin between ranked pairs. Once the vector is found, new data set is plotted on the space and is projected onto the vector W . Based on the value, a user finalizes the matching.

4.6.4 Expert decision - Finalizing the match

When the algorithm finishes the ranking, the domain expert, the user, receives the list of candidate concepts in the order of similarity, and among the list, the user is to choose the real correspondence for the given source concept. It is expected that the corresponding concepts should be on the high ranks, so that the user can finish the matching by assessing only small number of concepts with high rank.

Besides the rank itself, similarity value helps the decision. For example, if there is marginal similarity between rank 1, 2 and 3, but big difference in 3 and 4, it is likely that the corresponding concepts are among the concepts with rank 1, 2 or 3. Further truncating strategies can be found in [39]. In this research, however, we do not have such truncation. In product domain, we believe, it would not be very difficult to decide whether a concept is a correspondence or not without going further with lower rank pairs.

Once the user finishes the selection, the partial rank information is fed into the

ranking algorithm.

Next section shows case study where the framework and its result is presented with specific matching problem with assembly constraints.

4.7 Case study

We are not aware of other research/literature in this area that could be used to evaluate our approach objectively. Therefore, for this chapter, we develop a case study, and evaluate the resulting semantic maps manually, but also in comparison with other methods/measures.

A better management of assembly information can reduce the cost of manufacturing drastically - Lohse et al. [53] state that assemblies account for 80 percent of the cost of manufacturing a product. Yet, there have been few research on automating the matching of assembly level information.

Figure 4.7 shows an example of translating cell phone assembly between two CAD software tools - NX and CATIA. For the translation, assembly concepts from both software are matched to the standard representation ISO 10303-108. In this case study, we will demonstrate the procedure to match the assembly concepts from NX to ISO 10303-108 (part enclosed in a dotted line). Specifically, 9 assembly constraints in NX will be matched to the corresponding concepts in ISO 10303-108. In the matching, only 20 candidates are considered for brevity.

4.7.1 Ontological formulation

Ontology has been used to represent various types of product information. NSF's Cyberinfrastructure report on Engineering Design specifies the need to develop com-

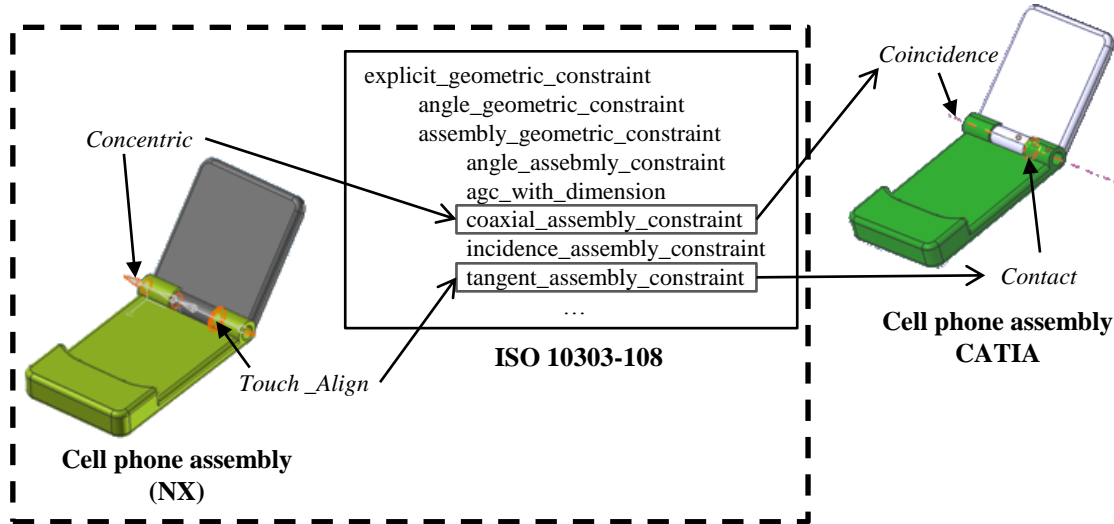


Figure 4.7: Example of a cell phone assembly model with matching concepts to capture constraints in the software NX, the former Unigraphics, and CATIA. The translation is through the standard representation ISO 10303-108. The case study considers only the part from NX to ISO 10303-108 for brevity (enclosed with dotted line).

mon languages and ontologies to capture technical domain such that they leverage and extend web services and semantic web standards [54]. The Process Specification Language (PSL), proposed by NIST, provides a shared ontology to enable information of different manufacturing processes to be built upon and exchanged [53, 55].

Currently, to the best of our knowledge, there is no major CAD systems use ontology for their representation. To demonstrate the matching between two different assembly representations, we construct ontologies based on the existing assembly representations of NX and ISO 10303-108, and the matching will be on those ontologies. Specifically, we use Web Ontology Language (OWL) [56] as a specific language, which is a standard ontology language endorsed by World Wide Web Consortium (W3C). The part of hierarchical structures of the concepts are shown in Figure 4.8.

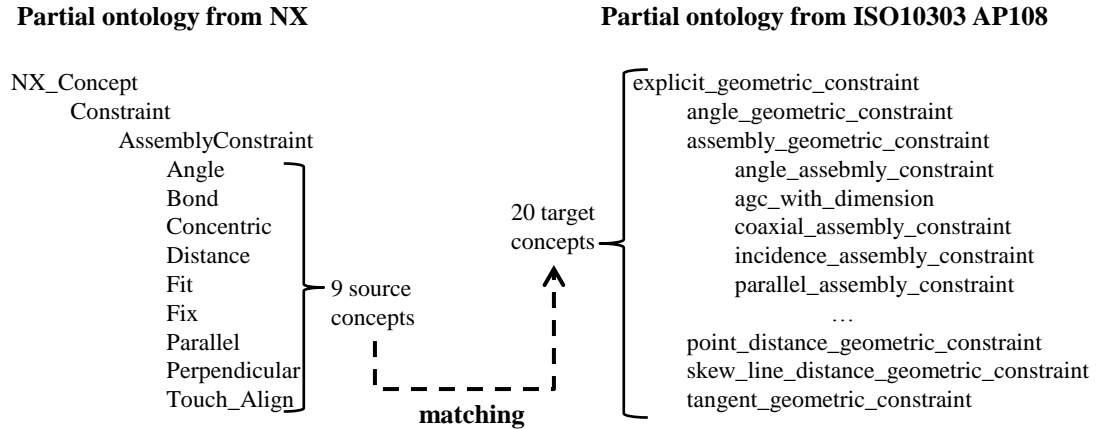


Figure 4.8: Partial concept hierarchies of the ontologies modeling assembly constraints in (a) NX and (b) ISO 10303-108 assembly constraints

Ontologies are created by analyzing both representation system: [57] has been referenced to build ISO 10303-108 assembly constraint ontology and NX user manual has been referenced for NX ontology.

4.7.2 Concept ordering

Now, we order the source concepts through concept ordering process so that the matching does not require any information yet to be identified. The source ontologies are converted into Directed Acyclic Graph (DAG) with the dependency relations. Partial graph including *Distance* and *Concentric* concepts are shown in Figure 4.9 (a). Then, a linearization can be found through topological sorting. The final ordering is shown in Figure 4.9 (b).

Note that the data types, i.e., *float* and *int*, in this case, are common to any (OWL) ontology, so its corresponding concepts are always themselves. In this case study, we assume that we already have a mapping between properties and concepts

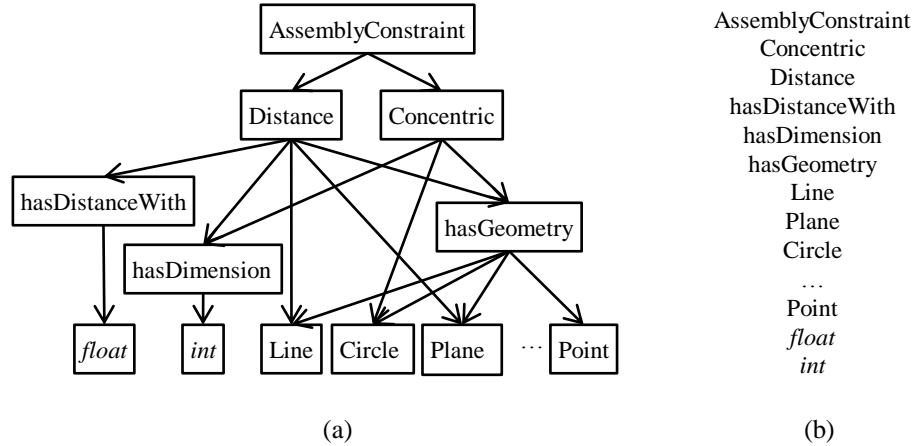


Figure 4.9: Concept ordering example: (a) dependency network for NX assembly constraints is shown in DAG, (b) a linearization is generated through topological sorting. In the order set, concepts with larger indices are independent from the concepts with smaller indices.

at the axiom level such as *Line* or *Circle*. It can also be incorporated into the same matching process, but we only focus on the main assembly constraint concepts for brevity.

4.7.3 Individual measures

In this case study, we use 3 individual matchers: Name, PathName and Property matchers. Following sections explain the matchers.

Name matcher

Concept name is determined in a way that it shows the meaning of the concept in very concise way. Therefore, name is simple yet important information to compare the concepts, and most of the matching applications includes name matcher. The matcher uses different types of string matching algorithms depending on the way the names are represented.

In this case study, we use 20 string matching algorithms from *SimMetrics* [58]

which includes Levenshtein distance, cosine similarity, Jaccard similarity. Different string match algorithms have different strategy. But we use all the available algorithms, and the proposed method using Ranking SVM is supposed to automatically decide which algorithms are actually helpful in the matching.

PathName matcher

PathName matcher considers the hierarchical information of the ontology. In this case study, we concatenate the name of the concept and its parent concept into one string, and compare the concatenated strings. Although there are many different hierarchical matchers [11], this gives enough hierarchical information for this case study because the case does not have complicated hierarchical structure in the ontologies.

As string matching algorithms, we used the same set of string matchers used for the name matchers.

Property matcher

Property matcher is a simplified version of the concept description matcher proposed by Patil [14]. The idea is to calculate the ratio between the common description fields and total number of description fields between concepts. In this simplified version, we only compare based on the property itself, not considering the actual value the property takes.

Note that the matching is on the “flattened” concepts. In [14], concepts are mostly in flat structure and the descriptions are explicitly defined for each concepts. In this case, we have some hierarchical structure and some logical descriptions are contained

in parent concepts. Therefore, we “flatten” the concepts before the comparison.

Figure 4.10 shows an example for *Angle* concept in NX ontology. In the ontology, the concept has three ancestor concepts: *AssemblyConstraint*, *Constraint*, and *NX_Concepts*. Each of them has its own logical expression except for *NX_Concepts* because it is the most fundamental concept in this ontology and it does not have specific logical expression.

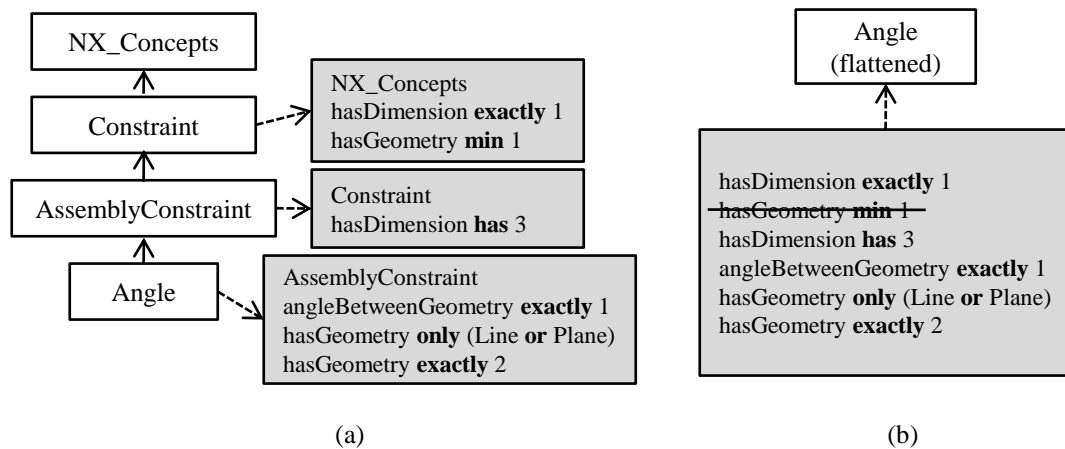


Figure 4.10: Concept flattening example: (a) *Angle* constraint has three ancestors and they have their own logical descriptions, (b) flattened *Angle* inherit the logical descriptions from its ancestor concepts. Note that the inherited description can be overridden by its own description. In the example, *hasGeometry min 1* is overridden by *hasGeometry exactly 2*.

4.7.4 Result

Figure 4.11 shows the process of identifying matching concepts with FEMFIT: there are all 9 source concepts to find correspondences from 20 target concepts. Each user selection is followed by the ranking algorithm, and the algorithm ranks the target concepts according to similarity values. The number below each target concept is a value used to rank the concepts; only the relative values are considered

here.

In the figure, only the result of three iterations are shown for brevity. The correspondences for the concepts shown in the example of translating cell phone assembly (*Concentric* and *Touch_Align*) are ranked 1 and 2 in the recommendation list. Detailed ranking results and comparison with other algorithms are given in the following sections. Throughout the result description, we will use *Rank* to measure the matching accuracy. In ideal case, the corresponding concepts are desired to be ranked 1 all the time; in that case, there is no further effort but choosing just first target concept in the list. In practice, *better* combination method is more likely to have higher *rank* throughout the iterations so that the user can find the corresponding concept without checking all the concepts down the ordered list.

Iteration	1	...	3	...	9
Source concept	Angle	...	Concentric	...	Touch_Align
Recommendation	(None)	...	coaxial_ac (0.09) parallel_ac_wd (0.08) incidence_ac (0.08) surface_distance_ac_wd (0.07)	coaxial_ac (0.06) tangent_ac (-0.51) incidence_gc (-0.66) agc_wd (-1.24) ...
User selection	angle_ac_wd	...	coaxial_ac	...	tangent_ac

ac:assembly_constraint, *gc*:geometric_constraint, *wd*:with_dimension

Figure 4.11: Table shows how FEMFIT confirms matching over iteration: for each iteration, that is, for each source concept, the recommendation is given as ranked list. When the user select the correct match, it trains them further and gives the improved recommendation for the ensuing iterations. Note that there is no recommendation in the first iteration; learning starts from the input of the user after the end of first iteration.

Comparison with conventional combination methods

We compare the result of FEMFIT with conventional combination methods, i.e., Max/Average combination and weighted average with linear regression. Figure 4.12 shows the rank of the corresponding concept in each iteration for different combination methods. Note that all 3 types of individual matchers, which are 41 in total, are used for combination.

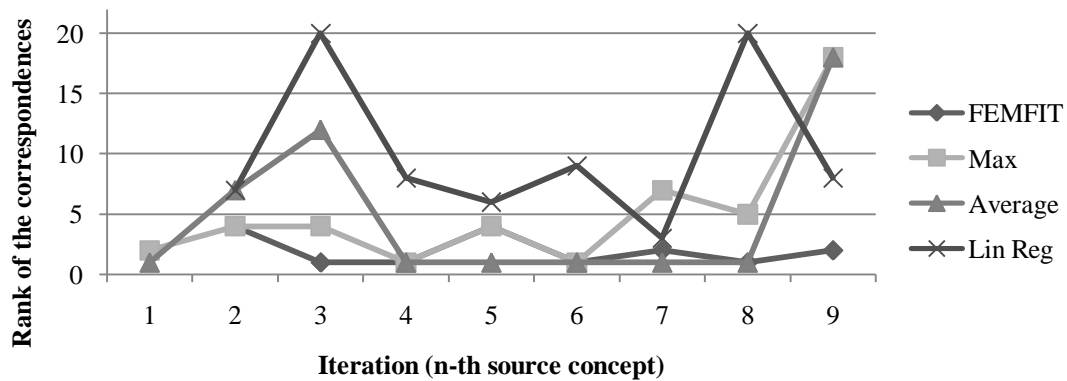


Figure 4.12: Ranks of the real corresponding concept in the ordered set by each combining method for 9 matching processes.

As shown in the figure, the corresponding concepts are ranked very high (4^{th} at worst) for the set generated with FEMFIT. This means, with this method, users will find all the corresponding concepts by checking only a few high rank concepts in the ordered set.

The Max and Average combination show fair performance, but not as good as FEMFIT. While FEMFIT considers the relative weighting among the individual matchers implicitly, Average combination uses the same weight for all. Therefore, when there are some poor measures included, those measures can deteriorate the overall performance. In addition, any highly important measures are treated

equal with all other mediocre or poor measures, and its information does not affect the overall result as much as it should.

Max combination blindly chooses the highest similarity value. The requirement for this algorithm to work well is that all the individual measures has similar range and distribution; that is, when individual measure A and B says 0.8, the similarity should really be similar. Although setting to the same range is simple, ensuring the linearity between all the individual measures and the overall measures are not always possible [13].

The weight values for weighted average combination is statistically found through linear regression. While previous approaches like Doan et al. [19] create a training data set to first find the regression coefficients, we do the regression at the end of every expert decision assigning training data of 1 (match) and 0 (non-match). In other words, in the same feedback framework, we compared Ranking SVM and linear regression. As shown in the figure, the performance is very poor in this experiment. Note that the Average combination is a special case of weighted average, but the result of weighted average with linear regression is much worse. This means, although inability to handle nonlinearity would be a partial cause for poor performance compared to FEMFIT, major reason for the failure is that linear regression with relatively small data but with large dimension (in this case, 41 matchers) cannot find a stable set of weights or regression coefficients.

Figure 4.13 shows the averages and standard deviations of the ranks of 9 matching iterations for different combination methods with different combinations of individual

measures. Note that the values are equivalent when only property matcher is used. Because there is only one matcher in that type, there is no combination to apply. We show this specific result as a comparison with other combinations.

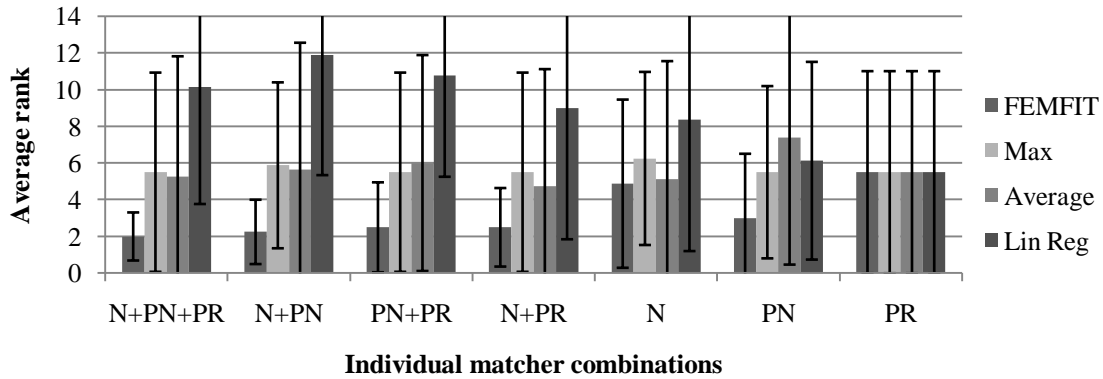


Figure 4.13: Averages and standard deviations of the ranks for different combination algorithms with different combinations of individual measures: Name (N), PathName (PN), and Property (PR) matchers

From the result shown in the figure, we have some major observations:

- Ranking SVM shows better performance - higher rank with less standard deviation - in most cases. We can conclude that the algorithm successfully combines different individual measures - giving proper weighting to individual measures, although not explicit. The only exception is the case with name matchers only. The matcher is not as good as others, but it does help the overall performance when combined with others.
- Considering the total number of candidate, which is 20, Max and Average combination also show fair average rank values. However, standard deviation is large - the performance is not stable over the different matching pairs.
- Combination not necessarily improves the result for Max and Average com-

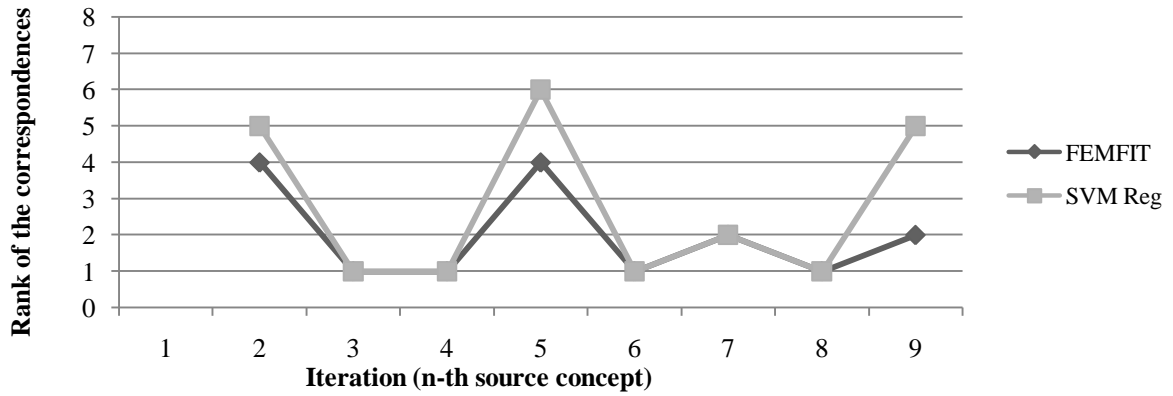
ination. Because we have many different string match algorithms, some of them are not very accurate in similarity estimation, and those will deteriorate the overall performance while Ranking SVM is capable of weeding out poor measures.

- The result for linear regression shows an interesting point: there is a tendency that linear regression works better with less number of matchers combined. We think the major reason for such trend is the number of training data. In this experiment, the number of independent variables, which is number of matching algorithms, is up to 41 when all 3 types of matchers are combined. To correctly decide the intercepts for all those 41 variables, it will require much larger data set, while each human decision only adds 20 data points.

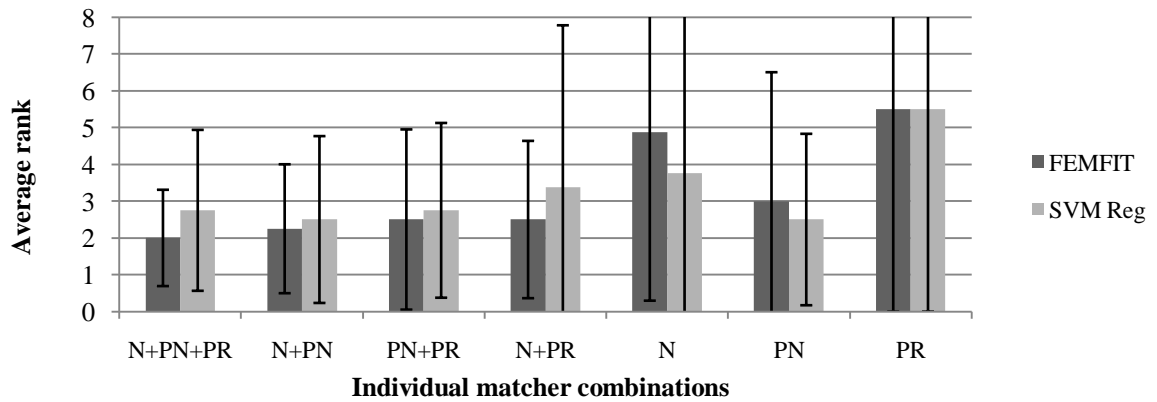
Comparison with SVM regression

We also compare the result of FEMFIT with the combination using SVM regression, which is similar to our previous work [13], but not exactly the same. In this experiment, we feed the same input for SVM regression as we provide for FEMFIT. That is, there is no pre-training before the matching. Also, the in-between training only provides match(1) or non-match(0), rather than the similarity value in continuous scale of $[0, 1]$. Figure 4.14 shows the result.

The figure shows that the result of FEMFIT is better than SVM regression overall. The superiority of FEMFIT is as expected: while FEMFIT uses the expert input as is, i.e., chosen pair has larger similarity than the other pairs, SVM regression set 1 for the chosen pair and 0 for the other, which is a *bias* to some extent.



(a)



(b)

Figure 4.14: Comparison between the result of FEMFIT and SVM regression: (a) Rank of the corresponding concepts for each iteration. Note that there is no result for the first iteration because there is no input before the first expert decision, (b) Averages and standard deviations of the ranks for different combinations of individual measures: Name (N), PathName (PN), and Property (PR) matchers

The performance difference is not significant, however. Also, in some cases, SVM regression shows (slightly) better result. We think there are two reasons for this. First, we just used same default parameter for both FEMFIT and SVM regression regardless of different combinations of matchers. When both algorithms are fine tuned separately for individual cases, FEMFIT is improved further than SVM regression. But we think the above figure is enough to show the overall trend in comparison. Second, the case is relatively small in dataset compared to the real product concept matching problem. We think as the problem becomes more realistic, i.e., larger, factors from noise will fade out and the superiority of FEMFIT would become more evident.

Other nonlinear machine learning technique, Neural Network, can be used in similar fashion as Sheikholeslami et al. [59] used it to combine the results for the heterogeneous data to cluster visual data. But we do not include Neural Network in this experiment; there are some known advantages of SVM over Neural Network: SVM, as compared to Neural Network, require less manual parameter tuning [60] and are less likely to converge to a local solution. In addition, it minimizes structural risk rather than empirical risk, and that keeps it from overfitting of the observed data.

4.8 Summary

Matching concepts, i.e., determining semantic maps across heterogeneous resources, relies on similarities with multiple views on product data; better approximation of the similarity between concepts requires a combination of different measures/methods. Typically, a domain expert chooses or develops a set of similarity

measures, and combines them to capture the similarity between the product concepts.

Finding a correct combination is, however, not a trivial task. First of all, there is no formal ground to decide relative importance of the individual similarities. Furthermore, accurate approximation typically requires nonlinear combination of the similarities which makes the identification extremely complicated.

In this research, we have proposed FEMFIT, a feedback matching framework; it finds the combination relation automatically from the expert decision without any additional cost. The experiment has shown very accurate results in matching heterogeneous assembly data resources. In addition, the performance of the framework is not degraded by poor measures; it will save the overall implementation time because the user does not need to be concerned much to select only the most appropriate individual matchers.

We expect that the proposed method will combine multiple measures to find a correct semantic map between the concepts. Once the concept map is identified, the translation requires a syntax translation; the data in one system should actually transform to the syntax in the receiving system. Next chapter presents our research toward the translation.

CHAPTER V

Translating the concepts

As discussed in chapter I, beyond the concept matching, there is a need for a method that actually translates the data from the sending system to the receiving system. This chapter explains the approach developed in this research to enable the same.

5.1 Motivation

So far, the standard approach toward translation has been through ISO 10303, informally known as STandard for the Exchange of Product model data (STEP); most CAD/CAM systems supports translation from/into ISO 10303. In spite of wide acceptance and success, its coverage is rather limited and important data semantics is lost during translation [14, 30]; there have been effort toward new standards, such as CPM [31, 32, 6] and PSRL [7].

As the new, semantically expressive, representations emerges, efforts toward automated matching has increased, and a similar effort toward automated translation would be a natural consequence.

So far, however, there has been little effort toward automated product data translation, and this research aims to present an initial approach toward the translation.

Next section presents the objective of this research.

5.2 Objective

The objective of this research phase is to develop a method that translates the data from the sending system to the receiving system. The translator is an aggregation of the translation rules for every matching concepts and attributes which transforms one syntax into another while preserving the same semantics.

The challenge is that every pair requires specific translation rule and finding the rule not only requires domain knowledges but also is very labor-intensive process. Therefore, the objective is, specifically, to develop a method to automatically find the translation rules for any given pair of concepts/attributes.

Next section defines the specific research problem.

5.3 Product concept translation and related work

In modern product development, collaboration of functionally and geographically distributed teams is an essential aspect to deliver the products in shorter time with less cost. However, different teams have different roles, and they typically use different software. For example, Figure 1.1 in page 3 shows the list of software involved in automotive industry, and a product-centric enterprise requires data translation between the software they use.

Currently, there are translators between the product development software. ISO

10303 is an industry standard as a product representation, and most CAD software support import/export of their native data format from/into the standard. However, translation through ISO 10303 does not preserve the semantics of the data [14, 30], and there have been efforts to a better solution.

There are commercial translation modules between some major CAD software, for example, translators between Siemens NX6 and CATIA V4/V5. However, it takes long manual process to build such translators and the coverage is very limited considering the number of CAD systems. Furthermore, such development process has not been published and no formal procedure behind the program has been reported.

Academia also has some approaches toward translation. In CAD domain, a promising approach is history-based translation [61, 62, 63] where a command history to build a geometric model in one system is translated into a set of command history, or macro, of another system through an interlingua. Advantages include that it can directly capture the user intent which has rich semantics, and that the neutral files can be maintained as relatively small-sized document files, i.e., eXtensible Markup Language (XML).

Such approach, however, cannot be applied when it comes to a broad spectrum of product information such as PDM or PLM data where command history is not typically available. To address the issue, ontology-based approaches have been proposed [64, 7, 65]. Formalizing product information with ontology provides a foundation for automated reasoning, e.g., consistency checking and matching across heterogeneous resources, and the ontology concept matching has been an active area of

research [14, 12, 33, 13].

However, finding concept map is not enough for the data exchange, and it requires the discovery of actual translation rules between identified matching concepts. Figure 5.1 shows concept matching and translation between two instances in SolidWorks and NX, CAD software applications. For the data exchange, first the concept map should be identified, e.g., *Date created* is mapped to *Created date*, not *Modified date*, and once the map is found, actual data should be transformed, e.g., “6/1/2009 14:20:28 PM” into “01 Jun 2009 14:20”. To the best of our knowledge, there has been no research toward formalizing such process in product domain, and this research will deal with this translation issue.

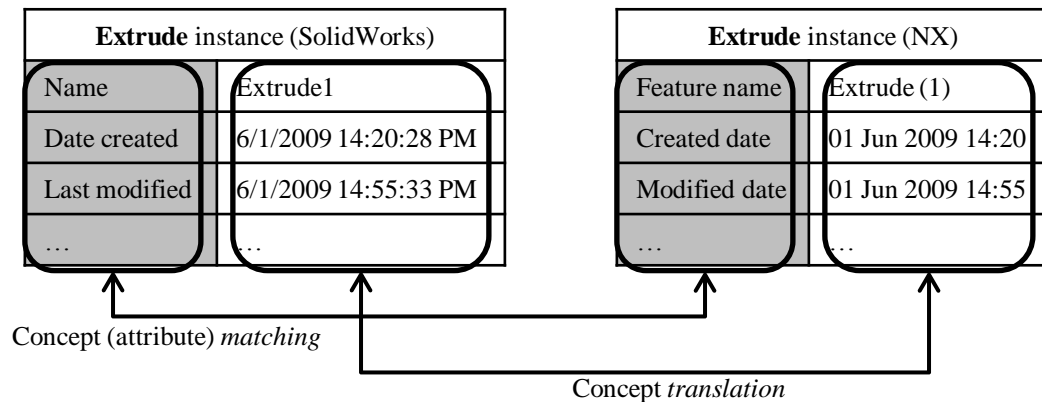


Figure 5.1: Extrude instance representations in SolidWorks and NX, CAD software applications. Concept (attribute) matching is to find correspondence between the concept/attribute names, and the translation is to actually transform the instance data for each attribute.

Next section presents a problem formulation.

5.4 Problem formulation

In the previous section, we explained product data translation problem in general.

This section, we define the problem as:

Given: A source concept A and its attributes a_1, a_2, \dots, a_n

A coSurreponding (target) concept B and its attributes b_1, b_2, \dots, b_m

A set of *aligned* instance $(i_{a1}, i_{b1}), (i_{a2}, i_{b2}), \dots, (i_{ak}, i_{bk})$

A set of *basis functions* $F = \{f_1, f_2, \dots, f_p\}$

For: Each target attribute b_i

Find: A translation rule R_i such that $b_i = R_i(a_1, a_2, \dots, a_n)$

where R_i can be represented with a combination of basis functions F

Note that this problem formulation deals with the translation between two *predetermined* source and its corresponding target concept. Finding a map between concepts from sending and receiving systems is an active area of research, and several solution approaches have been presented [14, 12, 33, 13].

In addition to the fundamental inputs for the translation problem, i.e., source and target attributes, this problem definition requires two additional resources: 1) a set of *aligned* instance, and 2) a set of *basis functions*.

By *aligned* instances we mean a pair of corresponding, that is, of same semantics, instances in the sending and receiving systems. For example, when an instance i_a in a sending system is successfully translated into i_b in the receiving system, they should have the same meaning, and i_a and i_b are said to be aligned. We believe that

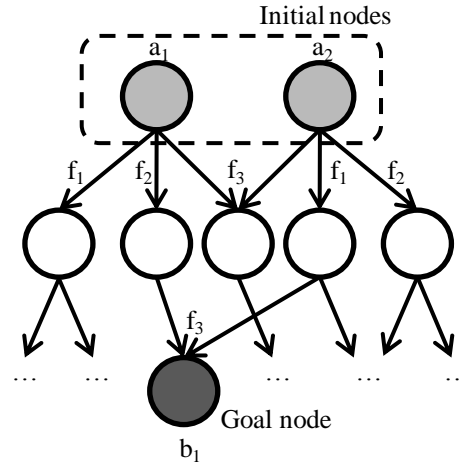
any organizations interested in automating the translation process would have used manual translation so far, and have collected a set of aligned instances needed for this problem formulation.

A *basis function*, in its original meaning, is an element of the basis for a function space [66]. That is, 1) any function in the space can be represented as a linear combination of basis functions, and 2) the basis functions are linearly independent with each other. In our problem formulation, we borrow the term, but loosely - a translation rule R , which is also a function, will be represented with a combination, not necessarily linear, of the *basis* functions F , and the functions are not necessarily linearly independent with each other. From this point on, we use the term basis function in this definition. Further explanation regarding the basis functions will be presented in the next section with the proposed approach.

5.5 Modeling concept translation with a directed graph

We model the concept translation problem with a directed graph. Figure 5.2 shows a schematic of the graph with a set of source attributes, a_1 and a_2 , and a target attribute b_1 . The basic idea is that the source attributes become initial nodes in the graph, and the graph is expanded as the basis functions are applied to the set of the existing nodes. When any created node is determined to be equivalent to the target attribute, it becomes a goal node, and the translation rule is returned with the functions in the path to the goal node.

A graph is an ordered pair $\langle N, E \rangle$, where N is a set of nodes (or equivalently, vertices), and E is a set of edges, or arcs when the graph is directed. Next sections



Translation rule R :

$$b_1 = R(a_1, a_2) = f_3(f_2(a_1), f_1(a_2))$$

Figure 5.2: Translation problems are modeled as a graph search. Functions in the path from the initial nodes to the goal node are combined to construct the translation rule for the given target attribute (b_1 in this case).

further explain how nodes and arcs are defined in our model.

5.5.1 Nodes

We define a **node** as a k -tuple, (v_1, v_2, \dots, v_k) , where k is a number of aligned instances and elements being any data types, e.g., numeric, string, date. In this problem, we have multiple **initial nodes** - the number is equal to the number of attributes in the source concept, and they are defined in following manner: when there is k instances for a specific source concept and the concept has n attribute, each instance is n -tuple, which can be represented as a column vector of n -dimension. Now, a data value matrix V can be created by concatenating those k vectors - from

the number of instances. Then, the matrix V is given as:

$$V = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1k} \\ v_{21} & v_{22} & \cdots & v_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ v_{n1} & v_{n2} & \cdots & v_{nk} \end{bmatrix},$$

where v_{ij} is data value for i^{th} attribute of j^{th} instance. Now, we have n initial nodes, and i^{th} initial node is characterized by i^{th} row vector in the matrix V .

There is one **goal node** - the problem is posed for each target attribute in the target concept as described in the previous section. We have the same number of instances as for the source concept because they are *aligned*, and thus, the goal node can be represented as a single k -tuple like all the initial nodes.

Intermediate nodes, shown as circles without any filling in Figure 5.2, are nodes that are created from the existing nodes with arcs. The graph initially starts with only initial nodes, and expands with intermediate nodes and arcs. The arcs and the expansion process is explained in the next section.

5.5.2 Arcs: basis functions

Each arc in a directed graph is an ordered pair of nodes, say (N_{out}, N_{in}) . In our approach, an arc is created when a node N_{out} is *expanded* which also creates a new node N_{in} . For each arc, one basis function is associated, and the value for the resulting node N_{in} is the function of N_{out} .

Basis functions can be divided into two types depending on the number of input - single input or multiple input. For example, $f_{SI} = ToUpper(str)$ is a function that returns the same string as the single input string, but all characters with uppercases.

When the function is applied to an existing node, say N_{out} , a new (intermediate) node N_{in} will be created, and the node has only 1 incoming arc.

On the other hand, there are functions that take more than one input. For example, $f_{MI} = concatenate(str1, str2)$ is a function that returns the concatenation of two input strings. When the functions are applied, two arcs are created - (N_{out1}, N_{in}) and (N_{out2}, N_{in}) with one intermediate node N_{in} . The node has two incoming arcs.

Note that the nodes are tuples, and the function is applied to each of the element in the tuple. For example, when a node N_{out} is expanded with a single input basis function f_{SI} , a new arc is defined as (N_{out}, N_{in}) such that $N_{in} = (f_{SI}(v_1), f_{SI}(v_2), \dots, f_{SI}(v_k))$ where $N_{out} = (v_1, v_2, \dots, v_k)$. We will also represent this as $N_{out} = f_{SI}(N_{in})$ for simplicity.

When a node N is expanded, typically multiple arcs and nodes are created because there are multiple basis functions. For example, when there are r single input basis functions, there will be up to r nodes and arcs are created - $f_1(N), f_2(N), \dots, f_p(N)$ and $(N, f_1(N)), (N, f_2(N)), \dots, (N, f_p(N))$ respectively. Note that not all basis functions can be applied to all the nodes, and certain functions are *not* applicable to the tuples of specific data types. For example, $f = ToUpper(str)$ is a function defined on strings, and cannot be applied to the tuples of numeric values.

The number of created nodes is different for the multiple input basis functions. When there are k nodes and r multiple input basis functions with multiplicity l , then, assuming all functions are applicable, there will be $r \times \mathbf{P}(k, l) = r \times \frac{k!}{(k-l)!}$ possible new nodes at the second *depth* in the graph (the term *depth* is typically

applicable only to tree structure which is a special type of graph, but the graph we use has initial nodes, and we can count the *depth* by using those initial nodes as root nodes). For example, when there are 4 two-input basis functions and 10 initial nodes, assuming all the functions are applicable, there will be $4 \times 10 \times 9 = 360$ new nodes at the depth of two, of which manual evaluation will be a time-consuming task.

Figure 5.3 shows an example of node expansion process - two single input basis functions are applied to an existing node creating two intermediate nodes. We do not show any multi-input cases here, but it will be discussed in later section for case study.

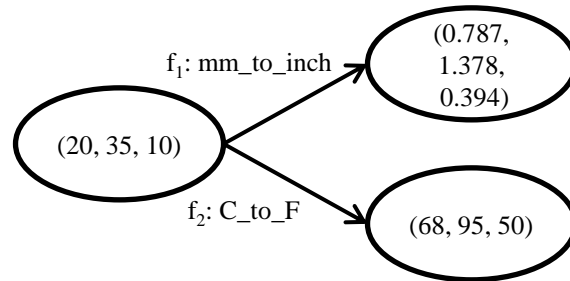


Figure 5.3: The figure shows a process to expand a node to create new nodes. In this example, there is two single input basis functions - function that converts *mm* into *inch* and the one that converts *Celsius* into *Fahrenheit*, and each produces a new node. The created nodes has a tuple of the same length with the original node - in this case, it is 3-tuple.

Next section presents how the solution is obtained using a graph search algorithm.

5.6 Graph search to find the translation rule

Previous section presented a graph model for the concept translation. In this section, we present an algorithm based on best-first search to traverse through the graph to find the translation rule.

Best-first search is a graph search algorithm where a node n with the lowest evaluation function $f_{eval}(n)$ is selected for expansion [67]. Figure 5.4 shows the overall procedure of the proposed method.

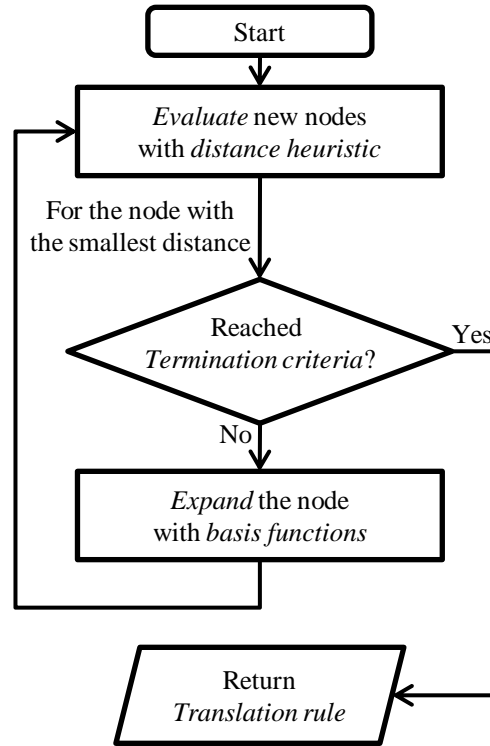


Figure 5.4: The flowchart shows a procedure for the proposed method to find a translation rule with an approach based on graph search algorithm. Specifically, best-first search is used - after evaluating the evaluation function $f_{eval}(n)$ for each node n , expand the node with the smallest value, i.e., approximately the closest to the goal node. The iteration goes on until it reaches termination criteria - either successful finding of the goal node or failure. Note that the figure shows only success cases for brevity.

Now, we explain the 3 components in the procedure: 1) node evaluation, 2) termination criteria, and 3) generating translation rule after the search.

5.6.1 Node evaluation

An evaluation function $f_{eval}(n)$ in graph search is a measure that estimates the *goodness* of the current node as a solution. A good selection of the function can

reduce the number of nodes to evaluate before finding the goal node, thus making the search faster.

According to the types of the evaluation, best-search is further classified: for example, greedy-best first search uses only heuristic function $h(n)$ which is an estimation of the cheapest path from the current node n to the goal node, and A^* search also considers the cost $g(n)$ to reach the node n from the start node, while breadth-first search uses only $g(n)$.

However, in the proposed graph modeling, there is no notion of cost derived from the path, neither to start (initial) node nor to goal node. In the modeling, each node is represented as a tuple, and the goodness or distance is only determined by the tuple itself compared to the tuple of the goal node without any path information. In fact, there is no path cost at all; no weight values are assigned to the edges/arcs. Note that we will still use the term *distance heuristic* for the evaluation function, which is typically used in many graph search problems.

The absence of weights for arcs implies a difference of this problem to other shortest-path graph search problem; as we explained in the previous section, the basis functions are not linearly independent in our problem, and there may exist two different paths to reach the same node. Because there is no weight associated with arcs, the search does not prefer the rules with 2 basis functions to the ones with 3 or more functions.

Specific distance function can be determined according to the type of goal node. For example, if the goal node is a tuple of string values, some string similarity

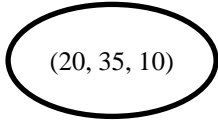
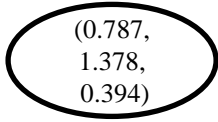
Node N to evaluate	Goal node
	
$D_{r_1}(20, .787) = \frac{ 20 - .787 }{\max(20 , .787)} = 0.96$	
$D_{r_2}(35, 1.378) = \frac{ 35 - 1.378 }{\max(35 , 1.378)} = 0.96$	
$D_{r_3}(10, .394) = \frac{ 10 - .394 }{\max(10 , .394)} = 0.96$	
$Dist(N) = (D_{r_1} + D_{r_2} + D_{r_3}) / 3 = 0.96$	

Figure 5.5: The figure shows a process to calculate the distance heuristic for an example node given a goal node. In this example, relative distance measure is used as distance heuristic. Relative distance is obtained for each pair of values, and averaged into one distance value.

measures, such as edit distance or Hamming distance [68] can be used. When the values are numeric, some measures like relative difference, Euclidean distance or its variations, such as standardized Euclidean distance (SED) [37], can be used to measure the distance. Figure 5.5 shows an example for the node evaluation with relative difference as a distance heuristic.

5.6.2 Termination criteria

There are two termination criteria - successful and unsuccessful ones. When any node is evaluated to have 0 distance to the goal node, then the search completes successfully.

It is not guaranteed, however, that the solution is found at all time. Not all product development systems have exactly the same features, and their concepts do

not match exactly. Therefore, it is possible that a specific information in the target concept does not exist at all in the source concept in any form, and there is no translation rule at all to bridge them. In such a case, the search will never finish by above mentioned criteria, and a termination criteria should be defined for this case which will complete the search unsuccessfully. The specific criteria should be defined depending on specific problems, and examples include a method limiting a number of iterations or processing time.

Sometimes, although the translation rule actually exists, but cannot be identified because of the lack of necessary basis functions, and it will terminate the search unsuccessfully. The proposed method does not provide any method to distinguish this case from the above case, and we believe some expert decision will be required for further clarification here.

When the search is unsuccessful, for either reasons, it should return the closest node(s), i.e., the node(s) with the smallest heuristic, during the search process so that the further manual process can follow accordingly.

5.6.3 Translation rule

When the search terminates successfully, the translation rule R for a given target attribute can be obtained recursively backtracking from the goal node to the initial node(s). The recursive procedure to find a translation rule R_{cur} for the current node N_{cur} can be defined as:

$$R_{cur} = \begin{cases} f_{inc}(R_{pred1}, R_{pred2}, \dots) & \text{if at least one predecessor and incoming arc} \\ a_i & \text{if no predecessor, and the node is } i^{th} \text{ initial node } a_i \end{cases}$$

When the search is unsuccessful, the user can start from the closest available node and its translation rule, and further basis functions can be added accordingly to complete the rule.

In the next section, we demonstrate how the method applies to some case examples.

5.7 Case study

In this section, we demonstrate case examples where the proposed method is used to find a translation rule for a given target attribute. We are not aware of other research/literature in this area that could be used to evaluate our approach objectively. Therefore, we develop a case study, and evaluate the resulting translation rules manually.

We first show how it applies in *element-level* translation, and discuss how *structure-level* translation is handled. By *element-level*, we mean that the translation is 1-to-1; to find the value of a specific target attribute, we will need only one source attribute to fill the target attribute although we have to find which one is needed. On the other hand, *structure-level* translation occurs when there is a structural difference between the concepts and one target attribute requires more than one attribute in the sending system to obtain its matching value.

There are other classifications for the types of translation, i.e., syntactic, semantic and pragmatic translations, as we discussed in Section 5.3. In our problem modeling, however, we do not care the specific content or operation inside basis functions; rather, the interface of the functions is concerned. That is, we borrow a set of basis

functions, and the functions are handled based on the number of inputs - either 1 (element-level) or more than 1 (structure-level).

5.7.1 Element-level translation

CAD is an important part of product development, and the translation between different CAD software is critical for modern product development. Figure 5.6 shows a schematic of translating a model information in *NX* (former Unigraphics) into *SolidWorks*. In this section, we will focus on demonstrating how the translation rule can be constructed for one target attribute (*Name* attribute in the figure).

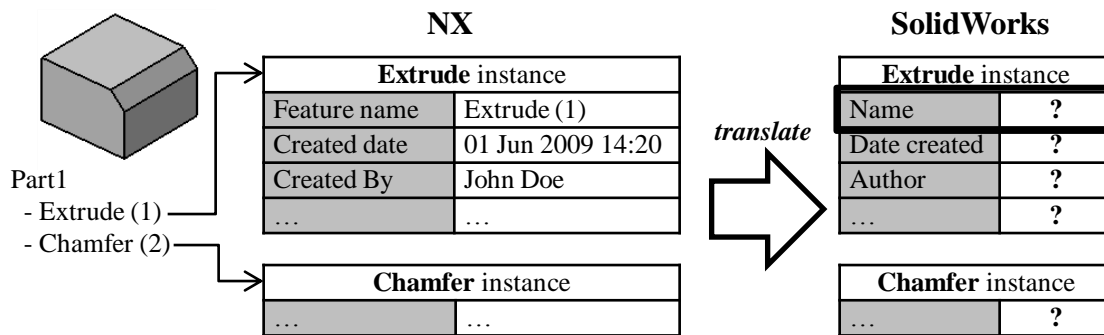


Figure 5.6: A schematic showing the translation of a part in *NX* into *SolidWorks*. To translate, the attribute values marked as “?” should be identified from the source instances. In the case study, the process to find only one attribute value - *Name* - enclosed with a thick box is demonstrated.

The proposed method requires an evaluation function and a set of basis function. For the evaluation function, a string similarity measure is needed because the goal node is string type, and edit distance (also known as Levenshtein distance), a well-known string similarity measure, is used. Note that the value is normalized with the string length so that the value is in the range [0, 1].

For the basis functions, only 3 functions will be used to highlight the procedure:

- f_1 : removes any parenthesis in the input string.
- f_2 : converts any numbers in the string into the words, e.g., 1 to one and 2 to two.
- f_3 : removes any spaces in the input string.

Figure 5.7 shows the search procedure. It starts with initial nodes (only 3 are shown for brevity), and expands the node with the smallest distance. When the search reaches a goal node, a translation rule can be reconstructed with the path information - basis functions associated with the arcs and the start node. As shown in the figure, the method successfully finds the translation rule for the given case. Also, the translation rule has been applied to a different set of aligned instances with the same attribute, and shows correct translations between the given instances.

There is a structure-level different between heterogeneous product development systems, and we discuss the case in the next section.

5.7.2 Structure-level translation

In the previous section, node expansion was explained that any new node is created from a single node with a set of basis functions. In other words, basis functions have single input and single output. In product development domain, however, not all attributes conform to the one-to-one translation pattern. For example, Figure 5.8 shows an *color* concept definition in different product schema - PLM XML by *Siemens* and 3D XML by *Dassault Systemes*. In PLM XML, a concept *RGBAType* is represented as a string with 4 float values separated with white space, while 3D XML has *RGBA-*

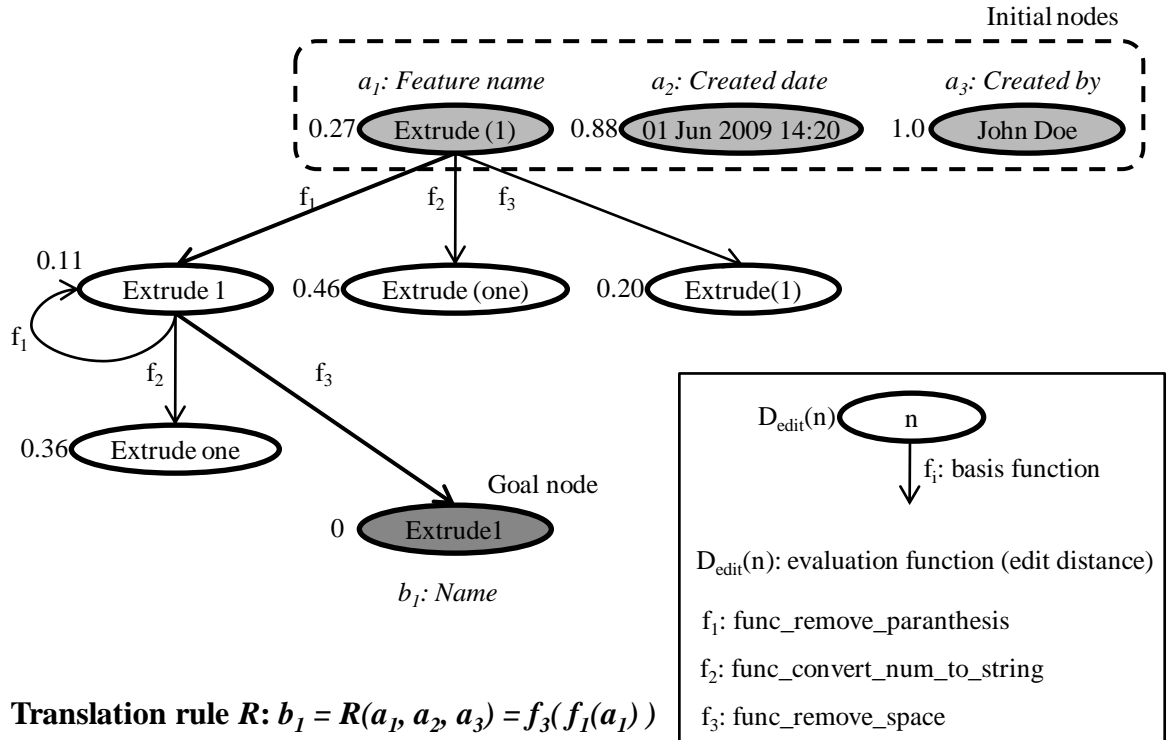


Figure 5.7: A translation rule is constructed with a graph search. There are 3 initial nodes, and the nodes with the smallest distance are evaluated. When the goal node is reached, the path information (arcs and the start node) is used to construct the translation rule. Note that the nodes are shown with only 1-tuple for brevity.

ColorType with 4 attributes - *red*, *green*, *blue* and *alpha* to represent the equivalent information.

In the example, the cardinality of the attribute matching is 4-to-1, and it cannot be represented with only 1-to-1 basis functions. In this example, a multi-input basis function is used to capture such relation. Specifically, a function f that merges two string values (separated with a space), is used.

Note that we handle the value of the nodes as string, and use the same edit distance used in the element-level translation. Figure 5.9 shows the procedure to

```
<... borderColour="0.4 0.7 1.0 0.5" ... >
```

(a) RGBAType in PLM XML: string with 4 float values

```
...
<Color xsi:type="RGBAColrType" red="0.4" green="0.7" blue="1.0" alpha="0.5"/>
...
```

(b) RGBAColrType in 3D XML: 4 attributes are used

Figure 5.8: Instances of color information used in (a) PLM XML by *Siemens* and (b) 3D XML by *Dassault Systemes*. In PLM XML, RGBA information is represented as a string of 4 values, while 3D XML shows them as separate attributes. This type of heterogeneity cannot be captured with the 1-to-1 conversion functions explained in the previous section and we need an extended function. Note that the nodes are shown with only 1-tuple for brevity.

find the translation rule.

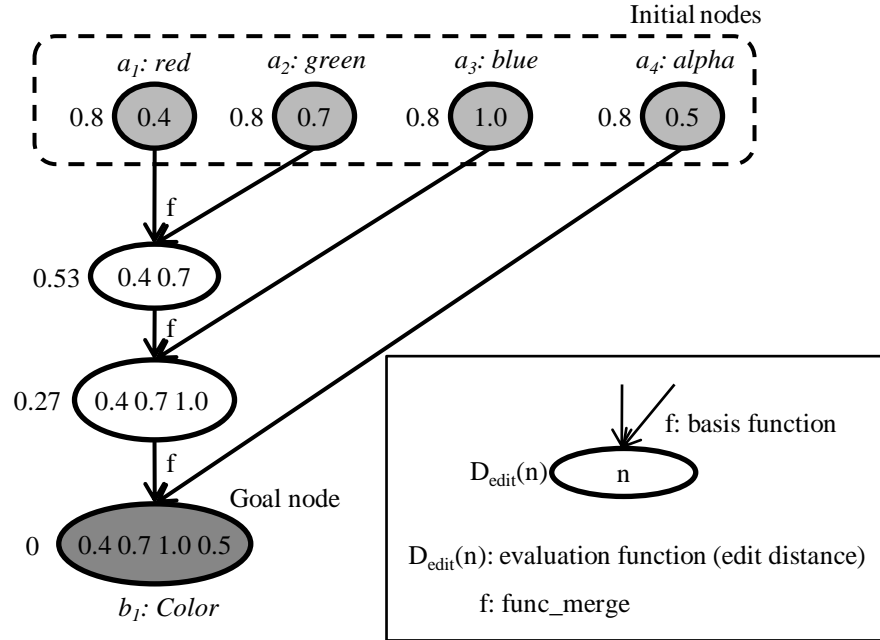
The rule successfully translates the source attributes into the goal attribute for the given aligned instance. The same rule also provides a correct translation between other pairs of aligned instances with the same attributes.

Note that in combining nodes to create a new node, we intentionally used a function that merges only two input strings at a time, rather than merging all 4 nodes at the same time. Using functions with multiple inputs should be considerate because it may increase search space drastically.

5.8 Summary

Product data translation is an essential part for integration of various product-centric activities, and there has been a need for a translation systems among the participating domains.

Creating a translation system is not a trivial task. Even with a semantic map fully identified, a syntax in the sending system should properly transform to the



Translation rule R : $b_1 = R(a_1, a_2, a_3, a_4) = f(f(f(a_1, a_2), a_3), a_4)$

Figure 5.9: A translation rule for one target attribute is constructed from 4 attributes. Only one basis function is shown to demonstrate the process; more functions can be included but will not change the result.

syntax in the receiving system while maintaining the same semantics. The challenge is that even within the same system different concept/attribute use different syntax, and different matching pairs require different translation rule. Even with the domain knowledges for both systems, it is time-consuming and error-prone task.

In this chapter, we have proposed a method to find the translation rule between any given pairs of concept/attributes. Given a set of basis functions - an atomic translation rule, we use a graph search method to automatically find a translation rule for a given pair of concepts as a combination of the basis functions.

The complete implementation of translation system will require further research:

First, the scope of the problem is limited to the translation between two concepts.

However, not all concept translation is 1-to-1, just like not all attribute translation is 1-to-1 as shown in this research. We believe it requires further analysis and modeling to address the issue. Extending the work to remove the limitation would be an important step toward the complete translators, and is left for the future work.

Also, we just briefly mentioned the size of problem space, and did not discuss any optimization issues. Given the multiple input basis functions, the number of nodes will increase combinatorially as the depth of the graph increases. The problem can be reduced with some optimizations, e.g., removing the initial nodes by finishing all element-level translation before dealing with any structure-level translation, but the further research is left for the future work.

CHAPTER VI

Conclusion

This chapter highlights important contributions of this thesis and discusses directions for future work.

6.1 Research contributions

The success of PLM relies on effective semantic interoperability of product information. This research has developed methodologies to determine matching between concepts in different product development systems.

In order to determine semantic maps, we have proposed a method - Instance-Based Concept Matching (IBCM) that can detect 1-to-n maps by exploiting implicit semantics captured in the instances of product models. The use of implicit semantics adds a new dimension in the area of product development, where most of the previous research has focused on using schema or data definition that are explicitly defined.

Any single matching method is not enough to determine the semantic maps across the different systems, since each method presents only one view. We have identified the challenges in combining multiple views provided by multiple methods and formalized the combination procedure for the first time in this domain. For this,

we have proposed a method - FEedback Matching Framework with Implicit Training (FEMFIT) to combine the different matching approaches using ranking Support Vector Machine (ranking SVM). The method overcomes the need to explicitly train the algorithm before it is used, and minimizes the decision-making load on the domain expert. Furthermore, it implicitly captures the expert's decisions on the matches without requiring him to input real numbers on similarity. The method can capture nonlinear relations between the individual matchers.

Finally, we have proposed a framework to automatically determine the translation rules to enable translation of concepts from one system to another. Even after the semantic maps are obtained, the syntax in the sending system should properly transform to the syntax in the receiving system. Existing work in product domain has focused mostly on finding semantic maps, but we identified the challenges in the translation process after the mapping. We use a graph search method that obtains the overall translation rule as a combination of multiple basic functions. Using such rules, concepts/instances from one system can be easily translated to another system.

6.2 Application to other areas

In this section, we project applications in other areas that can benefit from the technical contributions of this thesis.

- Other areas in PLM: Bill of Material (BOM) is a central component of product knowledge and is shared among different activities. Different BOMs view a product from different perspectives in different phases of the lifecycle, and therefore, they differ in structure and composition. Each of the three methods

(Instance-based matching, Combination of different views, and determination of translation rules) proposed in this research could be used to determine maps across the different BOMs.

- Matching/translation outside product domain: Interoperability is an issue not just in the product domain. Just like the heterogeneity issues in product domain, matching methods are required to integrate the scattered information in different schema, and the proposed concepts could be utilized. One example is the area of health-care, in particular the implementation of electronic records.
- Search/query problem: Finding similar documents, images, and videos are important issues in various fields. Searches are primarily based on some similarity between the query and the instances in the database. The proposed concepts, especially the framework to combine multiple views could be used to combine the existing similarity measures in these areas.

6.3 Future work

This section discusses future work that addresses certain limitations in the current work.

- Construction of ontological representations: Complete translation is not possible unless there are some explicit and open representations of the product development systems. We think, however, industry is moving toward more semantically rich representations where we do not need such manual construction of ontologies. For example, major CAD vendors have recently developed a

more expressive document representations, such as PLM XML by Siemens and 3D XML by Dassault Systemes. Such representations could reduce significant amount of manual work in parsing and reasoning about the concepts in the representations.

- Handling 1:n matches: While the IBCM can handle 1-to-in concept maps, the FEMFIT, used for combination of different individual methods, and the automated translation rule generation need to be improved to handle 1-to-n matches.
- Managing scalability: Further research is required to analyze and modify the proposed methods to handle scalability, particularly because product data is large and complex. Procedures should be designed to effectively use resources, such as parallel computing and the ability to store large datasets.
- Development of a repository: Currently, there is no formal basis to evaluate the correctness of the results of the research conducted in this dissertation. We have used our expert knowledge to determine the correctness of the results for the demonstrative cases that we have developed. However, competing techniques must be evaluated on a level playing field. There is a need for a repository that will provide standard datasets to benchmark/validate the different methods and tools developed by researchers in this field.

Bibliography

- [1] S. Brunnermeier and S. Martin. Interoperability cost analysis of the U.S. Automotive supply chain. Planning Report 99-1, Research Triangle Institute: Report prepared for the National Institute of Standards and Technology., 1999. URL <http://www.nist.gov/director/prog-ofc/report99-1.pdf>.
- [2] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pages 133–142, New York, NY, USA, 2002. ACM.
- [3] Farhad Ameri and Debasish Dutta. Product Lifecycle Management: Closing the knowledge loops. *Computer-Aided Design & Applications*, 2(5):577–590, 2005.
- [4] Michael P. Gallaher, Alan C. O’Connor, and Thomas Phelps. Economic impact assessment of the international standard for the exchange of product model data (STEP) in transportation equipment industries. Technical report, Research Triangle Institute, December 2002.
- [5] Raj Iyer and T. Gullledge. Product Lifecycle Management for the US Army weapon systems acquisition. In *Proceedings of the International Conference on*

Product Lifecycle Management (PLM'05): Emerging solutions and challenges for Global Networked Enterprise, pages 553–564, Lyon, France, 2005.

- [6] R. Sudarsan, S.J. Fenves, R.D. Sriram, and F. Wang. A product information modeling framework for product lifecycle management. *Computer-Aided Design*, 37(13):1399–1411, November 2005.
- [7] L. Patil, D. Dutta, and R. Sriram. Ontology-based exchange of product data semantics. *IEEE Transactions on Automation Science and Engineering*, 2(3): 213–225, 2005.
- [8] Q.Z. Yang, C.Y. Miao, Y. Zhang, and R. Gay. Ontology modelling and engineering for product development process description and integration. In *2006 4th IEEE International Conference on Industrial Informatics*, pages 85–90, 2006.
- [9] Alexandr Kuzminykh and Christoph Hoffmann. On validating STEP product data exchange. *Computer Aided Design*, Accepted for publication, 2008.
- [10] Peter Pin-Shan Chen. The entity-relationship model—toward a unified view of data. *ACM Trans. Database Syst.*, 1(1):9–36, 1976.
- [11] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal The International Journal on Very Large Data Bases*, 10(4):334–350, November 2001.
- [12] Min-Jung Lee, Min Jung, and Hyo-Won Suh. Semantic mapping based on ontology and a Bayesian Network and its application to CAD and PDM integration.

- In *Proceedings of IDETC/CIE 2006, ASME 2006 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, 2006.
- [13] Il Yeo, Lalit Patil, and Debasish Dutta. Semantic combination of matching methods for product data interoperability. In *5th International Conference on Product Lifecycle Management*, pages 167–176, 2008.
- [14] Lalit Patil. *Interoperability of formal semantics of product data across product development systems*. PhD thesis, University of Michigan, 2005.
- [15] Antoine Isaac, Lourens van der Meij, Stefan Schlobach, and Shenghui Wang. An empirical study of instance-based ontology matching. In *ISWC + ASWC '07: In proceedings of the 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference*, pages 253–266, November 2007.
- [16] Toralf Kirsten, Andreas Thor, and Erhard Rahm. Instance-based matching of large life science ontologies. In Sarah Cohen Boulakia and Val Tannen, editors, *DILS*, volume 4544 of *Lecture Notes in Computer Science*, pages 172–187. Springer, 2007.
- [17] Ryutaro Ichise, Hiedeaki Takeda, and Shinichi Honiden. Integrating multiple internet directories by instance-based learning. In *Proceedings of the eighteenth International Joint Conference on Artificial Intelligence*, pages 22–30, 2003.
- [18] Konstantin Todorov. Combining structural and instance-based ontology similar-

- ities for mapping web directories. In *Third International Conference on Internet and Web Applications and Services*, pages 596–601, 2008.
- [19] AnHai Doan, Pedro Domingos, and Alon Y. Halevy. Reconciling schemas of disparate data sources: a machine-learning approach. *SIGMOD Rec.*, 30(2):509–520, 2001.
- [20] AnHai Doan, Jayant Madhavan, Robin Dhamankar, Pedro Domingos, and Alon Halevy. Learning to match ontologies on the semantic web. *The VLDB Journal The International Journal on Very Large Data Bases*, 12(4):303–319, November 2003.
- [21] J.A. Larson, S.B. Navathe, and R. Elmasri. A theory of attributed equivalence in databases with application to schema integration. *IEEE Transactions on Software Engineering*, 15(4):449–463, 1989.
- [22] Wen-Syan Li and Chris Clifton. SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data & Knowledge Engineering*, 33(1):49–84, April 2000.
- [23] Lin Zhang and Jin-Guang Gu. Ontology based semantic mapping architecture. In *Proceedings of 2005 International Conference on Machine Learning and Cybernetics*, volume 4, pages 2200–2205, 2005.
- [24] Chao Wang, Jie Lu, and Guangquan Zhang. Integration of ontology data through learning instance matching. In *WI '06: Proceedings of the 2006*

- IEEE/WIC/ACM International Conference on Web Intelligence*, pages 536–539, Washington, DC, USA, 2006.
- [25] Hans Chalupsky. Ontomorph: A translation system for symbolic knowledge. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference (KR2000)*, pages 471–482, 2000.
- [26] Dejing Dou, Drew McDermott, and Peishen Qi. Ontology translation on the semantic web. *Journal on Data Semantics II*, LNCS 3360:35–57, 2004.
- [27] Oscar Corcho and Asunción Gómez-Pérez. ODEDialect: a Set of Declarative Languages for Implementing Ontology Translation Systems. *Journal of Universal Computer Science*, 13(12):1805–1834, 2007.
- [28] Fernando Silva Parreiras, Steffen Staab, Simon Schenk, and Andreas Winter. Model driven specification of ontology translations. In *ER 2008 - 27th International Conference on Conceptual Modeling*, volume LNCS 5231, pages 484 – 497, 2008.
- [29] Eswaran Subrahmanian, Rachuri Sudarsan, Abdelaziz Bouras, Steve Fenves, Sebti Fougou, and Ram Sriram. The role of standards in product lifecycle management support. Technical Report NISTIR 7289, National Institute of Standard and Technology, February 2006.
- [30] S. Szykman, S. J. Fenves, W. Keirouz, and S. B. Shooter. A foundation for interoperability in next-generation product development systems. *Computer-Aided Design*, 33(7):545–559, 2001.

- [31] Steven J. Fenves, Sebti Foufou, Conrad Bock, and Ram D. Sriram. CPM: A Core Model for Product Data. Technical report, National Institute of Standards and Technology, 2004.
- [32] Steven J. Fenves, Sebti Foufou, Conrad Bock, Rachuri Sudarsan, Nicolas Bouillon, and Ram D. Sriram. CPM 2: A Revised Core Product Model for Representing Design Information. Technical Report NISTIR 7185, National Institute of Standards and Technology (NIST), 2004.
- [33] Junhwan Kim, Michael J. Pratt, Raj G. Iyer, and Ram D. Sriram. Standardized data exchange of cad models with design intent. *Computer-Aided Design*, 40(7): 760–777, July 2008.
- [34] Yannis Kalfoglou and Marco Schorlemmer. Ontology mapping: the state of the art. *Knowledge Engineering Review*, 18(1):1–31, 2003.
- [35] Natalya F. Noy. Semantic integration: a survey of ontology-based approaches. *SIGMOD Record*, 33(4):65–70, December 2004.
- [36] Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer-Verlag New York, Inc., 2007.
- [37] J. D. Jobson. *Applied Multivariate Data Analysis*. Springer, 1991.
- [38] Jayant Madhavan, Philip A. Bernstein, , and Erhard Rahm. Generic schema matching with cupid. In *Proceedings of the Very Large Data Bases Conference (VLDB)*, pages 49–58, 2001.

- [39] Hong-Hai Do and Erhard Rahm. COMA: a system for flexible combination of schema matching approaches. In *VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases*, pages 610–621. VLDB Endowment, 2002.
- [40] Jérôme Euzenat and Pavel Valtchev. Similarity-based ontology alignment in OWL-Lite. In R. López de Mántaras and L. Saitta, editors, *Proceedings of the 16th European Conference on Artificial Intelligence*, pages 333–337. IOS Press, 2004.
- [41] A. Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, 1977.
- [42] Harold W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.
- [43] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics*, 5(1):32–38, 1957.
- [44] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:9–14, 1962.
- [45] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.
- [46] Horst A. Eiselt and Carl-Louis Sandblom. *Decision Analysis, Location Models, and Scheduling Problems*. Springer, 2004.

- [47] Kazuo Iwama and Shuichi Miyazaki. A survey of the stable marriage problem and its variants. In *ICKS '08: Proceedings of the International Conference on Informatics Education and Research for Knowledge-Circulating Society*, pages 131–136, 2008.
- [48] Michael Uschold and Michael Gruninger. Ontologies and semantics for seamless connectivity. *SIGMOD Rec.*, 33(4):58–64, 2004.
- [49] Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, 1974.
- [50] A. B. Kahn. Topological sorting of large networks. *Communications of the ACM*, 5(11):558–562, 1962.
- [51] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. *Proceedings of 18th International Conference on Data Engineering*, pages 117–128, 2002.
- [52] Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *Computational Learning Theory*, pages 144–152, 1992.
- [53] N. Lohse, H. Hirani, S. Ratchev, and M. Turitto. An ontology for the definition and validation of assembly processes for evolvable assembly systems. *The 6th IEEE International Symposium on Assembly and Task Planning: From Nano to Macro Assembly and Manufacturing*, pages 242–247, 2005.

- [54] Tim Simpson, Kemper Lewis, and Wei Chen. NSF: EXchanging Cyber-Infrastructure Themes in Engineering Design. Technical report, Report from the EXCITED Workshop held in Arlington, VA: National Science Foundation, 2005.
- [55] Line Pouchard, Nenad Ivezic, and Craig Schlenoff. Ontology engineering for distributed collaboration in manufacturing. In *Proceedings of the AIS2000 Conference*, 2000.
- [56] Deborah L. McGuinness and Frank van Harmelen. *OWL Web Ontology Language Overview*. World Wide Web Consortium (W3C), <http://www.w3.org/TR/owl-features/>, February 2004.
- [57] Michael J. Pratt. ISO/CD 10303-108, Product data representation and exchange: Integrated application resource: Parameterization and constraints for explicit geometric product models, 2001. URL http://www.tc184-sc4.org/SC4_Open/SC4_and_Working_Groups/WG12/N-DOCS/Files/WG12n940.pdf.
- [58] Sam Chapman. SimMetrics, 2006. URL <http://www.dcs.shef.ac.uk/~sam/simmetrics.html>.
- [59] G. Sheikholeslami, W. Chang, and Aidong Zhang. SemQuery: semantic clustering and querying on heterogeneous features for visual data. *Knowledge and Data Engineering, IEEE Transactions on*, 14(5):988–1002, 2002. ISSN 1041-4347.
- [60] Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, August 2004.

- [61] Duhwan Mun, Soonhung Han, Junhwan Kim, and Youchon Oh. A set of standard modeling commands for the history-based parametric approach. *Computer-Aided Design*, 35(13):1171–1179, 2003.
- [62] Jeongsam Yang, Soonhung Han, Joonmyun Cho, Byungchul Kim, and Hyun Yup Lee. An XML-Based Macro Data Representation for a Parametric CAD Model Exchange. *Computer-Aided Design and Applications*, 1:153–162, 2004.
- [63] Byungchul Kim and Soonhung Han. Integration of history-based parametric translators using the automation APIs. *Int. J. Product Lifecycle Management*, 2(1):18–29, 2007.
- [64] Jung-Ae Kwak and Hwan-Seung Yong. An approach to ontology-based semantic integration for PLM object. In *2008 IEEE International Workshop on Semantic Computing and Applications*, pages 19–26, 2008.
- [65] Alexander Smirnov, Michael Pashkin, Tatiana Levashova, and Nikolai Chilov. Ontology-based support for semantic interoperability between SCM and PLM. *Int. J. Product Lifecycle Management*, 1(3):289–302, 2006.
- [66] Kiyoshi Ito. *Encyclopedic Dictionary of Mathematics*. MIT Press, 2nd edition, 1993.
- [67] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: Modern Approach*. Prentice Hall, 2nd edition, 2003.

- [68] R. W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29(2):147–160, 1950.