Editorial Manager(tm) for Information Systems and e-Business Management
Manuscript Draft

Manuscript Number: ISEB-D-09-00028R1

Title: A Method for Comparing Legacy and Component-based Models in Re-engineering

Article Type: Special Issue on Design Science

Keywords: legacy systems; re-engineering; Bunge-Wand-Weber model; component-based systems; requirements models; design science.

Corresponding Author: Dr Aileen Cater-Steel,

Corresponding Author's Institution:

First Author: Raul Valverde

Order of Authors: Raul Valverde; Mark Toleman; Aileen Cater-Steel

Abstract: Recently, many organisations have become aware of the limitations of their legacy systems to adapt to new technical requirements. Trends towards e-commerce applications, platform independence, reusability of pre-built components, capacity for reconfiguration and higher reliability have contributed to the need to update current systems. Consequently, legacy systems need to be re-engineered into new component-based systems. This paper shows the use of the design science approach in information systems re-engineering. In this study, design science and the Bunge-Wand-Weber (BWW) model are used as the main research frameworks to build and evaluate conceptual models generated by the component-based and traditional approaches in re-engineering a legacy system into a component-based information system. The objective of this study is to develop a framework to compare a system designed and developed using traditional methods to a component-based system to verify that the re-engineered component-based model is capable of representing the same business requirements as the legacy system.

Response to Reviewers: see attachment

A Method for Comparing Legacy and Component-based Models in Re-engineering

Abstract

Recently, many organisations have become aware of the limitations of their legacy systems to adapt to new technical requirements. Trends towards e-commerce applications, platform independence, reusability of pre-built components, capacity for reconfiguration and higher reliability have contributed to the need to update current systems. Consequently, legacy systems need to be re-engineered into new component-based systems. This paper shows the use of the design science approach in information systems re-engineering. In this study, design science and the Bunge-Wand-Weber (BWW) model are used as the main research frameworks to build and evaluate conceptual models generated by the component-based and traditional approaches in re-engineering a legacy system into a component-based information system. The objective of this study is to develop a framework to compare a system designed and developed using traditional methods to a component-based system to verify that the re-engineered component-based model is capable of representing the same business requirements as the legacy system.

***Keywords**: legacy systems; re-engineering; Bunge-Wand-Weber model; component-based systems; requirements models; design science.*

1. INTRODUCTION

The objective of this study is to develop a framework to compare a system designed and developed using traditional methods to a component-based system to verify that the re-engineered component-based model is capable of representing the same business requirements as the legacy system. Design science is the research approach used. Design science has a history of providing good results in the evaluation of constructs and models in information systems (Hevner et al. 2004). This is in line with Nunamaker and Chen (1990) and Gregor (2002) who classify design science in IS as applied research that applies knowledge to solve practical problems. March and Smith (1995) define design science as an attempt to create things that serve human purposes, as opposed to natural and social sciences, which try to understand reality (Au 2001).

The business problem chosen to demonstrate the use of design science relates to the re-engineering of a legacy system in a financial institution. The vast majority of legacy information systems were implemented using the traditional paradigm. The traditional paradigm consists of modeling techniques used by system analysts including system flow charts and data flow diagrams (DFD) to capture, during the analysis phase, the activities within a system. However, with recent developments, particularly trends towards e-commerce applications, platform independence, reusability of pre-built components, capacity for reconfiguration and higher reliability, many organizations are realizing they need to re-engineer their systems. Given the limitations of legacy systems to adapt to these new technical requirements, new component-based systems are required to meet these trends. However, there is a high degree of interest and concern in establishing whether or not a full migration to a more portable and scalable component-based architecture will be able to represent the legacy business requirements in the underlying conceptual model of re-engineered information systems.

To address this concern, the research study re-engineered a sample process to derive a component model from the legacy system and addressed the question: *Can a framework be developed to enable the comparison of a system designed and developed using traditional methods with a component-based system?* In particular, it is important to ensure requirements are equivalent and to include a process within the framework that shows this equivalency.

In order to answer the research question, the project used a build/evaluate approach (Hevner et al., 2004). Conceptual models were generated by the component-based and traditional approaches in the re-engineering process in order to verify that the re-engineered component-based model was capable of representing the same business requirements of the legacy system. Design science is used as the central research approach for this project.

In this paper, the focus is not on reporting the outputs of the re-engineered business process but on the procedures and frameworks used by the researcher in comparing the requirements models of the traditional and component-based approaches.

In the first section, the BWW model is introduced as a tool for requirements model evaluation. The research method using the design science approach is then described and a framework proposed. The framework is applied to a case study. Both the building and comparison activities are described. The results of the comparison are provided and directions of future research are suggested in the conclusion.

2. BACKGROUND

Over the years, many different ontologies have emerged as a way to model reality. One general ontology that has been frequently applied for the evaluation of modeling methods in Systems Analysis and Design is the Bunge-Wand-Weber model (Wand and Weber, 1988, 1993, 1995).

The fundamental premise of the BWW (Bunge-Wand-Weber) model (Wand & Weber, 1988, 1993, 1995) is that any Systems Analysis and Design modeling grammar (set of modeling symbols and their construction rules) must be able to represent all things in the real world that might be of interest to users of information systems; otherwise, the resultant model is incomplete. If the model is incomplete, the analyst/designer will somehow have to augment the model(s) to ensure that the final computerized information system adequately reflects that portion of the real world it is intended to simulate. The BWW models consist of the representation model, the state-tracking model, and the decomposition model. The work reported in this paper uses this representation model and its constructs. The representation model defines a set of constructs that, at this time, are thought to be necessary and sufficient to describe the structure and behavior of the real world.

The BWW model is not the only ontology available to evaluate information systems since alternatives exist both in the form of general philosophical ontologies, for example, Chisholm (1996), or special enterprise and IS ontologies, for example, the enterprise ontology (Uschold et al., 1998) and the framework of information systems concepts (FRISCO) (Verrijn-Stuart et al., 2001). However, the use the BWW-model is justified for two reasons: first, the model is based on concepts that are fundamental to the computer science and information systems domains (Wand and Weber 1993). Second, it has already been used successfully to analyze and evaluate the modeling constructs of many established IS and enterprise modeling languages such as dataflow diagrams, ER models, OML and UML (Evermann and Wand 2001; Green and Rosemann 2000; Opdahl and Henderson-Sellers 2002; Weber and Zhang 1996) and for the evaluation of enterprise systems (Green et al. 2005) and business component frameworks (Fettke and Loos 2003).

For brevity, we do not introduce the BWW-model in detail. Instead, Table A.1 in the appendix summarizes its main constructs.

3. RESEARCH METHOD

For the chosen research problem, the design science approach is used to design an evaluation framework to help IS specialists in the verification of representation of the business requirements in re-engineered component-based models originally represented in legacy conceptual models.

March and Smith (1995) outline a design science framework with two axes, namely research activities and research outputs. Research outputs cover constructs, models, methods and instantiations. Research activities comprise building, evaluating, theorizing on and justifying artifacts.

Concerning research activities, March and Smith (1995) identify build and evaluate as the two main issues in design science. Build refers to the construction of constructs, models, methods and artifacts demonstrating that they can be constructed. Evaluate refers to the development of criteria and the assessment of the output's performance against those criteria. Theorize refers to the construction of theories that explain how or why something happens.

The building part of the research uses re-engineering methodologies to generate the conceptual models required for the research that will help to build the framework for re-engineering of legacy systems into component-based systems. There are many re-engineering methodologies that help to cope with the problem of transforming legacy systems originally developed with traditional methodologies into component-based systems.

The study covers the build and evaluate research activities and has a research output of constructs and models. Instantiations are not covered as the scope of this research is limited to conceptual models. Conceptual models do not include any implementation details that can be used for instantiation.

March and Smith (1995) propose a four by four framework that produces sixteen cells describing viable research efforts. The different cells have different objectives with different appropriate research methods. A research project can cover multiple cells, but does not necessarily have to cover them all.

The *build* activity of the framework will be used as part of this research since conceptual models need to be created for ontological evaluation and used to build the framework for the re-engineering of legacy systems into component-based systems. The main contribution of the research project will be the *evaluation* phase as it will allow identification of metrics to compare the performance of constructs and models.

Table 1 illustrates the cells at the intersection of research activities and research outputs of March and Smith's (1995) framework which are discussed in this paper. Each cell/intersection contains a specific research objective of the overall research. The *build* column covers the recovery of a conceptual model for a legacy system and the generation of a re-engineered component-based model used for the discovery of rules to build the objective re-engineering framework. Construct building is not required as existing constructs for both traditional and component-based are used.

The *evaluate* column in Table 1 includes evaluating the completeness of the component-based constructs (UML) in terms of ontological deficiencies that the constructs could have when modeling traditional constructs. Conceptual models need to be evaluated in order to measure the capacity of the component-based model to represent the same requirements as the legacy model.

Table 1. Research activities based on design science approach (adapted from March & Smith 1995)

	Build	**Evaluate**
Constructs	Not required	Identifying ontological modeling deficiencies of component-based constructs in terms of traditional construct representation
Model	Recover the legacy conceptual model of the case study Generate the re-engineered component-based model for the legacy system	Evaluate the capacity of the re-engineered component-based for representing the same business requirements embedded in the legacy model

As March and Smith explain, every cell and research objective may call for a different methodology. This makes it necessary to identify an adequate method for each specific research objective, resulting in an overall method mix. To achieve this, several methodologies were identified as part of the literature review. These methods are listed in Table 2.

Table 2. Methodologies selected for research project

Methodology	**Definition**
Case Study	Study of a single phenomenon (e.g., an application, a technology, a decision) in an organization over a logical time frame
Jacobson & Linstrom (1991)	Methodology for information systems re-engineering and legacy system conceptual model recovery
Fettke & Loos (2003)	Methodology for ontological evaluation of conceptual models
Interviews	Research in which information is obtained by asking respondents questions directly
Direct observation	This occurs when a field visit is conducted during the case study
Secondary Data	A study that utilizes existing organizational and business data, e.g., document, diagrams, etc.
Rosemann & Green (2002)	Meta Models methodology for Normalized Reference Models generation and comparison

4. A FRAMEWORK FOR CONCEPTUAL MODEL COMPARISON

The IS research problem chosen to demonstrate the use of design science involves three main parts: conceptual model recovery, system re-engineering, and ontological evaluation.

Methodologies selected for conceptual model recovery. The conceptual model recovery of the case study is one of the major challenges in the research since most of the legacy systems have very poor documentation in terms of models and technical design. In order to address this problem, the researcher captured the conceptual model of the legacy system by applying a reverse engineering approach as specified in the Jacobson and Lindstrom (1991) methodology. There are many re-engineering methodologies that help to cope with the problem of transforming legacy systems originally developed with traditional methodologies into component-based systems. The Jacobson and Lindstrom (1991) approach for re-engineering of legacy systems was chosen for the following reasons:

- It contemplates cases of a complete change of implementation technique and no change in the functionality, which is the case of this research;
- It does not require the use of source code. In the case study used for this research there is no access to the source code used to develop the system;
- It also covers reverse engineering. This is useful for this research given the need to capture the original conceptual model for the legacy system;
- It is relatively simple to use.

Although Jacobson and Lindstrom's original methodology was proposed for object-oriented systems, it can be easily adapted for component-based systems since components can be viewed as a higher level of abstraction based on object-oriented methodology. The methodology for this project uses data collection methods including interviews, direct observation and secondary data.

Methodologies selected for system re-engineering. Once the conceptual models from the legacy system are recovered, the system is re-engineered using the Jacobson and Lindstrom (1991) approach for re-engineering of legacy systems. The output of this step is the re-engineered component-based model as detailed in Valverde and Toleman (2007).

Methodologies selected for ontological evaluation. The legacy system and re-engineered models generated as part of the building part of the research are then evaluated based on the ontological evaluation of grammars (Wand & Weber 1993). As part of the evaluation research, an analysis is done using the Bunge-Wand-Weber (BWW) model. The BWW model is an ontological theory initially developed by Bunge (1977; 1979) and adapted and extended by Wand and Weber (Wand & Weber 1989; Wand & Weber 1995; Weber 1997). The BWW model is well founded on mathematical concepts. Prior research on the evaluation of grammars has shown it has been used successfully in information systems research (Evermann & Wand 2001; Green & Rosemann 2000; Opdahl & Henderson-Sellers 2002; Weber & Zhang 1996).

After developing the re-engineered model, it is necessary to compare both legacy and re-engineered models for equivalency of representation of business requirements. An ontological normalization methodology developed by Fettke and Loos (2003) is used for this activity. The Fettke and Loos (2003) methodology is considered appropriate as it provides a mechanism for the comparison of conceptual models; models can be compared based of their normalized referenced models; and it is simple to use.

In order to generate these normalized reference models in BWW terms, the Rosemann and Green (2002) BWW meta-model is used. This meta-model is based on the original entity relationship specification from Chen (1976) with extensions made by Scheer (1998). Scheer's version is called the extended ER model (eERM).

Once the legacy system and re-engineered models are generated, they can be evaluated based on an ontological evaluation of grammars (Wand & Weber 1993). An ontological normalization for the original and re-engineered models is generated. The two models are evaluated using the Fettke and Loos (2003) methodology based on their ontologically normalized models generated by the Rosemann and Green (2000) methodology. The result of the comparison reveals that the compared models are equivalent, complementary or in conflict (Fettke & Loos 2003). Table 3 displays the mapping of the retained methodologies to the activities.

Table 3. Research methodologies selected for the design science approach

	Build	**Evaluate**
Constructs	Not required	Fettke & Loos (2003)
Model	Case Study Interviews Secondary Data Direct Observation	Case Study Fettke & Loos (2003) Rosemann & Green (2002) Jacobson & Linstrom (1991)

In effect, the useful "components" from prior research were considered and assembled to provide a framework for conceptual model comparison.

5. BUILDING THE REQUIREMENTS MODELS

Research procedures in this study are divided into build and evaluation procedures. Both research procedures make use of the case study methodology; this methodology is chosen to evaluate the capacity of the re-engineered component model to represent the same requirements as the legacy traditional model (Benbasat, Goldstein & Mead 1987). The case-study system selected is a *Home Loan* information system developed by a consultant company in the Netherlands. The system was customized for a mid-sized home loan bank that specializes in the marketing, sales and administration of its own home loan products. The information system was designed for use on Unisys A-Series mainframes.

Build procedures are required to accomplish the build objectives of the design science approach while the evaluation procedures accomplish the evaluation objectives.

Data Collection (Build). Data gathering is an important part of this research as it is required to commence the building part of the research. For this research, observation techniques, interviews, and review of physical artifacts and system documents were used as the sources for data gathering.

In this study, the case study information system's site was visited and its functionality observed, that is a *complete observer* situation. The technique used to interview users, maintainers and designers was open-ended interviews. The final goal of the open interview is to interview system users, maintainers and designers of the legacy systems in order to find out how the system was developed, what are the functions of the system and the type of documentation used for the system development. The system owners consented to the participation of the developers in the interviews.

System documentation was collected in order to perform the reverse engineering analysis required to recover the conceptual models (Jacobson & Lindstrom 1991). The legacy information system can be described by using different elements such as requirements specifications, user operating instructions, maintenance manuals, training manuals, design documentation, source code files, and database schema descriptions (Jacobson & Lindstrom 1991). Information systems documentation is a valuable source of data. Documentation related to the system, including manuals, database schemas and system architecture diagrams was collected.

Conceptual Model Recovery (Build). In order to capture the conceptual model of the legacy system, the reverse engineering methodology, as specified in Jacobson and Lindstrom (1991) was applied. The following three steps were used:

1. Develop a concrete graph that describes the components of the system and their interrelationship;
2. Develop an abstract graph showing the behavior and the structure of the system;
3. Develop a mapping between the two, that is, how something in the abstract graph relates to the concrete graph and vice versa.

The abstract graph should be free of implementation details. For example, mechanisms for persistent storage or partitioning into processes should not appear on this graph. The concrete graph must, on the other hand, show these details. The mapping between the two should explain how the abstract graph is implemented by way of the concrete graph (Jacobson & Lindstrom 1991).

Use cases are an excellent tool for reverse engineering since they provide a sequence of user interactions with the system (Jacobson & Lindstrom 1991). In the context of reverse engineering, it is possible to explore a legacy system with use cases (Jacobson & Lindstrom 1991). Use cases were developed to create the concrete graph for reverse engineering. These use cases show the interrelationship between manuals, documentation, interviews, source code and researcher's observation of the system. The abstract graph described in the Jacobson and Lindstrom (1991) methodology is in fact an example of the legacy conceptual model. For this research project, the conceptual model was represented in terms of data flow diagrams (DFDs), a context diagram and entity relationship (E-R) diagrams.

The description of the business process, business events and responses is essential in generating a conceptual model (Whitten et al. 2001). The use cases employed to construct the concrete graph, document the business processes, events and responses required to construct this legacy abstract graph. In order to generate the DFDs required to construct the legacy conceptual model, business events to which the system must respond and appropriate responses were identified with the help of the use cases. According to Whitten et al. (2001) there are essentially three types of events:

- External events: are so named because they are initiated by external agents. When these events happen, an input data flow occurs for the system in the DFD;
- Temporal events: trigger processes on the basis of time. When these events happen, an input called *control flow* occurs;
- State events: trigger processes based on a system change from one state or condition to another. Information systems usually respond to external or temporal events. State events are usually associated with real time systems (Whitten et al. 2001).

Once these events were identified, DFDs were drawn with the help of the list of mapping transformations suggested by Whitten el al. (2001). The concrete graph represented by the use case can be mapped to the abstract graph represented by the DFD. The actor in the use case that initiated the event will become the external agent; the event identified in the use case will be handled by a process in the DFD; the input or trigger in the use case will become the data or control flow in the DFD; all outputs and responses in the use case will become data flows in the DFD.

The data model of the legacy conceptual model is generated by identifying the data stores in the DFD, examining the use cases, and finally documented by using an E-R Diagram.

Component-based Model Generation (Build). Once the model was reverse engineered from the legacy system, the legacy system was re-engineered for a complete change in implementation technique but no change in functionality by preparing an analysis model and then mapping each analysis object to the implementation of the old system (Jacobson & Lindstrom 1991).

In the first step, an analysis model was prepared with the help of the use cases prepared in the reverse engineering process. These use cases already contain the information that was assimilated from the manuals, system architecture documentation, open interviews and research observations described as *description* elements in the Jacobson and Lindstom (1991) methodology (Figure 1). Only the analysis model of the re-engineering process was required since the primary objective of the research project was the comparison of conceptual models and not the full implementation of the information systems.

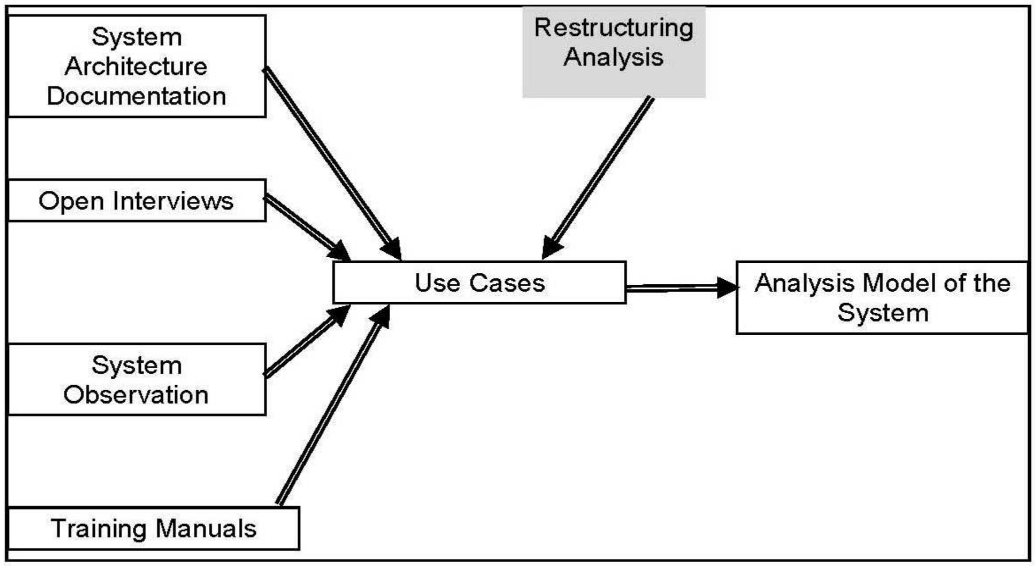

Figure 1. Preparation of the Analysis Model (adapted from Jacobson & Lindstrom 1991)

An analysis model only contains the logical aspects and is free of physical implementation details. The logical representation of a component is concerned with its logical abstraction, its relationship with other logical elements, and its assigned responsibilities. The logical representation of a component-based system was modeled by using the UML diagrams: use case diagrams; class diagrams; sequence diagram; and state diagrams (Houston & Norris 2001).

Actors were identified from the use cases and use case diagrams were constructed to identify the system scope and boundaries. The model should be free of physical implementation details. For the case of components, their logical representation was modeled using UML subsystems and identified inside the use case diagrams as proposed by Houston and Norris (2001). Class diagrams were prepared using the criteria for finding objects as described in Jacobson's (1987) object-oriented method. This step was accomplished by reviewing each use case to find nouns that correspond to business entities or events (Jacobson 1987). Not all the nouns in the use cases represent valid business objects. A cleansing process removed nouns that represent synonyms, nouns outside of the scope of the system, nouns that are roles without unique behavior or are external roles, unclear nouns that need focus or nouns that are really actions or attributes (Whitten et al. 2001). Once objects were identified, their relationships were modeled as part of the class diagrams and interfaces were identified.

Re-engineering framework Generation (Build): Once the ontological evaluation has been used to create mapping tables between UML diagrams and BWW constructs, a set of rules can be identified as part of a re-engineering framework. The next section provides the rules derived.

6. COMPARISON OF THE REQUIREMENTS MODELS

Ontological Evaluation (Evaluation). Once the legacy conceptual model was recovered and the component business analysis model represented with the use of UML diagrams, the Fettke and Loos (2003) methodology was used to evaluate these models for equivalency of representation of business requirements.

As part of this evaluation, the ontological normalization of the legacy and re-engineered component models was generated. The ontological normalization of a reference model consisted of four steps (Fettke & Loos 2003):

1. Develop a transformation mapping;
2. Identify ontological modeling deficiencies;
3. Transform the models; and
4. Assess the results.

In the first step of this method, a transformation mapping of the traditional and component-based (UML) diagrams used for representing the conceptual models was developed. This transformation mapping allowed converting the constructs of the traditional and component based (UML) diagrams to the constructs of the BWW model. The first step was based on the method for the ontological evaluation of grammars proposed by Wand and Weber (1993).

The transformation mapping consisted of two mathematical mappings. First, a representation mapping described whether and how the constructs of the BWW model are mapped onto the traditional and component-based (UML) constructs. Second, the interpretation mapping described whether and how the traditional and component based (UML) constructs are mapped onto the constructs of the BWW model (Fettke & Loos 2003). Table A.2 in the appendix shows the mapping between traditional and BWW constructs and Table A.3 the mapping between UML and BWW constructs.

All ontological deficiencies of the conceptual models were identified as part of the second step of the generation of the normalized ontological models. To identify the ontological deficiencies of the recovered model and re-engineered component-based model, all constructs of the models were reviewed. Each construct of the models analyzed was examined with respect to whether the construct was used correctly regarding the interpretation mapping.

Three classifications of deficiencies were used:

- Adequacy: the grammatical construct is ontologically adequate. Nevertheless an ontological deficiency can emerge by applying the grammatical construct to build the reference model. Therefore it must be examined whether the construct of the reference model is used correctly with respect to the interpretation mapping.
- Excess: construct excess is a modeling deficiency in general and needs special handling in the transformation step. Therefore, this construct should be marked as excessive in the reference model.
- Overload: construct overload is a modeling deficiency in general and needs special handling in the transformation step. Therefore, this construct should be marked as overloaded in the reference model (Fettke & Loos 2003).

Based on the representation mapping it was decided whether the traditional and component-based grammar are incomplete or redundant. An incomplete grammar suggests that specific facts of reality cannot be adequately represented in the model.

In the third step, the models were transformed to ontological models. The outcome of this step was two ontologically normalized models. The objective of both techniques was to represent the domain of interest in a normalized way by applying specific transformation patterns (Fettke & Loos 2003).

7. EVALUATION

Upon reflection, a response to the research question requires the answers to four research issues. The first deals with the possible conflict that might occur if one grammar construct in one diagram of the legacy requirements model can be mapped to more than one grammar construct in one diagram in the target re-engineered component-based requirements model.

The second deals with the accommodation of all legacy requirements model grammar constructs into the re-engineered component-based requirements model and the third with the possibility of the component requirements model being complementary to the legacy business model, which means that the re-engineered requirements model is able to accommodate all the grammar constructs of the legacy requirements model and complement in addition more constructs that were not able to be represented in the original requirements models. Finally, the fourth issue is to use the analysis revealed by the ontological evaluation in order to identify the rules that form part of the framework required to answer the research question for this study.

The research revealed that there was a conflict with the use of data flows as these can represent events (internal or external) and also couplings between processes to data stores, processes with processes and processes with external agents (Valverde 2008).

Although this might be seen as a potential conflict in the re-engineering process, the problem of mapping the data flow with UML triggers or UML messages can be eliminated if the interpretation is known before the legacy requirements model is re-engineered. The interpretation can be easily found by reading the use cases or business process descriptions of the legacy requirements model and a rule can be used to solve this conflict. The rule can require mapping the data flow as a UML trigger if it is interpreted as an event, and mapping it as a UML message if the data flow is interpreted as coupling (Valverde 2008).

The research also showed that the re-engineered component requirements model was capable of representing all the legacy requirements model constructs (Valverde 2008). Table 4 shows the mapping of all the legacy requirements model constructs onto the component-based requirements model as a proof of this.

Table 4. Traditional diagrams representation in UML component diagrams

Type of diagram	**Diagram element**	**UML representation**
Context Diagram	External Agents	Actor (Use case diagram)
	Data Flow	UML association (Use case diagram)
	System	System Boundary(Use case Diagram)
DFD	External agents	Actor (Use case diagram)
	Data stores	Object (Sequence diagram)
	Data flows (internal and external events) Data flows (external agent and process	Triggers (State diagram) UML message

Type of diagram	Diagram element	UML representation
	coupling) Data flows (process and data store coupling) Data flows (process and process coupling)	UML message UML message
	Process	Activities (Activity diagram) UML operations (Class diagram)
ERD	Entities	UML class (Class diagram)
	Cardinalities	UML multiplicity (Class diagram)
	Relationships	UML association (Class diagram)

In addition, the research revealed that the component-based requirements model is able to complement the legacy requirements model and therefore able to better represent requirements in ontological terms (Valverde 2008).

Based on the ontological analysis, a set of rules was identified in order to build a framework that can be used when re-engineering legacy systems in order to ensure the same representation of requirements in the re-engineered requirements models. The following rules were identified:

a) For the case of the context diagram in the legacy requirements model, this can be represented with the help of the use case diagram in the component-based model by following the rules below:
    1. For every external agent, create an actor that interacts with the system in the use case diagram.
    2. For every data flow, create a UML association that will bind actors with the system.

b) For the case of ERD in the legacy requirements model, these can be represented with the use of UML class diagrams in the component-based model by following the rules below.

    1. For every entity in the ERD of the legacy requirements model, a class should be created in the class diagram of the component-based model.
    2. Relationships in the ERD should be respected in the class diagrams and implemented with UML associations.
    3. Cardinalities in the ERD should be respected in the class diagrams and implemented with UML multiplicity constructs.

c) For the case of DFD in the legacy requirements model, these can be represented with the use of sequence diagrams, state diagrams and class diagrams by following the rules below:

1. For every external agent, create an actor in the sequence diagrams.
2. For every process, create an operation in an appropriate class of the class diagram that implements the process in the DFD.
3. For every data flow interpreted as an internal or external event, create a trigger in the state diagram of the appropriate object in charge of generating the event. If the event is external use a stereotype to indicate this in the trigger.
4. For every data flow interpreted as coupling, create a message in the sequence diagrams. Data flows used to couple external agents with processes should be represented in the sequence diagram as a message between the actor representing the external agent and the object that is in charge of implementing the process by using the operation created for this in rule 2. Data flows used to couple processes with data stores should be represented in the sequence diagram as a message between the object implementing the process and an object representing the data store. Data flows used to couple a process with another process should be implemented by a message between an object implementing the first process and another object implementing the second one. If both processes are implemented by the same object this could be represented by a message being sent from the object to itself.

The rules above provide a response to the research question. The framework identified can be used to re-engineer a legacy system into a component-based system and verifies that the resulting re-engineered component-based requirements model generated using UML grammar is able to represent the requirements encapsulated in a legacy system requirements model represented by the traditional DFD, ERD and Context diagrams models.

8. CONCLUSIONS

This study developed a framework to compare the requirements models generated by the component-based and traditional approaches in the re-engineering process. A legacy system was selected as part of the case study and re-engineered using the component-based paradigm with the help of UML notations. The study verified that the framework is effective by demonstrating its application. The re-engineered requirements model is capable of representing the same business requirements as the legacy system.

This study provides a relatively interesting example of design science being used to build a framework and it proved to be useful for the research of information systems re-engineering. The research activities that March and Smith (1995) identify for this methodology are build and evaluate and these were fundamental for this study as the first was used for construction of the re-engineering framework for the transformation of legacy systems into component-based systems, the requirements and BWW normalized models required for the evaluation and the second was used in the evaluation of these models for equivalency of business requirements.

The comparison part of the research revealed that the re-engineered requirements models in UML are capable of representing the same business requirements of the legacy system and this evaluation was used to build a set of rules that are part of the proposed re-engineering of legacy systems into component-based systems framework.

Future research can be concentrated in the development of automated tools for the re-engineering of information systems. A software tool could be constructed to build legacy and re-engineered conceptual models and evaluate them based on the methodology proposed. This software tool could translate the legacy and component models into ontological normalized reference models that could be used for comparison.

9. REFERENCES

Au, Y. A. (2001). Design Science I: The Role of Design Science in Electronic Commerce Research. Communications of the Association for Information Systems (CAIS), 7(1).

Bunge, M., (1977). Treatise on Basic Philosophy: Volume 3: Ontology 1: The furniture of the world. Reidel, Boston.

Bunge, M., (1979). Treatise on Basic Philosophy: Volume 4: Ontology II: A World of Systems, Reidel, Dordrecht.

Chen, P. P.-S., (1976). The Entity-Relationship Model: Toward a Unified View of Data. ACM Transactions on Database Systems, 1(1), 9-36.

Chisholm, R.M. (1996). A Realistic Theory of Categories: An Essay on Ontology, Cambridge University Press, UK.

Dussart A., Conseil R., Aubert B. & Patry M. (2004), An Evaluation of Inter-Organizational Workflow Modeling Formalisms, Journal of Database Management, .15(2), 74-104.

Evermann, J. & Wand, Y. (2001). An Ontological Examination of Object Interaction in Conceptual Modeling. In Proceedings of the 11th Workshop on Information Technologies and Systems, (WITS 2001). New Orleans, Louisiana.

Fettke, P. & Loos, P. (2003). Ontological evaluation of reference models using the Bunge Wand-Weber-model. In Proceedings of the Ninth Americas Conference on Information Systems. pp. 2944-2955, Tampa, FL, USA.

Green, P. & Rosemann, M. (2000). Integrated Process Modelling: an ontological evaluation. Information Systems, 25(2), 73-87.

Green, P. & Rosemann, M. (2005), Business Analysis with Ontologies, Idea Group Publishing, New York. US.

Gregor, S. (2002). Design Theory in Information Systems. *Australasian Journal of Information Systems*, 10(1), 14-22.

Hevner, A.R., March, S.T., Park, J. & Ram, S. (2004). Design Research in Information Systems Research. MIS Quarterly, 28(1), 75-105.

Houston, K. & Norris, D. (2001). ‘Software Component and the UML’ Chapter 14 in Component Based Software Engineering , Addison-Wesley, 243-263.

Jacobson, I. & Lindstrom, F. (1991). Re-engineering of Old Systems to an Object-Oriented Approach. In Proceedings of Conference on Object-Oriented Programming Systems, Languages and Applications OOPSLA 1991, 340-350.

Jacobson. I. (1987). Object Oriented Development in an Industrial Environment. In Proceedings of OOPSLA. Orlando, Florida.: ACM Press, 183-191.

March, S. & Smith, G. (1995). Design and Natural Science Research on Information Technology. Decision Support Systems, 15(4), 251 - 266.

Nunamaker, J.F. & Chen, M. 1990, Systems development in information systems research, in Proceedings of the Twenty-Third Hawaii International Conference on System Sciences, IEEE Computer Society Press, 631-639.

Opdahl, A.L. & Henderson-Sellers, B. (2002). Understanding and improving the UML metamodel through ontological analysis. Journal of Software and Systems Modelling (SoSyM), Springer, 1(1), 43–67.

Rosemann, M. & Green, P. (2002). Developing a meta model for the Bunge-Wand-Weber Ontological Constructs. Information Systems, 27, 75-91.

Scheer, A.-W. (1998). ARIS–Business Process Frameworks. 2nd edn. Springer-Verlag, Berlin.

Uschold, M., King, M., Moralee, S. & Zorgios, Y. (1998), The enterprise ontology. The Knowledge Engineering Review, 13(1), 31-89.

Valverde, R. & Toleman, M. (2007). Ontological Evaluation of Business Models: Comparing Traditional and Component-Based Paradigms in Information Systems Re-engineering. In Kishore, R., Ramesh, R & R Sharman, R. (Eds), Ontologies in the Context of Information Systems, Springer-Verlag, Berlin.

Valverde, R. (2008). The ontological evaluation of the requirements model when shifting from a traditional to a component-based paradigm in information systems re-engineering. DBA Thesis, University of Southern Queensland.

Verrijn-Stuart, A.A. (2001), A Framework of Information System Concepts, Proceedings of the IFIP TC8/WG8.1 Working Conference on Information System Concepts, Brussels, November 15-16.

Wand, Y. & Weber, R. (1988). An ontological analysis of some fundamental information systems concepts, Proceedings of the Ninth International Conference on Information Systems, Minneapolis USA, November 30–December 3.

Wand, Y. & Weber, R. (1989). An ontological evaluation of systems analysis and design methods. In Proceedings of the IFIP WG8.1 Working Conference on Information Systems Concepts: An In-Depth Analysis (Falkenberg, E. & Lindgreen, P. Eds.), Namur, Belgium, 79–107, North-Holland, Amsterdam.

Wand, Y. & Weber, R. (1993). On the ontological expressiveness of information systems analysis and design grammars. Information Systems Journal, 3(2), 217–237.

Wand, Y. & Weber, R. (1995). On the deep structure of information systems. Information Systems Journal, 5(2), 203–223.

Weber, R. (1997). Ontological Foundations of Information Systems. Coopers and Lybrand Accounting Research Methodology. Monograph No. 4. Melbourne.

Weber, R. & Zhang, Y. (1996). An analytical evaluation of NIAM's grammar for conceptual schema diagrams. Information Systems Journal, 6(2), 147-170.

Whitten, J. L., Bentley D. L. & Dittman K.V. (2001), Systems Analysis and Design Methods, McGraw-Hill, New York.

Appendix

Table A.1. Constructs of the BWW-model (source: Wand and Weber 1993; Weber and Zhang 1996)

Ontological Construct	Definition
THING	The elementary unit in our ontological model. The real world is made up of things. A composite thing may be made up of other things (composite or primitive).
PROPERTY	Things possess properties. A property is modeled via a function that maps the thing into some value. A property of a composite thing that belongs to a component thing is called a hereditary property. Otherwise it is called an emergent property. A property that is inherently a property of an individual thing is called an intrinsic property. A property that is meaningful only in the context of two or more things is called a mutual or relational property
STATE	The vector of values for all property functions of a thing
CONCEIVABLE STATE SPACE	The set of all states that the thing might ever assume.
STATE LAW	Restricts the values of the property functions of a thing to a subset that is deemed lawful because of natural laws or human laws
EVENT	A change of state of a thing. It is effected via a transformation (see below).
EVENT SPACE	The set of all possible events that con occur in the thing.
TRANSFORMATION	A mapping from a domain comprising states to a co-domain comprising states.
PROCESS	An intrinsically ordered sequence of events on, or state of, a thing.
LAWFUL TRANSFORMATION	Defines which events in a thing are lawful.
HISTORY	The chronologically ordered states that a thing traverses.
ACTS ON	A thing acts on another thing if its existence affects the history of the other thing.
COUPLING	A thing acts on another thing if its existence affects the history of the other thing. The two things are said to be coupled or interact
SYSTEM	A set of things is a system if, for any bi-partitioning of the set, couplings exist among things in the two subsets.
SYSTEM COMPOSITION	The things in the system.
SYSTEM ENVIRONMENT	Things that are not in the system but interact with things in the system.
SYSTEM STRUCTURE	The set of couplings that exist among things in the system and things in the environment of the system.
SUBSYSTEM	A system whose composition and structure are subsets of the composition and structure of another system
SYSTEM DECOMPOSITION	A set of subsystems such that every component in the System is either one of the subsystems in the decomposition or is included in the composition of one of the subsystems in the decomposition
LEVEL STRUCTURE	Defines a partial order over the subsystems in a decomposition to show which subsystems are components of other subsystems or the system itself
STABLE STATE	A state in which a thing, subsystem or system will remain unless forced to change by virtue of the action of a thing in the environment (an external event)
UNSTABLE STATE	A state that will be changed into another state by virtue of the action of transformation in the system.
EXTERNAL EVENT	An event that arises in a thing, subsystem or system by virtue of the action of something in the environment on the thing, subsystem or system. The before-state of an external event is always stable. The after-state may be stable or unstable.
INTERNAL EVENT	An event that arises in a thing, subsystem, or system by virtue of lawful transformations in the thing, subsystem, or system. The before-state of an internal event is always unstable. The after state may be stable or unstable.
WELL DEFINED EVENT	An event in which the subsequent state can always be predicted given the prior state is known
POORLY DEFINED EVENT	An event in which the subsequent state cannot be predicted given the prior state is known.
CLASS	A set of things that possess a common property.
KIND	A set of things that possess two or more common properties.

Table A.2. Mapping between traditional and BWW constructs (Source: Valverde and Toleman 2007 p. 65)

BWW construct	**Context Diagram**	**DFD**	**ERD**
Thing	External agents External data stores	External Agents External Data Store Data Stores	
Property: In particular In general Intrinsic Mutual Emergent Hereditary Attributes			Attribute type
Class			Entity type
Kind			Specialization/ generalization (IS-A)
Conceivable state space			
State law			Specialization/ generalization descriptors; [Min., max.] cardinalities
Lawful state space			
Event		Data flow	
Process		DFD	
Conceivable event space			
Transformation		Process	
Lawful transformation			
Lawful event space			
History			
Acts on			
Coupling: Binding mutual property	Ext. Agent->Data Flow-> System System->Data Flow-> External Data store	Process->Data Flow->Ext. Agents Ext. Agent->Data Flow-> Process Process->Data Flow-> Data store Data stores ->Data Flow-> Process	Relationship type (no symbol for relationship in grammar)
System	System	DFD	
System Composition		External agents and data stores in a DFD	
System Environment	External Agent External data stores	External Agent External Data Stores	
System structure		DFD	
Sub-system		DFD	
System decomposition		DFDs and sub diagrams	
Level structure		Series of processes decomposed at different levels	
External event		Data flow	
Stable state			
Unstable state			
Internal event		Data flow	
Well-defined event			
Poorly defined event			

TableA.3. Mapping between UML diagrams and BBW constructs (Source: Dussart et al. 2004 p.85)

BWW construct	**Use Case**	**Sequence**	**Class**	**State**	**Activity**
THING	Actor Use Case	Object		Object	Object Swimlane Actor
PROPERTY: IN PARTICULAR IN GENERAL INTRINSIC MUTUAL EMERGENT HEREDITARY ATTRIBUTES			UML attribute		Activity Swimlane
CLASS			Class		
KIND	Use Case		Generalization UML aggregate class UML composite class		
STATE				State	
CONCEIVABLE STATE SPACE				State machine	
STATE LAW			UML-multiplicity	State>Transition>State	
LAWFUL STATE SPACE				Sub states	
EVENT				Trigger	Activity
PROCESS	Use Case				Activity diagram Activity
CONCEIVABLE EVENT SPACE				All triggers	
TRANSFORMATION			UML operation		Activity
LAWFUL TRANSFORMATION					Guard conditions On transitions
LAWFUL EVENT SPACE					
HISTORY				Shallow history state construct	
ACTS ON					
COUPLING: BINDING MUTUAL PROPERTY	UML association UML extend UML include	Messages	UML association UML interface		.
SYSTEM	System Boundary	Sequence Diagram	Package with <<system>>		
SYSTEM COMPOSITION	System Boundary Sub-system Boundary	Object			
SYSTEM ENVIRONMENT	Actor	<<Stereotype>>			Actor
SYSTEM STRUCTURE		Messages			
SUBSYSTEM			Package with <<subsystem>>		

BWW construct	**Use Case**	**Sequence**	**Class**	**State**	**Activity**
SYSTEM DECOMPOSITION			Composition		
LEVEL STRUCTURE			Generalization		
EXTERNAL EVENT				<<Stereotype>	
STABLE STATE				Final State	
UNSTABLE STATE				Initial State	
INTERNAL EVENT				<<Stereoype>>	
WELL-DEFINED EVENT				Trigger	
POORLY DEFINED EVENT					