

## Association for Information Systems AIS Electronic Library (AISeL)

---

PACIS 2009 Proceedings

Pacific Asia Conference on Information Systems  
(PACIS)

---

July 2009

# Synthesizing System Integration Requirements Model Fragments

Narasimha Bolloju

*City University of Hong Kong*, [narsi.bolloju@cityu.edu.hk](mailto:narsi.bolloju@cityu.edu.hk)

Chuan Hoo Tan

*City University of Hong Kong*

Follow this and additional works at: <http://aisel.aisnet.org/pacis2009>

---

### Recommended Citation

Bolloju, Narasimha and Tan, Chuan Hoo, "Synthesizing System Integration Requirements Model Fragments" (2009). *PACIS 2009 Proceedings*. 65.

<http://aisel.aisnet.org/pacis2009/65>

This material is brought to you by the Pacific Asia Conference on Information Systems (PACIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in PACIS 2009 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact [elibrary@aisnet.org](mailto:elibrary@aisnet.org).

# SYNTHESIZING SYSTEM INTEGRATION REQUIREMENTS MODEL FRAGMENTS

Narasimha Bolloju

Associate Professor  
Department of Information Systems  
City University of Hong Kong, Hong Kong  
[narsi.bolloju@cityu.edu.hk](mailto:narsi.bolloju@cityu.edu.hk)

Chuan Hoo Tan

Assistant Professor  
Department of Information Systems  
City University of Hong Kong, Hong Kong  
[ch.tan@cityu.edu.hk](mailto:ch.tan@cityu.edu.hk)

## Abstract

*Systems integration is an enduring issue in organizations. Many organizations have often been faced with the predicament of managing large and complex IT infrastructures accumulated over the years. Before proposing suitable integration architecture and selecting appropriate implementation solutions, a holistic and clear understanding of the enterprise-wide integration requirements among various internal and external systems is needed. This paper builds on prior literature on conceptual modelling of integration requirements to present an algorithm that synthesizes model fragments, i.e., piecemeal sections of the integration requirements. The details of the algorithm, for synthesizing two or more model fragments into a single integration requirements model, are detailed in this paper. An empirical assessment of the algorithm's generated integration solution is made by comparing it against that performed manually.*

Keywords: systems integration, requirements modeling, conceptual modeling, and synthesizing requirements

## 1 INTRODUCTION

*“...the economic slowdown may trigger a rise in mergers and acquisitions--which could create significant opportunities for IT consulting and systems integration organizations.”* Natasha Lomas, ZDNet Asia (Oct 29, 2008)

Systems integration is an enduring issue faced by organizations with large and complex Information Technology (IT) infrastructures accumulated over the years. Indeed, with more organizations undergoing restructuring, facing acquisitions and mergers, and reengineering of the systems, there is a growing need for organizations to embark on the system integration task. Such needs lead to the emergence of integration technologies such as the Service-Oriented Architectures (SOA) and Event-Driven Architectures (EDA). SOA denotes an architecture that offers an open, agile, extensible, composable platform comprised of autonomous, interoperable, discoverable and potentially reusable services. EDA builds on the principle that events trigger messages to be sent between systems that are completely unaware of each other and the systems interested in specific types of events react. While SOA and EDA help organizations to move toward more flexible and agile IT infrastructures, the capacity of an organization to have a thorough understanding of the integration requirements could directly affect the choice and implementation strategies of the integration task.

Our review of the extant literature suggests that there is a startling shortage of studies conducted that could serve to assist systems integrators and architects to understand integration requirements, and to construct a conceptual model of interactions among the systems and databases. One of the exceptions

is the work by Bolloju (2009), which presents a conceptual modeling technique that comprises of a set of modeling constructs and a method for obtaining them. Due to the method's focus on identifying the logical requirements from each system's perspective, the individuals or teams responsible for those systems are able to capture and model those requirements more efficiently and effectively. However, the issues related to synthesizing the model fragments (e.g., how to reconcile the differences in interpretation of the fragments) are not addressed. In our view, the synthesizing process requires significant effort in dealing with semantic and structural differences in representation across different model fragments. Since the number of model fragments to be merged or synthesized can be quite large, algorithmic support is expected to contribute to the efficiency and effectiveness of this activity.

This study presents an algorithm to merge model fragments reconciling the naming differences among those fragments in formulating a synthesized model of the integration requirements represented by individual model fragments. Section 2 includes background material related to approaches to modeling integration requirements and associated limitations. Section 3 presents an overview of the integration requirements model synthesizing process, the details of the synthesizing algorithm and its implementation. We conclude the paper in Section 5 after providing details of an empirical evaluation of the implemented algorithm in Section 4. We believe this study will contribute to the cumulative knowledge of system integration, which could yield practical and tangible benefits to organizations which are in search of tested system integration requirements modelling techniques.

## 2 BACKGROUND

System integration refers to the construction of linkages among computer systems and databases (Markus 2000). System integration is necessary as organizations often have to add new functionalities into the organizational Information Systems (IS) portfolio, which frequently entails dealing with indispensable legacy systems. Research on system integration primarily builds on the theoretical *raison d'être* to focus on the conceptual modeling technique with which an integrated representation of the selected phenomena in the focal domain is constructed (Wand and Weber 2002). To construct a proper conceptual system integration model, it is imperative that the system modelers are informed of and educated on the appropriate ways of representing the requirements, reconciling the differences in requirements and deriving a well-integrated representation of the systems as a whole.

Our review of the extant studies suggests that research on system integration requirements modeling has been scarce; among those studies conducted, a significant number of them focus on data integration (Bergamaschi et al. 1999; Batini et al. 1986; Batini and Lenzerini 1984). For instance, Schmitt and Saake (2005) built on the formal concept analysis to propose a generic integration modeling method that integrates schemata from heterogeneous databases to form a homogenized schema. Although these studies have been instrumental in enhancing our understanding of data integration, there is a growing realization that we should not undermine the importance of understanding and proposing modeling mechanisms that connect and integrate among heterogeneous applications (Bolloju 2009; Bass and Lee 2002). One of the challenges facing researchers is that different architectural domains entail different modeling techniques and concepts, making it difficult to integrate applications or even systems across the entire organization (Jonkers et al. 2004). Indeed, due to the complexities involved, several related studies have been conducted on the basis of practical wisdom rather than from theoretical angle (see e.g., Gold-Bernstein and Ruh 2004). The result of these expeditions is that system modelers often construct topological representations that do not accurately reflect the focal domain due to: (1) modeling incompleteness, i.e., discarding important details, (2) ambiguity, i.e., inconsistency in the modeling of processes and parameters passing, and/or (3) detail excessiveness, i.e., inclusion of irrelevant details that may not aid in the model clarity (Greca and Moreira 2000). The root of the problem, as highlighted by Weber (2003), is that conceptual modeling attempt has been undermined because it suffers from a lack of sound theoretical foundations to underpin its research, pedagogy, and practice.

Conceptual modeling is rooted by the seminal work of Brachman (1979) who conceived conceptual modeling as composed of a small set of language-independent elements and relationships, which could be used to capture and express semantic meanings of the focal domain. This set of language-

independent elements and relationships could yield desirable modeling outcome properties, such as modeling correctness. Building on this notion, researchers have proposed various forms of modeling techniques, such as service invocations in the sequence and communication diagrams of Unified Modeling Language (UML) and the dataflow diagram that forms layers of abstraction, which are traditionally focused on modeling interactions within a system (Kendall and Kendall 2002). Moreover, these techniques are designed to be used during the system development stage, making them less applicable for situations in which established operational systems are to be integrated.

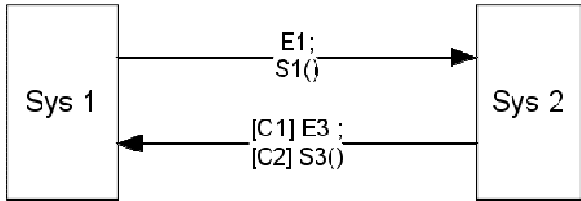
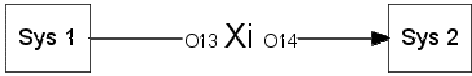


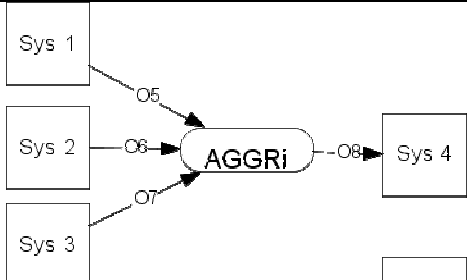
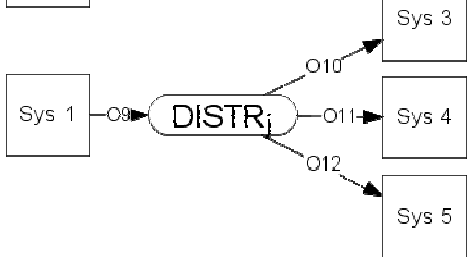
Construct	Illustration
<p>Nodes Sys1 and Sys2 represent systems; Links (directed arrows) indicate interactions: services (S1, S3), events (E1, E3). Guard conditions C1 and C2, filter service requests and events reaching the destination.</p>	
<p>X defines translation of O13 to O14 (O<sub>i</sub> represents service request or event)</p>	
<p>BATCH defines batching or grouping of several O1s into one O2.</p>	
<p>SPLIT defines separating or splitting O3 into individual service requests/events O4.</p>	
<p>AGGR defines aggregation or combining O5, O6, O7 into a single aggregated or composite service request or event O8.</p>	
<p>DISTR defines distribution or slicing a composite O9 into several service requests or events.</p>	

Table 1. Summary of System Integration Constructs (source: Bolloju, 2009)

A more recent attempt by Bolloju (2009), which seeks to address the challenges of system integration, proposes a set of constructs. The proposed technique builds on the two-stages modeling approach, in which the modeler first creates model fragments representing integration requirements for each system perspective, and subsequently synthesizes the fragments to form an integrated, holistic model of the requirements of the entire set of systems studied. To facilitate the modeling, a set of constructs,

namely the nodes which represent individual or group of systems and data sources, and the links which denote the service requests and events between a pair of nodes, is proposed (see Table 1 for a summary of the constructs). Section 3 presents two examples of model fragments and a corresponding manually synthesized requirements model. An evaluation of the constructs and method for creating model fragments and synthesizing the model fragments indicated that the process is effort intensive and error prone, and consequently highlighted a strong need for algorithmic support for synthesizing a given set of model fragments. Specifically, the manual synthesizing process included reconciliation of differences among the model fragments that were created by numerous modelers. In this research, we seek to address this limitation of the technique proposed by Bolloju (2009).

### 3 SYNTHESIZING INTEGRATION REQUIREMENT MODELS

Modeling system integration requirements is an iterative process that involves several steps (see Figure 1). The team of the system analysts and integrator would first need to gather and study the interaction, e.g., procedural invocation and message passing, among the various systems. The output of this step is a set of model fragments that entail the various interactions among systems, which could include both internal and external systems. The model fragments are consolidated to form an integrated model, known as the enterprise integration requirements model. Next, the overall enterprise integration architecture (e.g., SOA and EDA that build on top of an enterprise service bus) is derived based on the enterprise integration requirements model. Finally, based on the overall enterprise integration architecture specific solutions meeting the requirements of individual systems can be developed.

It is to be noted that this form of bottom-up approach, i.e., gathering various integration requirements and then synthesizing the model fragments to form an overall enterprise requirement, poses two challenges. First, discrepancies in naming the nodes, services, events, attributes of events, parameters of services could occur across model fragments. For instance, different names could be used to describe employee number, e.g., empNo, emplNo, emp\_no, empNumb, empID, emp-no. Second, structural differences in parameters of services and attributes of events could occur. For instance, number of arguments, order of the arguments, types of arguments of a service between two systems could differ, making the integration of the model fragments challenging. In this study, we propose an algorithm that seeks to synthesize and amalgamate the model fragments to form the enterprise integration requirements model.

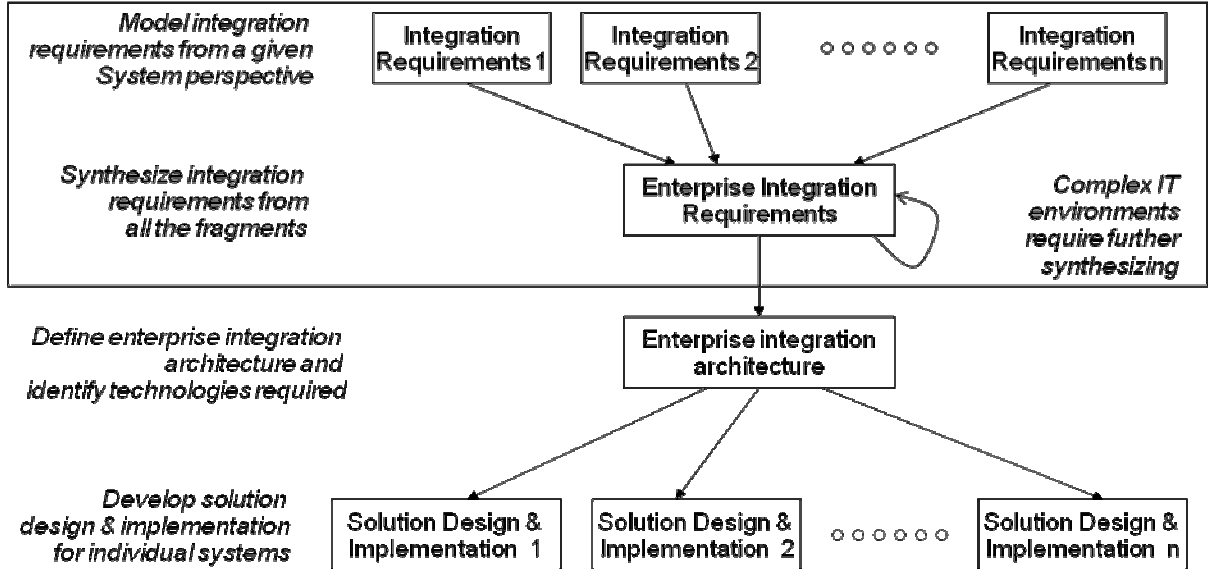


Figure 1. Bottom-up method for modeling integration requirements

### 3.1 Synthesizing Process

Synthesizing algorithm merges a given set of model fragments (see Figures 2a and 2b for example) into a single synthesized integration requirements model (see Figure 3 for example) in two steps. First, a combined model is created with one additional intermediary transformation node group to collect all transformation nodes from the input models. In this process, interactions such as System A consuming service s() from System B are mapped to two interactions, one System A to consume s() X and the other X to consume s() from System B where X is a null transformation added to the intermediary transformation node group (see Table 2 for code fragment).

#### Loan System Model Fragment

##### Integration Requirements

1. get a batch of loan applications from IBPS at the end of every working day;
2. gets credit score for a given loan applicant from External Credit Rating System (ECRS);
3. send loan applications status changes (rejections/approvals) to IBPS at the end of the day;
4. Credit instructions are sent by LS to the CBS to credit approved loan amount to the customer account

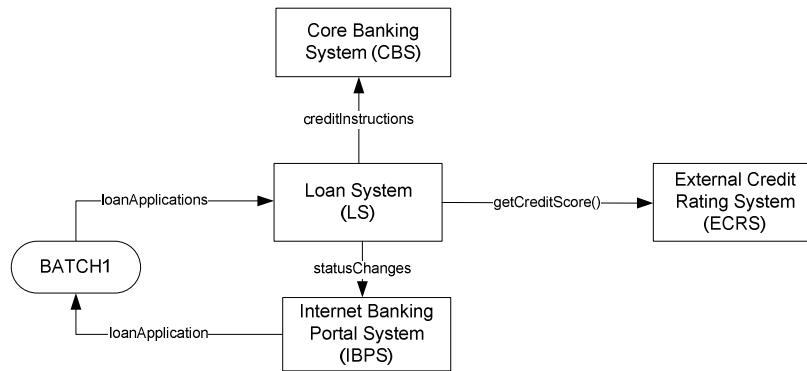


Figure 2a. Sample model fragment A.

#### Mutual Fund Trading System (MFTS)

##### Integration Requirements

1. send closing quotes to Internet Banking Portal System (IBPS) at the end of each trading day;
2. get buy/sell orders from IBPS;
3. forward batches of buy/sell orders to mutual fund company order systems;
4. send instructions to Core Banking System (CBS) to hold the amount due in customers account;
5. send debit or credit instructions to CBS;
6. forward transaction results (purchases and sales) to IBPS.

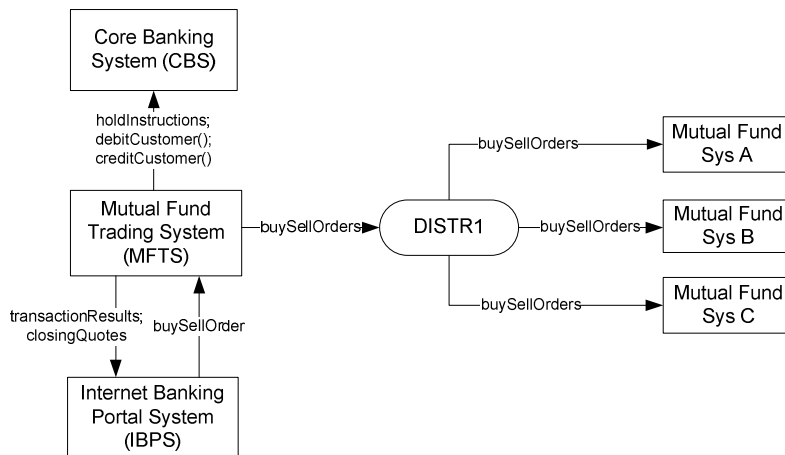


Figure 2b. Sample model fragment B.

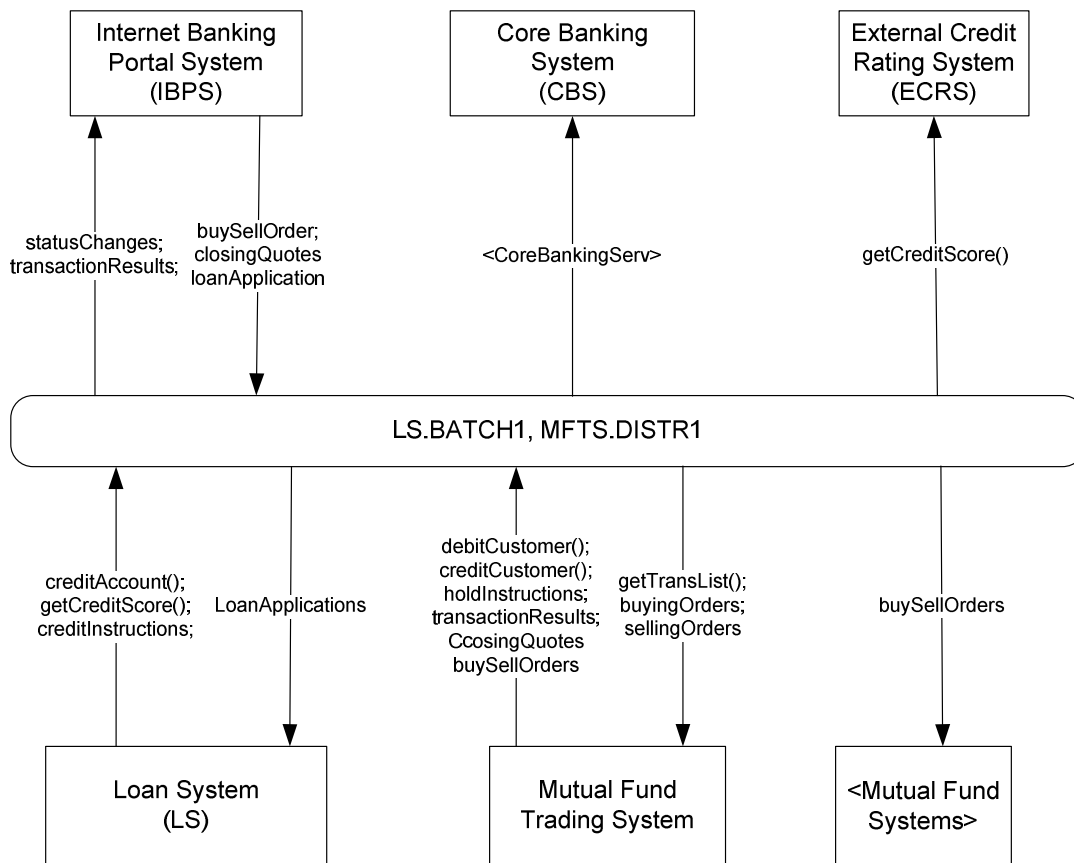


Figure 3. Sample enterprise integration requirements model.

The second step resolves similarities among nodes and similarities among interactions in the intermediary requirements model (see Table 3). This step makes use of the hierarchical clustering technique to group similar nodes into node clusters, and similar interactions between a system node and intermediary transformation interaction clusters based on predefined threshold values for similarity scores. For this purpose, any pair of elements that match exactly will be considered equal (similarity score 1) and those that do not match exactly will receive a score between 0 and 1. The similarity score for names is computed using a string similarity function. As part of this resolution process, names of the system nodes are resolved completely before resolving interactions (i.e., services and events) using the interaction similarity function described in Table 3. .

Current implementation of the synthesizing algorithm uses a composite string similarity function to compute a similarity score of a given pair of strings (in their lowercase form) as an average of the following commonly used metrics: 1) Euclidean distance, 2) Chapman's Matching Soundex, 3) Jaro, 4) Levenshtein, 5) Monge Elkan, and 6) Smith-Waterman<sup>1</sup>. After experimenting with several individual similarity functions using several sets of labels selected from different projects, the combined score produced using the above six metrics was found to result in more representative similarity scores (neither too high nor too low) for typical variations expected in naming of various elements used in integration models. A weighted combination of different similarities was used for the calculation of interaction similarities - for services a combination of name, return value and parameter similarities, and for events a combination of name and attribute similarities.

<sup>1</sup> Due to the space constraint, we will not elaborate on the six metrics. For a quick understanding of the various metrics, please refer to <http://www.dcs.shef.ac.uk/~sam/stringmetrics.html>.

The resolution process uses user-defined threshold values (minimum cut-off scores) for automatic matching of a given type of elements. Any values above the threshold value require manual resolution. The system maintains the history of manual resolutions and makes use of that "knowledge" for identifying future similarities.

```

synthesize(in Mf: set of IntReqModels; out Ms: IntReqModel)
begin
    Ms := null;
    IntMs := mergeFragments(Mf, Ms);
    MS := resolveSimilarities(IntMs);
end;
mergeFragments(in Mf: IntReqModel, inout Ms: IntReqModel)
// merge model fragment Mf into synthesized model Ms
begin

    add a transformation node group T to Ms;           // to collect transformations in various model fragments
    for each interaction <N1,Int,N2> in Mf do         //N1, N2 node pair, Int is either service request or event
    begin
        if node N1 is a transformation node
            then addNodeToGroup(N1,T)
            else addNode(N1,Ms);                     // adds node only if it does not exist in Ms
        if node N2 is a transformation node
            then addNodeToGroup(N2,T)
            else addNode(N2,Ms);                     // adds node only if it does not exist in Ms;
        if interaction <N1,Int,T> does not exist in Ms
            then addInteraction<N1,Int,Xi>;          // add a null transformation node Xi
        if interaction <T,Int,N2> does not exist in Ms
            then addInteraction<Xi,Int,N2>;
    end;
end; // merge

```

Table 2. Pseudocode for merging model fragments.

```

resolveSimilarities(Ms: IntReqModel)
begin
    create system node clusters with nodeSimilarity above nodeThreshold value;
    create interaction clusters with interactionSimilarity above intThreshold value;
end;
nodeSimilarity(N1:Node,N2:Node): float
begin
    if N1 and N2 are identical or defined equivalent, or belong to an existing cluster
    then return 1
    else begin
        morphologicalEq(N1.name,MN1);
        morphologicalEq(N2.name,MN2);
        return stringSimilarity(MN1,MN2);
    end
end;
interactionSimilarity(Int1:Interaction,Int2:Interaction): float
begin
    if Int1 is a service and Int2 is a service
    then return(0.6*stringSimilarity(Int1.name,Int2.name)
        +0.2*returnValueTypeSimilarity(Int1.returnType,Int2.returnType)
        +0.2*argumentSimilarity(Int1.arguments,Int2.arguments))
    else return(0.7*stringSimilarity(Int1.name,Int2.name)
        +0.3*attributeSimilarity(Int1.arguments,Int2.arguments))
end;

```

Table 3. Pseudocode for resolving similarities.



### 3.2 Implementation

The current system implementation aims at validating the synthesizing process. This system facilitates 1) capturing descriptions of model fragments along with related data such as user-defined equivalences via a synonym list, and threshold values for merging algorithm, and 2) merging of selected model fragments into one synthesized integration requirements model.

This system represents the collection of model fragments as a project which is internally represented as a Java object (see the meta-model shown in Figure 4) and saved as an XML document using XStream conversion library. The same converter is also used to load a saved project (XML document) and convert it into a Java object for further manipulation. The system supports entry of model fragment descriptions through a simple forms-based user interface (final version will provide a graphical drawing facility). This system is implemented using Java 5.0 with SimMetrics 1.6.2 library for computing string similarities and XStream 1.3 for converting Java objects to XML documents and vice versa, in order to provide persistence.

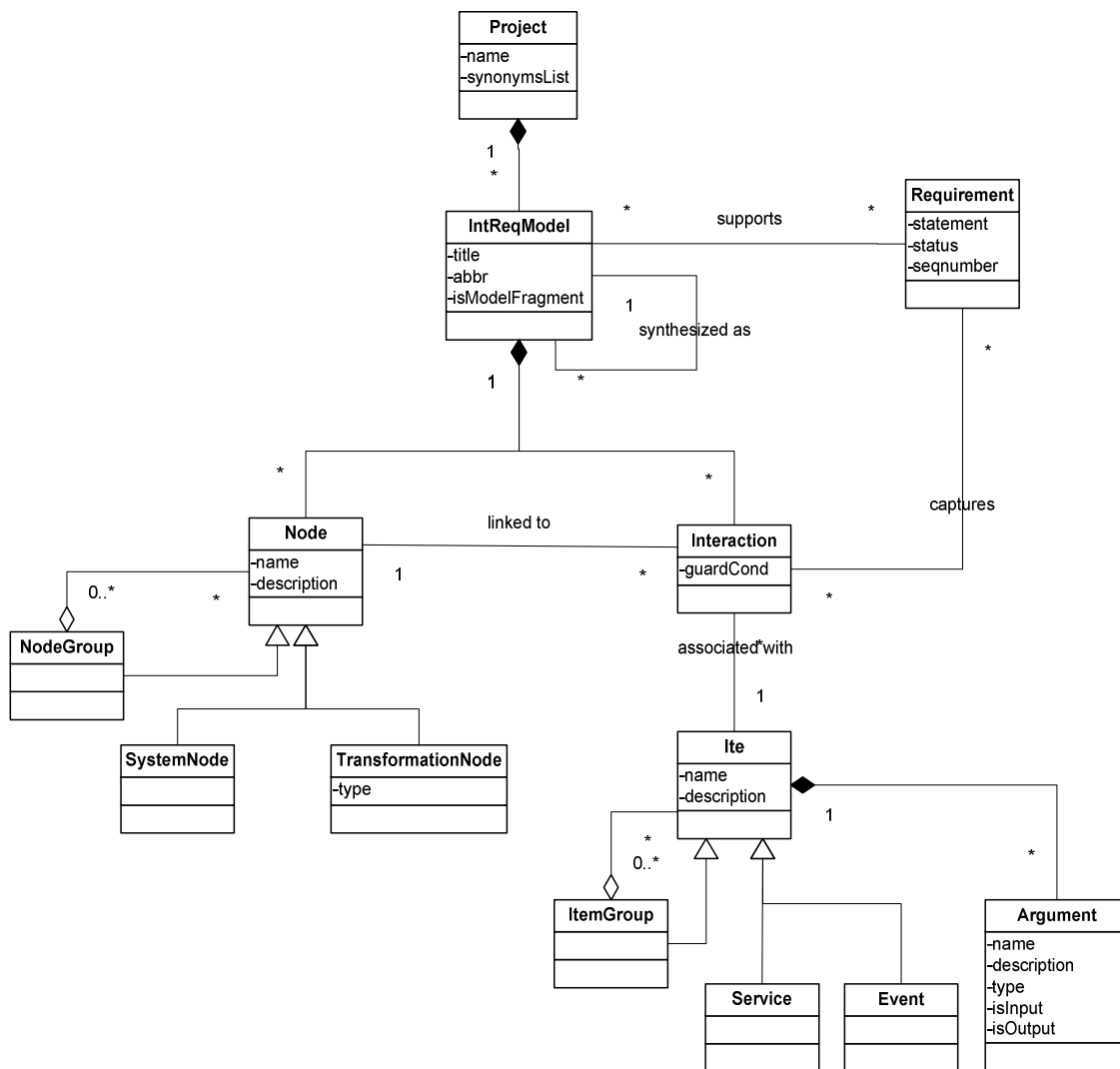


Figure 4. Meta model of integration requirements model

## 4 EVALUATION

An evaluation of the synthesizing algorithm was conducted to understand its effectiveness by comparing three manually synthesized models with the synthesized models generated by the system. The model fragments and synthesized models were created and submitted as project reports by teams

of graduate students of a course on systems integration. Each project report, in addition to one synthesized model, had several model fragments created by the student teams and descriptions of services, events and transformations. Table 4 depicts the details of these projects where the last column indicates the total number of links present in model fragments and the total number of services and events (some links are associated with multiple interactions). The descriptions of these model fragments were input by a research assistant into the system to generate the synthesized models.

Project #	# of Model Fragments	# of Systems	# of Interactions (services and events)
P09	9	11	56 (links 53)
P12	6	17	57 (links 34)
P14	5	16	51 (links 25)

*Table 4: Contents of project reports used for evaluation*

Since the implemented system is expected to address the effort intensive and relatively straight forward merging of a set of model fragments, the evaluation is focused on the effectiveness (rather the efficiency) of the synthesizing process – especially in dealing with various labels or identifiers of model elements that could have slight variations. The effectiveness is measured using recall and precision measures which are widely used in information retrieval area. The recall (relevant/existing) is defined as the ratio of the number of interactions in the generated synthesized model in the manually synthesized model and the number of interactions in the manually synthesized model. The precision (relevant/retrieved) is defined as the ratio of the number of interactions in the generated synthesized model that are present in the manually synthesized model and the number of interactions in the generated synthesized model. Finally, the F-measure which is defined as the harmonic mean of recall and precision ( $2 * recall * precision / (recall + precision)$ ) is used to arrive at a single measure of effectiveness of the synthesizing algorithm.

Project #	Exact match			Approximate match		
	Recall	Precision	F-measure	Recall	Precision	F-measure
P09	86.79	82.14	84.40	98.11	92.86	95.86
P12	70.59	42.11*	52.75	73.53	43.86*	54.95
P14	100	49.02*	65.79	100	49.02*	65.79

*Table 5: Evaluation results of synthesizing algorithm*

Table 5 lists the values of recall, precision and F-measure for two sets of comparisons: a) exact match where the elements in the generated models must be identical to those in the manually synthesized models, and b) approximate match where minor differences are ignored. The low precision score in the second and third rows of the above table were identified as a result of problems such as missing systems, services, and events in the manually synthesized models. The lower scores, resulting from the problems associated with manual synthesizing process, also indicated the need for automated support for synthesizing model fragments especially when large number of models to be merged.

## 5 CONCLUDING COMMENTS

This paper builds on the prior work on conceptual modeling of systems integration requirements to present an algorithm that synthesizes model fragments, i.e., piecemeal sections of the integration requirements. The process of synthesizing two or more model fragments into a single integration requirements model is detailed in this paper. The algorithm makes use of string similarity metrics and parameter matching to arrive at clusters of similar elements in the model fragments. The effectiveness of the implemented algorithm is demonstrated by comparing the synthesized integration models generated by the system against those performed manually. This study, thus, contributes an effective solution to the effort-intensive and error-prone synthesizing process.

One of the limitations of this study is related to employing student project reports to validate the synthesized algorithm. Although these project reports contained a good number of model fragments, they may still be a distance from those in the real world IT infrastructure in terms of complexity and sheer size. While it is quite natural that the models generated by the implemented system may not exclude any elements from the input model fragments, the process of combining similar elements is likely to be superior if it is done manually.

Based on the findings from this study, we have developed a tool to support the synthesizing process. We are currently in the process of revising the tool and when this tool is completed, the human role in the synthesizing process will be limited to manually identifying and resolving a small set of elements from several model fragments.

In conclusion, we believe this work is unique for it builds on the conceptual modeling techniques used in systems development to system integration. Further research in this regard could assist organizations' IT architects in eliciting and representing system integration requirements.

## References

- Bass, C. and Lee, J. M. (2002). "Building a Business Case for EAI", *eAI Journal*, January, 18-20.
- Bolloju, N. (2009). "Conceptual Modeling of System Integration Requirements", *IEEE Software*, forthcoming.
- Batini, C., & Lenzerini, M. (1984). "A methodology for data schema integration in the entity relationship model", *IEEE transactions on software engineering*, 10(6), 650-664.
- Batini, C., Lenzerini, M., & Navathe, S. (1986). "A Comparative Analysis of Methodologies for Database Schema Integration", *ACM Computing Surveys*, 18(4).
- Bergamaschi, S., Castano, S., & Vincini, M. (1999). "Semantic integration of semistructured and structured data sources", *ACM SIGMOD Record*, 28(1), 54-59.
- Brachman, R. J. (1979). "On the Epistemological Status of Semantic Networks", In: N. V. Findler (ed.): *Associative Networks: Representation and Use of Knowledge by Computers*, Academic Press, New York, 3-50.
- Greca, I. M., & Moreira, M. A. (2000). "Mental Models, Conceptual Models, and Modeling", *International Journal of Science Education*, 22(1), 1-11.
- Jonkers, H., Lankhorst, M., and Buuren, R. V. (2004). "Concepts for Modeling Enterprise Architectures", *International Journal of Cooperative Information Systems*, 13(3), 257-287.
- Kendall, K. E. & Kendall, J. E. (2002). *Systems Analysis and Design*. Prentice Hall, 7th Edition
- Markus, M. L. (2000). "Paradigm Shifts - e-Business and Business/ Systems Integration", *Communications of the AIS*, 4(1), 1-45.
- Schmitt, I., & Saake, G. (2005). "A comprehensive database schema integration method based on the theory of formal concepts", *Acta Informatica*, 41(7), 475-524.
- Wand, Y. & Weber, R. (2002). Research Commentary: Information Systems and Conceptual Modeling – A Research Agenda. *Information Systems Research*, 13(4), 363-377.
- Weber, R. (2003). "Conceptual Modeling and Ontology: Possibilities and Pitfalls", *Journal of Database Management*, 14(3), 1-20.