ONTOLOGICAL EVALUATION OF BUSINESS MODELS: COMPARING TRADITIONAL AND COMPONENT-BASED PARADIGMS IN INFORMATION SYSTEMS RE-ENGINEERING

Raul Valverde [1] and Mark Toleman [2]
[1] Concordia University University 1455 de Maisonneuve Blvd. W. H3G 1M8 Montreal, QC Canada [2] University of Southern Queensland West Street Towoomba Qld 4350 Australia

Abstract: The majority of current information systems were implemented using traditional paradigms which include business modeling techniques applied during the analysis phase such as Data Flow Diagrams and Entity-Relationship Diagrams. These legacy systems are now struggling to cope with recent developments, particularly trends towards e-Commerce applications, platform independence, reusability of pre-built components, capacity for reconfiguration and higher reliability. Many organizations now realize they need to re-engineer their systems using new component-based systems. Although the traditional and component-based approaches have different grammars for representing business models, these business models can be compared, based on their ontological grammars. This paper illustrates how an ontological evaluation of business models by using the Bunge-Wand-Weber model can be used to compare them for equivalency of representation of business requirements, when re-engineering legacy systems into component-based information systems.

Key words: Re-engineering; component-based systems; structured systems; business models, ontological evaluation; legacy systems, bunge-wanda-weber model, BWW.

1. INTRODUCTION

The vast majority of Information Systems were implemented in the early days of computing using traditional software development paradigms with procedural computer languages such as Cobol (Longworth, 2003). The traditional paradigm consists of modeling techniques such as Data Flow Diagrams (DFD) and Entity Relationship (ER) Diagrams used by system analysts during the design phase to capture the activities within a system. These particular models have been used since the early times of computers and were considered, for the most part, the documentation of legacy systems (Longworth, 2003).

However, with recent developments, particularly the trends towards e-Commerce applications and platform independence, many companies are realizing that they have to migrate their systems to new improved systems in order to meet these trends since legacy systems are not capable of coping with these new challenges. The migration of a *legacy* system to a new target system is a process of *re-engineering* that requires the system’s examination and alteration to reconstitute it in a new form (Chikofsky and Cross, 1990). This new form may result in the need for a shift in the information systems paradigm serving the architecture and the business domain. Modern computer languages such as Java, offer many advantages in such a re-engineering process; in particular, good web-application development capabilities, platform independence for applications and security.

Although object technology has become the vogue for re-engineering information systems, many projects regarded as being object-oriented have failed in recent years due to organizational and technical troubles. Wolfgang (1997) mentions some the problems associated with the object-oriented paradigm as: classes/objects implemented in one programming language cannot interoperate with those implemented in other languages, some object-oriented languages require the same compiler version, composition of objects is typically done at the language level, and composition support is missing, that is, visual/interactive tools that allow the plugging together of objects.

On the other hand, the *Component-Based (CB) paradigm* is now heralded as the next wave to fulfill the technical troubles that object technology could not deliver (Wolfgang 1997). In addition, the component-based paradigm allows a fast delivery of information systems due to its capacity of reconfiguring and reassembling pre-built business components, easy maintainability and higher quality due to the reusability of pre-tested components (Szyperski, 1999). However, regardless of the systems paradigm used in the development of information systems, business models need to be created in order to describe the requirements (Jacobson Christerson and Jonsson, 1993) collected by the system analyst.

These business models can be used as the blueprint for information systems development, however, they can become quite different depending on whether the project team uses the traditional or the component-based paradigm. The former maintains a process-oriented view of systems, providing a decomposition based on processes (namely, data flow diagrams), whereas the component-based paradigm constructs a system around a set of interacting components (Satzinger, Jackson and Burd 2002).

Within the context of an information systems paradigm shift, the continuity, robustness and integrity of the business processes and functions of the system are of prime concern when re-engineering legacy systems. This means that the business model of the re-engineered information system should represent the same business requirements as the ones from the original legacy system in order to preserve this integrity.

Although the traditional and component-based approaches have different grammars for representing business models, these business models can be compared for equivalency of representation of business requirements by using the Bunge-Wand-Weber model (Wand and Weber 1993). An ontological evaluation of business models would reveal the limitations of representing the legacy system business requirements in the component-based re-engineered model.

The paper is structured as follows: The problem definition and scope is described in the next section. In section three, the Bunge-Wand-Weber (BWW) model (Wand and Weber 1988, 1993, 1995) is discussed and justified for this research. In the next section the research approach is described and then the case study software system is explained. The following section describes the ontological evaluation, finally preliminary results are shown and conclusions are drawn.

2. PROBLEM DEFINITION AND SCOPE

During the life cycle of the information system, changes can occur that require a change of its scope. One of these changes is the availability of better technology (Whitten, Bentley and Dittman, 2000). A decision analysis would need to be performed in order to assess if the new technology would be feasible in the system (Whitten, Bentley and Dittman 2000). If the analysis reveals that the implementation is not feasible, the information systems will require re-engineering in order to adapt to the new technology requirement.

This re-engineering of a legacy information system would require a paradigm shift. However, there is a high degree of interest and concern in establishing whether or not a full migration to a more portable and scaleable component-based architecture will be able to represent the legacy business requirements in the underlying business model of the re-engineered information systems.

The aim of research therefore becomes an evaluation of the business models of the traditional and component based information systems in the re-engineering process in order to verify that both models are equivalent and represent the same business requirements.

The main purpose of research would be to investigate the following research question:

Is the resulting component based business model equivalent to the legacy business model when shifting paradigms in the re-engineering process?

A substantial information systems evolution can be a major concern of any company considering a paradigm shift since this represents the ability of the new information system to accommodate the company's essential business processes.

The study concentrates on re-engineering projects that do not include new requirements. The main reason for this is to simplify the comparison of business models that should be equivalent in this particular case.

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).

In the next section, the BWW model will be discussed as tool for business model evaluation in order to detect the limitations of representing the legacy system business requirements in the component-based re-engineered model.

3. BUNGE-WAND-WEBER MODEL

The BWW (Bunge-Wand-Weber) model's (Wand & Weber, 1988, 1993, 1995) fundamental premise 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 chapter 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, e.g., Chisholm (1996), or special enterprise and IS ontologies, e.g., 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 (Wanda 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 2003b).

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

Table ***Error! No text of specified style in document.****-1.* Constructs of the BWW-model (source: (Wand and Weber 1993; Weber and Zhang 1996 1996)

Ontological Construct	Ontological Construct
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

Ontological Construct	Ontological Construct
	codomain 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 some thing 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.

Ontological Construct	Ontological Construct
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.

4. RESEARCH METHODOLOGY

In order to address the research question, the case study methodology is chosen to emphasize and explore factors, which may lead to directions for the question (Benbasat, Goldstein and Mead, 1987).

There are many reengineering methodologies that help to cope with the problem of transforming legacy systems originally developed with structural methodologies into component-based systems. However, the Jacobson & Lindstrom (1991) approach for reengineering 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 business model for the legacy system.
- It is relatively simple to use.

Although the 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 that is based on object oriented methodology.

In order to capture the business model of the legacy system, the researcher will apply reverse engineering as specified in the Jacobson and Lindstrom (1991) methodology. To do this, the following steps need to be used:

1. A concrete graph that describes the components of the system and their interrelationship.
2. An abstract graph showing the behavior and the structure of the system.

3. A mapping between the two, i.e. 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 tell how the ideal world of analysis is implemented by way of the concrete graph (Jacobson & Lindstrom 1991).

This abstract graph is in fact the business model. The business model will be represented in terms of Data Flow diagrams, Context Model, Functional Decomposition Diagram and Entity Relationship Diagrams. Once the business model is reverse engineered from the legacy system, the legacy system will be re-engineered by using the following steps (Jacobson & Lindstrom 1991):

1. Prepare an analysis model.
2. Map each analysis object to the implementation of the old system.

In order to prepare the analysis model step, it is important to assimilate the existing information about the system. The existing information has many different forms, e.g. requirements specifications, user operating instructions, maintenance manuals, training manuals, design documentation, source code files, and database schema descriptions. These are called description elements (Jacobson & Lindstrom 1991).

From the set of description elements, an analysis model can be prepared. This is done by using the criteria for finding objects that are described in the object-oriented methodology of Jacobson et.al. (1993). After the analysis model is completed, a map of each analysis object to the implementation of the old system is required. The map must show that all analysis objects and dependencies must be motivated by at least one primitive description element. We can express that with *is-motivated-by*, a mapping from the analysis model to the set of primitive description elements. All the dependencies in the analysis model must be motivated by at least one primitive description element.

A methodology by Fettke & Loos (2003a) is used to evaluate the business models generated by the reverse engineering (legacy business model) and the one generated by the reengineering process (component model), for the following reasons:

- It provides a mechanism for evaluation of business models
- Business models can be compared based of their normalized referenced models
- Its simplicity
- It is based on the BWW model

The ontological normalization of a reference model consists of four steps (Fettke and Loos 2003a):

1. Developing a transformation mapping,
2. Identifying ontological modeling deficiencies,
3. Transforming the reference model, and

4. Assessing the results.

In the first step of this method, it is necessary to develop a transformation mapping for the grammar used for representing the business model. This transformation mapping allows converting the constructs of the used grammar to the constructs of the BWW-model. The first step is based on the method for the ontological evaluation of grammars proposed by Wand and Weber (1993).

The transformation mapping consists of two mathematical mappings: First, a representation mapping describes whether and how the constructs of the BWW-model are mapped onto the grammatical constructs. Second, the interpretation mapping describes whether and how the grammatical constructs are mapped onto the constructs of the BWW-model (Fettke and Loos 2003a).

In the third step, the reference model will be transformed to an ontological model. The outcome of this step is an ontologically normalized reference model. The objective of both techniques is to represent the domain of interest in a normalized way by applying specific transformation patterns (Fettke and Loos, 2003a). The two models will be compared based on their ontologically normalized models. The result of a comparison will be that the compared models are equivalent, complementary or in conflict.

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

The methodology used for the study is summarized in figure 1.

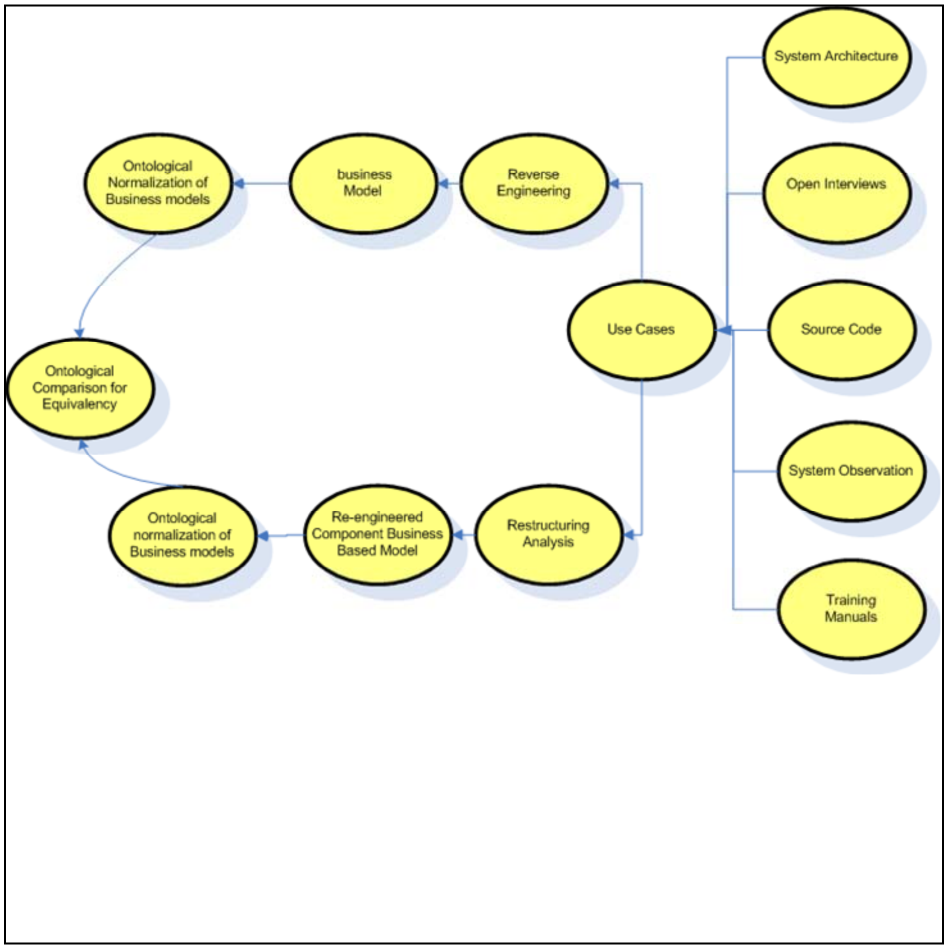
System Architecture
Ontological Normalization of Business models
business Model
Reverse Engineering
Open Interviews
Use Cases
Source Code
Ontological Comparison for Equivalency
System Observation
Ontological normalization of Business models
Re-engineered Component Business Based Model
Restructuring Analysis
Training Manuals

Figure ***Error! No text of specified style in document.****-1.* Methodology used for study

5. CASE STUDY

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.

Due to the large scale and complexity of the system, the study is focused on one sub-system that is representative of the main types of business processes. This includes an on-line user interactive component, a procedural business flow component and a batch-processing component. The sub-system for this research is the Offer and Application system.

The technique used to recover the original business model from the legacy system is the one proposed by Jacobson and Lindstrom (1991). For this case study, the following elements were collected to describe the information system:

1. Architecture diagram

2. Database files
3. Manuals
4. Interviews with users and developers

The description of the business process, business events and responses are essential in recovering the business model (Whitten et. al 2000). One of the most popular and successful approaches for documenting business processes, events and responses is a technique called *use cases* developed by Dr. Ivar Jacobson (Jacobson et al. 1993). Use cases describe the business process, which documents how the business works and what the business goals are of each interaction with the system. Use cases are not just useful to document business processes, they are also used to generate the target component based business model.

The following use-cases were identified to describe the Offer and Application Sub System:

- Process Application
- Process Offer Regional office
- Process Offer Head office
- Maintain Offer
- Loan Generation

In order to generate the DFD diagrams required to construct the legacy business model, business events to which the system must respond and appropriate responses were identified with the help of the use cases. For example, in the case of the process application use case, the applicant (actor) responds to the event “Completes loan application” that was triggered by a “new application” with “storing the application in a file cabinet” response.

Once these events were identified, Data Flow Diagrams were drawn with the help of the list of transformations suggested by Whitten el. Al (2001). The list of recommendations is:

- The actor that initiated the event will become the external agent.
- The event will be handled by a process.
- The input or trigger will become the data or control flow
- All outputs and responses will become data flows

Figure 2 depicts the context diagram for the events of table 1. The data model was generated by examining the database files and by identifying the data stores in the DFD diagram. An ERD diagram for the process application is shown in figure 3

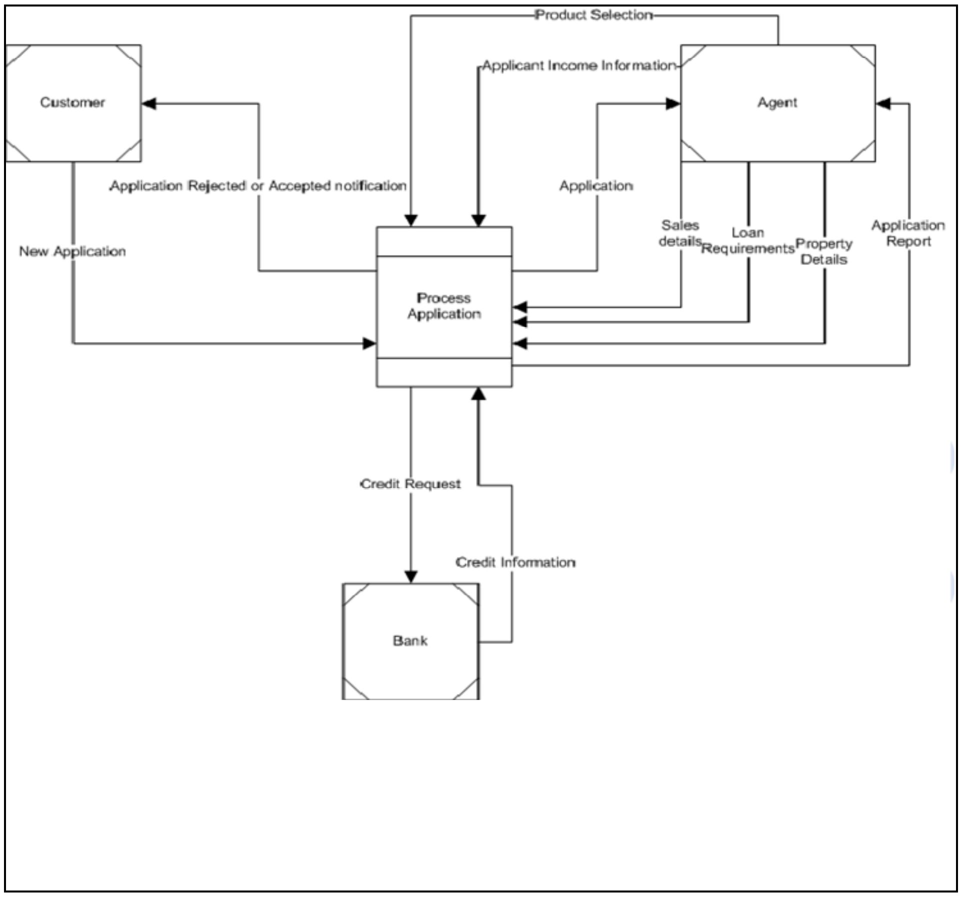

Figure ***Error! No text of specified style in document.****-2.* Context diagram for the Process Application use case

After the business model is recovered, the system is re-engineered and component based models are generated with the help of the use cases. UML was chosen to model the target system as it is the most accepted standard modeling language based on current best practice (Reed 2002). The type of UML diagrams used for this purpose were:

1. Use case diagrams
2. Sequence diagrams
3. State diagrams
4. Activity Diagrams
5. Class diagrams

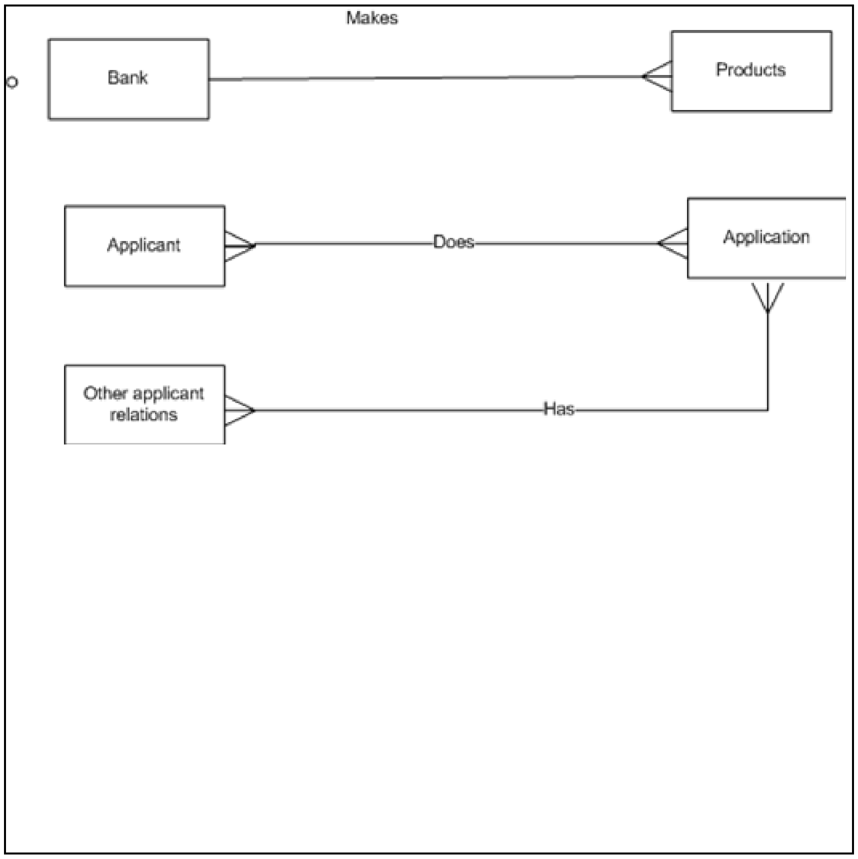

Figure ***Error! No text of specified style in document.****-3.* ERD for the Process Application Use Case

Figure 4 shows the use case diagram and figure 5 the class diagram for process application of the re-engineered system. One of the main differences of class diagrams for component-based systems is the inclusion of interface classes. These classes are used by components in order to interact with each other.

Once the legacy business model is recovered and the re-engineered business model generated, an ontological evaluation of business models by using the BWW model will be applied in order to verify that the requirements captured in the legacy business model are also reflected in the re-engineered business model.

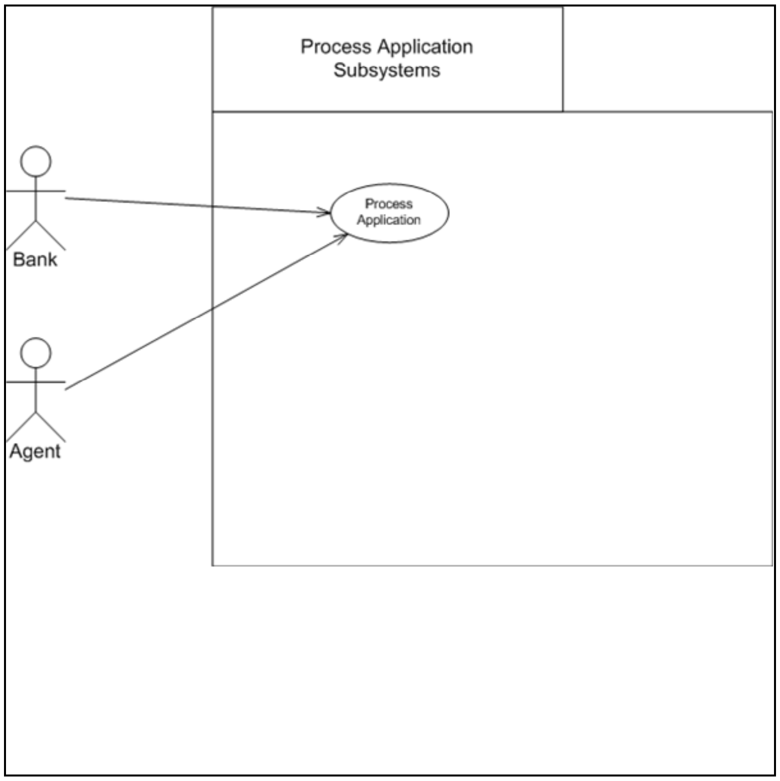

Figure ***Error! No text of specified style in document.****-4.* Use case diagram for the process application use case

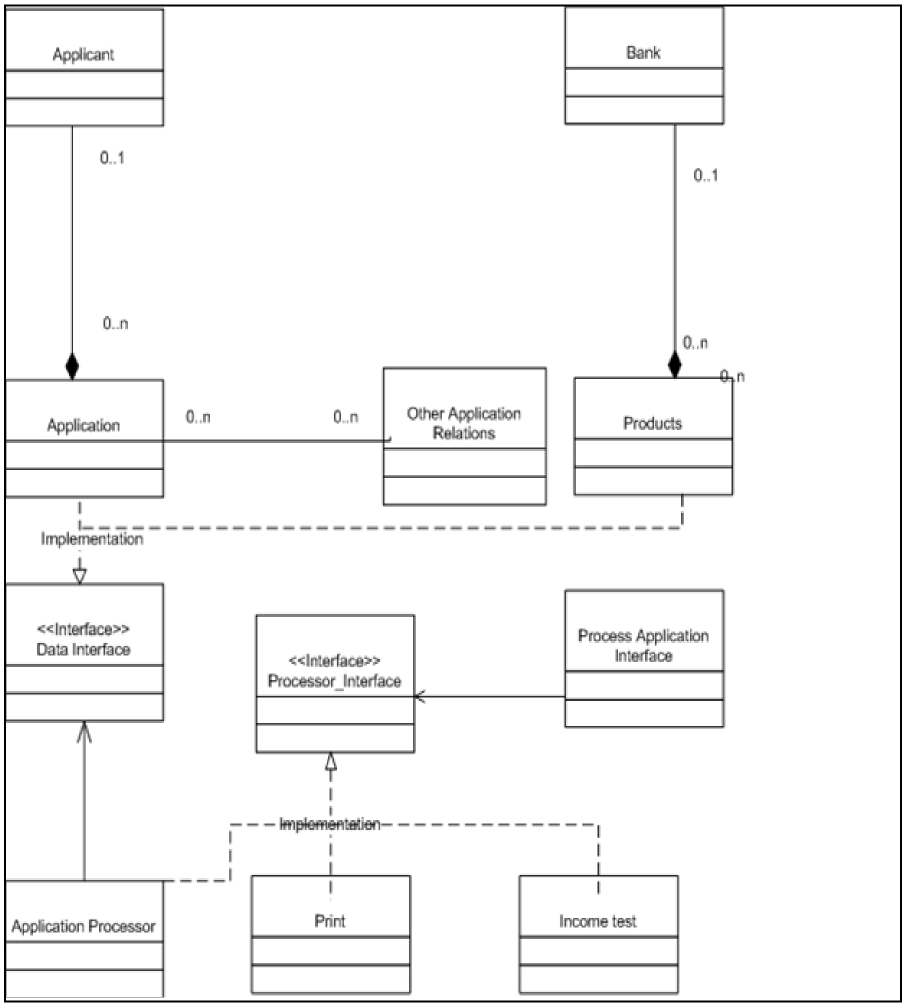

Figure ***Error! No text of specified style in document.****-5.* Diagram for the process application use case

6. ONTOLOGICAL EVALUATION

In this section of the study, diagrams are mapped into BWW constructs and then normalized reference models generated as part of the ontological evaluation of business models. There are four types of diagrams that depict legacy systems by using the Yourdon (1989) structured analysis:

1. Context diagram
2. Functional decomposition diagram
3. Data flow diagram
4. Entity relationship diagram

There are two types of modeling in DFDs: 1) Physical DFDs- where the diagram describes the physical components of the information system and 2) Logical DFDs –that describe the meaning or the 'what' of the components of the information systems (Wand & Weber 1989).

Since this research deals only with the business models, only logical DFDs will be analyzed. The logical DFD to BBW construct mapping is based on the work of Wand & Weber (1989). In logical DFDs, data store represent state information, data flows represent external and internal events. Properties of real things may be represented by data elements described in data dictionaries but not in data flows and data stores.

There is no explicit representation of the states of the real system in a DFD. Rather, the possible and allowed states of the information systems are defined implicitly in terms of possible and allowed values of the data elements described in the data dictionary and therefore not represented in the DFD diagrams (Wand & Weber 1989).

Events of the information system are represented by data flows. External events are represented by data flows coming from a source while internal events are represented by internal data flows that are generated because the system responds to an external event. Data linked to a process, a process linked to another process and a process linked to an external agent may be interpreted as coupling (Wand & Weber 1989).

A DFD diagram represents a proper system if and only if there is a path between every two processes. If this is not the case, then the DFD represents two or more disconnected information systems. External agents and data stores are represented by things and they form part of the environment in the BWW model (Wand & Weber 1989).

In DFDs, decomposition involves breaking a process "bubble" into a number of sub-processes. DFDs conform to the BWW model notion of a good decomposition (Wand & Weber 1989). In their analysis, Wand & Weber (1989) did not include the transformation construct mapping as this was added in the BWW model after the publication of this analysis. In this paper, transformations were interpreted as processes because they represent a procedure by which data inputs are transformed into data outputs (Satzinger et. al 2002 pp 196).

Entity-relationship diagrams (ERD) contain entities and relationships. Although Wand & Weber's (1989) interpretation that both can be viewed as representing things of a real system, the interpretation of Green & Rosemann (2000) of the entity being representing a class was used as entities can represent multiple instances of things . Properties are represented directly in the entity relationship diagram via de notion of attributes. Coupling between things can be represented by the lines between the lines and relationships (Wand & Weber 1989). The interpretation of the functional decomposition diagram mapping was taken from Green & Rosemann's (2000) work.

Limitations of process modeling with traditional models are acknowledged by Rosemann et. al (2005) and Green & Rosemann (2000). Functional decomposition diagrams are ontologically redundant when compared to the combination of DFDs, ERDs, and context diagrams.

No representations exist for conceivable state, state space, lawful state space, conceivable event space, lawful transformation or lawful event space. Accordingly, problems may be encountered in capturing all the potentially important business rules of the situation.

No representations exist for *stable* state, unstable state, well defined event and poorly-defined event. Again, the usefulness of traditional diagrams for defining the scope and boundaries of the system being analyzed is undermined.

Component based models were generated by using UML diagrams and mapped into BBW constructs in table 3.

UML actors in use case diagrams can be interpreted as BWW things since they act on the proposed system. UML-extend and UML-include can be mapped as a BWW-binding mutual property and UML-use cases as a BWW processes. The UML-system and UML-sub-system boundaries can be mapped as a UML system composition. UML actors can be also considered BBW environment since they are external entities that interact with the system (Opdahl and Henderson-Sellers 2004).

Irwin and Turk (2005) propose that the use case construct represents a BBW thing and a process at the same time and therefore ontologically overloaded. The system construct in a use case diagram represents a BWW composite thing (Irwin and Turk 2005). The UML actor is also overloaded as it also corresponds to a kind in the BWW ontology (Irwin and Turk 2005).

Table ***Error! No text of specified style in document.****-2.* Mapping between traditional and BWW constructs

BWW construct	Context Diagram	DFD	Functional	ERD
THING	External agents External data stores	External Agents External Data Store		
PROPERTY: IN PARTICULAR IN GENERAL INTRINSIC MUTUAL EMERGENT HEREDITARY ATTRIBUTES			Function	Attribute type
CLASS				Entity type
KIND			Specialization/ generalization (IS-A)	Specialization/ generalization (IS-A)
CONCEIVABLE STATE SPACE				
STATE LAW			Specialization/ generalization descriptors	Specialization/ generalization descriptors; [Min., max.] cardinalities
LAWFUL STATE SPACE				
EVENT		Data flow		
PROCESS		DFD diagram Process	Function type Process oriented function	

BWW construct	Context Diagram	DFD	Functional decomposition	ERD
CONCEIVABLE EVENT SPACE				
TRANSFORMATION		Process	Function Type	
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->Process Ext. Agent->Data Flow-> Process Process->Data Flow-> Data store		Relationship type (no symbol for relationship in grammar)
SYSTEM	Context Diagram	DFD diagram		
SYSTEM COMPOSITION	External agents and external data stores in a context diagram	External agents and data stores in a DFD diagram		
SYSTEM ENVIRONMENT	External Agent External data stores	External Agent External Data Stores		
SYSTEM STRUCTURE		DFD diagram		
SUBSYSTEM		DFD diagram		
SYSTEM DECOMPOSITION		DFD Diagrams and sub diagrams		
LEVEL STRUCTURE		Series of processes decomposed at different levels	Series of function type decomposition indicators	
EXTERNAL EVENT		Data flow		
STABLE STATE				
UNSTABLE STATE				
INTERNAL EVENT		Data flow		
WELL-DEFINED EVENT				
POORLY DEFINED EVENT				

The UML-system is consistent with the BWW definition, and thus there is technically no ontological discrepancy with BWW system construct (Irwin and Turk 2005). In a use case diagram, an association represents a specific type of relationship; namely that an actor initiates a use case, or that an actor interacts with the system to accomplish the goal of the use case. Association in use case diagrams corresponds to a BWW binding mutual property (Irwin and Turk 2005).

The interpretation for the mapping of BWW constructs for the activity, state, class and sequence diagrams comes from the work of Dussart et. al. (2004). The BWW ontological construct "Thing" can be associated with the object in the sequence, activity and state diagrams. The activity chart can show the transformations made on objects during activities and therefore interpreted as BWW transformation and property constructs (Dussart et. al 2004).

Class diagrams can contain symbols for classes, associations, attributes, operations, generalizations. Class and Kind are respectively represented in the UML in the class diagram with the class and the generalization constructs (Deursan et. al 2004). UML operations can be depicted by BWW transformations and UML-attribute a BWW-characteristic intrinsic property (Opdahl, A.L. and Henderson-Sellers 2004).

Class diagrams can also show the subsystem architecture, where the primary elements are UML system and subsystems. Subsystems represent components during development (Opdahl and Henderson-Sellers 2004). Subsystems and systems can be represented using a stereotyped package (Dussart et. al. 2004).

Relations between classes are depicted by UML associations, these can be represented by using the BWW mutual binding property construct and UML-multiplicity represented by state law. (Opdahl & Henderson-Sellers 2004)

Table ***Error! No text of specified style in document.****-3.* Mapping between UML diagrams and BBW constructs]

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		

BWW construct	Use Case	Sequence	Class	State	Activity
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	UML association	Messages	UML association		·
MUTUAL PROPERTY	UML extend UML include		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>>		
SYSTEM DECOMPOSITION			Composition		
LEVEL STRUCTURE			Generalization		
EXTERNAL EVENT				<<Stereotype>	
STABLE STATE				Final State	
UNSTABLE STATE				Initial State	

BWW construct	Use Case	Sequence	Class	State	Activity
INTERNAL EVENT				<<Stereoype>>	
WELL-DEFINED EVENT				Trigger	
POORLY DEFINED EVENT					

States of the thing are represented by the state of the object in the activity diagram or by the state construct in the state diagram. A state machine in the state diagram represents the conceivable State Space, defined as all the states that a thing may ever assume. A Lawful State Space can be represented in a state diagram using substates. Stable States and Unstable States can respectively be represented by the final state or the initial state in a state diagram (Dussart et. al. 2004).

Events are represented as the trigger for a transition in the state diagram. But events can also be represented as an activity in the activity diagram. There is no grammatical differentiation for External and Internal events but the use of the Uses Cases for human-machine interaction diagram or the use of stereotypes could help make the differentiation possible. The Conceivable Event Space can be observed on the state machine of a thing by looking at all transitions triggers. There exists no construct for a poorly defined event, and well-defined events use the same grammatical construct as a normal event (Dussart et. al. 2004).

Lawful transformations are represented by guard conditions on transitions. There is no grammatical construct for Lawful event space. History can be modeled using the shallow history state construct in the state diagram. Acts on cannot be represented in the same way as it is defined in the definitions of the ontological constructs but could eventually be associated to the composition relationship in the class diagram, for example, in a composition relation between a thing “Activity” and a thing “Project” (Dussart et. al. 2004)

A system can be represented using the sequence diagrams, the System composition is represented using the object construct and the System environment, that is to say external and internal things to the system, cannot be differentiated without a stereotype. The System structure BWW construct is represented using the UML-message in the sequence diagram and the decomposition by UML-composition (Dussart et. al. 2004).

For an analysis of ontological completeness, several constructs cannot find representation in any diagrams: lawful event space, acts on, poorly defined event. A construct overload is found for the activity construct in the activity diagram that can represent a transformation, a process, a property in general or an event. Construct overload was also observed for the swimlane of the activity diagram that can represent either a thing (such as an organization) or a hereditary property of the thing (a user of the organization). Finally, overload was also identified for the trigger construct (that can represent either an event or a well-defined event). There is construct redundancy in the case of the process ontological construct that can be either represented by a complete activity diagram or by the activity construct in an activity diagram. In the case of the activity diagram, construct excess can also be identified since the branching construct could not find any matching ontological construct. Overlaps occur in the activity diagram and the state diagram (Dussart et. al. 2004).

As for the transformation of legacy business models using traditional diagrams into UML models, the ontological analysis reveals that all the BWW constructs represented in the traditional models can be represented in the UML models. Context diagrams can be depicted by use case diagrams as these contain all the BWW constructs required for equivalent representation. ERD diagrams are represented by property, class, kind, state law and coupling constructs. The class diagram is able to represent the same constructs therefore able to represent the same requirements. DFD diagrams are able to represent thing, property, transformation, process, coupling, system composition, system environment, system decomposition and level structure constructs. These could be represented with the help of activity, class and use case diagrams. Finally, functional decomposition diagrams can be depicted with the use of the same diagrams.

The use of state and sequence diagrams is redundant in the representation of structured diagrams. The main reason is that structured traditional diagrams are not able to represent states and the overlap of sequence with use case diagrams.

Although the ontological mappings provide evidence that component-based models derived from the traditional models of a legacy system are able to represent the same BWW constructs, a normalized reference model comparison can be used as a tool to verify that the same requirements captured in the legacy system traditional business models are represented in the component-based models. Figures 6, 7, 8 and 9 show the BWW normalized reference models for the ERD, class, context and use case diagrams of the legacy and reengineered systems. The reference models were created by using the Rosemann and Green (2002) meta models for the representation of BBW constructs.

By comparing the normalized reference models from the ERD and class diagram (figures 6 and 7), it is possible to verify that the same classes represented in the ERD diagram are represented in the class diagram. The same relationships are present in both reference models. On the other hand, the class diagram reference model is able to complement the ERD by including classes for things that are also part of the original system but were not able to be represented in the traditional diagrams and by including classes that will help to implement the interfaces needed for component interaction.

The comparison of normalized reference models for the context and use case diagrams (figures 8 and 9) reveals that the applicant (BWW thing) originally represented in the legacy system is not included in the use case diagram of the re-engineered system. Although this actor is included in the original use case, process modeling includes manual functions while use cases diagram boundaries only include automated systems. Since the applicant fills an application manually, this component is left out of the use case diagram but it is part of the context diagram. The representation of the applicant is therefore no longer part of the re-engineered system.

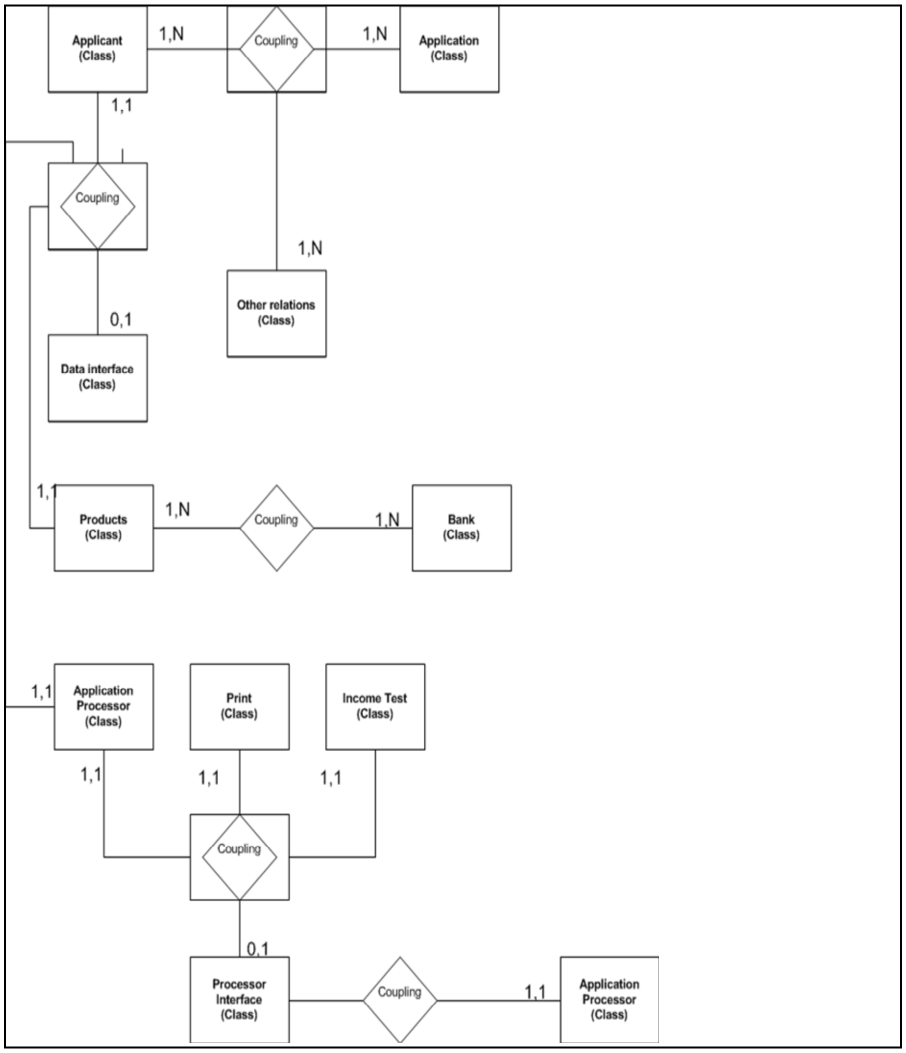

Figure ***Error! No text of specified style in document.****-6.* Normalized referenced model for the class diagram of the case study

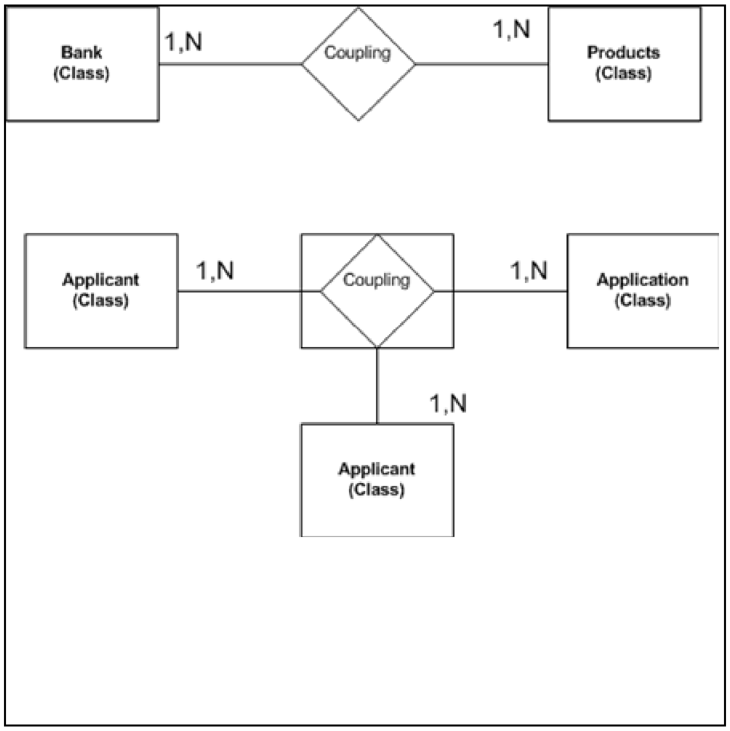

Figure ***Error! No text of specified style in document.****-7.* Normalized referenced model for the ERD diagram of the case study

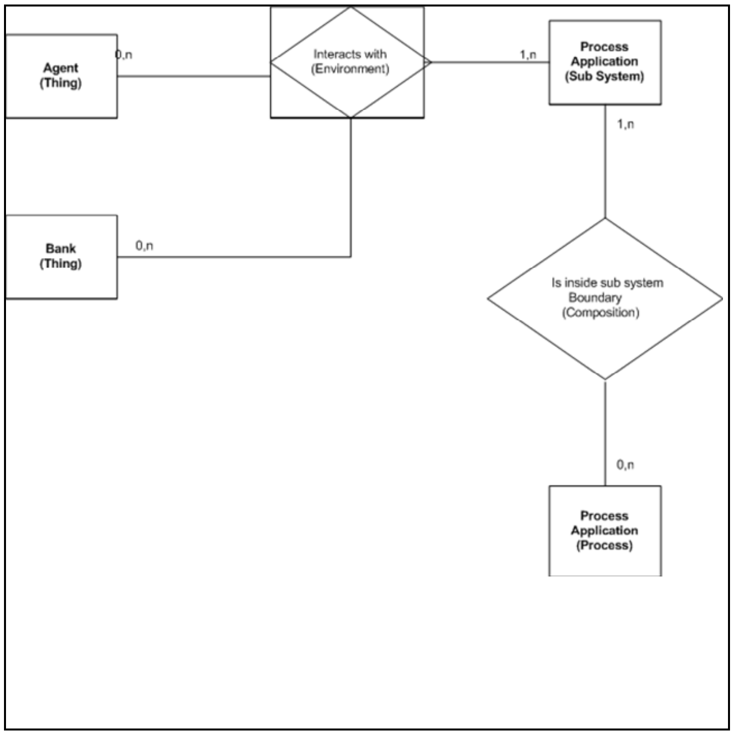
Agent (Thing)
0,n
Interacts with (Environment)
1,n
Process Application (Sub System)
1,n
Bank (Thing)
0,n
Is inside sub system Boundary (Composition)
0,n
Process Application (Process)

Figure ***Error! No text of specified style in document.****-8.* Normalized referenced model for the use case diagram of the case study

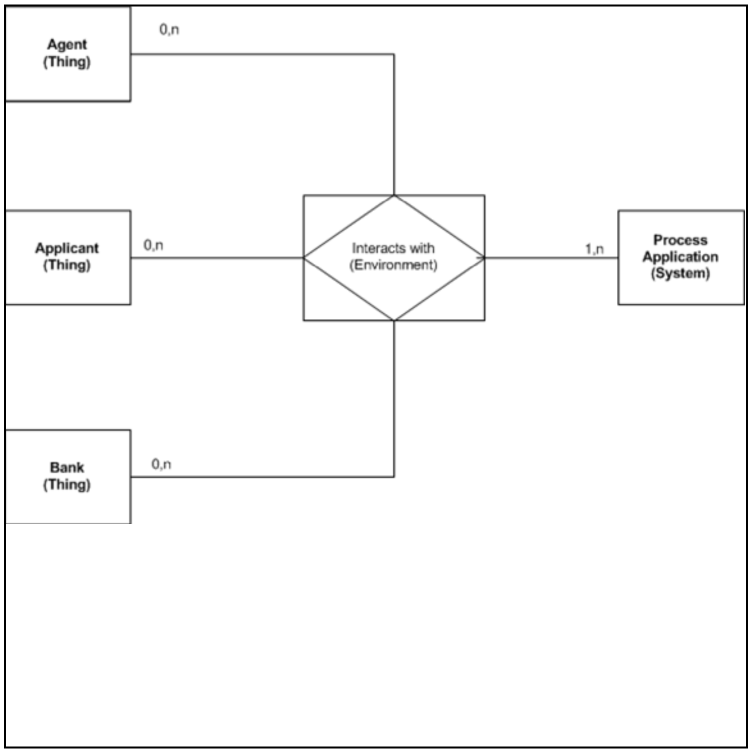

Figure ***Error! No text of specified style in document.****-9.* Normalized referenced model for the context diagram of the case study

7. CONCLUSIONS

The study evaluated the business models generated by the Component-Based and Traditional approaches when shifting paradigms in the re-engineering process in order to verify that the re-engineered business model is capable of representing the same business requirements of the legacy system. A legacy system was selected as part of the case study and re-engineered by using the Component-Based paradigm with the help of UML diagrams. The business model of the legacy system was recovered by using reverse engineering and compared to the component-based business model by using normalized reference models generated with the help of BWW transformation maps. These maps revealed that the re-engineered business models in UML are capable of representing the same business requirements of the legacy system. The identified UML diagrams required to represent the legacy models were class, use case and activity diagrams.

Normalized reference models in BWW terms were generated for the ERD, context, use case and class diagrams as a way to illustrate their use for business model comparison. A conflict was detected in this comparison as actors in the context diagram might not appear in the use case diagram as the latter only depict automated systems while context diagrams depict manual processes. On the other hand, component models complement legacy models by including objects that were not represented before re-engineering.

Although the study showed that these normalized reference models are useful tools to verify that component-based models represent the same requirements as the original traditional models of the legacy systems, further research will need to be conducted in order to find how to generate normalized reference models for the complete suite of traditional and UML component based models. As a result, business rules could be elaborated in order to re-engineer legacy systems into modern component-based systems for business requirements equivalency.

8. REFERENCES:

Benbasat, I., Goldstein, D. and Mead, M. 1987, The Case Research Strategy in Studies of Information Systems *MIS Quarterly*, 4, pp 368-386.

Chikofsky E.J and Cross J.H. 1990, Reverse Engineering and Design Recovery - a Taxonomy. *IEEE Software*. No7 Vol 1. pp13-17.

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

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

Dussart, A., Aubert, B. A. and Patry, M. 2004, An Evaluation of Inter-Organizational Workflow Modelling formalisms, *IDEA group publishing*

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

Fettke P., Loos P. 2003a, Ontological Evaluation of Reference Models using the Bunge-Wand-Weber Model, *North America Conference in Information Systems.*

Fettke, P.; Loos, P. 2003b, Onthological Evaluation of the Specification Framework proposed by the "Standardized Specification of Business Components" Memorandum - Some Preliminary Results. *In S. Overhage, K. Turowski (Eds.): Proceedings of the 1st Int. Workshop on Component Engineering Methodology*. Erfurt, pp. 1-12.

Green, P and Rosemann, M. 2000, Ontological Analysis of Integrated Process Modeling: Some Initial Insights, in *Proceedings of the Australian Conference on Information Systems (ACIS 2000)*, Brisbane, Australia, 6-8 December.

Green, P, Rosemann, M & Indulska M 2005, Ontological Evaluation of Enterprise Systems Interoperability Using ebXML *IEE transactions in knowledge and data engineering* Vol. 17. No 5.

Jacobson, I. ,Christerson, M., Jonsson, P. and Overgaard, G. 1993, Object-oriented Software Engineering: A Use Case Driven Approach, *Addison-Wesley*, Wokingham, Englad

Jacobson and F. Lindstrom 1991, Re-engineering of Old Systems to an Object-Oriented Approach, *Proceedings OOPSLA 1991*, pp. 340-350.

Irwin, G. Turk D, 2005, An Ontological Analysis of Use Case Modeling Grammar, *Journal of the Association for Information Systems* Vol. 6 No.1, pp.1-36

Longworth, R. 2003, Modeling Events for Today's System Requirements available from Internet (http://www.cips.ca/it/resources/eventviews.pdf (255Kb)) Accessed 20 Oct 2003.

Opdahl, A.L., and B. Henderson-Sellers 2002 Ontological Evaluation of the UML Using the Bunge-Wand-Weber Model, *Software Systems Model*, 1, pp 43-67.

Opdahl A and Henderson-Sellers B 2004, A template for defining enterprise modeling constructs *Journal of Database Management*, 15(2), 39-73, April-June 2004

Reed, Jr, P.R. 2002, *Developing Applications with JAVA and UML*, Addison-Wesley, Boston.

Rosemann, M., Recker J., Indulska M. and Green P. 2005, Process Modeling – A Maturing Discipline? *BPMI.org: Business Process Modeling Notation (BPMN)*, available at: http://www.bpmi.org/ accessed June 5 2005

Rosemann, M. and Green, P. 2002 Developing a meta model for the Bunge-Wand-Weber Ontological Constructs, *Journal of Information Systems*, 27, 75-91.

Satzinger, J.W., Jackson R.B., and Burd S.D. 2002, *Systems Analysis ad Design in a Changing World,* 5th ed. Course Technology, Boston, Mass.

Serrano, M., Carver, D., de Oca, C. 2001, Reengineering legacy systems for distributed environments, *Journal of Systems and Software*, 64(1), 37-55.

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

Szyperski. C., 1999. *Component Software: Beyond Object-Oriented Programming*, 1th edn Addison-Wesley, New Jersey.

Uschold, M., King, M., Moralee, S. & Zorgios, Y. 1998, The enterprise ontology. *The Knowledge Engineering Review*, vol 13.

Yourdon, E. 1989 *Modern Structured Analysis*. 1th edn Yourdon Press, Englewood Cliffs.

Wand, Y. and Weber, R. 1988,. An ontological analysis of some fundamental information systems concepts. In DeGross, J.I. & Olson, M.H. (eds.), *Proceedings of the Ninth International Conference on Information Systems, Minneapolis/ USA, November 30-* December 3,1988,213–225.

Wand, Y.; Weber, R. 1989, An Ontological Evaluation of Systems Analysis and Design Methods, *In: E. D. Falkenberg; P. Lindgreen (ed.): Information Systems Concepts: An In-Depth Analysis*. North-Holland, pp. 79-107.

Wand, Y. & Weber, R. 1990. An ontological model of an information system. *IEEE Transactions on Software Engineering (TSE)*, 16(11), 1282–1292.

Wand, Y.; Weber, R. 1993, *On the ontological expressiveness of information systems analysis and design grammars*, Journal of Information Systems (3:4), 1993, pp. 217-237.

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

Wand, Y.; Weber, R. 2002, Research Commentary: Information Systems and Conceptual Modelling - A Research Agenda, *Information Systems Research* (13), 2002, pp. 363-377.

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

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

Whitten, J. L., Bentley D. L. and Dittman K.V. 2000, *Systems Analysis and Design Methods*, 5th edn, McGraw-Hill, New York

Wolfgang P. 1997, Component-Based Software Development - A New Paradigm in Software Engineering? *Software-Concepts and Tools Software - Concepts and Tools* vol 18 no. 4 pp. 169-174.

Verrijn-Stuart, A.A. 2001. A Framework of Information System Concepts — *The Revised FRISCO Report*. Web document, draft version.