Portrayal Triumph’s Manual

Prem Raval, Nishi Patel, Manasa Nagavalli Kondibovina, Vineet Singh Bisht

Humber College

Etobicoke, ON M9W 5L7

Canada

**Abstract**

Requirement engineering is considered the most important phase of a software development process, as through this phase only the client as well as developers get the primitive idea of what is about to me made. Accepted that fetching the requirements from the client and meeting the expectations of the users is an iterative and never-ending process, but the first time before starting the process is very crucial. The core understanding of what has to be done is must and for that purpose modelling came into existence. Presenting data in the form of pictures or models enhances the understandability of the underlying information. Models provide a graphical view of the organizational procedures thereby reducing the complex reality of the work. Such models help improving organizational procedures and reduce the complexity hidden beneath. Hence, the importance of conceptual modelling and visual modelling is increasing day-by-day in software development process, hence grasping the understandability of this models are very significant to truncate the chances of errors to a minimal level. Thus, comprehending modelling and its relevant types is *vital* when developing a software. This paper talks about how a diagram overcomes the flaws of theoretical explanations such as language barriers, complexity in functionalities, priority functions and many more.

Keywords: modelling, requirement engineering, diagram, model understandability, model comprehending, software development, conceptual model.

# Introduction

“*A picture is worth a thousand words*” is an idiom which very well applies to conceptual model

in requirements engineering. Without following any procedures and techniques, developing any

software or to that matter any product will lead to a quality less outcome, with lots of issues in

the implementation processes. This can be avoided by following standard procedures, steps, models, principles, architectures of system provided by software engineering (KN 2018). The above said criteria is met through conceptual models, which represent a system through a composition of concepts, helping to understand the subject a model represents.

A model is a representation of an existing reality or a reality to be created. Modeling may be applied to material or immaterial objects. The processing and usage of information within the organization is done through different techniques, like natural language or conceptual models. While conceptual models lead the techniques being used, as it follows same standards throughout, depicting the requirements much more compactly and reducing the ambiguity. However, depending on the perspective, the type of model varies and also requires specific knowledge of modeling for analyzing and designing the software. Conceptual models and their visualizations, i.e., as diagrams, to represent abstract data, play an important role in the Information Systems (IS) field (Jens Gulden and Hajo A. Reijers, 2015) and such a visual approach stimulates natural characteristics of visual processing in human cognition(Jens Gulden and Hajo A. Reijers, 2015). There have been hundreds of conceptual modelling languages proposed as tools to understand and communicate software project information over the past few decades (John Mylopoulos, 1998). Among the wide range of existing languages are Event driven Process Chains (EPCs) (Keller, Nüttgens, & Scheer, 1992; Scheer, 2000), Unified Modeling Language (UML) Activity Diagrams (Object Management Group, 2011a), Yet Another Workflow Language (YAWL) (van der Aalst & ter Hofstede, 2005), and Business Process Model and Notation (BPMN) (Object Management Group, 2011b). When analyzing and improving complex organizational processes, Visual Process models are helpful (Kathrin Figl, Jan Mendling, Mark Strembeck, 2013). The effectiveness of software models depends on a number of communication qualities such as: Cost of production, comprehensibility, speed of ‘decay’ (loss of synchrony with the content it represents), and steepness of their learning curve. If a language is deficient in several of these qualities, even having high expressive power or well-formalized semantics, will not make it usable for communication. (Jorge Aranda, Neil Ernst, Jennifer Horkoff, and Steve Easterbrook, 2007). Practicing software modelling by implementing principles such as use of abstraction, problem decomposition, separation of concerns has proven to be one of the ways to combat complexity of software development. Model-driven approaches to systems development move the focus from third-generation programming language (3GL) code to models (in particular models expressed in UML and its profiles), (Shane Sendall and Wojtek Kozaczynski, 2003).

# Model and its importance

To understand the criticalness of the model and modelling concepts, first we need to know what does model actually mean.

## Defining Model:

When a system has to be created or changed, the requirements to proceed with the process are jotted and analyzed, but implementing them directly will get expensive and may not be feasible, so a mimic of the future system with its main functionalities is created to ensure that the requirements requested by the client and requirements understood by the analyst are the same. This prototyping of the system before creating an actual system is called a Model. Models are made considering the essential details and eliminating the non-essential details (also called abstraction), so that the main idea is nailed without any hindrance. Once a model is created, it is reviewed by the team subjecting it to various testing processes to ensure that it is meets the standards, is complete and satisfies the tests and is feasible without any requirements contradicting. Same is given to the client for experience and once the client assures that the requirements are met, original system is built. Modelling a system is done to avoid errors during the build of actual system, as in any case the requirement remains undetected, the cost to rectify it could be higher than the whole build cost and involves a lot of man power. This whole process could be done by just documenting the requirements using natural language alone, but due to natural language ambiguity increases and the requirements being misinterpreted by the client, modelling the system is the best approach.

Each model addresses some number of subject matters. Which subject matters to be included and which to be ignored can be decided by the software engineer depending on the requirements, but this might as well be required to be woven in several models together. (Mellor, Stephen J., Clark, Tony and Futagami, Takao, 2003). Model is a three- or two-dimensional representation of anything or of a proposed structure, typically on a smaller scale than the original. Modeling will simulate the real conditions as close as possible so that particular behavior can be predicted precisely (Shivanand Mahadeo Mali, 2017). The use of abstraction, problem decomposition, and separation of concerns help to combat the complexity of software development. As each of this is a way to segregate the problem into smaller parts and work on it, it helps to focus more on individual units and when done all of the solutions are integrated to make a complete solution.

### Modeling:

As we have seen earlier that models are an abstract of a system, in the same way context modeling is the process of developing those models for a particular system. When developing a new software or a system there are some principles that need to be taken care of, like requirement fulfilling, accuracy, customer satisfaction, minimizing rework costs, longer software lifespan. Modeling has become a major way of implementing these principles. E.g. Software modeling. Developers build different models throughout the development process in order to verify that the eventual software will meet the requirements (Michael Jackson, 1995). This is done through several stages of trial and error, where the prototypes of solutions are made and verified through various tests performed at various stages of development. These Model-driven approaches to systems development move the focus from third-generation programming language (3GL) code to models (in particular models expressed in UML and its profiles). (Shane Sendall and Wojtek Kozaczynski, 2003). It is about representing a system using some kind of graphical notation, which is almost always based on notations in the **Unified Modeling Language (UML).** Such models help the analyst to understand the shallow as well as deep functionalities provided by the system and also these models are used as a communication portal with customers. Models can explain the system from different perspectives:

* An **external** perspective, where model is based on the context or environment of the system.
* An **interaction** perspective, where model is based on the interactions between a system and its environment, or between the components of a system.
* A **structural** perspective, where model is based on the organization of a system or the structure of the data that is processed by the system.
* A **behavioral** perspective, where model is based on the dynamic behavior of the system and how it responds to events.

### System Model:

System model is a term used to define model for a particular system. Now what do we mean by a “system” precisely? A system is a group of interacting or interrelated entities forming a unified whole, delimited by boundaries. In a simpler sense, connection of every object that influences, either directly or indirectly, to the main objective is a system. In requirements engineering, everything that needs to be modelled is considered to be a system, e.g., a bank is a system with all its entities forming subsystem. The easiest way to understand is the way John Krogstie and Arne Solvberg tell, that is “System model: A model of a system.”. Though many times one does need understand some specific ideologies to create an accurate model for the proposed system. Following are some steps that might be taken care of:

• As an ***upper ontology*** (common metamodel), standardized ontological models for each domain of interest is built, using system concepts (entity, relationships, attributes, roles).

• Each element (entity, attribute, relationship) in the overall system model has to be associated with a ***semantic type*** referencing the corresponding element in the appropriate synthesis domain ontological model. Thereby reversing the earlier referred semantic flattening.

• The semantics of the element in the system model needs to match precisely with the element in the acceptable ***domain knowledge***, so as to avoid challenges that may arise in doing the above. And if any such mismatches have occurred, would have to be resolved by the engineers of the specific system.

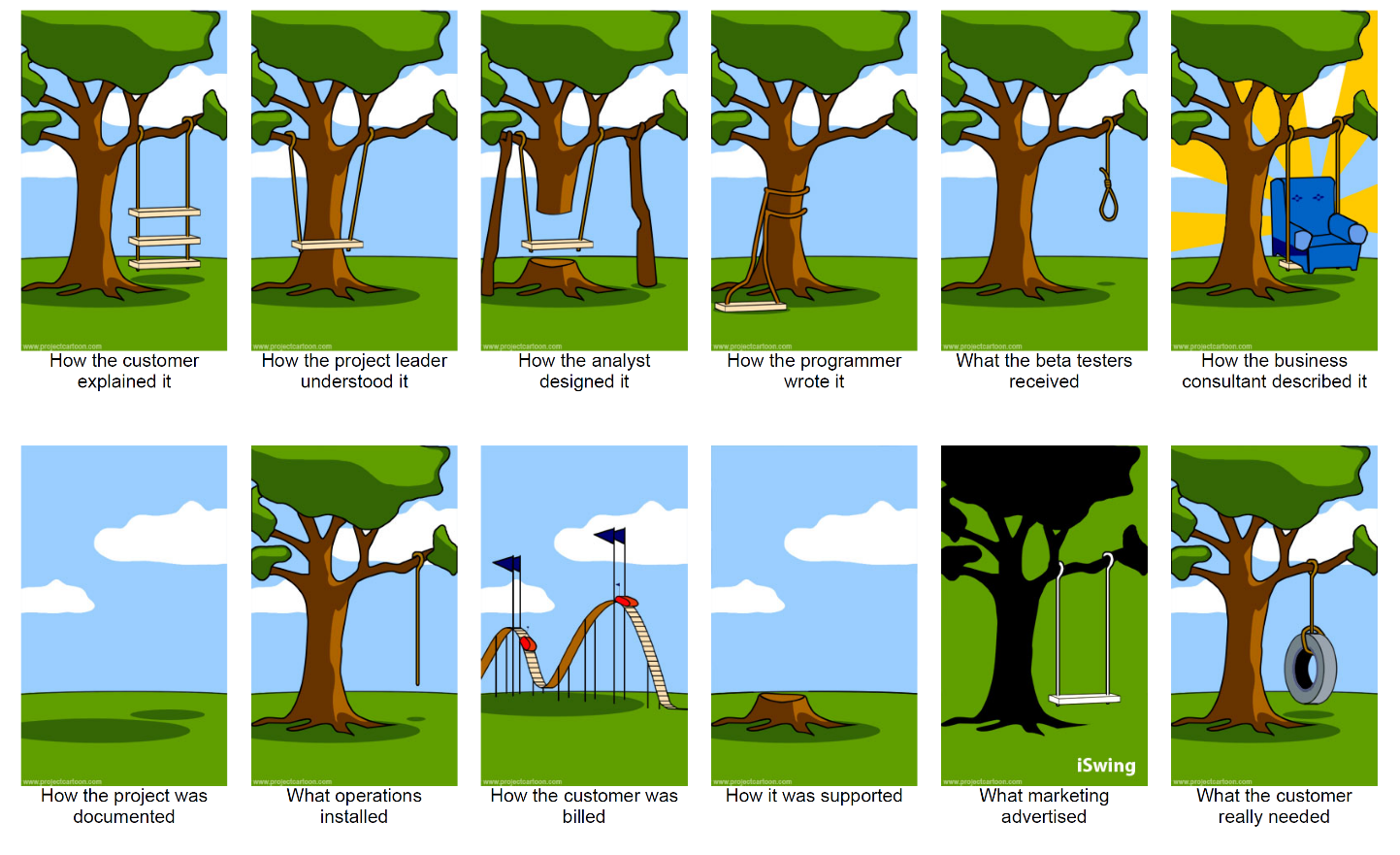
• For each relationship between whole **o*bjects*** in the system, appropriate view mapping has to be used to ***bind the abstractions*** to corresponding concrete elements in the other whole.

• ***Domain-specific view*** can be generated from the overall system model to integrate domain-specific tools. And any new elements introduced in the domain-specific tool will be integrated into the overall system model by mapping back to the view.

(Swaminathan Natarajan, Kesav Nori, Anand Kumar, Viswanath Kasturi, Subhrojyoti Roy Chaudhuri, Venkatesh Choppella, 2018)

# Software design prior to designing a software

As the definition given by Sharon Andrews White and Mohammad “Tuan” Amith, “*Software design is the activity between the requirements gathering phase and implementation phase, which results in an abstract model of the system*”. Why it comes in between is because it is a representation on how the requirements and domains will be implemented as a design. They also put up a question asking, “why design, why not just start coding because, is not the code the design itself? Is design necessary when requirements are always changing, when time-to-delivery is important, and when the software is continuously evolving?” Design is the representation of the future system. It shows all the entities and features of the system to be, giving a feel of how it would be and what can be expected when completed. Just in case if any of the features are not satisfactory, it is easier and cost efficient to change or replace it in the designing phase as compared to that in the build phase.



*Fig 1*

The cartoon shown in fig 1 (www.projectcartoon.com , 2006) has been widely shared regarding managing scope. The above figure gives a gist of how system requirements would transverse from the customer to the analyst and how the end product would be looking like the opposite to what it was supposed to look like. This happens when ambiguity, misunderstanding and unrelated requirements are documented. Documenting of requirements is very important but more importantly is the method used to document the requirements, in the above diagram, requirements were not documented properly and when the same requirements were passed down the lane, information was misinterpreted leading to undesired outcomes at each stage and hence the output is not what was expected. Conceptual model plays an important role in fighting incorrect data interpretation, but this also calls for model understandability by the personnel involved in the system including the stakeholders.

A good design should be understandable, clear without ambiguity and meet all the requirements. It is key to understanding and communication. Benefits of good design are as follows:

• Good design **captures documentation** and enables knowledge transfer

• Good design **can communicate** with customers and clients.

• Good design can **enable understanding** of the product and problem domain.

• Good design can be an **economic asset**.

A poor design holds a lot of ambiguity causing multiple interpretation of a single design, the mentioned requirements may not be met, thereby increasing the work repetition. Some of the results of the poor design are as follows.

• Poor design can result in failed projects.

• Delivery of a poor system.

• Poor design results in a system that is very hard to change, or rigid.

• Poor design becomes very brittle.

• Poor design is not portable.

# Conceptual Models

A conceptual model is a representation of a system, made of the composition of concepts which are used to help people know, understand, or simulate a subject the model represents. It is also a set of concepts. In the context of designing technical and organizational aspects of information systems (IS), conceptual models are considered to be capable tools used as methodical instruments for initial IS development and management.

## Quality of conceptual models:

'Quality' is a difficult notion, and within the field of information systems, many approaches to quality have been proposed. Quality of conceptual models are highly relevant in research and practice as their purpose is fulfilled only when used with appropriate quality. These models have high application potential in various fields like database design and management, business and process management, implementation and customization of standard software. To communicate the model content and verify if the important requirements are met, it is mandatory for the user or stakeholder to have knowledge about conceptual models. Therefore, it is important to identify potential principles, characteristics or relationships influencing the understandability of conceptual models in order to improve the success of conceptual modeling. (Understanding Understandability of Conceptual Models – What Are We Actually Talking about? Constantin Houy, Peter Fettke, and Peter Loos). Quality of conceptual models, requirement specifications and model quality has been divided into seven main areas:

* ***Physical quality:*** The relationship between knowledge of those performing modeling and the model.
* ***Empirical quality:*** The relationship between the model and another model containing the same statements being somehow regarded as better through different arrangement or layout.
* ***Syntactic quality:*** The relationship between the model and the language used for modeling.
* ***Semantic quality:*** The relationship between the model and the domain of modeling.
* ***Perceived semantic quality:*** The relationship between the knowledge of the modelers and their interpretation of the model.
* ***Pragmatic quality:*** The relationship between the model and the modeler’s interpretation of the model.
* ***Social quality:*** The relationship between different model interpretations.

The business process models or process models for short, may be used to “make decisions about where, how, and why changes to the processes should be enacted to warrant improved operational efficiency” (Sánchez-González, García, et al., 2013, Toward a quality framework for business process models, p. 1) The interest in the value proposition of process models is growing (Bandara and Gable, 2012, A Formative Measurement Model Of Business Process Model Quality), because process modelling is always accompanied with many costs( such as purchasing of a modelling tool, opportunity costs etc.). Various studies exist that have contributed to the knowledge about the quality of process models (Moreno-Montes de Oca et al., 2014, A systematic literature review of studies on business process modeling quality). Some of the studies focus on the development of theoretical models that support the concept of process model quality (e.g., Krogstie et al., 2006, Quality of Business Process Models ). On the other hand, other studies focus on the development of metrics that characterize particular model properties (e.g., Vanderfeesten, Cardoso and Reijers, 2007, Quality metrics for business process models). Yet the other studies (e.g., Mendling et al., 2010, Seven Process Modeling Guidelines) aim to develop practical guidelines as to assure or improve the quality of the process models. What aggregates the build knowledge is that there is no broad overview that brings together these and potentially other, directions of research about process model quality, which is also a known research gap (Dikici et al., 2017, Factors Influencing the Understandability of Process Models; Recker and Mendling, 2007, How Collaborative Technology Supports Cognitive Processes in Collaborative Process Modeling: A Capabilities-Gains-Outcome Model). However, an overview like this is valuable because it helps forming a solid foundation on which new research can build in order to contribute to the development of the knowledge domain (Figl, 2017, Comprehension of Procedural Visual Business Process Models).

The research domain currently, is characterized by a lack of homogeneity concerning the nomenclature and by an abundance of quality dimensions and classifications a(Sánchez-González, García, et al., 2013). Software perspective approach is preferred by some authors for process model quality (e.g., Vanderfeesten, Cardoso and Reijers, 2007), while others make use of quality frameworks for conceptual models (e.g., Sánchez-González, García, et al., 2013). For some concepts different terms are used moreover, while sharing the same semantics (e.g., comprehensibility and understandability). It is possible however, that authors have different (meticulously built) opinions about these aspects, this lack of consensus results in unambiguity and confusion, which poses a threat to the progress of the domain (Moody, 2005, Theoretical and practical issues in evaluating the quality of conceptual models: Current state and future directions).

Based on the categories proposed by Mendling (2007) and by Reijers and Mendling (2011), model-related factors also have a crucial role. Based on their relativity to the abstract (structure) or the concrete syntax (graphical layout) of the models, they are subdivided in factors. Personal as well as a small number of other factors were also found, rather than only context-related factors.

**Model-related factors:** Factors relating to the abstract syntax of a process model is what the majority of the selected studies discuss. The structure of process elements and their mutual relations is described by this structure (La Rosa, Wohed, et al., 2011, Managing Process Model Complexity). Model-related factors that relate to the concrete syntax of process models also have a noteworthy role. The graphical representation of the models is what the focus is on here.

**Personal factors:** The factors that influence the understandability of process models was studied by Reijers and Mendling (2011). The authors in their work, direct attention to the important role of personal factors, which assert that understandability implies a human interpretation.

**Other factors:** Personal and model-related factors as shown in majority of the literature were drivers of process model quality. And yet, some authors still identify other drivers (Mendling and Strembeck, 2008, Understanding Business Process Models: The Costs and Benefits of Structuredness; Soffer et al., 2012, Modeling Styles in Business Process Modeling), such as modelling language and representation medium.

**Conceptual modeling in information systems**

ISs can be analyzed in at least three distinct and complementary perspectives which adds to the difficulty [44]:

*• Contributions Provided.*

*• Structure and Behavior.*

*• Functional Behavior.*

An IS does not exist for itself. Examples of such a definition would be: An IS designed system supports operations, management, and decision-making in an organization or An IS is a communication facilitating system among its users. The major issue with such a definition is that it fails to provide a clear characterization of ISs. A system designed to collects, stores, processes, and distributes information about the state of a domain can be characterized as an Information System. As easy as it is to agree on those functions, the problem is that they are too general and are not related to the purpose for which the IS exists. Due to these very reasons, majority of authors prefer a more specific definition of the functions, one that captures the nature of ISs more neatly. To that end, it is considered that an IS has **three** main functions [46]:

1. **Memory function** which is to maintain a representation of the state of a domain;

2. **Informative function** which provides information about the state of a domain;

3. **Active function**, which performs actions that change the state of a domain.

A memory function and some informative function is what all conventional ISs perform.

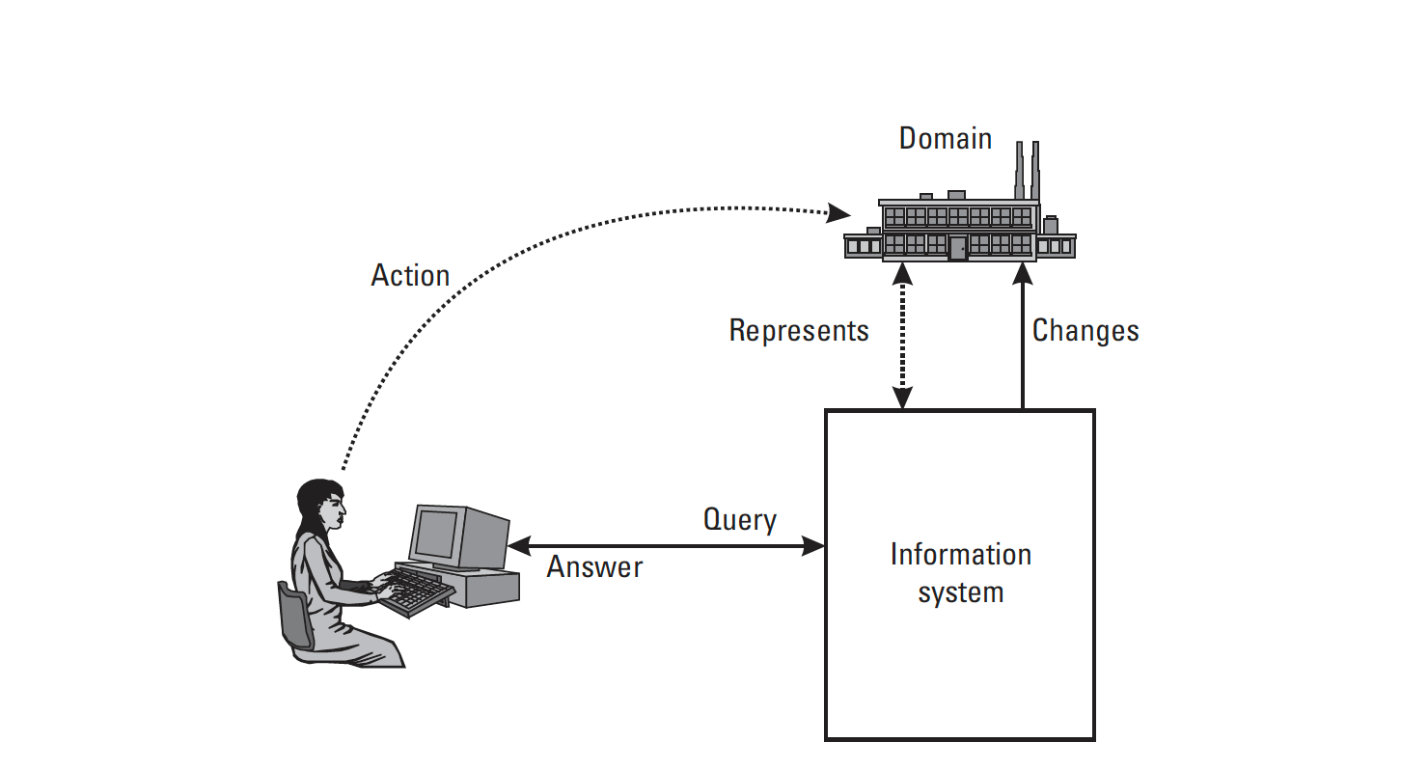


Fig. 2 (An Introduction to Conceptual Modeling of Information Systems, Antoni Olivé)

According to the fig 2, the *information function* is used by the system to provides users with information about the state of the domain. The state of the domain can often be observed directly in the domain, and at the same time can also be represented in the IS.

The *memory function* on the other hand is needed by the other two functions. Maintaining an internal representation of the state of the domain is the objective here. This representation however, must be accurate and complete [45]. This memory function can be performed in two different execution modes: **on request or autonomously**. The first mode comprises of the system memorizing the state and order because of a user explicitly telling the system that (normally implicitly). The second mode autonomously, has the system memorizing the state of the domain instead without having to accept an explicit request from the user. Since the memory function does not perform anything that directly affects users or the domain, it is considered passive. It still is however required by the other functions and constraints to function.

The main objective behind conceptual modeling is the elicitation about a domain and formal definition of the general knowledge required by the IS to perform the required functions.

This section describes the kinds of knowledge required by most ISs.

This is the line of reasoning we will follow:

• We must define which is the concrete state that must be represented if the ISs memory function is to maintain the representation of the state of the domain.

• Defining the causes of changes along with the effects of those changes on the state is required since the state of most domains varies with time.

• Consistency must be followed while representing the state in the IS; therefore, defining what it means to be a consistent representation, is a must.

• Many a times, some inference capability on the part of the IS can be required to answer queries posed by users. Derivation rules is what this capability makes use of, which also must be defined.

The term conceptual model is defined in the IS field as commitment corresponding to viewing the domains in a particular way. The same conceptual model can be applied to several different domains in principle, and various conceptual models could also be applied to the same domain.

*Conceptualization* of that domain is defined as the set of concepts that are used in a particular domain. *Ontology* is the term that is used in some languages for the specification of that conceptualization in that domain [47, 48]. Several conceptualizations for a given domain may be present and, thus, several ontologies as well. An ontology can also be termed as a concrete view of a particular domain. For the persons that observe and act on that domain, it is therefore also an ontological commitment.

# Understandability of a Model

“Understandability can be interpreted as a kind of pragmatic quality of conceptual Models” [Gemino, A., and Wand, Y. A,2004]. Related to the ease of use respectively, the effort for reading and correctly interpreting a conceptual model, is also a cognitive process of assigning meaning to the different parts of a conceptual model [Gemino, A., and Wand, Y.,2003]. Only if they possess an appropriate quality, can they fill their function and purpose, which makes the topic of conceptual model quality highly relevant for research and practice. In this context, the ‘Understandability of Conceptual Models’ is considered an important quality criterion (Arisholm, E., Briand, L.C., Hove, S.E., and Labiche, Y., 2006). In order to support the communication about and a collective understanding of the functionality and structure of information systems (IS), a high understandability is important when conceptual models are used during the analysis, design and usage, which is one of the main purposes of conceptual models having a long tradition in information systems (IS) research and the conceptual modeling community. Based on this fact, research results on model understandability are likely to be ambiguous if the underlying conceptualization and operationalization are not considered in detail during the interpretation of results.

“*Model Understandability*” as the ease with which the model can be understood (Canfora, Garc, Piattini, Ruiz, & Visaggio, 2005; Moody, 1998) definitions, emphasize the relevance of cognitive load theory in this context. Understanding the complex control flow logic of process models is a task likely to demand high cognitive effort. Cognitive activities like visual perception, attention, short and long-term memory processing, reasoning, and problem solving have to be performed. According to the cognitive load theory, high cognitive load during problem-solving exercises impairs learning and knowledge acquisition (Sweller, 1988). The theory differentiates between three types of cognitive load: **intrinsic, extraneous, and germane**. Intrinsic cognitive load is determined by the complexity of information (i.e., the number of elements, and their relations and interactions). Very complex and large models therefore, are likely to increase cognitive load and may adversely affect understanding (Gruhn & Laue, 2009; Nordbotten & Crosby, 1999). Individual competencies of users concerning process modeling in general and each specific symbol set along with the domain of a model in particular (Lowe, 1989; Winn, 1993) may improve understanding and information extraction when reading a model. Notational deficiencies in symbol sets may lead to heavier cognitive load for users, hampering model comprehension. Cognitive challenges have been studied extensively to better match system engineering methods and human cognitive capabilities in programming – and in reading and understanding data and process models (Burton-Jones et al., 2009; Gemino and Wand, 2004; Hoc et al., 1990). Computers, can easily process program code and translated conceptual models of arbitrary size, whereas a human understanding is influenced by cognitive bias and irrational beliefs (Green et al., 2009). Human understanding of process models is particularly relevant because process models involve a lot of tasks, which “must be enacted by a human rather than a machine” (Curtis et al., 1992). Plenty of attempts have been made to identify influence factors of process model understanding (e.g., Figl et al., 2013a; Figl et al., 2013b; Mendling et al., 2012; Reijers and Mendling, 2011) and process model creation (e.g., Recker et al., 2012). The fact that a high interactivity between model elements can make a model more difficult to understand has been revealed by Empirical Studies (Guceglioglu and Demirors, 2005; Vanderfeesten et al., 2008). Previous research on process models has suggested that understanding their control-flow is generally difficult for humans (Mendling et al., 2010a). Mendling et al. (2012) results on the other hand have shown a positive influence of modeling knowledge and experience on the ability to understand process models. Therefore, it can be said that modelers with higher process modeling knowledge will be better in solving deductive reasoning tasks and will also experience these tasks as easier than modelers with lower process modeling knowledge (Kathrin Figl and Ralf Laue,2015). Maes et al. (Ann Maes and Geert Poels, 2007) even defines the perceived ease of understanding of a model as “the degree to which a person believes that using . . .would be free of mental effort”. A thorough review and reconstruction of 42 experiments on conceptual model understandability conducted in the research (Constantin Houy, Peter Fettke, and Peter Loos, 2012) shows that there is a variety of different understandings and conceptualizations of the term *model understandability*. This results in the term remaining ambiguous. Research results on model understandability are hardly comparable and partly imprecise, which shows the necessity of clarification about what the conceptual modeling community is actually talking about when the term model understandability is used. The research classifies the different observed dimensions of model understandability in a reference framework, in order to overcome this shortcoming. (Constantin Houy, Peter Fettke, and Peter Loos, 2012).

# Model Comprehensibility

Comprehensibility, is an essential quality of communication artifacts. Cryptic, misleading, or vague documents and diagrams will not serve their communication purpose. Even though studying the overall comprehensibility of a model is important, from a language evolution perspective it is even more relevant to discover which elements of a notation work well and which do not. If there are important gaps between the hypotheses and relevant cognitive perspectives, the evaluator could be missing some comprehension-related benefits, and may wish to refine the hypotheses to include them. With varying degrees of empirical rigor has the topic of model comprehensibility been previously addressed. Pseudocode and flowcharts also yield no difference in comprehension as reported by two early studies (Ramsey, H.R., Atwood, M.E., and Van Doren, J.R.,1983), but Scanlan (Scanlan,1989) countered that flowcharts actually outperform pseudocode. No theoretical foundation to ground the evaluation has been given by either studies, and their methodological problems cast serious doubt on their validity (Jorge Aranda, Neil Ernst, Jennifer Horkoff, and Steve Easterbrook, 2007). Both comprehension and correct interpretation of models are relevant for many different tasks (Burton-Jones et al., 2009). Therefore, comprehension questions the most common way to measure comprehension of process models (e.g. Mendling et al., 2012; Reijers and Mendling, 2011). Such comprehension questions can be characterized as deductive reasoning tasks, since correct answers can be derived from general knowledge on process-flow logic and the specific process model. The questions require deductive reasoning, which is defined as the “*mental process of making inferences that are logical*” (Johnson-Laird, 2010). The work of Bodart et al. (Bodart, F., Patel, A., Sim, M., Weber, R., 2001) defines different depths of understanding which formatively operationalize model understanding, viz. surface-level (“correctly recalling model parts”) and deeper-level understanding (“correctly answering questions concerning the model content and problem solving”) contrasts to the work of Agarwal et al. (Agarwal, R., De, P., Sinha, 1999) which only considers one dimension of understandability (accuracy of model comprehension) and measures it by means of comprehension questions concerning the model content (Mendling et al., 2010b; Mendling et al., 2012; Reijers and Mendling, 2011). Little is known about *what exactly* makes it difficult for humans to reason on the basis of a process model. The ability to deduce correct conclusions based on the ‘**premises**’ expressed in a process model is relevant to almost any situation in which the comprehension of the model is necessary for analysis or reengineering tasks (Burton-Jones et al., 2009). Generalizability may be limited by abstract labels because research revealed that material with abstract labels has a slightly different effect on human reasoning than material with content labels (Beller and Spada, 2003; Markovits et al., 2002), which also eases model comprehension (Mendling et al., 2012).

Research by Mendling has also shown that theoretical knowledge is more important to syntactical process model comprehension than other factors such as practical experience (Mendling et al., 2012). It is, however, likely that cognitive load increases even further for higher element interactivities based on theoretical considerations and empirical research on the negative effect of model size on comprehension (Mendling et al., 2007a; Mendling et al., 2010a). Empirical evidence that cognitive load and lower comprehensibility of process models can be improved by high interactivity of elements has also been expressed.( Lothar Krempel, 1999).

There are many ways to understand the level of understandability about a specific model and its characteristics among the industry. Many people are performing various experiments in order to understand that which model is best for which purpose. Some of those experiments and their research outcomes could be found in the following table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Source | Type | N | Independent  Variables | Dependent  Variables | Outcome |
| (Pinggera, Zugal et al. 2011) | Cheetah Experimental Platform (CEP) to record each user interaction along with corresponding time stamp | 26 graduate students | Modeler’s sequence of actions | Analyzing the process of process modelling | Modeling phase diagram(analysis technique) |
| (Figl and Laue 2015) | Research model to capture the influence of different control patterns in process model and interactivity of model elements | 155 | Cognitive difficulty of understanding specific parts of a process model | human understanding of process models | Evaluate a Dataset of comprehension questions |
| (Agarwal, De et al. 1999) | Laboratory experiment  +replication, two groups of students with knowledge on modelling | 71, assigned two experiments | Qualitative pattern analysis | Accuracy of model comprehension | comprehension test: comprehension score rating participants’ answers on 8 comprehension questions each |
| (Figl, Mendling et al. 2013) | Paper questionnaire with four sections | 188 business school students(100males,88females) | perceptual discriminability deficiencies, semiotic clarity deficiencies | Effects of notational deficiencies on process model comprehension | Cronbach’s α |
| .. | .. | .. | .. | .. | .. |

**Conclusion**

We set out with this research to develop a better insight into the factors that helps in making the process models more understandable for humans. We concluded that if you get, understand and select the best possible model for your software development than the resulting software would be accurate but still error prone. A good model can overcome language as well as understandability barriers among humans and make the software development a more easier and comprehensive task. Our research is characterized by other limitations, in particular the small set of models being considered and the limited set of participants. With larger sets in future replications of the experiment, we can investigate the impact of secondary factors in greater detail.

# References

1. Agarwal, R., P. De, and A.P. Sinha, Comprehending object and process models: An empirical study. IEEE Transactions on Software Engineering, 1999. 25(4): p. 541-556.

2. Aranda, J., et al. A framework for empirical evaluation of model comprehensibility. in International Workshop on Modeling in Software Engineering (MISE'07: ICSE Workshop 2007). 2007. IEEE.

3. Aranda, J., et al. A framework for empirical evaluation of model comprehensibility. in International Workshop on Modeling in Software Engineering (MISE'07: ICSE Workshop 2007). 2007. IEEE.

4. Arisholm, E., et al., The impact of UML documentation on software maintenance: An experimental evaluation. IEEE Transactions on Software Engineering, 2006. 32(6): p. 365-381.

5. Beller, S. and H. Spada, The logic of content effects in propositional reasoning: The case of conditional reasoning with a point of view. Thinking and Reasoning, 2003. 9(4): p. 335-378.

6. Boehm-Davis, D.A., On the use of psychology in evaluating software documentation. Behavior Research Methods & Instrumentation, 1983. 15(2): p. 217-221.

7. Burton-Jones, A., Minimizing method bias through programmatic research. MIS quarterly, 2009: p. 445-471.

8. Canfora, G., et al., A family of experiments to validate metrics for software process models. Journal of Systems and Software, 2005. 77(2): p. 113-129.

9. Détienne, F., Expert programming knowledge: a schema-based approach, in Psychology of programming. 1990, Elsevier. p. 205-222.

10. Figl, K. and R. Laue. Cognitive complexity in business process modeling. in International Conference on Advanced Information Systems Engineering. 2011. Springer.

11. Figl, K. and R. Laue, Influence factors for local comprehensibility of process models. International Journal of Human-Computer Studies, 2015. 82: p. 96-110.

12. Figl, K., J. Mendling, and M. Strembeck, The influence of notational deficiencies on process model comprehension. Journal of the Association for Information Systems, 2013. 14(6): p. 1.

13. Gemino, A. and Y. Wand, A framework for empirical evaluation of conceptual modeling techniques. Requirements Engineering, 2004. 9(4): p. 248-260.

14. Green, M.A., et al., Death by dissolution: sediment saturation state as a mortality factor for juvenile bivalves. Limnology and Oceanography, 2009. 54(4): p. 1037-1047.

15. Gruhn, V. and R. Laue. Reducing the cognitive complexity of business process models. in 2009 8th IEEE International Conference on Cognitive Informatics. 2009. IEEE.

16. Guceglioglu, A.S. and O. Demirors. Using software quality characteristics to measure business process quality. in International Conference on Business Process Management. 2005. Springer.

17. Gulden, J., et al. Toward Advanced Visualization Techniques for Conceptual Modeling. in CAiSE Forum. 2015.

18. Heeager, L.T. and J. Rose, Optimising agile development practices for the maintenance operation: nine heuristics. Empirical Software Engineering, 2015. 20(6): p. 1762-1784.

19. Houy, C., P. Fettke, and P. Loos. Understanding understandability of conceptual models–what are we actually talking about? in International Conference on Conceptual Modeling. 2012. Springer.

20. Houy, C., P. Fettke, and P. Loos. Understanding understandability of conceptual models–what are we actually talking about? in International Conference on Conceptual Modeling. 2012. Springer.

21. Johnson-Laird, P.N., Mental models and human reasoning. Proceedings of the National Academy of Sciences, 2010. 107(43): p. 18243-18250.

22. KN, D., The Evaluation of Efficient Process Model Based on Survey. 2018.

23. Krempel, L., Visualizing networks with spring embedders: two-mode and valued data. 2000.

24. Kühne, S., et al., Business Process Modelling with Continuous Validation, in Business Process Management Workshops. 2009. p. 212-223.

25. Lowe, N.K., Explaining the pain of active labor: the importance of maternal confidence. Research in Nursing & Health, 1989. 12(4): p. 237-245.

26. Maes, A. and G. Poels. Evaluating quality of conceptual models based on user perceptions. in International Conference on Conceptual Modeling. 2006. Springer.

27. Markovits, H., C. Doyon, and M. Simoneau, Individual differences in working memory and conditional reasoning with concrete and abstract content. Thinking & Reasoning, 2002. 8(2): p. 97-107.

28. Mellor, S.J., T. Clark, and T. Futagami, Model-driven development: guest editors' introduction. IEEE software, 2003. 20(5): p. 14-18.

29. Mendling, J. and M. Strembeck. Influence factors of understanding business process models. in International Conference on Business Information Systems. 2008. Springer.

30. Moody, J., et al., Performance functions and reinforcement learning for trading systems and portfolios. Journal of Forecasting, 1998. 17(5‐6): p. 441-470.

31. Moreira, A., J. Araújo, and A. Rashid, A Concern-Oriented Requirements Engineering Model, in Active Flow and Combustion Control 2018. 2005. p. 293-308.

32. Mylopoulos, J., Information Modeling in the Time of the Revolution. Information systems, 1998. 23(3-4): p. 127-155.

33. Natarajan, S., et al. A Conceptual Model of Systems Engineering. in INCOSE International Symposium. 2018. Wiley Online Library.

34. Nordbotten, J.C. and M.E. Crosby, The effect of graphic style on data model interpretation. Information Systems Journal, 1999. 9(2): p. 139-155.

35. Pinggera, J., et al. Tracing the process of process modeling with modeling phase diagrams. in International Conference on Business Process Management. 2011. Springer.

36. Recker, J., Scientific research in information systems: a beginner's guide. 2012: Springer Science & Business Media.

37. Scanlan, D.A., Variables and Cognitive Factors Predicting Intermediate‐Level Programming Success: A Preliminary Study. Computer Science Education, 1990. 1(4): p. 347-354.

38. Sendall, S. and W. Kozaczynski, Model transformation: The heart and soul of model-driven software development. IEEE software, 2003. 20(5): p. 42-45.

39. Sweller, J., Cognitive load during problem solving: Effects on learning. Cognitive science, 1988. 12(2): p. 257-285.

40. Vanderfeesten, I., H.A. Reijers, and W.M. Van der Aalst, evaluating workflow process designs using cohesion and coupling metrics. Computers in industry, 2008. 59(5): p. 420-437.

41. Vogel-Heuser, B., Usability experiments to evaluate UML/SysML-based model driven software engineering notations for logic control in manufacturing automation. Journal of Software Engineering and Applications, 2014. 7(11): p. 943.

42. White, S.A., Software Design, a 2011 Primer.

43. Winn, W., Instructional design and situated learning: Paradox or partnership. Educational Technology, 1993. 33(3): p. 16-21.

44. Langefors, B., .Information Systems, Proc. IFIP .74, North Holland, 1974,

pp. 937.945

45. Wand, Y., and R. Weber, .An Ontological Analysis of Some Fundamental Information

Systems Concepts,. Proc. 9th. Intl. Conf. on Information Systems, Minneapolis,

MN, Dec. 1988, pp. 213.225.

46. Boman, M., et al., Conceptual Modelling, New York: Prentice-Hall, 1997.

47. Uschold, M., and M. Gruninger, .Ontologies: Principles, Methods, and Applications, Knowledge Engineering Review, Vol. 11, No. 2, 1996, pp. 93.136.\

48. Mylopoulos, J., .Information Modeling in the Time of the Revolution,. Information

Systems, Vol. 23, No. 3/4, 1998, pp. 127.155.

**Portrayal**

[The body of your paper uses a half-inch first line indent and is double-spaced. APA style provides for up to five heading levels, shown in the paragraphs that follow. Note that the word Introduction should not be used as an initial heading, as it’s assumed that your paper begins with an introduction.]

# [Heading 1]

[The first two heading levels get their own paragraph, as shown here. Headings 3, 4, and 5 are run-in headings used at the beginning of the paragraph.]

## [Heading 2]

[To add a table of contents (TOC), apply the appropriate heading style to just the heading text at the start of a paragraph and it will show up in your TOC. To do this, select the text for your heading. Then, on the Home tab, in the Styles gallery, click the style you need.]

### [Heading 3]

[Include a period at the end of a run-in heading. Note that you can include consecutive paragraphs with their own headings, where appropriate.]

#### [Heading 4]

[When using headings, don’t skip levels. If you need a heading 3, 4, or 5 with no text following it before the next heading, just add a period at the end of the heading and then start a new paragraph for the subheading and its text.]

##### **[Heading 5]**

[Like all sections of your paper, references start on their own page. The references page is created using the Citations & Bibliography feature, available on the References tab. This feature includes a style option that formats your references for APA 6th Edition. You can also use this feature to add in-text citations that are linked to your source, such as those shown at the end of this paragraph and the preceding paragraph.]