Universidade Federal do Espírito Santo

Bernardo Ferreira Bastos Braga

Cognitive effective instance diagram design

Vitória - ES, Brazil
July, 2011

Universidade Federal do Espírito Santo

Bernardo Ferreira Bastos Braga

Cognitive effective instance diagram design

Monografia apresentada ao curso de Ciência da Computação do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação. Orientador: Prof. Dr. João Paulo Andrade Almeida

Vitória - ES, Brazil
July, 2011

Acknowledgements

First of all, I’d like to thank my family for all the love and support they gave me. My mother Ruth and her husband João, my father Antonio Carlos and his wife Rozeli and my brother Pedro. They have been with me most of my life and their involvement in my upbringing was essential to defining my character, specially my love for all types of information and knowledge. Likewise, my friends have immensely helped me overcoming my troubles and limitations and will probably never realize how important they have been and how much they have affected me. I’ve had the pleasure of loving large number of friends through my life and listing the names of a few seems unfair to me. I’m also grateful to my friends and teachers from the Federal University of Espírito Santo for their participation in my academic experience. In particular, I’ve had the opportunity to participate in the NEMO laboratory and work with Giancarlo, Alessander and Veruska, who were key in my research with long discussions about ontologies and conceptual modeling. Last but not least, I’d like to thank my advisor and friend João Paulo Andrade Almeida for everything we’ve been through. He is a standard for excellence and competence and my role model for being a competent researcher.

RESUMO

Essa monografia investiga o design de diagramas, em particular, diagramas de instância. Primeiramente, revisamos a linguagem de modelagem conceitual OntoUML, a atividade de avaliação de modelos e os trabalhos que conduzimos anteriorment nesta área. Em seguida, analizamos a teoria de percepção e processamento cognitivo de imagens para fazer o *design* de diagrams que sejam eficientes do ponto de vista cognitivo e aplicamos este design em um exemplo prático para criar diagramas de instância para a ontologia MusicBrainz. Terminamos o trabalho com conclusões e direções futuras para pesquisa.

Abstract

This monograph investigates diagram design, in particular, instance diagram design. First, we review the OntoUML conceptual modeling language, model assessment and previous work we have produced in this area. Following, we analyze human graphical information processing mechanisms to design cognitive effective diagrams and we apply such design in a practical example to create instance diagrams for the MusicBrainz ontology. We end with a conclusion and directions for future work.

Index

1 Chapter 1 – Introduction

1.1 Motivation

In his 1972 ACM Turing Award Lecture entitled “The Humble Programmer”; E. W. Dijkstra (DIJKSTRA, 1972) discussed the sheer complexity one has to deal with when programming large computer systems. His article argues that the increase in computer’s processing power leads to an increase in the expectation of the use of such power; which leads to an increase in the complexity of programming computer systems that could meet such expectations.

According to Dijkstra, we as computer scientists should take a humble position towards such complexity, account for human’s limited cognitive capacities and use whatever possible resources to deal with such complexity, as opposed to supposing that we can deal with it using our minds alone. He opened the eyes of theorists and practitioners to the fact that programming computers is an extremely complex task which should not be taken lightly. Although his lecture was explicitly addressed to the act of computer programming, we may read into his words more broadly and see that the act of codification, the representation of ideas is generally a very difficult task, and his plea to humility can be applied in many different levels when we regard transferring information and transforming it for physical out-of-mind representation. At time of such lecture, there was a crisis in software, which motivated the speech; projects were often over budget, delayed, low quality, unmanageable and hard to maintain. Dijkstra alludes to the fact that the explicit account for complexity would be key to solving the crisis.

From the software crisis emerges the necessity of reliable development. Many approaches appear and are claimed to be the ultimate solution to the crisis. Contrary to these claims, in 1986, Fred Brooks states that there are “no silver bullets”; i.e. no single development method can solve all the inherent problems of creating software. “I believe” – he says –“the hard part of building software to be the specification, design, and testing of this conceptual construct, not the labor of representing it and testing the fidelity of the representation. We still make syntax errors, to be sure; but they are fuzz compared with the conceptual errors in most systems.” (BROOKS, 1987) The hard part of building software is to correctly specify what is supposed to be built; not building it.

The practical relevance of thorough requirements analysis is emphasized by evidence provided by the empirical software engineering community, which states that it is much cheaper to find and fix software problems during the requirements and design phase than after delivery (BASILI; BOEHM, 2001). In this context, properly acquiring knowledge on a problem domain prior to detailed design has justified several efforts in conceptual modeling.

In this work we address conceptual models, artifacts that may be reasoned upon and that represent what should be built prior to actually building it; a means to analyze the idea behind the software prior to codification. In a general sense, we address what Brooks called a “conceptual construct” that can be specified, designed and tested. More specifically, the following paraphrase from John Mylopoulos defines conceptual modeling as adopted in this work:

“Conceptual modelling supports structuring and inferential facilities that are psychologically grounded. After all, the descriptions that arise from conceptual modelling activities are intended to be used by humans, not machines... The adequacy of a conceptual modelling notation rests on its contribution to the construction of models of reality that promote a common understanding of that reality among their human users." (MYLOPOULOS, 1992)

Such definition should make clear that our research is directed towards humans, not machines. In order to support the communication purposes of conceptual modeling, a formal conceptual model must capture a modeler's intention and convey a precise message with unambiguous semantics. This is particularly important if conceptual models are to be used effectively as a basis for the construction of an information system.

The need to express conceptual models with precise real-world semantics has justified the revision of a portion of UML into the OntoUML conceptual modeling language. This revision enables modelers to adopt a widely employed language (UML) while making fine-grained distinctions between, among other things, different types of classes according to the UFO foundational ontology (GUIZZARDI, 2005). These ontological distinctions reflect, in turn, different manners an object can be an instance of a type. In particular, the language focuses on the different modal (temporal) consequences implied by the different modes of instantiation.

Regardless of the quality of the conceptual modeling language employed, conceptual modeling itself remains a challenging activity, requiring additional methodological and tool support for ensuring that the modeler's intention is properly reflected in the models. Dijkstra's plea to humility should apply for conceptual modelers as well; tools that ease the task and make it less error prone should be treasured.

The development tool to aid the construction of OntoUML conceptual models presented in previous work (BENEVIDES, A. B.; GUIZZARDI, 2009) (an OntoUML model editor) has given us so far the opportunity to verify models for ontological well-formedness, i.e., adherence to ontological consistency rules defined at the language-level. While this guarantees some quality for conceptual models by enforcing ontological consistency via domain-independent syntactic rules, it does not serve to increase the modeler's confidence in the correct representation of the intended domain conceptualization, i.e. it does not support modelers in answering the question "have we built the right model for this particular domain?". Similar to the way a spellchecker may correct text but doesn't aid in the construction of phrases that make sense; or an IDE may correct code syntax, but doesn't predict what behavior is interpreted as a bug; the model editor may correct the syntax, but it doesn't predict what the intended message was. It is left to the human conceptual modeler to define the message and assess whether the model conveys what was meant.

In (BENEVIDES, A. et al., 2009; BRAGA et al., 2010) we have proposed an approach to facilitate the validation process of conceptual models defined in OntoUML by transforming these models into specifications in the logic-based language Alloy (a "lightweight formal method" (JACKSON, 2006) and using its analyzer to generate instances of the model and possibly produce assertion counter-examples. Validation is defined here as "the process of determining the degree to which a model is an accurate representation of the real-world from the perspective of the intended uses of the model" (DEPARTAMENT OF DEFENSE, 2007). Our approach supports validation by allowing the observation of sequences of snapshots of model instances. We argue that the visualization of instances confronts the modeler with the implications of modeling choices. Should the instances reveal inadmissible states-of-affairs (or sequences thereof), the model may be analyzed to identify opportunities for correction in an iterative validation approach.

Moreover, we believe that this can also be used as means to identify missing or over restrictive domain rules. This tool has been called OntoUML2Alloy to allude to similar work done for the UML language, UML2Alloy (ANASTASAKIS; BORDBAR, 2005).

Although OntoUML2Alloy facilitates the task of validation of OntoUML conceptual models, our initial experience with the tool showed that its usability and readability decreased rapidly as the complexity of the models increased. In the simulation of complex models, individuals may be classified into several classes and may engage in various types of relationships at once. All this may be overwhelming for the novice user or, in some cases, even for the experienced user. For the tool to have practical relevance, we must take Dijkstra's humble position again, this time considering the difficulty of the task of validation. The view of John Mylopoulos on conceptual models as means of communication between humans (not machines), indicates that such improvements should be directed at how humans communicate, how humans perceive the world and how human learning takes place.

1.2 Objective

Our objective is to facilitate OntoUML model validation by improving the effectiveness of model simulation. In particular, we consider the opportunities for improvement which arise from the usage of visualization techniques.

1.3 Approach

Although this monograph was developed at an Informatics department and plays the role of a final assignment in a Computer Science Bachelor degree, it is mostly motivated by topics beyond computation. Since we are concerned with the humans involved in computer interactions, our approach is interdisciplinary and aims at applying research from Cognitive Sciences and Psychology to improving (human) understanding of conceptual model simulations used in Computer Science and Ontology Engineering. Areas investigated include visual perception, sight interpretation, visual querying, working memory and multimedia learning. These theories are used to predict the effect (on humans) of visual design choices for model visualization at the instance level and to define assessment strategies.

Inspired by Daniel Moody's "The Physics of Notations: Evidence-based principles for designing cognitively effective visual notations" tutorial on the ER2009 conference, we started by exploring the visual aspects of a concrete syntax that could facilitate model assessment. Our interest with visual aspects has been motivated primarily by observations in the fields of visual processing and cognition.(WARE, 2004) (MOODY, D. L., 2010) In particular, one can observe that human graphical information processing exhibits properties that can be exploited in careful design to contribute to information transferring. The use of visual elements such as shape, color and texture (to name a few) affect perception and interpretation of sight prior to conscious attention e.g. a closed shaped contour is perceived as an entity, while connecting lines are perceived as relationships (WARE, 2004). Thus, the visual syntaxes proposed in this work address the concern that symbols should be naturally distinguished and easily remembered. However, the choice of visual elements in itself does not entail symbol meaning understanding, contributing mostly to perceiving symbol difference and similarity. Symbols are meant to represent something and making symbols distinguishable can only do so much for this purpose.

To help understanding of symbol meaning, we provide presentation strategies that promote gradual (on demand) learning of domain symbols. Concerns towards limitations in working memory take a central role in this approach and instructional design based on Cognitive Load Theory (MAYER; MORENO, 2003) is applied, aligned with Moody's ideas for complexity management (MOODY, D. L., 2010). We present information in chunks of constrained complexity, simulating (an expert's) conscious acts of attention towards the selection of relevant elements for cognitive processing. These chunks constitute contexts; highly connected concepts are selected and grouped for presentation, promoting concise learning of model elements on the basis of already known related concepts. We argue that the use of these complexity controlling mechanisms, contribute not only to learning of domain symbols, but also to facilitating problem solving. By removing distracting information, limited working memory resources may be directed to addressing more focus to a specific aspect of model assessment.

In addition, we believe that our presentation strategy leads to the construction of highly connected mental schemas, allowing deep understanding of the model and promoting the creation of mental strategies for visually segmenting a complex simulation in understandable chunks. Regarding working memory limitations, atomic symbols may compose patterns which are dealt with as single atomic elements; making these concepts more suitable for cognitive processing, considering human's limited-capacity working memory. This not only benefits the current validation task but also eases subsequent, more complex tasks; patterns can be combined to create other (higher-level) patterns.

We approach the problem of improving OntoUML model validation by building a prototype to test the effects of these ideas in practice. This prototype can be regarded as a proof-of-concept of the application of visualization techniques to model assessment. We consider this a first step towards defining a prescriptive methodology for OntoUML model assessment based on visualization.

1.4 Structure

This work is structured as follows: Chapter two introduces OntoUML language, focusing on the instance level. Chapter three discusses model assessment and the differences between model validation and verification. It also discusses the problems we faced so far in our approach to model assessment. Chapter four introduces human graphical information processing models and discusses the perceptual processing of graphics. Additionally, there is a discussion on symbol design, visual pattern design and visual search strategies. Chapter five discusses cognitive processing and the role of working and long-term memory. Complexity management mechanisms are presented as means to reducing load on working memory and to promote diagrammatic reasoning skill development. Chapter six exemplifies the application of the presented theory in the design of instance-level diagrams for the MusicBrainz ontology, as well as the complexity management mechanisms and a presentation strategy to promote skill development. Finally, chapter seven brings final conclusions. Chapter eight presents the bibliography used in this work.

2 OntoUML

2.1 Introduction

"Conceptual modelling is the activity of formally describing some aspects of the physical and social world around us for purposes of understanding and communication"(MYLOPOULOS, 1992) Given this definition, a conceptual modeling language should be able to promote de construction of models that can be used to transfer conceptualizations (means of conceiving reality) between human users of the language. UML is the *de facto* standard for conceptual modeling activities and such widespread use has revealed that the language lacks a precise definition of its formal semantics (GUIZZARDI et al., 2004). In order to model conceptualizations of reality, a language should be founded on formal upper-level ontologies (GUIZZARDI et al., 2004). Upper-level ontologies describe very general concepts like "space, time, matter, object, event, action, etc., which are independent of a particular problem or domain" (GUARINO, 1998).

Abstractions of a given portion of reality are means for humans to organize their experience for reuse and cooperation. A *conceptualization* is a manner in which the world is perceived and involves the way in which concepts are formed; it is defined by Guizzardi (2005) as "the set of concepts used to articulate abstractions of state of affairs in a given domain". Figure 2-1 illustrates the relation between *conceptualizations*, *domain abstractions*, *models* and *modeling languages* (the figure was originally extracted from (GUIZZARDI, 2005) and had the terminology adapted in (CARRARETO, 2010)). *Domain Abstractions* are instances of such conceptualization. Using a *conceptualization* to define some portion of perceived reality is creating abstractions on that domain of reality. The physical representation of a *domain abstraction* is called a *model*, which is composed using a *modeling language*. Such *modeling language* can be interpreted in terms of the *conceptualization* and represents the *conceptualization*. Both *conceptualizations* and *domain abstractions* are abstract entities that exist only in the mind of someone who perceives and understands reality in some way. The *modeling language* and the *model* are concrete entities which can be manipulated, realized and evolved.

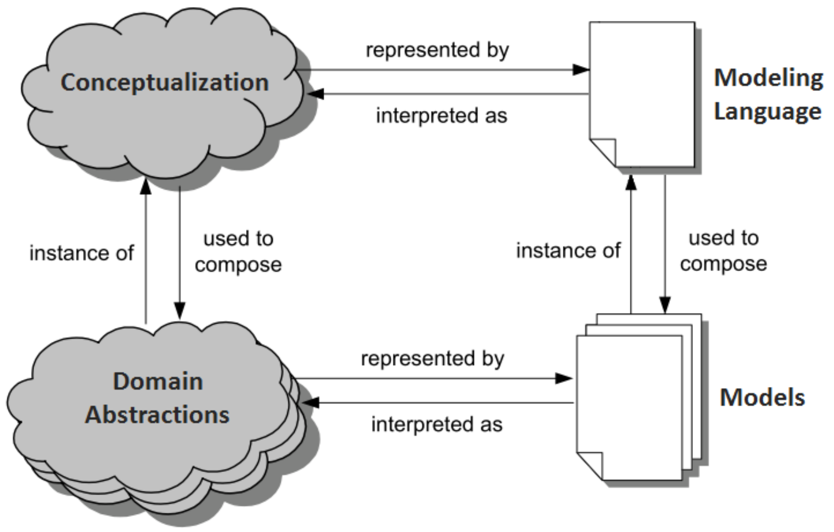


Figure 2-1 Relationships between terms used in the conceptual modeling literature (extracted from (CARRARETO, 2010))

OntoUML conceptual modeling language is an extension of UML with specific concerns for the semantics behind the modeling constructs. More specifically, it is founded on the Unified Foundational Ontology (GUIZZARDI, 2005). The stereotypes offered in OntoUML allow the construction of ontologically well-founded conceptual models, meaning they express precise semantics. Figure 2-2 exemplifies (in gray) some ontological distinctions that are realized as stereotypes in OntoUML.

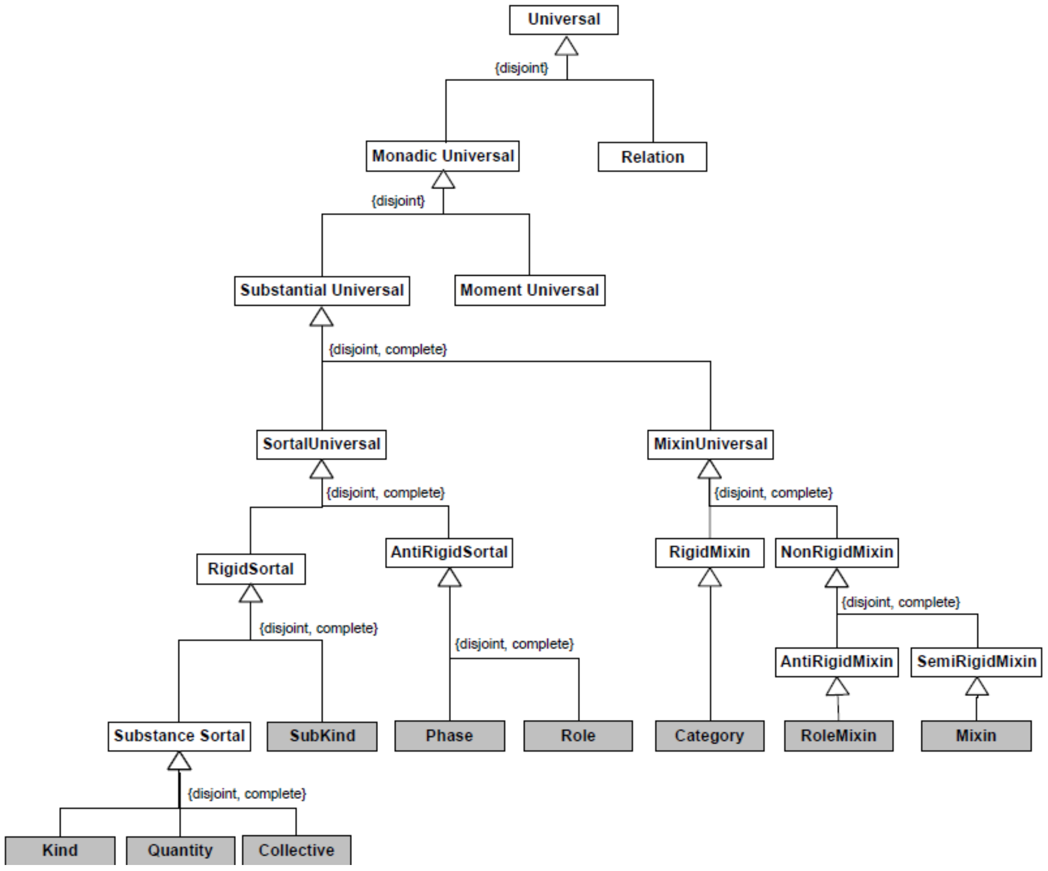


Figure 2-2 A fragment of OntoUML showing ontological distinctions among *Substantial Universals*

This chapter aims at presenting a fragment of OntoUML language to those unfamiliar with the language. Detailed definitions of OntoUML can be found elsewhere, such as in (GUIZZARDI, 2005), (BENEVIDES, A. et al., 2009), (BENEVIDES, A. B., 2010) and (ZAMBORLINI, 2011) (in Portuguese). Here, we take a different approach to explaining OntoUML, focusing on the instance level, where our diagram design may take place. Figure 2-3 exemplifies a model specification using OntoUML. This will be used later on this chapter to demonstrate how the language may be used.

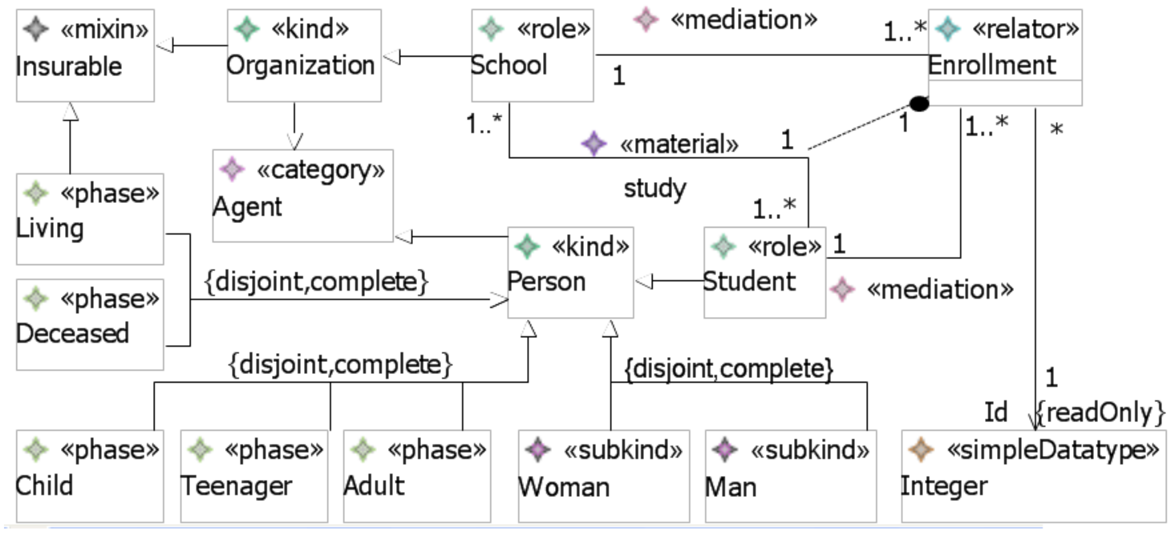


Figure 2-3 Example OntoUML model

2.2 *Individuals and Universals*

The distinction between Universals and Individuals (roughly, types and instances) is fundamental in OntoUML conceptual modeling. Individuals are entities that exist in reality possessing a unique identity, and Universals, conversely, are space-time independent pattern of features, which can be realized in a number of different Individuals (GUIZZARDI, 2006). For example, a particular person (such Barack Obama or Osama Bin Laden), would be an Individual, while the concept of Person would be a Universal.

States of affairs consist of *individuals (instances of monadic universals)* and the relations between them; each individual classified by some (one or more) *Universals*. For example, consider the situation depicted in Figure 2-4 An example instance diagram (an informal representation of a possible instantiation of the model presented in Figure 2-3) where a person, Bernardo, studies at Móbile, a school. Bernardo and Móbile are *individuals* and instantiate *Monadic Universals* such as Person, Man, Student, Organization and School. Also, a *relation* holds between them and such *relation* is also classified by (instantiates) *Universals*, such as Enrollment.

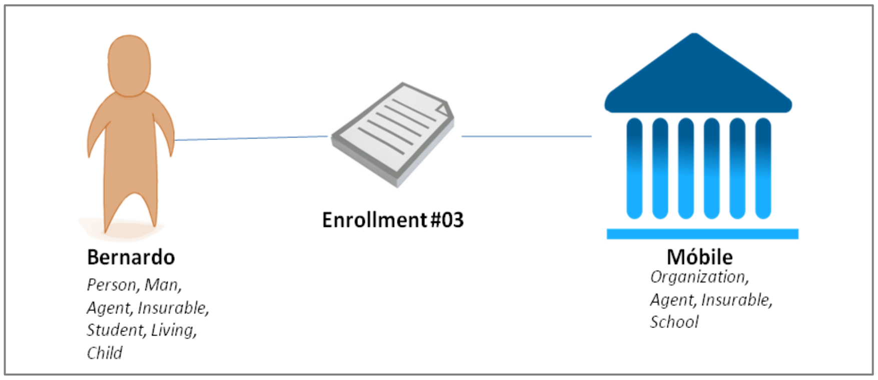


Figure 2-4 An example instance diagram. Bernardo studies at Móbile

2.3 Dependence

Frequently, some individuals depend on others, requiring for their very existence a relationship with some other entity e.g. consider how a person depends on her brain. Dependence is an extraordinarily common and varied phenomenon and OntoUML language offers constructs that reflect distinctions in types of dependence. For example, an enrollment depends on the student and on the school in a way which is not symmetrical: an enrollment can only exist as long as the participants exist, but the same cannot be said conversely. Moreover, for a person to be a student, he must be enrolled in (and therefore depends on) some school and not on a specific school, while an enrollment depends on specific individuals to exist: if you change the person or the school, it cannot be said to be the same enrollment (the identity of the enrollment is connected to the participants). This exposes a fundamental distinction between two types of dependency relation: generic dependence and specific dependence. The student depends generically on **some** school to be a student, a generic dependence; Enrollment#03 depends specifically on Bernardo and Móbile, constituting a specific dependence.

A particular yet very important type of specific dependence is existential dependency. Existential dependency means an individual depends necessarily (in a modal sense) on some specific individual to exist e.g. Bernardo depends existentially on his brain; there is no situation where he can maintain his identity without having his specific brain as a part of him. Existentially dependent relations include *characterization*, *mediation* and *essential* or *inseparable* parthood. We refrain from detailing these types of relations for the sake of conciseness.

Definition 1 (*existential dependence*) an individual *x* is *existentially dependent* on another individual *y* iff, as a matter of necessity, *y* must exist whenever *x* exists. In other words, in every world *w*, if *x* exists in *w* then *y* must also exist in *w*.

2.4 Moments and Moment Universals

A special kind of existentially dependent individual is called a *moment*, derived from the German word *Momente* from the writings of Husserl and denotes what is called also called a “trope”, “abstract individual” or “property instance”. *Moments* can be understood as objectified properties of an individual and are *inherent* to them (ZAMBORLINI, 2011). For example the color of an apple is a *moment* inhering on the apple; John’s headache is inherent to John, as the intensity of the headache is inherent to the headache. Individuals that do not inhere in other individuals are called *substantial individuals*. *Universals* that describe *substantial individuals* are called *Substantial Universals*; likewise universals that describe *moments* are called *Moment Universals*; both are specializations of *Monadic Universal*.

2.5 Modal properties of Universals

Necessity is also strongly connected to the fundamental concept of rigidity; one that applies to *Universals*. A rigid universal applies necessarily to its instances, while an anti-rigid universal applies contingently. For example, consider a domain where Person is a rigid class and Student is an anti-rigid class; in such scenario any individual Person cannot cease to be a person but may become and cease to be a Student.

Definition 2 (*Rigidity*) A type T is rigid if for every instance x of T, x is necessarily (in the modal sense) an instance of T. In other words, if x instantiates T in a given world w, then x must instantiate T in every possible world w’:

R(T) =def $\Box(\forall x\ T(x) \rightarrow \Box(T(x)))$.

Definition 3 (*Anti-rigidity*): A type T is anti-rigid if for every instance x of T, x is possibly (in the modal sense) not an instance of T. In other words, if x instantiates T in a given world w, then there is a possible world w' in which x does not instantiate T:
AR(T) =def $\Box(\forall x\ T(x) \rightarrow \Diamond(\neg T(x)))$.

The set of rigid universals in any conceptual model defines its *backbone taxonomy*, the most important features of an ontology (GUARINO; WELTY, 2000). The term refers to the structural role these concepts perform in conceptual modeling: considering only the elements of the backbone gives someone a survey of the entire universe of possible instances (GUARINO; WELTY, 2000). These are divided in three types: *categories* which do not carry identity (-I), *kinds* which supply identity (+O) and *subkinds* which carry but do not supply identity (-O+I). Universals that carry identity are called Sortals, while those that do not are called Mixins. Mixins classify individuals that obey different principles of identity (e.g., Agent in Figure 2-3 which classifies different kinds of entities such as Persons and Organizations). Hence, mixins are types which provide properties to (characterize) individuals which have already being individuated by sortal-supplied principles.

Anti-rigid universals describe the characteristics that apply contingently to individuals. In OntoUML there are two types of anti-rigid universals: Roles and Phases which are differentiated with regard to their specialization conditions. For the case of Phases, the specialization condition is always an intrinsic one. For instance, in Figure 2-3, a Child is a Person within a certain age. For Roles, in contrast, their specialization condition is a relational one: a Student is a Person who is enrolled in (has a study relation to) a School, etc. If the relation is no more, the person ceases to be a student; i.e. the class instantiation depends on the relation to another individual. Formally speaking, this distinction is based on a meta-property named Relational Dependence:

Definition 4 (*Relational Dependence*) A type T is relationally dependent on another type P via relation R iff in every world w, for every instance x of T there is an instance y of P in that world such that x and y are related via R in w.

Additionally, as discussed in (GUIZZARDI, 2005), Phases (in contrast to Roles) are always defined in a partition set. For instance, in Figure 2-3, the universals Child, Teenager and Adult define a phase partition for the Kind Person. As consequence, we have that in on each world w, every Person is either a Child, a Teenager or an Adult in w and never more than one of these.

3 OntoUML Model Assessment and Tool Support

3.1 Introduction

Model assessment is an important part of conceptual modeling in which one inspects the model for correctness and adequacy to its purposes. Model development is a human centered activity; more specifically, it can be seen as a communication activity. Human activities are naturally error-prone but communication activities are even worse; the difficulty of proper communication appears even in ancient texts, such as in the etiological myth “The tower of Babel” in Genesis 11.

To illustrate the need for model assessment, imagine the activity of writing text in natural language. Be the text long or short, the writer usually feels the need to read what was written, to assess the meaning of the words he wrote and evaluate if the message conveys his intentions, i.e., if it says what he meant. Models convey messages which must be assessed for correctness as well. Two key aspects are analyzed: syntax and semantics. We call the process of assessing the syntax “model verification”, and of assessing the semantics “model validation”. *Caveat lector* - verification and validation are words with similar meaning; their usage to convey these specific meanings, as we do, is not completely agreed upon.

In this chapter, we discuss the tasks involved in OntoUML model assessment. More specifically, in section 3.2 we discuss syntax verification and in section 3.3 we discuss intention validation, i.e., understanding the model semantics and determining its degree of accuracy in conveying an intended meaning. The main purpose of sections 3.2 and 3.3 is to differentiate between these two aspects of model assessment so that the context in which this work is immersed may be clearer. In section 3.4, we proceed to presenting simulation as means to validation and discuss the OntoUML2Alloy tool, which is our current tool support for validation. Section 3.5 exposes some of the problems in our tool and motivates directions that were followed in this work in order to address these problems.

3.2 Syntax verification

The word “verification” is widely used in this work and elsewhere in software engineering literature as the activity of verifying model adherence to some set of formal rules (or constraints). Previous efforts on model correctness led to the development tool called “OntoUML editor” (BENEVIDES, A. B.; GUIZZARDI, 2009) . The editor aids the construction of OntoUML conceptual models by providing a visual interactive environment to construct OntoUML conceptual models and by allowing automatic syntax verification. While that guarantees some quality to the model, namely the adherence to the language’s syntactic rules, it does not serve to increase the modeler’s confidence in the correct representation of the domain (BRAGA et al., 2010). In other words, it helps determining if the model was built correctly, but it does not help determining if the right model was built. The OntoUML model editor was built on the Eclipse Modeling Framework, is open-source and available online via Google Code (BENEVIDES, A., 2011).

3.3 Intention Validation

Validation is defined here as “the process of determining the degree to which a model is an accurate representation of the real-world from the perspective of the intended uses of the model" (DEPARTAMENT OF DEFENSE, 2007). The understanding of the meaning of verification is so important to understanding validation because it makes clear what validation isn't. Models may be invalid or inaccurate representations of the real-world and still be syntactically correct, i.e., still pass verification. One way to illustrate the difference between verification and validation is to make an analogy to the software development process. In such analogy, verification would be the process of checking the syntax prior to compiling a code and validation would be debugging. Most compilers will find syntax errors and some IDEs will even suggest a correction, nevertheless, they are hopeless in understanding what behavior the code should exhibit. It is only by running and testing the code that the actual behavior of the code will be revealed. Before testing, the developer usually has expectations of such behavior, which can be (and frequently are) false, constituting what is called a bug in the code, i.e. a fragment of code that does not compute as expected. The same applies to model verification and validation, in a way: verification may guarantee that the model is correct in terms of syntax, and validation can be done by testing the model (in our case, simulating it) and seeing how it behaves. Similarly, "bugs" may emerge in models in two modes: as state-of-affairs that should be possible, but are not implied by the model and as state-of-affairs that shouldn't be possible, but are implied by the model.

Our previous efforts in OntoUML model validation resulted in the design and implementation of OntoUML2Alloy (BRAGA et al., 2010), a software that uses Model-Driven Development (MDD) techniques to automatically transform models in the OntoUML language to the logic-based language Alloy (JACKSON, 2006). The product of this transformation is an Alloy specification that can be fed into the Alloy Analyzer to generate a sequence of instance-level states which are valid according to the language axioms. The analyzer may be further used to produce assertion counter-examples, i.e. to query possibilities within the model's constraints. Our approach supports validation by allowing the observation of sequences of snapshots of model instances. We argue that the visualization of instances confronts the modeler with the implications of modeling choices. Should the instances reveal inadmissible states-of-affairs (or sequences thereof), the model may be analyzed to identify opportunities for correction in an iterative validation approach. Moreover, we believe that this can also be used as means to identify missing or over restrictive domain rules (BRAGA et al., 2010).

3.4 Problems in visualization and opportunities for improvement

The available functionalities offered by the tool support can be very effective in helping to improve the quality of OntoUML conceptual models; by correcting syntax mistakes, by helping to find missing or over restrictive constraints or even by helping to identify missing domain concepts. However, through our experience, we have identified problems regarding the tool's usability.

The current version of the OntoUML2Alloy tool relies on the Alloy Analyzer's built-in visualization functionalities to allow the visualization of model instances. When working with our transformed models, the Alloy Analyzer usually present images with high visual complexity (all information is displayed at once) and is therefore inappropriate for most users involved in OntoUML model assessment. **Error! Reference source not found.**Figure 3-1 shows an example of how it can be difficult to assess information this way.

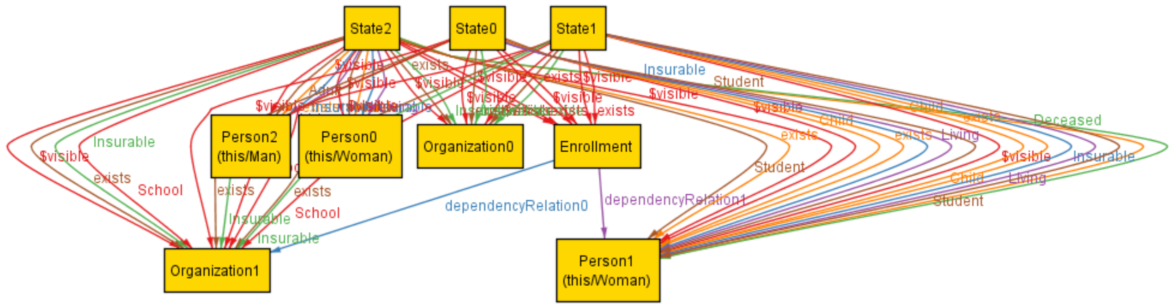
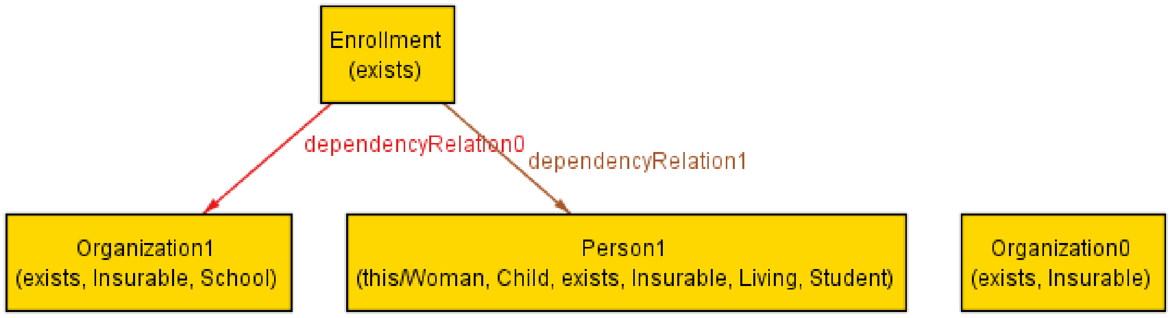
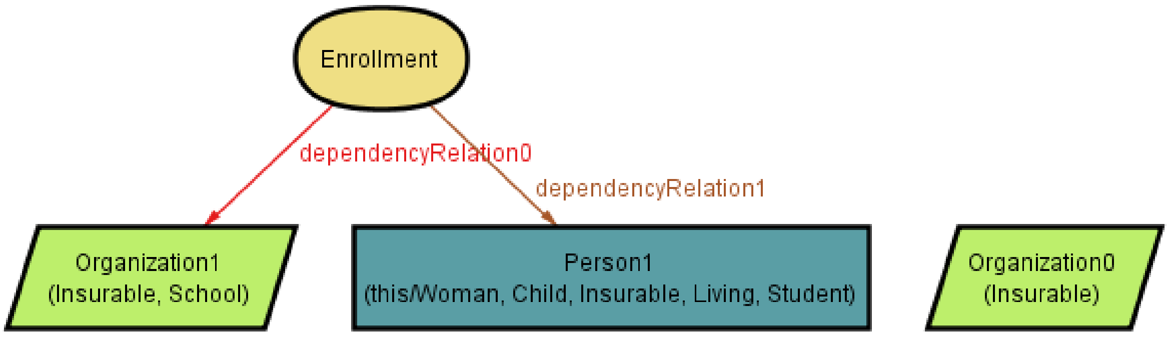


Figure 3-1 Instances generated by the Alloy Analyzer are hard to assess if untouched

It is possible to customize the visualization, reducing its visual complexity and thus augmenting its usefulness. Projecting over the states reduces the amount of information displayed at a time, only the information relevant to that state is displayed. Figure 3-2 shows an example of projecting over the state signature.

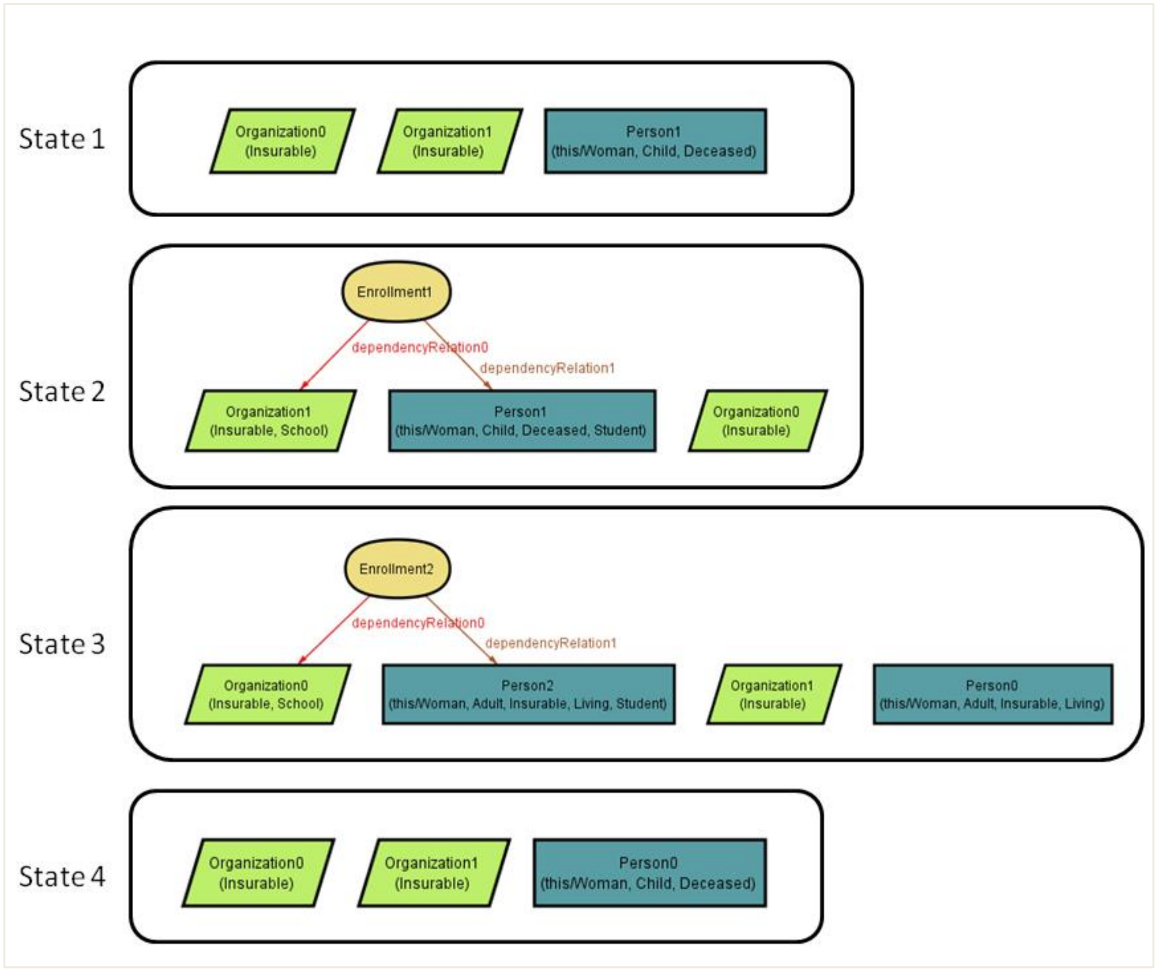

Enrollment (exists)
dependencyRelation0
dependencyRelation1
Organization1 (exists, Insurable, School)
Person1 (this/Woman, Child, exists, Insurable, Living, Student)
Organization0 (exists, Insurable)
Figure 3-2A projection on State0

Regarding the symbols, themes can be generated automatically. However, the result is seldom satisfactory and some customization is necessary. Figure 3-3 exemplifies the result of applying the automatic theme. The customization process requires knowledge in the transformation strategy, as it is conducted on terms of the Alloy model obtained from the transformation.


Enrollment
dependencyRelation0
dependencyRelation1
Organization1 (Insurable, School)
Person1 (this/Woman, Child, Insurable, Living, Student)
Organization0 (Insurable)
Figure 3-3The result of applying the automatic theme

The visualization of model instances could be improved if there were more options to designing symbols. There are a limited number of available symbols, color and line types and no support to add more. One cannot create custom icons or set texture to nodes.

Additionally, the default visualization theme uses the classical Sugiyama hierarchical layout algorithm (SUGIYAMA et al., 1981), slightly modified (CHANG, 2009). There are no means to change the layout algorithm and the result is unsatisfactory. Nodes are arranged in rows and the layout algorithm is run in each state individually, making it difficult to find individuals across multiple snapshots. To illustrate the problem, consider Figure 3-4. Although State 1 and State 4 look very similar, one can only notice the difference by reading the nodes labels and realizing Person1 does not exist in State 4, rather Person0 is occupying the same space. The same thing happen in States 2 and 3, individuals change positions and this can only be noticed by reading the labels, which is slow and error-prone. The problem is minimized here since one can all the states simultaneously; however, in the Alloy Analyzer, one can only see one state at a time, and would have to change back and forth to compare the states. We believe the visualization could benefit from different layout algorithms that can help the assessment of change between snapshots.


State 1
Organization0 (Insurable)
Organization1 (Insurable)
Person1 (this/Woman, Child, Deceased)
State 2
Enrollment1
dependencyRelation0
dependencyRelation1
Organization1 (Insurable, School)
Person1 (this/Woman, Child, Deceased, Student)
Organization0 (Insurable)
State 3
Enrollment2
dependencyRelation0
dependencyRelation1
Organization0 (Insurable, School)
Person2 (this/Woman, Adult, Insurable, Living, Student)
Organization1 (Insurable)
Person0 (this/Woman, Adult, Insurable, Living)
State 4
Organization0 (Insurable)
Organization1 (Insurable)
Person0 (this/Woman, Child, Deceased)
Figure 3-4 A sequence of states.

4 Improving perceptual processing of instance diagrams

4.1 Introduction

Model assessment is central in this monograph and in the previous chapter we discussed, particularly in section 3.4, the problems we faced when visualizing the simulations. In this chapter, we present design principles and theory to aid the application of visualization techniques to model assessment, following the approach presented in section 1.3. We consider this a first step towards defining a prescriptive methodology for OntoUML model assessment based on visualization. Our efforts are focused on cognitive effectiveness, not on aesthetics aspects (although we do try to make the diagrams beautiful). In aesthetic terms, the diagrams could be regarded as sketches.

The problems with the visual analysis of the state of affairs were obvious early in our work; it has even been mentioned on papers (BENEVIDES, A. B., 2010). Some ad-hoc solutions were applied (based on Alloy themes) but there is still much ground for improvement. A tutorial in ER conference in 2009, namely Daniel Moody's "The Physics of Notations: Evidence-based principles for designing cognitively effective visual notations" tutorial motivated us on working in this direction. The tutorial alerted us on the dangers of ill-designed diagrams and the benefits of well-designed ones. In his work, Daniel Moody presents a theory for the design of visual notations, which we consider in this work.(MOODY, D. L., 2010)

Our approach for improving cognitive effectiveness of information assessment goes through a great range of topics: from the design of the diagrams as a graphical artifact to subsequent perceiving, recognizing and understanding by the human mind. These topics are addressed in this chapter, which is further structured as follows: Section 4.2 discusses some models of human graphical information processing: the perceptual and cognitive processes humans engage in when looking at and seeking visual information; section 4.3 discusses diagrams and visual queries, i.e., how the diagrams may be designed to provide distinctive elements, to increase ease of use and to facilitate visual thinking; finally, section 4.4 discusses how the theory can be applied to OntoUML diagram design in particular.

4.2 The language of graphics and human graphical information processing

Moody prescribes that to produce more cognitively effective diagrams, one needs to understand two things: the language of graphics and human graphical information processing (MOODY, D., 2007). The language of graphics refers to the "techniques available for encoding information graphically" (IBID). In "the semiology of graphics" (BERTIN, 1983), a seminal work in the field of Graphical communication, the author Jacques Bertin identified eight elementary *visual variables* (MOODY, D. L., 2010) which can be used to encode information (Figure 4-1). These can be used in the design and analysis of visual languages, as Moody prescribes. Bertin's original theory has sprouted in different directions; in (WARE, 2004) some of the original features are extended, along with neurological and psychological "human graphical information processing" research. In Ware's book, it becomes clear that the language of graphics is in fact very much intricate with human physiology i.e. to the functioning of human eye and brain e.g. some of Bertin's variables are justified in terms of eye nerve functioning. Therefore the line between the concepts of “language of graphics” and “human graphical information processing” is thin, but still, the first refers to the graphical, physical representations of information while the latter refers to the response of the brain to the visual stimuli.

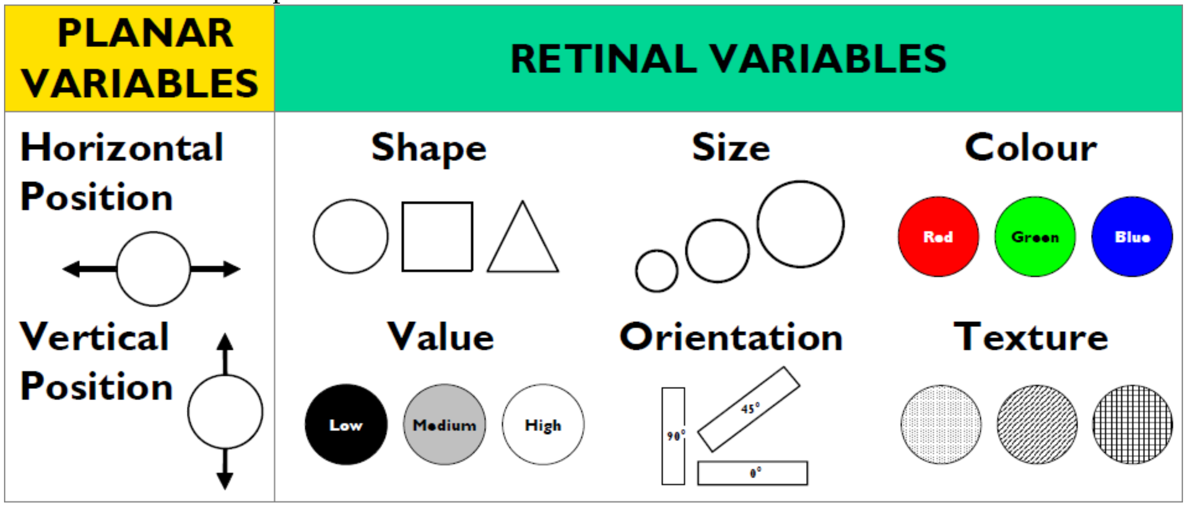


Figure 4-1: Bertin’s visual variables. Extracted from (MOODY, D. L., 2010)

Human graphical information processing is very complex and there are many different theories about how it works. Nevertheless, the perceptual features of human vision are one of the most well understood parts of the brain. Here we use two different but similar models. Ware’s model (Figure 4-2) describes the human graphical information processes in three stages: parallel processing to extract low-level properties of the visual scene, pattern perception and sequential goal-directed processing. Moody’s model (Figure 4-3) describes the human graphical information process using roughly five major stages: perceptual discrimination, perceptual configuration, attention, working memory and long term memory. We explain the two models below, revealing their many similarities.

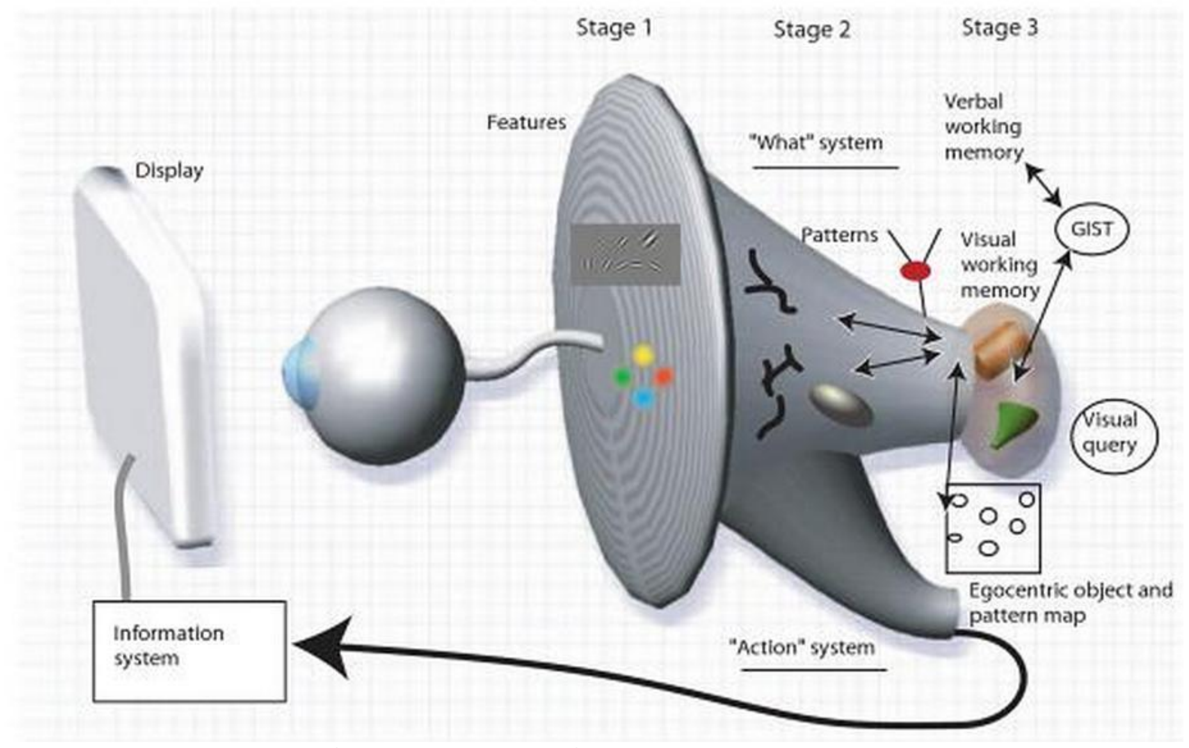


Figure 4-2 The three stages of human graphical information processing (WARE, 2004)

The two models are very similar in their first two stages: Ware's parallel processing to extract low-level properties of the visual scene and Moody's perceptual discrimination both refer to the rapid extraction of features (e.g. orientation, color, texture and movement patterns) of the retinal image. "Visual information is first processed by large arrays of neurons in the eye and in the primary visual cortex at the back of the brain. Individual neurons are selectively tuned to certain kinds of information, such as the orientation of edges or the color of a patch of light."(WARE, 2004) These arrays of neurons act in parallel to rapidly extract information of the visual field preattentively i.e. prior to conscious attention, "independently of what we choose to attend to (although not of where we look)"(WARE, 2004).

The second stage (Ware's pattern perception and Moody's perceptual configuration) both refer to the usage of the data extracted in the first stage. In the second stage the mind separates the visual field into areas of similar color or texture and combines features to generate perceptual units based on their visual characteristics; Ware also calls this stage the "what" system (Figure 4-2). The *Gestalt laws* define how visual stimuli are grouped together.(MOODY, D. L., 2010)

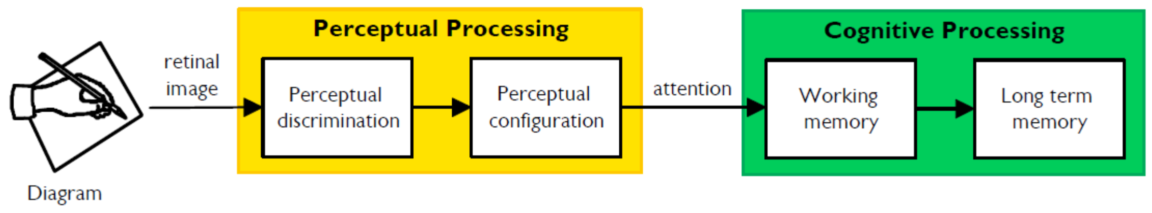


Figure 4-3: The five stages of human graphical information processing according to (MOODY, D. L., 2010)

In the third stage, the two models start to show their differences. While Moody separates acts of attention in a different stage (see Figure 4-3) as a sort of bridge between perceptual processing (seeing) and cognitive processing (understanding), Ware considers the second stage to be both perceptual and cognitive: “The pattern-finding stage of visual processing is extremely flexible, influenced both by the massive amount of information available from Stage 1 parallel processing and by the top-down action of attention driven by visual queries.” Ware also states explicitly that the second stage involves use of both working memory and long-term memory, which are separated as a fourth and fifth stages in Moody’s model.

Despite the differences, both models describe the remaining stages of graphical information processing as attention-centered and bottlenecked by working memory limits. It is agreed that to be optimized, diagrams should be designed to accommodate memory limitations. However, one key difference between the two models should be emphasized: Ware’s third stage offers richer explanations for interacting with visualizations as it is also concerned with user reaction such as eye movements to visual query the diagram. This directs the diagram designer’s attention towards the use of the diagram, its very purpose.

4.3 Diagram graphical design and visual queries

A diagram is a symbolic representation of information according to some visualization technique. To design effective diagrams, we must understand what information has to be encoded in the diagrams and which visualization technique will be used i.e. how to present the information visually. The manner in which we present the information is of particular importance when we regard diagrams as cognitive externalizations, components that complement the human user in a problem-solving system. Our design is based on the aforementioned human graphical information processing, since “if we want people to understand information quickly, we should present it in such a way that it could easily be detected by these large, fast computational systems in the brain” (WARE, 2004).

In OntoUML model assessment, the diagrams produced display possible sequences of states of affairs, instances of the model. The sequences reveal object creation, classification and destruction. In this sense, our design should favor understanding a simulated world structure on the basis that one cannot validate a world structure before identifying its contents; we intend to offer diagrams that are easier and faster to interpret.

When considering diagrams as problem solving artifacts, we must think of the problem we are trying to solve using such artifacts. Visual queries are activities conducted by the eye and brain to seek for information in the form of visual patterns. The effort to retrieve the information is a good indicator of the usability of a diagram; good diagrams make their tasks easy to perform. For example, consider a person using a map (Figure 4-4) to plan a trip between Vitória and São Paulo. Visual queries are conducted first to find the two dots corresponding to the cities in the map and later considering the lines that connect the two dots. This query could be rather loose, for example: minimize driving time and maybe find some interesting cities along the way. This could involve conducting new queries to find cities (dots) along the way and establishing a connection to the knowledge the user has about that city. Realistic travelling is influenced by many other types of information such as which roads that are in good conditions, where are good places to stop to rest, traffic intense areas, natural disaster sites, toll prices, roads with high speed limits, etc.

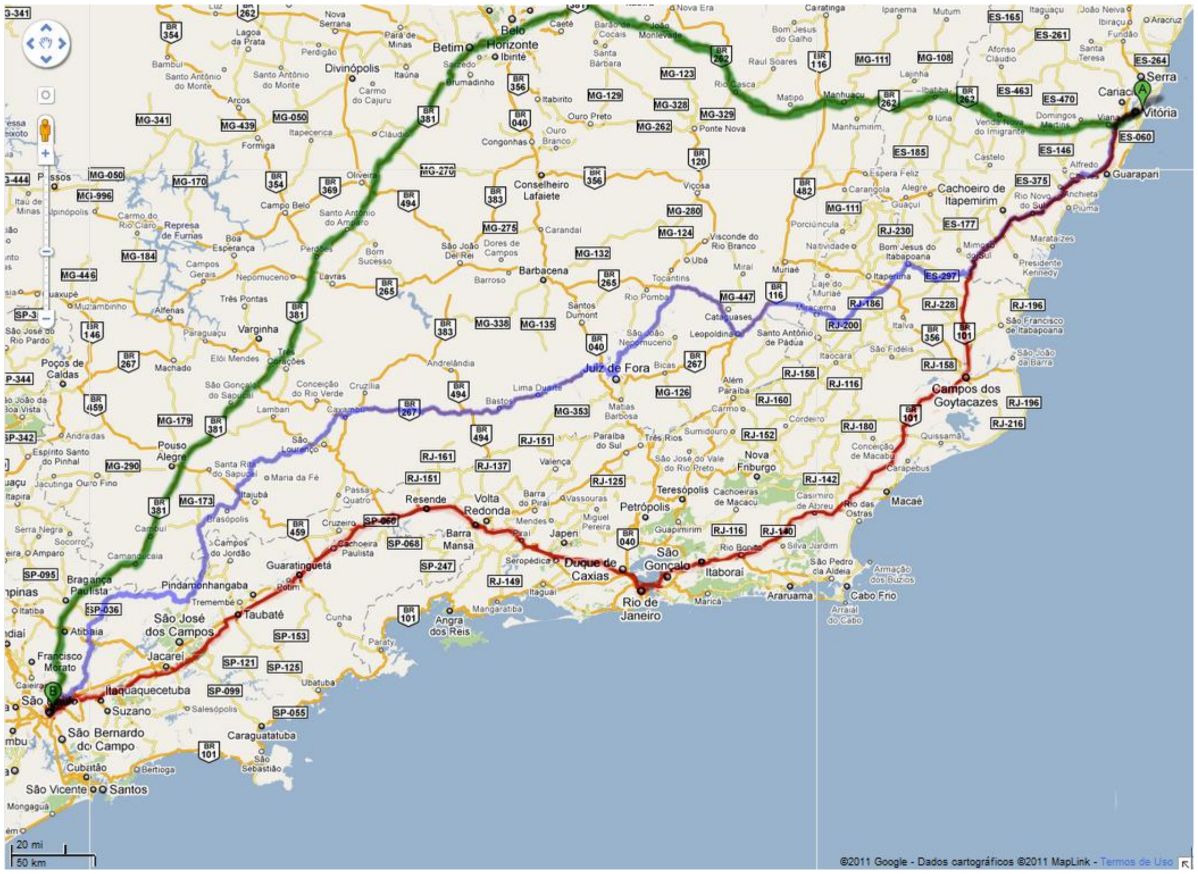


Figure 4-4 A roadmap showing paths between Vitória and São Paulo

The map in Figure 4-4 has its own purpose of abstracting geographical location and supporting travel planning. Other types of maps support other types of tasks; their visual characteristics are planned accordingly. In our case, the task is not as straightforward as finding a good route between two cities; the user will parse through the information available (entities, relationships and their classification) to assess whether there is "something wrong" with the state-of-affairs or if it conveys the expected.

The user facing our instance diagrams should be able to assess: (i) which entities exist, (ii) their classification, (iii) the relationships between them, (iv) the type of relationship and (v) their persisting identity through sequences of snapshots. Ultimately, we believe this will lead to understanding of what is formally possible according to the model, in the context provided by the diagram.

Figure 4-5 may be used to illustrate the tasks conducted by a user in understanding an instance diagram. Each token (symbol instance) on the diagram represents an individual instance of the conceptual model (instance of a universal in the model); the connecting lines represent their relationships. The reference and the labels reveal the types each individual instantiates. In this case, the diagram depicts the state-of-affairs between artists (individual artists and their bands), as well as the albums they produced with different releases. The diagram reflects the rules enforced by the conceptual model; after understanding what the diagram means the user may assess if it represents a valid state-of-affairs. This particular diagram depicted in Figure 4-5 displays a context of Album authorship in the musicbrainz ontology; one may navigate the diagram and find artists that collaborated in bands, assess which is the most prolific author, if artists released solo work, how many releases a particular album has and so forth. It reveals some characteristics of the domain: some artists have no solo work, albums may have multiple releases. Additionally, all visible albums have at least one release; an indication that the relationship may be mandatory. Also, all bands have some members, but the small number of instances does not provide a hint on the multiplicity constraints for band membership.

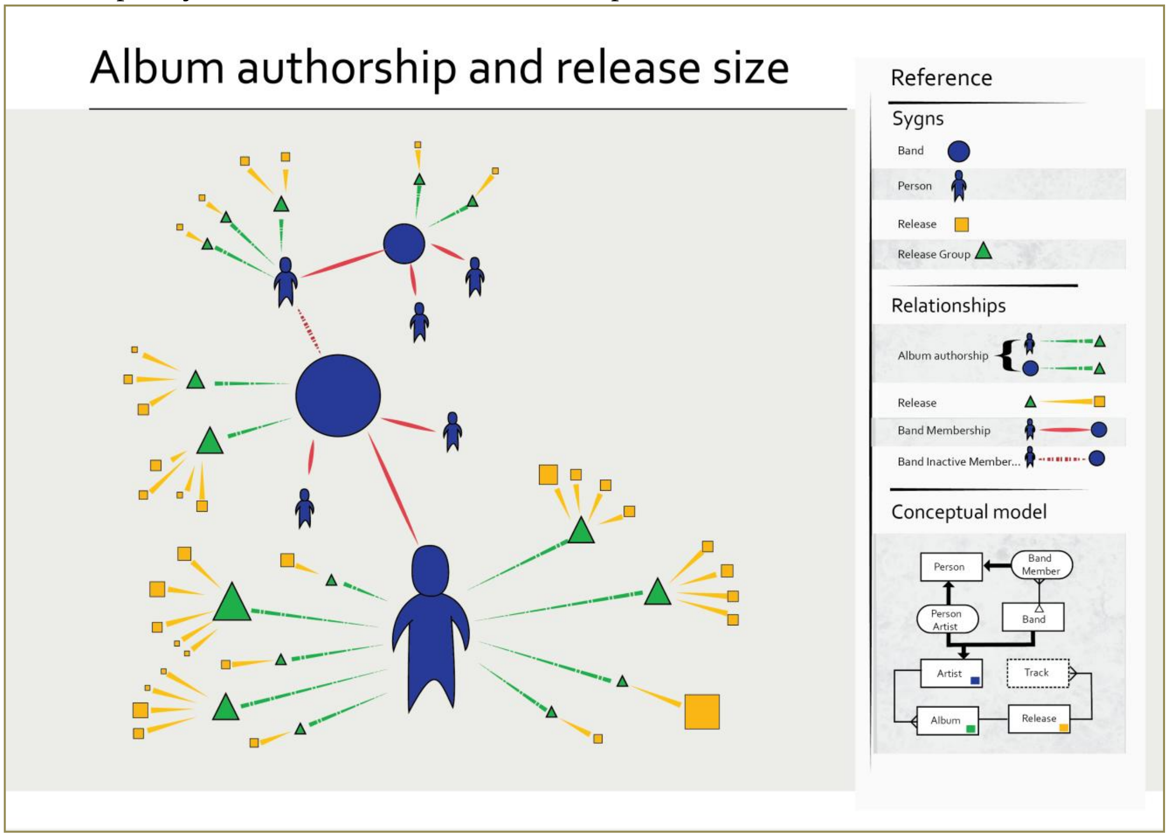


Figure 4-5 An instance diagram based on the MusicBrainz Ontology

But how does the brain generate a strategy for performing a query? First the entire field of view is processed in parallel, generating a feature map weighted adequately to the task. Next, fast eye movements are executed in sequence, visiting the possible targets according to the weighted feature map and the query algorithm. The query algorithm may be provided (like we did in the example above), generated creatively by the user or inadvertently discovered (perceived through experience).

Each query may be analyzed in terms of the time it takes to acquire information. In visual assessment, there are two types of eye movements: switching attention between objects within a region that can be perceived with no significant eye movement, which takes 50ms, and between objects that require a ballistic, saccadic eye movement to change the attention point, which takes 200ms.

In the following subsections we address OntoUML instance diagrams in particular and how we may design symbols that can be easily distinguished. These can be grouped together to form visual patterns which can be used to perform visual queries.

4.4 OntoUML instance diagram design

An instance diagram is a diagram that shows the structure of a modeled domain in a specific time. Instance diagrams are useful for exploring “real world” examples of a class diagram. The concrete syntax is similar to that of a node-link diagram: individuals are represented as nodes and relationships as arcs between the nodes In Chapter 3 we have discussed how instance diagrams may aid model validation efforts and here we apply visualization techniques improve their effectiveness.

OntoUML instance diagram design is a specific case of diagram design, meaning this section applies some of the theory presented previously. It is important to notice that the design decisions have collateral effects and often involve favoring the assessment of one type of information over the other. Consistently with our objective presented in Section 1.2, we advocate that our design facilitates model validation activities.

The section is structured as follows: Section 4.4.1 introduces atomic symbol design, while Section 4.4.2 discusses visual patterns that emerge when symbol tokens interact and provide an example. We also address the reasoning and inference made possible through the patterns.

4.4.1 Symbol design

Instance diagrams convey state-of-affairs about individuals and their relationships. Therefore, a central concept that surrounds instance diagrams is that of instance identity. But how do humans identify individuals? “There are a variety of criteria for individuation and tracing identity over time. In the case of ordinary physical objects, the most fundamental criteria are spatiotemporal. A single object cannot be in two places at the same time; and two objects cannot occupy the same space at the same time. Moreover, objects move on spatiotemporally continuous paths; if no continuous path exists between two appearances of what might be a single object or two distinct objects, we infer there must be two.” (XU; CAREY, 1996) Therefore individuals are identified by their *planar variables* (see Figure 4-1: Bertin’s visual variables. Extracted from (MOODY, D. L., 2010)) and these are stored in memory in an *egocentric spatial map* (WARE, 2004).

Spatial positioning is one of the main reasons why the Alloy Analyzer’s native visualization is unfit for our purposes as it does not account for keeping individuals in place between multiple snapshots. In that situation, to understand change between two states, one must first search serially for the individual by reading the nodes’ name labels, which is inefficient and error prone. Keeping individuals in the same position through all snapshots in a sequence allows us to take advantage of our spatial location memory. Research suggests “it may be possible to remember some information about approximately nine locations” (POSTMA et al., 1998 *apud* WARE, 2004) “Three of these may contain links to object files, whereas the remaining ones specify only that there is something at a particular region in space, but very little more.”(WARE, 2004) An o*bject file* is a temporary grouping of a collection of visual features together with the links to their meaning; these are stored in the egocentric spatial map. The egocentric spatial map is a type of memory that maintains information about the position of objects in the world, using an egocentric frame of reference. “The egocentric frame of reference is, roughly speaking, our subjective view of the world.” (WARE, 2004, p. 333). By keeping individuals in the same position on the diagram, we may perceive if a token appeared (was created), if it disappeared (was destroyed) or simply if it changed (e.g., instantiated or ceased to instantiate a class).

Each individual may instantiate many classes; we support visual coding in two modes: a node’s shape and a node’s decoration such as contour width, color, texture filling and attached symbols. Shape is reserved to indicate which Substance Sortal class the individual instantiates. Each individual instantiates one and only one Substance Sortal class and, likewise, individuals are represented in the instance diagram by a token which has one and only one shape. “Of all visual variables, shape plays a special role in discriminating between symbols as it represents the primary basis on which we classify objects in the real world” (MOODY, D. L., 2010). The token may be decorated, which indicates the other classes the individual instantiates. The idea that shapes should be assigned to rigid concepts and node decoration to anti-rigid concepts has been defended in (GUIZZARDI, 2005, p. 43)

The choice of which shape to use may be difficult. Iconic representations are occasionally interesting to use because of *perceptual directness*, i.e., when appearance suggests (correct) meaning (MOODY, D. L., 2010). An example of use of an icon to cause perceptual directness is provided in Figure 4-6, in which the human-like icon represents an individual Person. While in some cases iconic representations may work effectively to promote perceptual directness, icons are culturally learned and may cause more harm than good if they are *perceptually perverse*, i.e. if their appearance suggests different meaning (MOODY, D. L., 2010, p. 9). For example, using musical notes as icons to represent tracks could be *perceptually direct* for some users, but *perverse* to others, as it could suggest multiple different (music related) meanings, such as “song”, “musical recording”, “musical performance”, “note”, “music”, “album”, “release”, “tone”, “key” etc. Figure 4-6 shows an example; although the “author of” relationship may eliminate some possible interpretations (such as “note”, “tone” and “key”) it is still ambiguous as it could represent, to name a few, “song”, “musical recording”, “album” or “release”.

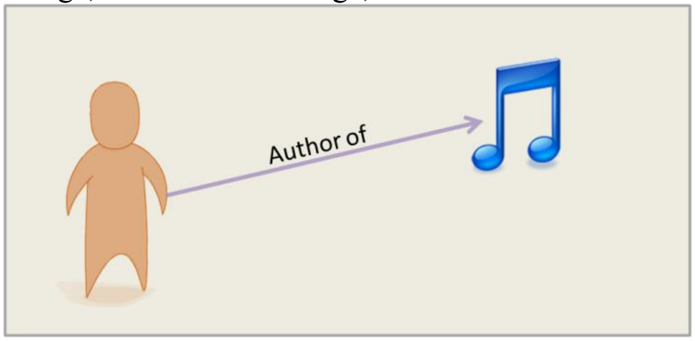


Figure 4-6 Iconic representations can be perceptually direct or perverse

Node decoration (such as color, texture or attached symbols) is used to visual code the secondary class instantiation; those that provide a principle of application, but not a principle of identity. “It is useful to think of color as an attribute of an object rather than as its primary characteristic.” (WARE, 2004) In that sense, differences in the coloring of two person tokens mean they are different types of persons, but persons nonetheless. If the coding is done properly, users may find individuals with specific shape-color combinations preattentively. For example, in Figure 4-7, it is possible to separate preattentively the Man and Woman tokens.

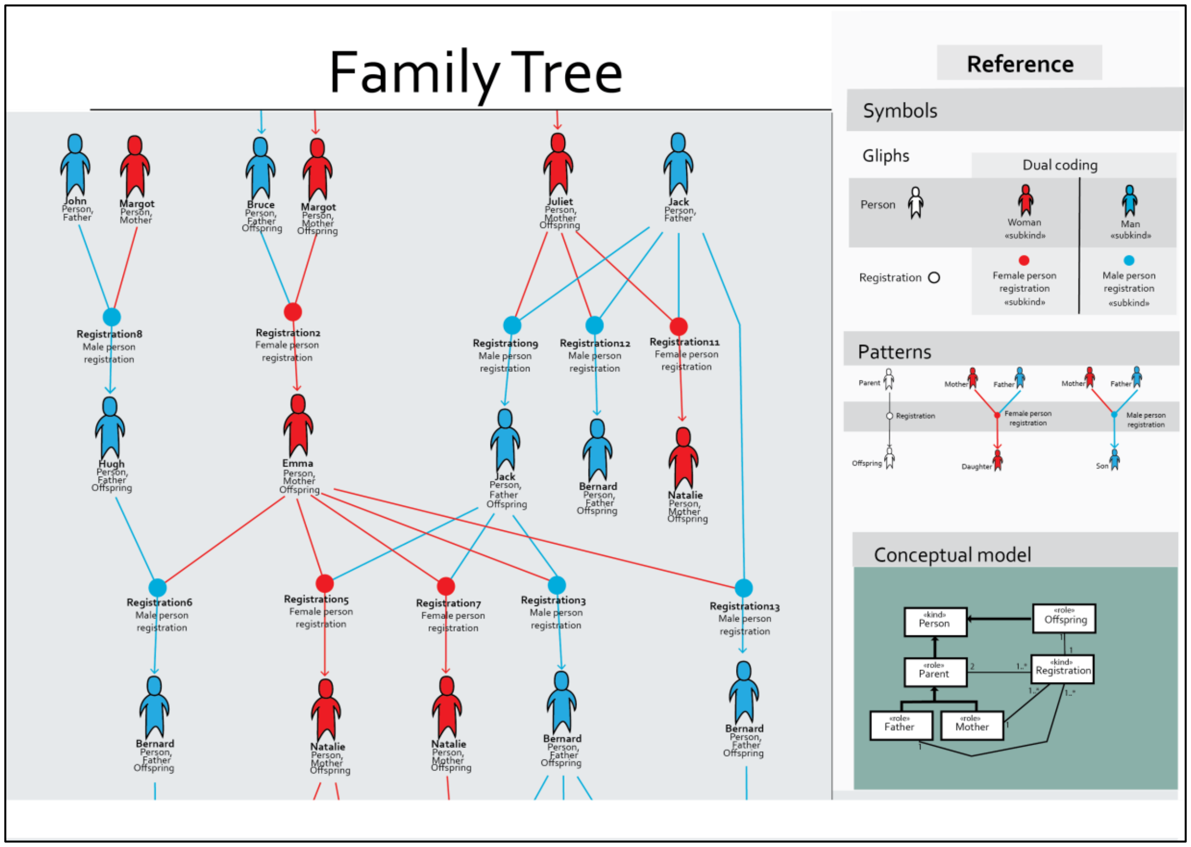

Family Tree
Reference
Symbols
Gliphs
Dual coding
Person
Woman «subkind»
Man «subkind»
Registration
Female person registration «subkind»
Male person registration «subkind»
Patterns
Conceptual model
«kind» Person
«role» Offspring
«role» Parent
«kind» Registration
«role» Father
«role» Mother
John Person, Father
Margot Person, Mother
Bruce Person, Father Offspring
Margot Person, Mother Offspring
Juliet Person, Mother Offspring
Jack Person, Father
Registration8 Male person registration
Registration2 Female person registration
Registration9 Male person registration
Registration12 Male person registration
Registration11 Female person registration
Hugh Person, Father Offspring
Emma Person, Mother Offspring
Jack Person, Father Offspring
Bernard Person, Father Offspring
Natalie Person, Mother Offspring
Registration6 Male person registration
Registration5 Female person registration
Registration7 Female person registration
Registration3 Male person registration
Registration13 Male person registration
Bernard Person, Father Offspring
Natalie Person, Mother Offspring
Natalie Person, Mother Offspring
Bernard Person, Father Offspring
Bernard Person, Father Offspring

Figure 4-7 In this diagram it is possible to distinguished preattentively Woman and Man tokens

Additionally, to imply conceptual similarity of the classes such as those in a *phase partition*, the set of assigned colors could imply such proximity. For instance, consider coding an ordered set of *phases*, such as Child, Teenager and Adult. To do so, one could choose a color for the set of *phases* and assign different values of saturation and lightness to each class, so one symbol seems lighter than the next. This situation is depicted in Figure 4-8. This technique is not restricted to phase partitions; alternatively, the color sequence could imply hierarchical differences for a chain of class specializations e.g. Parent and Grandparent.

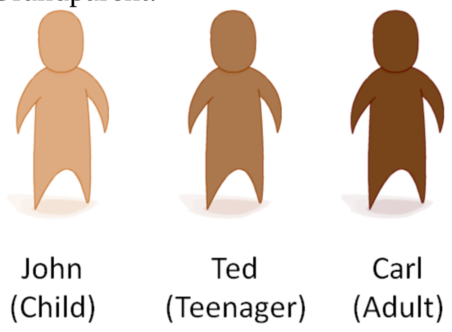


Figure 4-8 Differences in brightness can indicate conceptual similarity

The suggestion of similarity that color produces can be further explored to expose Mixin instantiation: individuals with different shapes sharing the same color imply there is something in common with them. This technique, however, may cause the creation of conjunctions and implies a trade-off in the type of query one wishes to benefit: either Sortal discriminability or Mixin similarity. Mixin classes are necessarily *abstract*, meaning the *concrete* Sortal class instantiation is sufficient to imply the Mixin instantiation; meaning the information can be deduced and it is not strictly necessary to color code Mixin instantiation. However, if the Mixin instantiation is relevant to the context and purpose of the diagram, the coding may be justified for its support in a type of visual query. For example, in Figure 4-5 Person and Band share the blue color to indicate a common *Mixin* class, namely the Author class. In this diagram, the Author *Role Mixin* is emphasized, since the purpose of the diagram is to visualize albums and their different releases. The design emphasizes the relevance of the similar role the tokens play in the diagram but still indicates their essential differences with the different shapes.

There are two major concerns regarding color coding: avoiding conjunctions and finding a set of colors that are distinct from one another but still keep the set aesthetically pleasant. Colin Ware recommends a list of 12 colors for use in coding: Red, Green, Yellow, Blue, Black, White, Pink, Cyan, Gray, Orange, Brown and Purple. The first six colors would normally be used before choosing any from the other set of six because they mark the ends of the opponent color axes. This palette should not be considered the only (or best) available option. Choosing the palette is a design choice that provides a set of non-functional qualities for the diagram. Variations in the color palette can favor a certain aesthetic purpose, or to support some type of assessment i.e. degrees in similarity and difference e.g. using opponent colors for drastically different types of entities or using slight variations in hue to imply type similarity.

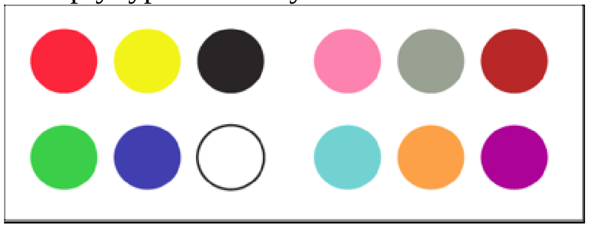


Figure 4-9 Palette for color coding

Inevitably, one color must be used for the background. We recommend using white, gray or black for the background color, to avoid color interference by contrast. Colors are perceived in their context and placing certain colors near others changes the way we perceive them. For example, Figure 4-10 depicts a color contrast illusion. The X's are the same color on both sides, but they seem bluer on a red background and pinkier on a blue background

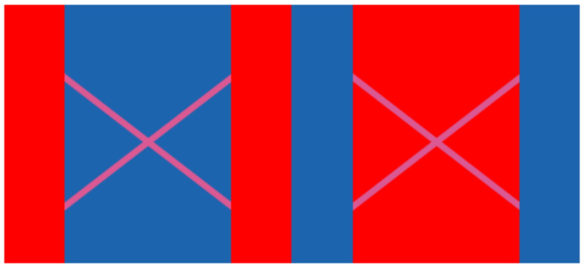


Figure 4-10 Color contrast illusion. The X's are the same color on both sides, but they seem bluer on a red background and pinkier on a blue background. Adapted from (WARE, 2004)

Finally, relationships should be subject to symbol design as well. Relationships are represented as connecting lines between two individuals. Being one-dimensional, a line cannot have a shape the same way a closed contour does; however, one may use different types of lines (e.g. dashed lines) to encode the type of relationship. Additionally the lines may be dual-coded by applying color and texture to it, the same way we described for individual symbol design.

Since the size of the class model is indeterminate (and may be very large), visual coding every possible class instantiation from the model hierarchy (in a single diagram) would not be fit to novices; it would increase drastically the complexity of our symbol set which would be inappropriate since novices have more difficulty discriminating between symbols, have to consciously remember their meaning and are more affected by visual noise. Moody's *Principle of Graphical Economy* advises that to deal with graphical complexity directly, we may *modularize* the whole into sets of diagrams and introduce *symbol deficit*. Modularization results in displaying a fragment of the information at a time. The modularization strategy will dictate what information can be assessed in each module, meaning modules can be constructed to a specific validation purpose and the corresponding diagram can be optimized by design to aid such validation purpose, for example, by employing adequate s*ymbol deficit*.

4.4.2 Pattern based reasoning and inference

Visual patterns allow the rapid assessment of complex information about the interaction of individuals in an instance diagram. The most elementary modes of retrieving information from the diagram (e.g. node identity; *kind* classification) have been introduced in the previous section, along with the design decisions for the symbol set and concrete syntax. We have advocated that a symbol set based on the aforementioned design principles support a number of visual queries: to find and identify specific individuals, to perceive their creation, change, and destruction, to find an individual with a specific class instantiation and to perceive similarities (as well as differences) in class instantiation of multiple individuals. These elementary modes of diagram reasoning cooperate to generate higher level search patterns (for details on expertise acquisition and higher level schemas, please refer to chapter 5).

Gestalt psychology describes organizing phenomena that occurs in visual perception. The word *gestalt* simply means *pattern* in German, we use Gestalt theory to describe visual patterns that occur in our instance diagrams and the interpretations we assign to them.

One of the key principles in the Gestalt system of perception is emergence. Emergence describes how visual stimuli are grouped together in a meaningful unity; the process of pattern formation from simpler elements. This principle is demonstrated in Figure 4-11, where the figure of a sniffing Dalmatian dog emerges in the mist of patches of black. In this case, the simpler elements are the patches of black, which are grouped together when the figure of the Dalmatian dog (the pattern) is perceived.

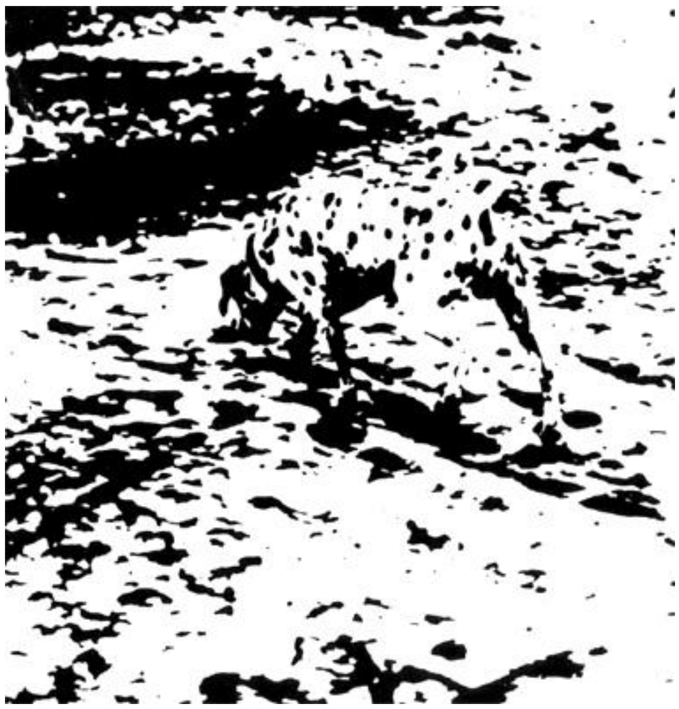


Figure 4-11 Emergence: A Dalmatian dog sniffing the ground in the shade of overhanging trees. (GREGORY, 1970)

For the purpose of analyzing our instance diagrams, we can use some “gestalt laws”, such as connectedness, similarity and proximity, to predict to how visual stimuli are grouped together.

- Connectedness: things that are connected by lines are perceptually grouped together
- Similarity: things that are similar are perceptually grouped together
- Proximity : things that are close together are perceptually grouped together

These principles can describe how multiple symbol tokens interact, forming meaningful units. Following, we detail some patterns of symbol tokens and the role they play either as search strategies to retrieve information or as cues that suggest otherwise unspecified properties of the model (such as overconstraining).

A material relationship becomes visible as a pattern of *substantial individuals* mediated by a *relator*. As mentioned in Chapter 2, material relationships are founded on relators. Figure 4-12 exemplifies the pattern, presenting a conceptual model, the corresponding instance diagram and legend. In the instance diagram, John is treated in MedUnit#1, MedUnit#2 and MedUnit#3. The objectified instance of this relationship (the relator individual) is identified by the name Treatment#223. Variations in line type indicate the different types of mediation relationships.

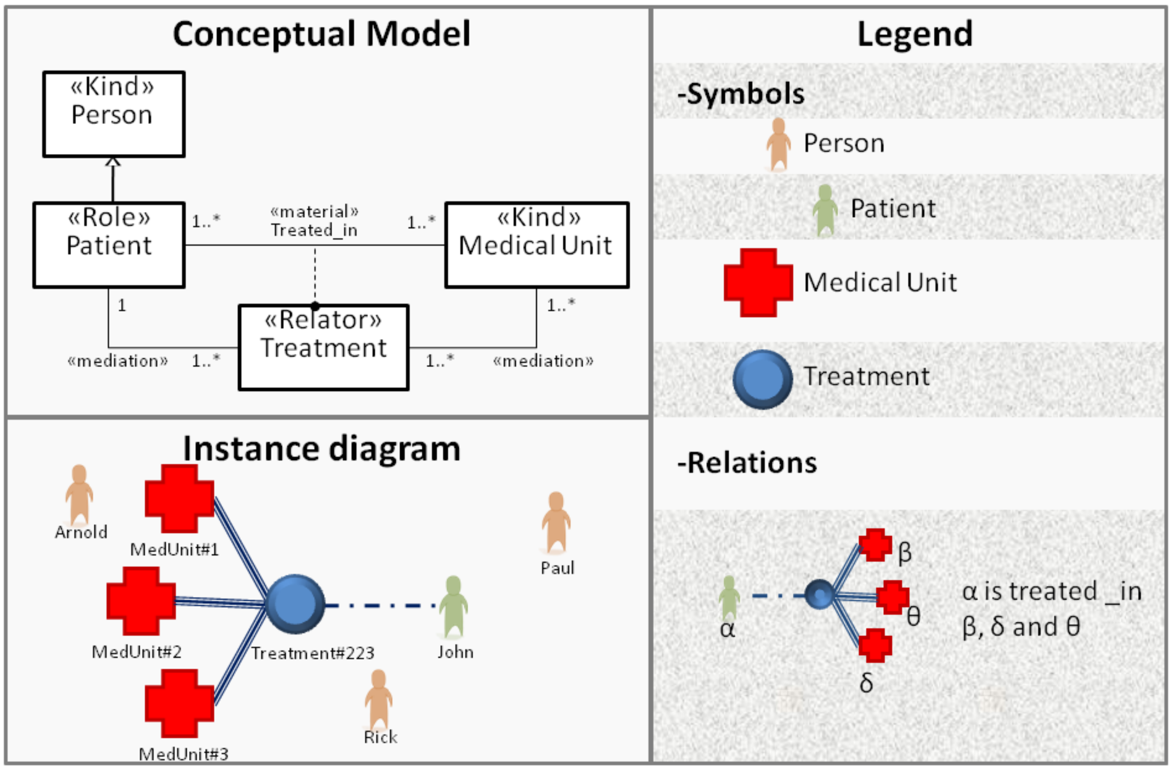


Figure 4-12 Medical Treatment material relationship example. Conceptual model extracted from (GUIZZARDI, 2005, p. 260)

The material relationship higher level pattern is constituted by the tokens decorated with the corresponding roles and connected by the line tokens of the corresponding mediation relationship. Figure 4-13 details the combination of tokens into higher level patterns. Figure 4-13 -A lists the atomic symbols which are used as components of the material relationship pattern. Figure 4-13 -B and C describe the first level patterns, formed with the possible combinations of the atomic components. Figure 4-13 -D and E describe the second level patterns, formed by combining multiple occurrences of first-level pattern B.

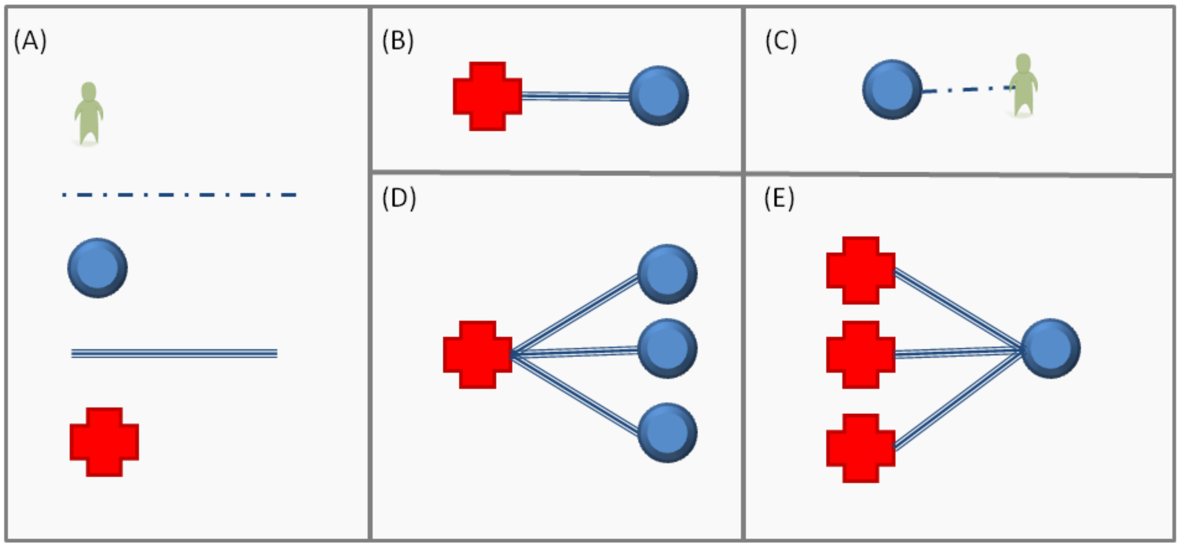


Figure 4-13 Components of the "Treated_in" material relationship pattern.

Examples of the complete material relationship pattern can be seen in Figure 4-14. In particular: Rick is treated in MedUnit#3, MedUnit#4 and MedUnit#5 and the visual pattern for this material relationship is a combination of patterns (C) and (E) from Figure 4-13. Visual queries may be conducted on this pattern e.g. is there a Medical Unit that has treated all patients? Which Medical Units have delivered more, less or an equal number of treatments compared to MedUnit#3? These questions or queries can be answered by performing visual queries based on the pattern; therefore it can serve as basis for retrieving information.

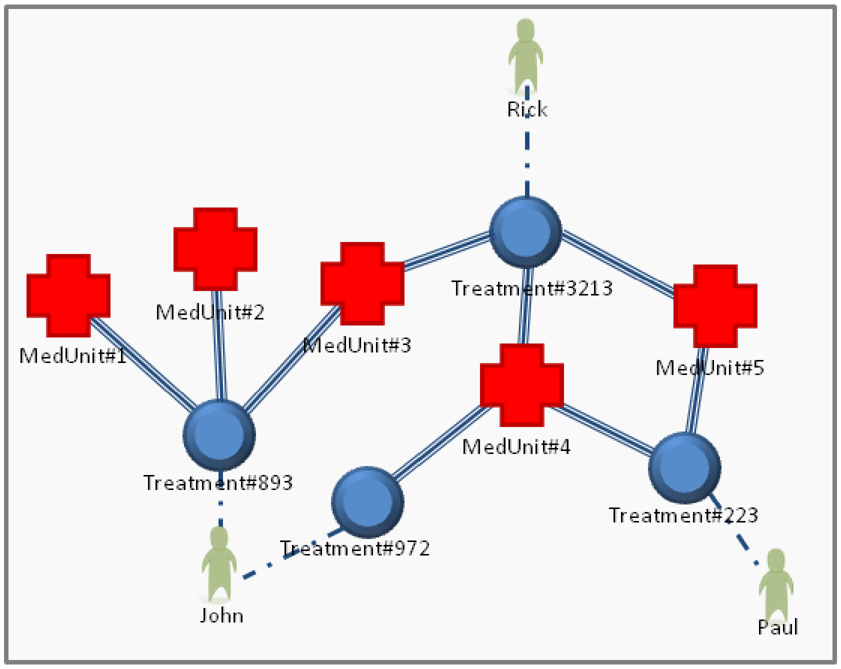


Figure 4-14 Multiple instances of the "Treated_in" material relationship

Additionally, the pattern interactions can serve as cues that suggest otherwise unspecified properties of the model. For example: there are no occurrences of patients which were treated in the same Medical Unit in different Treatments. Does this mean that the situation is prohibited? Formally speaking, this is not a sufficient condition; there could be some state-of-affairs in which the situation holds. In our approach it is possible to verify this by issuing a command to the Alloy Analyzer to generate state-of-affairs in which the situation holds. If the situation is permitted by the model, the Analyzer will successfully display a valid occurrence of the situation; otherwise it will deliver a warning stating that the situation is impossible according to the model. Regardless of the result, the user was cued towards investigating an aspect of the model not due to some visual arrangement, but due to the lack of such arrangement. A detailed discussion on *implicatures* and *pragmatics* is highly relevant to explaining how the user may be cued to otherwise unspecified characteristics of the model, but it is out of scope and we leave it to future work.

5 Improving cognitive processing of instance diagrams

5.1 Introduction

Cognitive processing is a stage of human graphical information processing where sensory input gathered by perceptual processing is selected by conscious acts of attention and stored in a limited working memory and combined with prior knowledge stored in long-term memory. Concerns for working memory limitations and long-term memory retrieving and storing of information are relevant in our efforts for improving model assessment since "almost any interesting task is too difficult to be done pure mentally. Information visualization enables mental operations with rapid access to large amounts of data outside the mind" (WARE, 2004).

Human memory can be a tricky subject to study and define. On one side it is capable of storing enormous amounts of information in long term memory, and on the other it is only capable of holding a small number of chunks of information at a time in short term memory. One of the most highly cited papers in psychology is an investigation on the limits of our short term memory: George Miller's "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information" (MILLER, 1956). Although there is nothing magical about the number (Miller used the expression only ironically), the paper is often cited as a justification for a number of simplifications in visual design, such as limiting PowerPoint slide's bullet points. The paper is often misinterpreted; contrary to the common generalizations, Miller's paper did not offer rules about the limits of human memory that could be applied in any situation. Instead, his paper reviewed psychological experiments on holding "unrelated pieces of really dull data at once (…) the deep point of Miller's paper is to suggest strategies, such as placing information within a context, that extend the reach of memory beyond tiny clumps of data." (TUFTE, 2000) Diagrams can support thinking by extending memory, for they can rapidly evoke information and cause it to be loaded from long-term memory into working memory (WARE, 2004).

In our studies of the relations between diagram and memory, we must consider the differences between experts and novices as it has been argued that diagrams cannot be optimized to suit both (MOODY, D. L., 2010, p. 16). As a consequence, one should choose who the intended users are prior to diagram development. Information that may be essential for novices may be distracting for experts and visual complexity may be overwhelming for novices but important for multi-tasking experts. Given our original objectives, we focus on the design of diagrams to favor usage by novices. Moody stresses the following key differences between diagram novices and experts:

- Novices have more difficulty discriminating between symbols;
- Novices are more affected by complexity;
- Novices have to consciously remember what symbols mean;
- Novices are more affected by visual noise; and,
- Novices lack strategies for processing diagrams.

Clearly, a diagram for the novice user must be carefully designed to account for these deficits. In this section we explore opportunities to extend working memory limits for novice users. We hope to improve the usability of the diagrams with a redesign that reduces demands on prior knowledge such as knowledge on the language, notation or domain. Novices must keep this knowledge in working memory, occupying precious limited resources. Gradual, contextualized exposure to information also benefits long-term learning, contributing further to reducing working memory demands (MAYER; MORENO, 2003).

5.2 Working memory and diagram complexity management

According to Cognitive Load Theory, reducing the demands on working memory improves speed and accuracy of understanding and facilitates cognitive processing of information, a prerequisite for meaningful learning (SWELLER et al., 1998). Our approach to reduce demands on working memory involves (but is not limited to) managing complexity. We do this by dividing the conceptual models in "packages", i.e. "contexts" or "modules" that can be dealt with separately. In fact, "complexity management mechanisms should be essential components of all SE notations" (MOODY, D. L., 2010). -Dealing with a fraction of the model at a time effectively reduces the volume of prior knowledge needed simultaneously.

The OntoUML instance diagram user faces two types of complexity: the inherent complexity of the conceptual distinctions underlying OntoUML and an arbitrary complexity that depends on the complexity of the domain (universe of discourse). These complexities are of a different nature: the aforementioned distinctions are domain-independent and finite and may be learned fully, while the possible domains are infinite. Visual complexity for the symbols depends both on the size of the model (domain complexity) and on the type of constructs used to compose such model (inherent complexity).

Our modularization strategy is largely based on domain-independent distinctions and patterns that emerge naturally in the use of OntoUML as a conceptual modeling language. General purpose rules are defined for modularizing instance diagrams, based on such distinctions. For example, Figure 5-1 is an instance diagram for a module of the MusicBrainz ontology (explained in detail in Section 6) and displays authorship relationships between author and releases. Apart from the obvious reduction in demands of domain knowledge (compared to displaying all the available information); this example only uses a fraction of the OntoUML language: the concepts used are *kinds, role mixin* and *material relationship*. This consistently deals with both types of complexity simultaneously: each module represents only a fragment of the domain, using a fragment of the modeling language. The instance diagram is also less visually complex as a consequence.

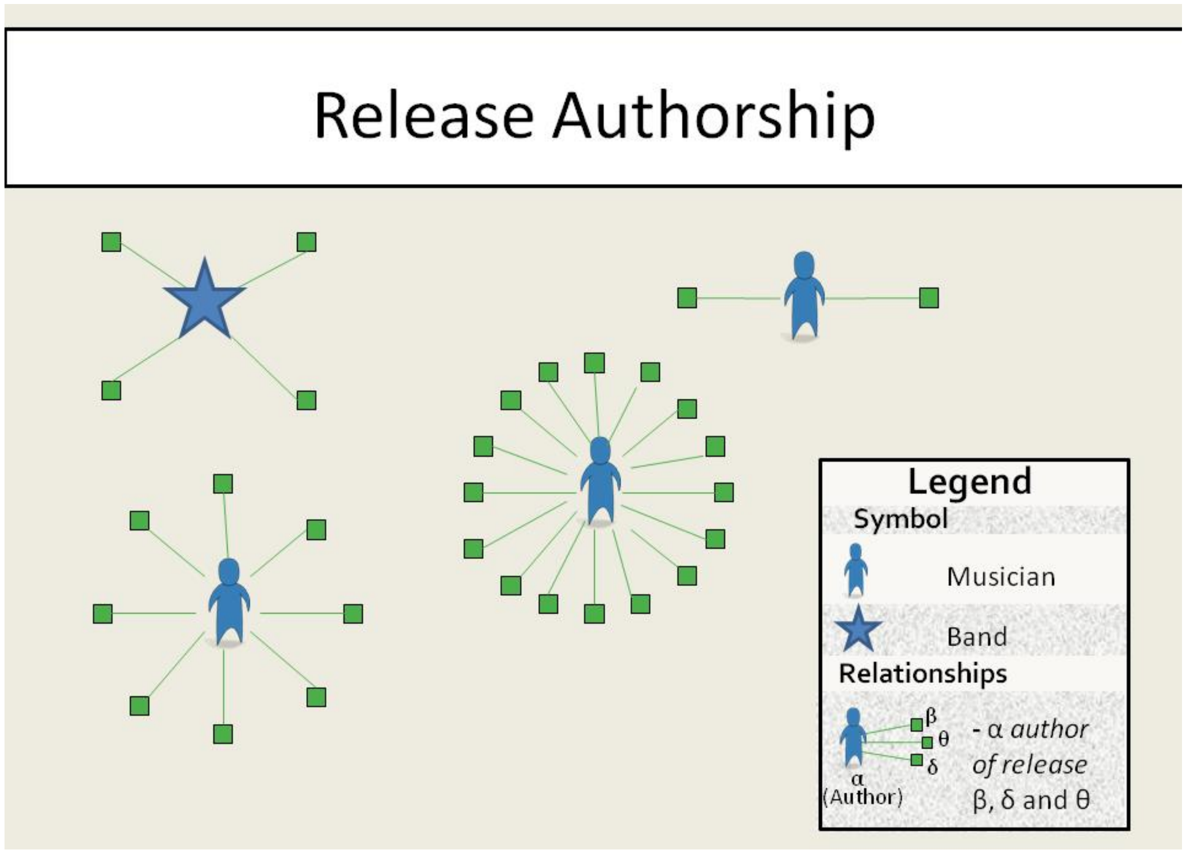


Figure 5-1 The instance diagram reveals but a fraction of the domain

Our general purpose rules for modularizing the class model (and therefore the instance diagram) are based on the conceptual distinctions underlying OntoUML. Here, we group these distinctions into three types of information: relational information, individual intrinsic information and subsystem information. The table below divides OntoUML conceptual distinctions into the three types of information described. The relational aspects level covers relationships between individuals and the patterns of relations that emerge. Figure 5-2 shows an example of John and Paul and John's bicycle, John being older than Paul and owning John's bicycle. The individual external aspects level exposes individual information: their characterizing moments and *phase* classification. Figure 5-3 exemplifies this, displaying moments (such as John's headache and John's bicycle color) and phase classification (John is dead and Paul is alive). The individual internal aspects level is a breakdown of the individual: wholes are divided into their parts. (i.e. a system is divided into subsystems). Figure 5-4 exemplifies this aspect, displaying John and Paul's heart and brain, and the bicycle's crank; the parts of the individuals.

Relational information	Intrinsic information	Subsystem information
-Role & role mixin classification -Formal relationships -Material relationships -Intransitive part-whole relations (*memberOf* and intransitive *componentOf*)	-Phase classification -Modes, qualities and characterization relations -Rigid classification (*kind, subkind, category*)	-Transitive part-whole relations (*subCollectionOf, subQuantityOf* and transitive *componentOf*) -Constitution relations

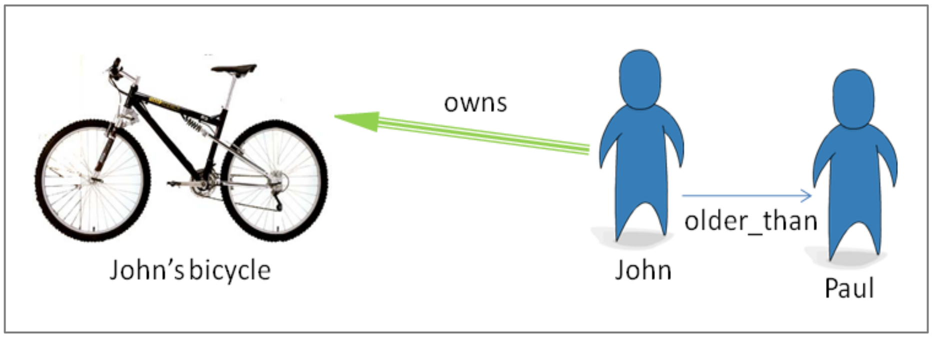


Figure 5-2 An example instance diagram focusing on relational information. The material relation "owns" is represented as a single arrow, omitting the relator that mediates the relationship.

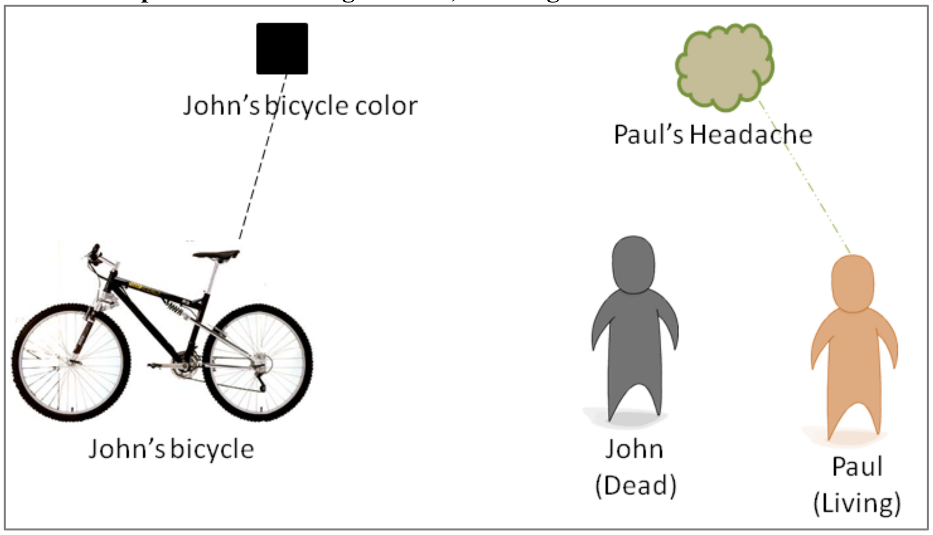


Figure 5-3 An example instance diagram focusing on intrinsic information

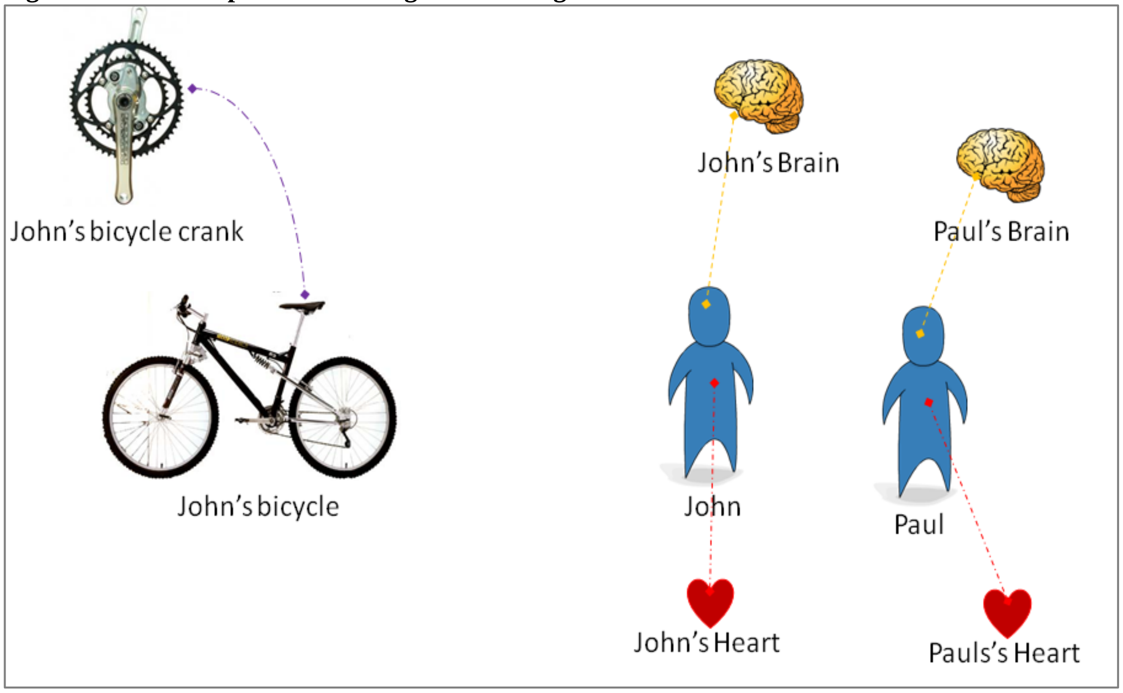


Figure 5-4 An example instance diagram focusing on subsystem information.

A given diagram module may focus on any of these granularity levels or on a combination of these, with different trade-off effects between complexity and expressive power. For example Figure 5-5 combines the subsystem and intrinsic information; while Figure 5-6 combines all of the types. Notice how these last two figures are more expressive and complex than the last three.

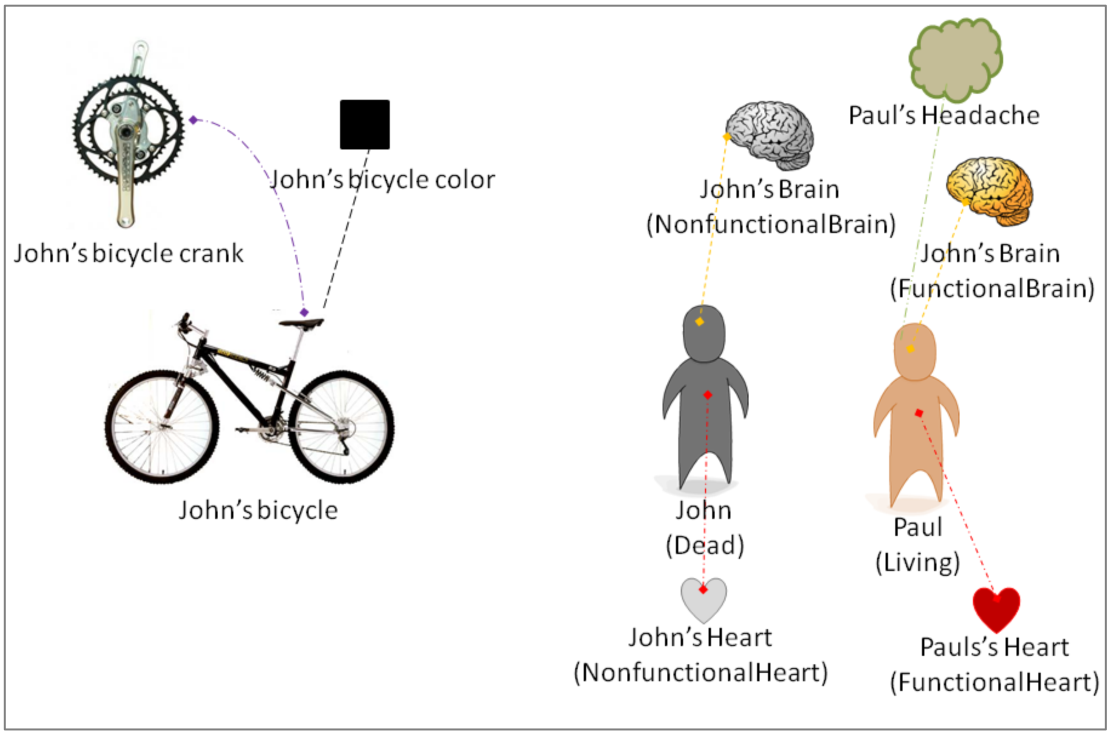


Figure 5-5 Combining two different types of information focus: subsystems and intrinsic information

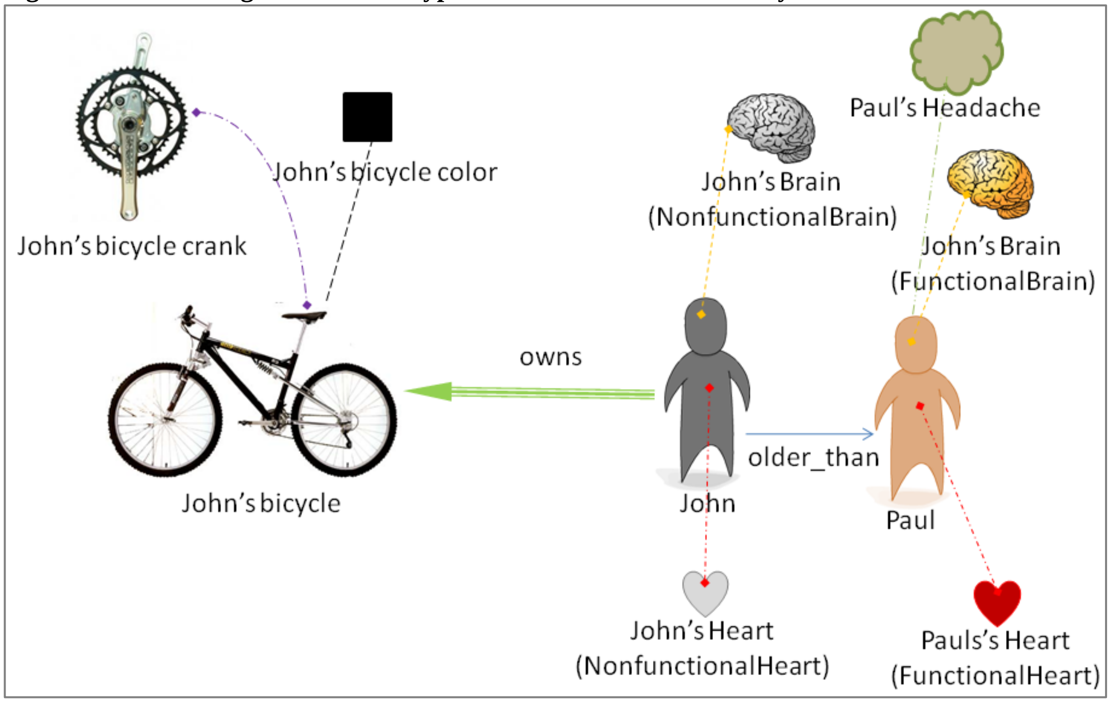


Figure 5-6 All three types of information combined

Additionally, diagram modules can focus on displaying information for multiple individuals or simply about a single individual, either way, one may still divide information using these three types of information. Exhibiting information about a single individual may involve a different type of design: all the information is **necessarily** about a single node, so the links don't offer much information in this case and the information may be organized differently. These may be interesting for exploring the possibilities in classification for a single kind of individual.

Additionally, the modularization strategy attends to another previously mentioned problem, that "Novices lack strategies for processing diagrams" (MOODY, D. L., 2010). By dividing the model in contexts, we can effectively lead the attention of the user to focus on a single relevant aspect of the model at a time. Once such aspect has been exhausted, the user may move on to specialized aspects of the same concepts or move on to a different part of the diagram. Besides reducing demands on working memory, this strategy promotes meaningful learning of domain concepts and support expertise development in validating OntoUML models, allowing one to deal with larger parts of the diagram at a time. In the next section we discuss long term memory and learning, and the role our modularization strategy plays in supporting this.

5.3 Long term memory and learning

Long term memory (as opposed short-term memory) "is the information that we retain from everyday experience, perhaps for a lifetime." (WARE, 2004) Humans retain information in many different forms (e.g. visual, auditory, verbal etc..) and what we experience as long term memory is usually a combination of these different types of memory.

The study of long term memory is relevant in our investigations seeing that prior knowledge holds a powerful effect on working memory limitations. "Humans are particularly poor at complex reasoning unless most of the elements with which we reason have previously been stored in long-term memory. Working memory simply is incapable of highly complex interactions using novel (i.e. not previously stored in long-term memory) elements." (SWELLER et al., 1998, p. 254) Therefore, diagram design that considers long term memory and learning facilitates the use and reasoning done with the diagram.

Educational psychology offers some strategies and models to stimulate learning and skill development. One such model involves perceiving knowledge in the form of mental schemas: "A [mental] schema categorizes elements of information according to the manner in which they will be used (…). Thus, chess grand masters have schemas that categorize board pieces into patterns that tell them which moves are appropriate. (…) According to schema theory, it is through the building of increasing numbers of ever more complex schemas by combining elements consisting of lower level schemas into higher level schemas that skilled performance develops" (SWELLER et al., 1998). Schemas hold as much space in working memory as a single element (or "chunk", in Miller's terms) would, so even though the concept of a restaurant, for example, is extremely elaborate, it poses little effort to the memory of those familiar with it. A clear example of the elaboration and automation of such mental schemas is development of reading skill: "In early school years, children construct schemas for letters that allow them to classify an infinite variety of shapes (as occurs in hand writing) into a very limited number of categories. These schemas provide the elements for higher order schemas when they are combined into words that in turn can be combined into phrases, and so forth. Ultimately, this process allows readers to rapidly scan a page filled with a hugely complex array of squiggles and derive meaning from it." (SWELLER et al., 1998)

To support meaningful learning, we must offer lower level schemas and gradually build higher level schemas, but what characterizes a schema as low or high level? "The elements of low element interactivity material interact minimally and so can be learned serially without imposing a heavy working memory load (…) Material that is high in element interactivity is hard to understand because understanding requires working memory to process many interacting elements simultaneously, rather than serially. Understanding may only occur fully once the interacting elements have been incorporated into a higher-order schema that can be held more easily in working memory." (SWELLER et al., 1998, p. 290) It appears that our working memory concerns will operate in synergy with our learning concerns: reducing cognitive load facilitates learning, which in turn reduces cognitive load.

Learning can be supported by gradually introducing domain-specific, modeling language and notational knowledge as lower level schemas to allow a smooth construction of higher level schemas. In the case of instance diagrams, individuals are an example of a lower level schema. Understanding the concept of an individual is (concerning working memory loads) very straightforward and atomic. From that starting point, other concepts may be combined into higher level schemas and learned. The *material relationship pattern* presented in Section 4.4.2 is an example of a higher level schema: a given set of symbol tokens combined offer a wider meaning but its understanding requires understanding the lower level elements that constitute it.

It is useful to think of diagram visual queries as mental schemas that may be learned and stored in long term memory; constituting a state of expertise in the use of such query. As expertise is acquired, the memory load to execute the visual queries reduces and working memory may be used to engage in other types of reasoning. This is especially relevant if the user is solving a problem: as previously stated, automated schemas in long term memory significantly reduce working memory loads. Some of these schemas can be perceived visually, once the user is skilled. In that case, the visual recognition triggers the mental schema: visual images can activate memory in as little as 100 msec (WARE, 2004). Once the user is skilled in the visual patterns, they can be perceived, assessed and held in memory in atomic chunks; in the same way letters are chunked together into words by the skilled reader. In the previous section we introduced the term "object file", sometimes used to describe a kind of summary of the properties of an object or a scene that is held in working memory. The long-term memory counterpart of an object file is called "gist". "Gist is used mainly to refer to the properties that are pulled from long-term memory as the image is recognized."(WARE, 2004)

Once the user is skilled in using the visual queries, the schema is said to be automated: the conclusions can be "seen" i.e. what otherwise could be reached with reasoning becomes *perceptive* in the visual scene: "At higher levels of [visual] processing, perception and cognition are closely interrelated, which is why the words *understanding* and *seeing* are synonymous". (WARE, 2004)

To aid in the construction of low level schemas, the starting point for meaningful learning, we may apply a technique for reducing cognitive load known as *pretraining* (MAYER; MORENO, 2003). "Constructing a mental model involves two steps—building component models (i.e., representations of how each component works) and building a causal model (i.e., a representation of how a change in one part of the system causes a change in another part, etc.)” (IBID) Concepts which can be defined on their own (don’t depend on the definition other concepts) would constitute a component model; understanding the relationships between them and the consequences of these would be the causal model. Naturally, domain specific characteristics define different degrees of element interactivity.

The next chapter could be regarded as an example of pretraining. In it, we first describe domain concepts that are most elementary (lower level schemas) and gradually introduce the concepts that build upon them (higher level schemas). As the domain is explained, the symbol set is also introduced, providing examples that the audience is probably familiar with.

6 Example of application on designing instance diagrams

6.1 Introduction

Our prototype model simulation is based on the MusicBrainz project; a free database of information about music artists, their recorded works and the relationships between them. Recorded works capture, generally, the album title, track tiles and the length of the tracks.(METABRAINZ, 2011a) The data is supplied by the community either through a web interface or through Picard Music Tagger software. “The Tagger will attempt to automatically identify your files and walk you through the process of matching up the unidentified files” (METABRAINZ, 2011b). The user input may be validated by volunteer editors that try to spot mistakes, duplicates or out-of-style data.

Figure 6-1 The MusicBrainz logo

The musicbrainz.org community offers extensive documentation on the complex concepts around the MusicBrainz metadata. These concepts are sometimes very abstract and hard to define and as the metadata proposal expanded, philosophical issues raised concerning the concepts. In the next section we introduce the MusicBrainz data terminology and exemplify instance diagram design. We use the site’s documentation to generate a conceptual model that represents it. Figure 6-2 shows the resulting OntoUML conceptual model. It provides some specialized terms that do not exist in the MusicBrainz terminology; nevertheless they are ontologically well founded concepts that emerge because of OntoUML’s formal rules. The names of the Universals in the conceptual model that are part of the MusicBrainz terminology are marked in the text using italics; MusicBrainz terminology which is not present in the conceptual model is not marked. Visual coding for the instance diagrams is available in Figure 6-3.

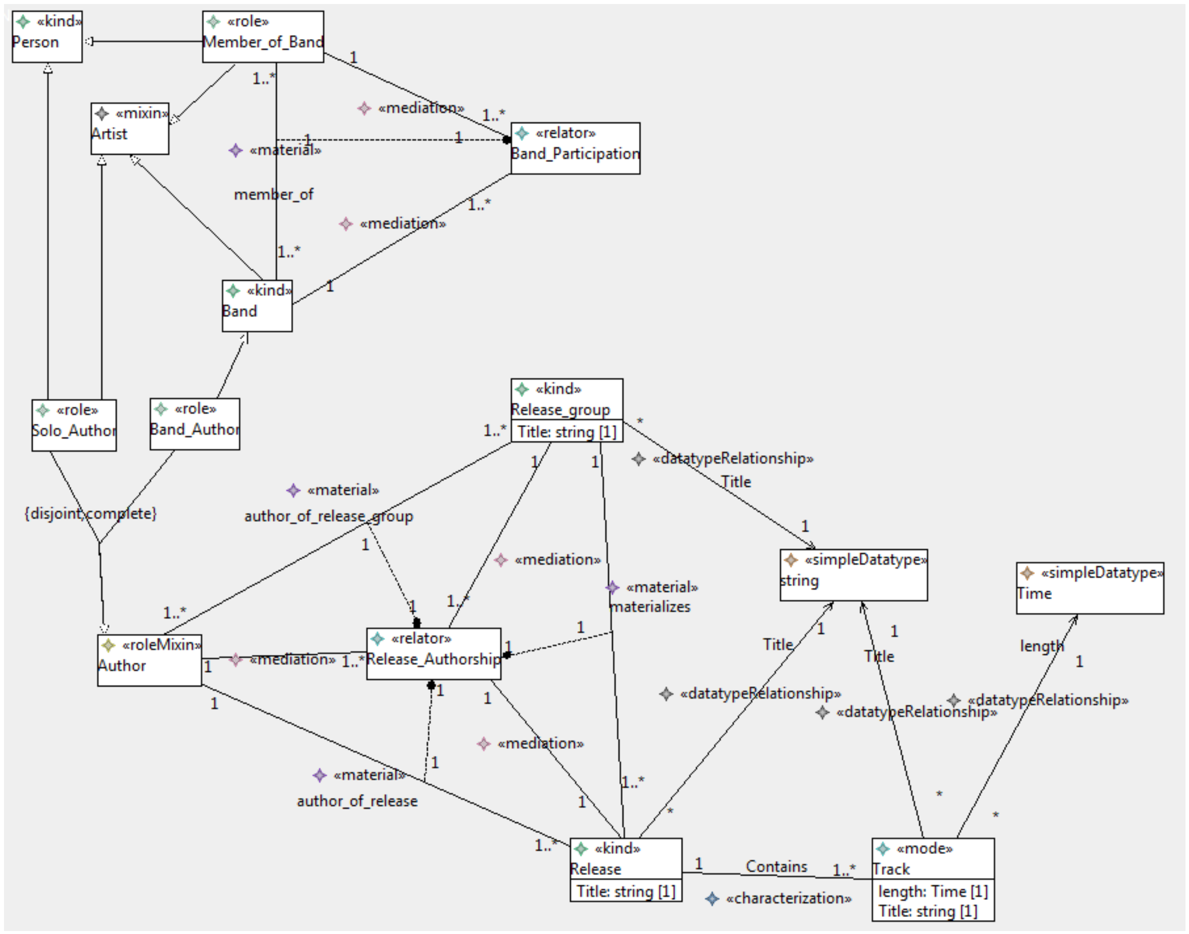


Figure 6-2MusicBrainz conceptual model

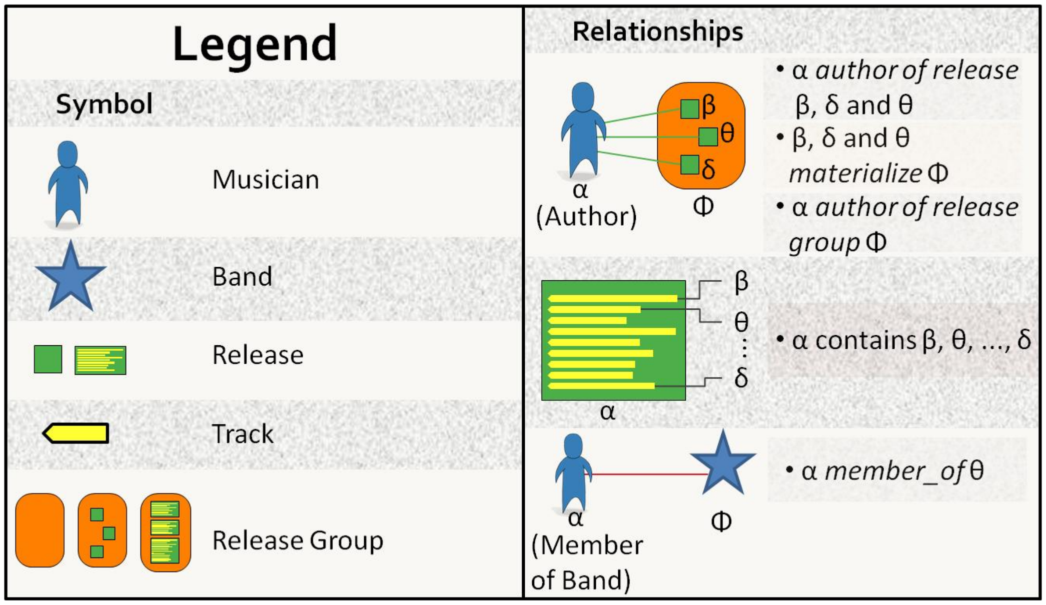

Legend
Symbol
Musician
Band
Release
Track
Release Group
Relationships
• α author of release β, δ and θ
• β, δ and θ materialize Φ
• α author of release group Φ
α (Author)
Φ
β
θ
δ
α
• α contains β, θ, ..., δ
• α member_of θ
α (Member of Band)
Φ

Figure 6-3 Legend for the MusicBrainz instance diagrams

Color coding was done with aid from the “Colorbrewer” (available at http://colorbrewer2.org/). Color specifications in RGB are available in Figure 6-4.

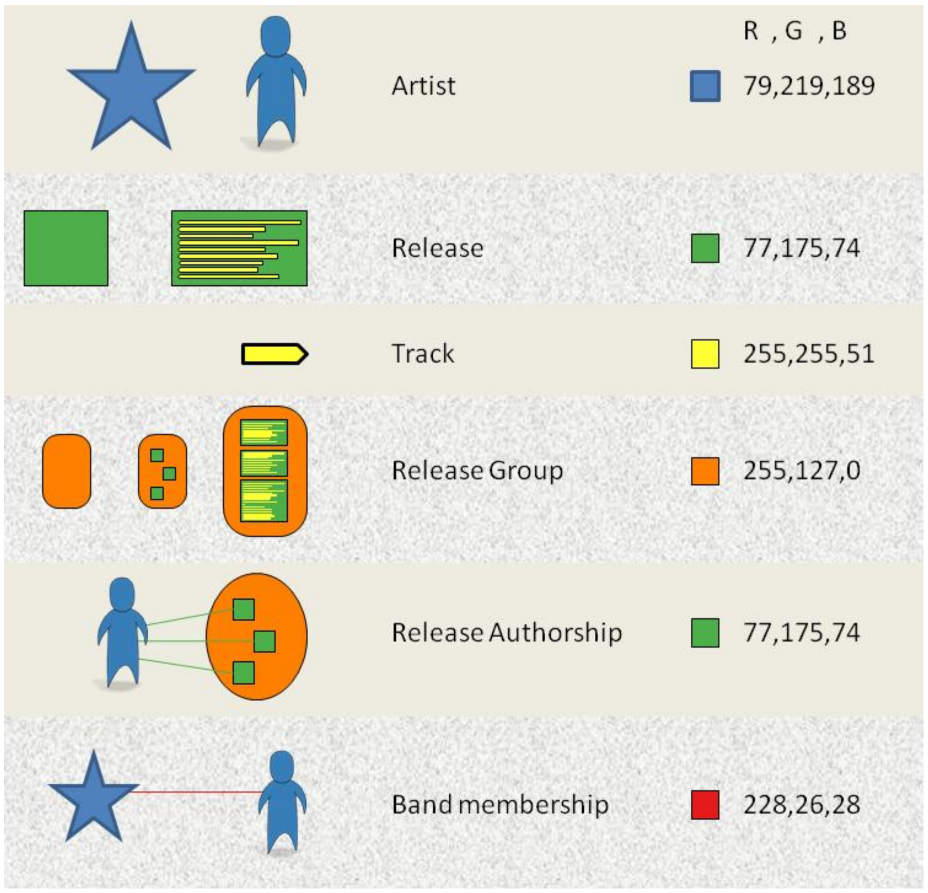


Figure 6-4 Color assignment in RGB to symbols

6.2 Domain concepts

MusicBrainz collects data about recorded music and the people involved, which are called *Artists*. An *Artist* is either a *Person* or a group of people who participate in the realization of a *Release* (a unique set of recorded *Tracks*) (METABRAINZ, 2011a). For example, Figure 6-5 exemplifies both Bands and Musicians.

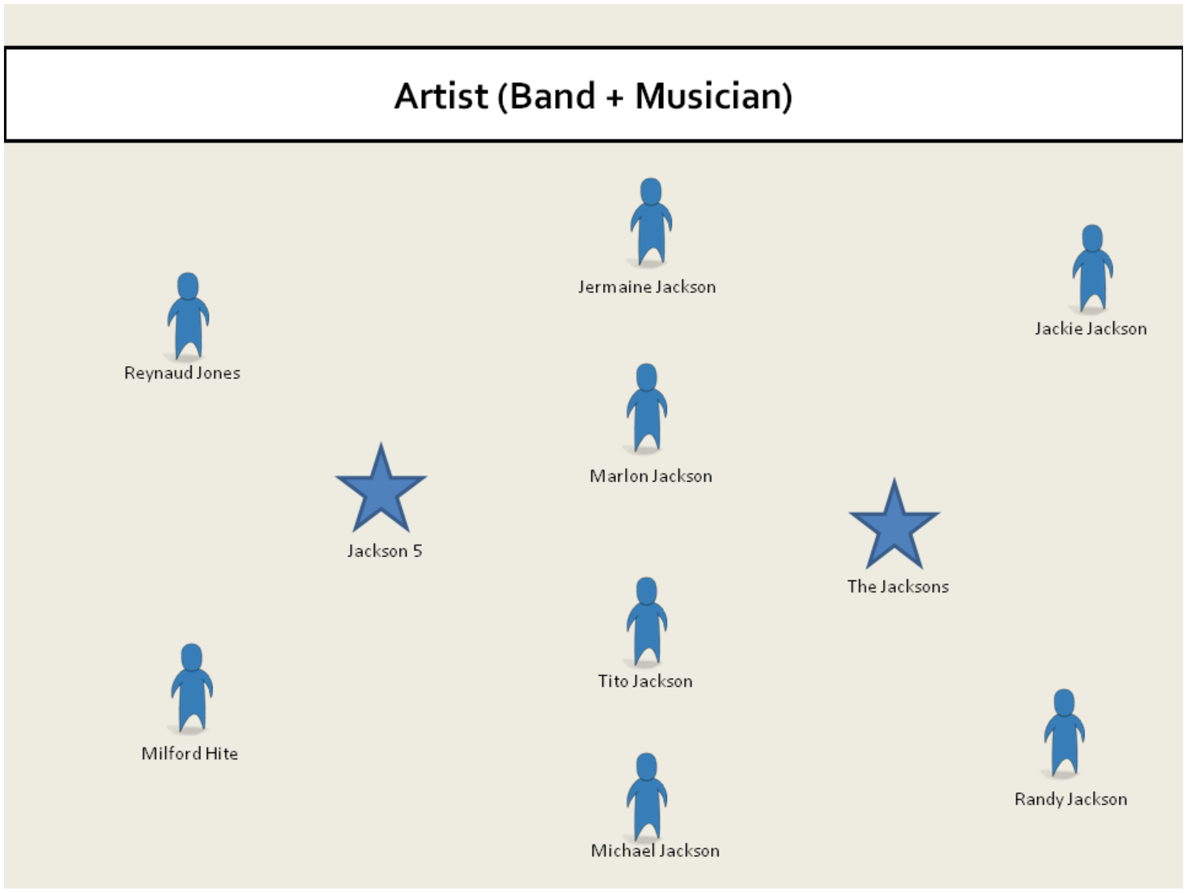


Figure 6-5 Music artist instances

These individual musicians may be related to constitute what is called a *Band*. Figure 6-6 exemplifies this relationship relating Artists such as Michael Jackson and the Jackson 5 from Figure 6-5.

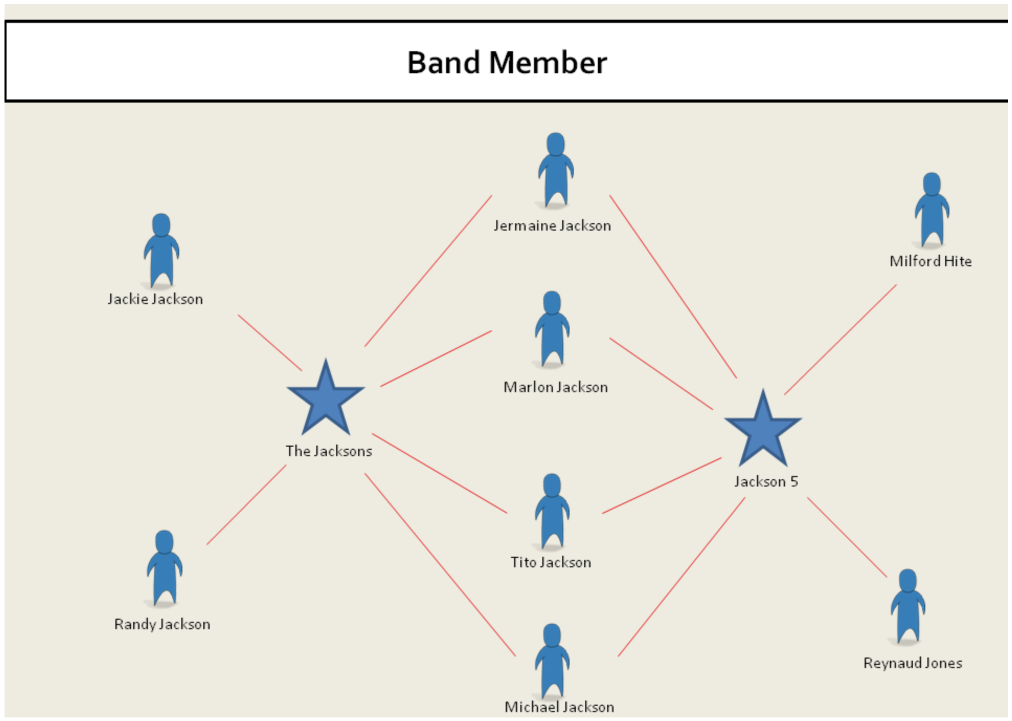


Figure 6-6 Band membership relations

Artists can fulfill all kinds of roles, from producers, publishers, lyricists, conductors and many more. Here we concentrate on *Artists* that can be authors of music albums, which may either be a *Band* or an individual person, what we call a Solo Author. Figure 6-7 exemplifies album authorship.

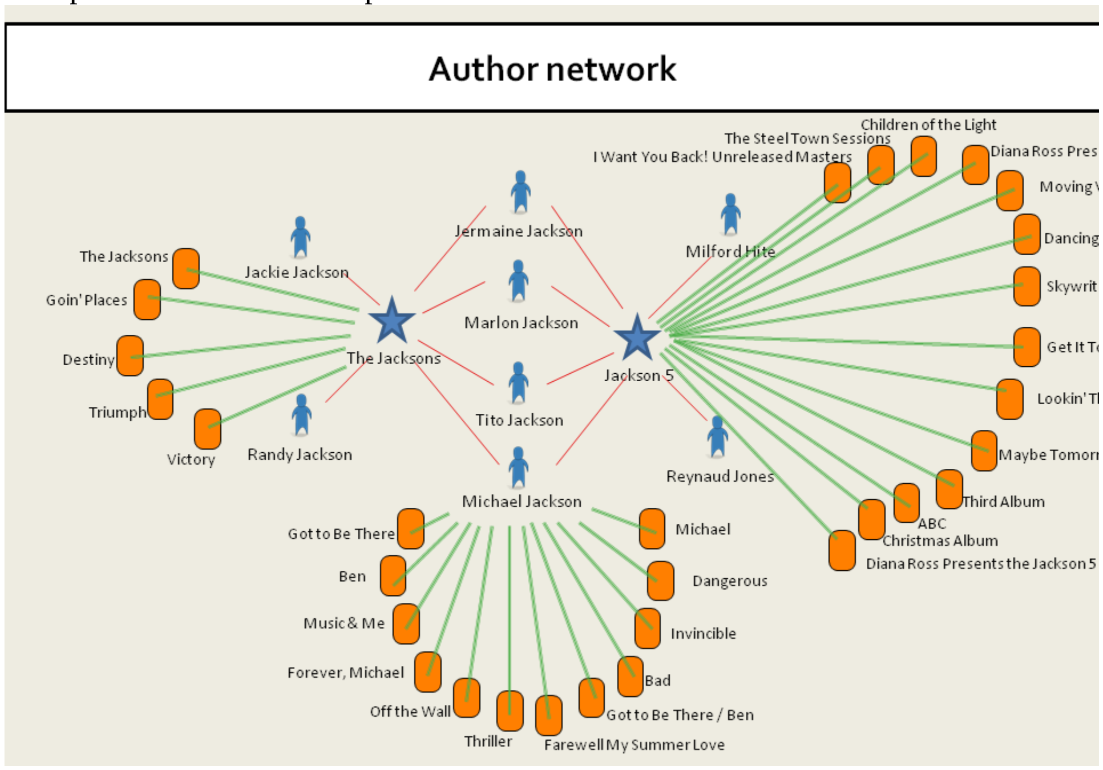


Figure 6-7 Albums can be authored by both Bands and Solo Authors

The most interesting information about *Artists* is the *Releases* they were involved in. A *release* in MusicBrainz is identified by its *track*-set e.g. one release of Thriller by Michael Jackson has 9 tracks and another has 15. To the average music consumer, they are both the same album, namely, Thriller. To respect such conceptual unity and group several different releases into a single logical entity, the concept of a *Release Group* was introduced; every *release* belongs to one, and only one *release group*. Figure 6-8 exemplifies Michael Jackson's different releases, for each release group.

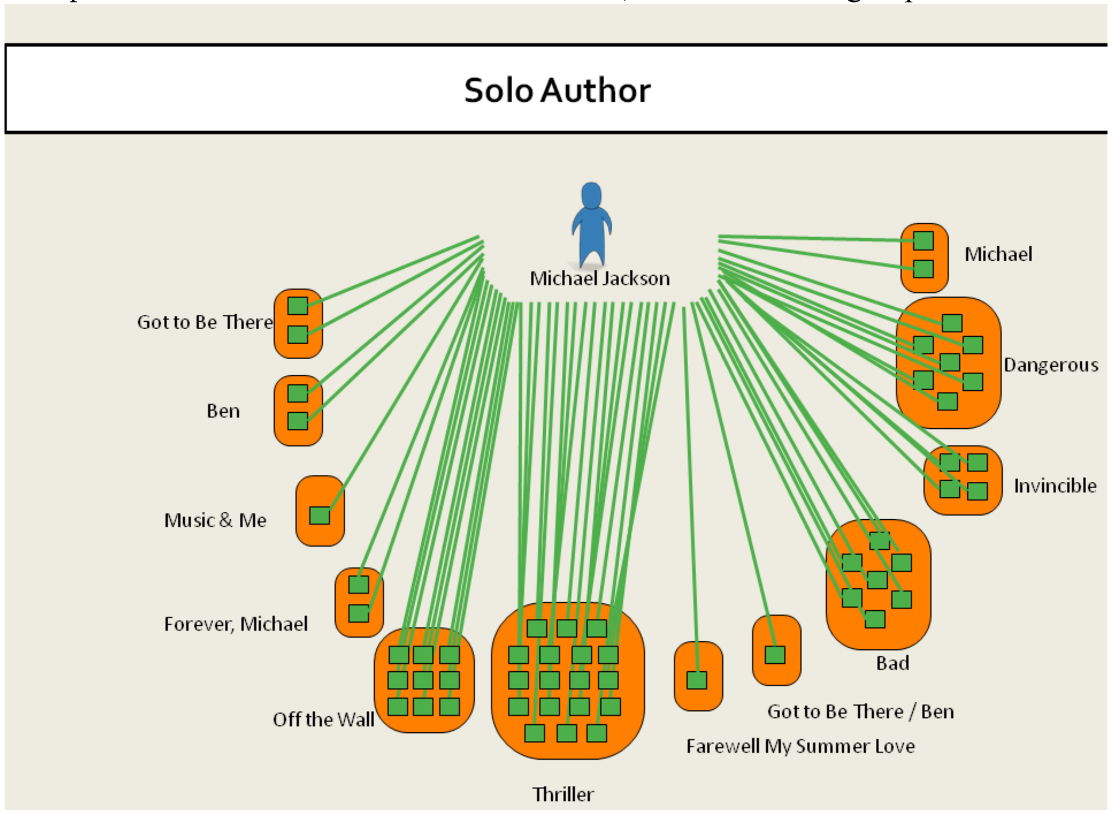


Figure 6-8 Different releases for Michael Jackson's release groups

Usually, releases and release groups have the same name, but on occasion, some releases are named differently; for example, on Thriller's 25th anniversary a special edition reissue was released under the name "Thriller 25". These releases are grouped together under the same release group in MusicBrainz. Figure 6-9 exemplifies this situation. Figure 6-10 details a single release, showing the track names.

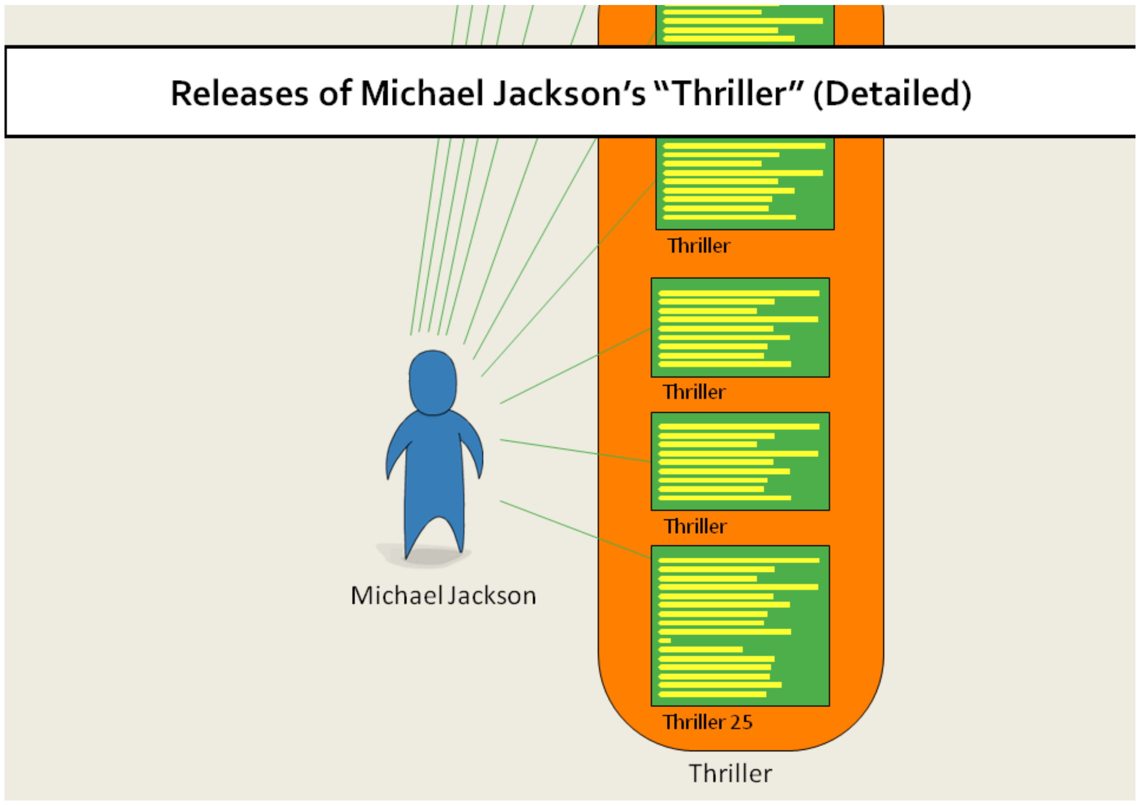


Figure 6-9 Similar releases with different track-sets constitute the same release group

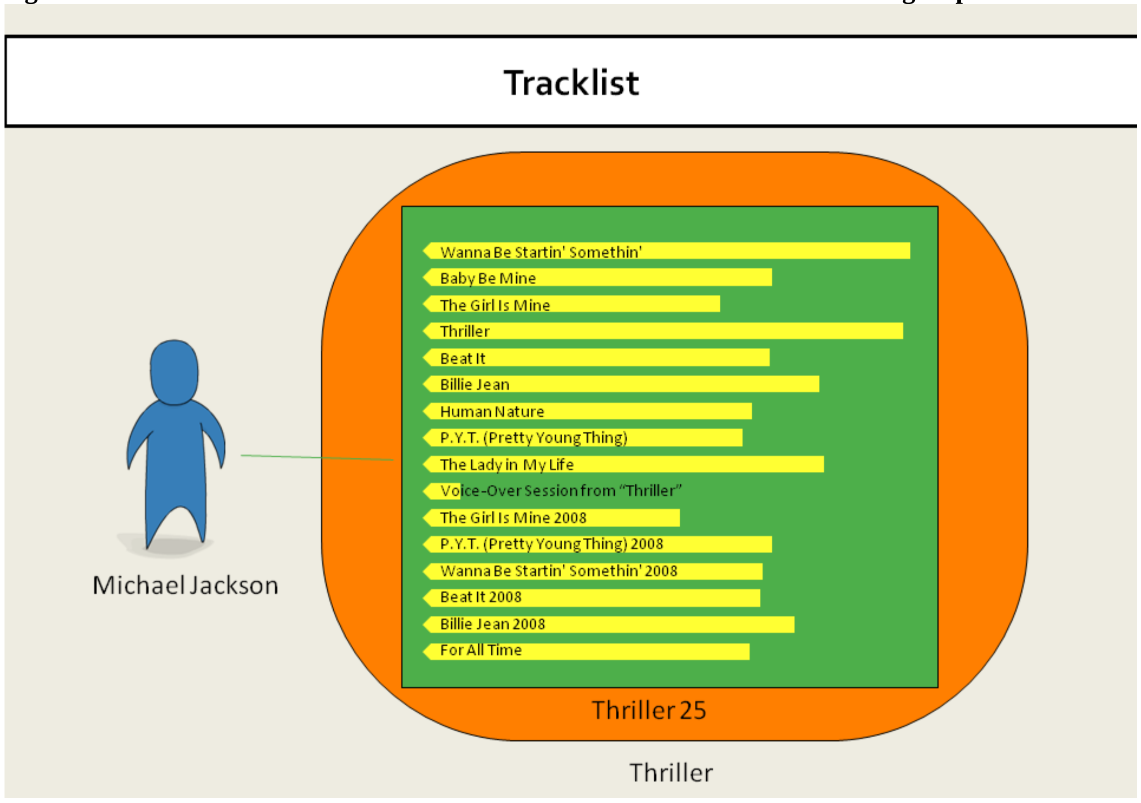
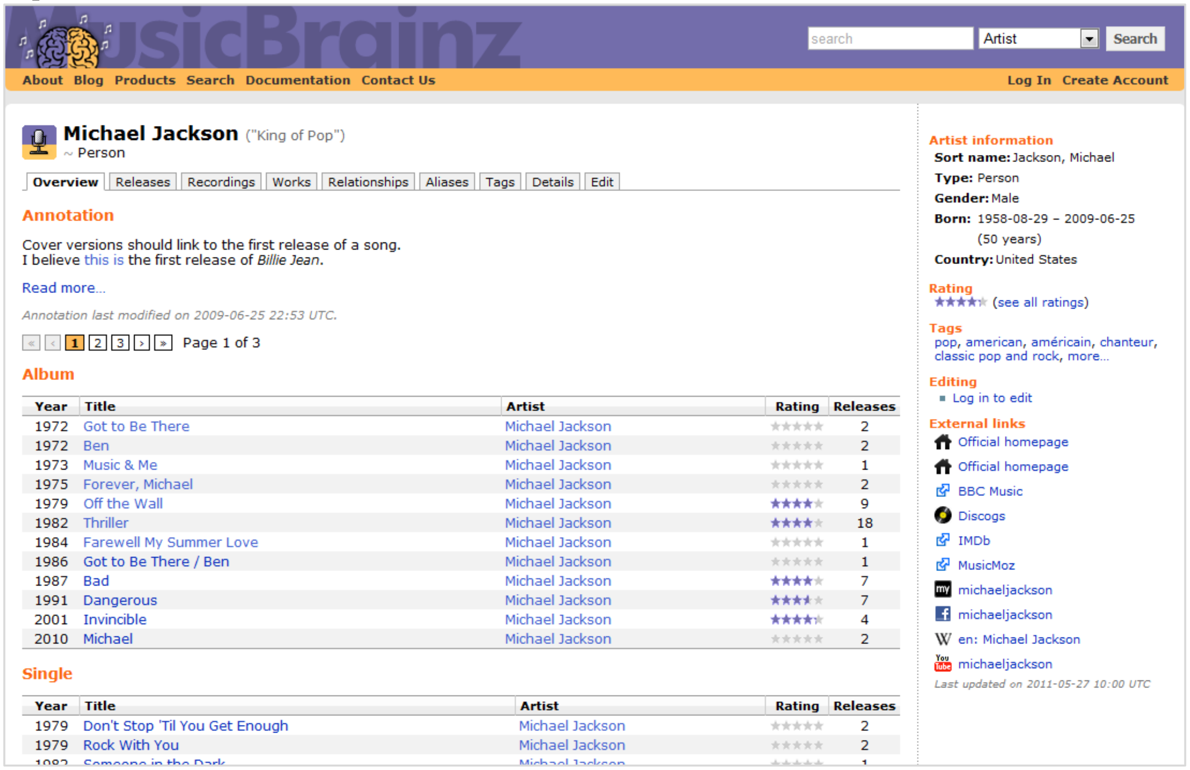


Figure 6-10 A release detailing showing tracks and their lengths

Both release groups and releases are "albums" in a general sense, but with a slight difference: a release is something you can buy as media, e.g. a CD, a vinyl record etc. on its own, while a release group embraces the concept of an album -- it doesn't matter how many CDs or editions/versions it had. When an artist tells you "We've released our new album", they're talking about a release group. When his publisher says "This new album gets released next week in Japan and next month in Europe", they're talking about the different releases that belong in the release group that the artist told you about. Still, one release may have different release events, and be released in different countries even in different media. Extensive discussion on the concept is available in the documentation (METABRAINZ, 2011d).

The currently available form of data visualization is through html pages for each of the individual belonging to one of the main data kinds: Artist, Release, Release Group and Track. Links in these pages embody the relationships between the individuals, e.g. in any Person Artist's page there is a link to the bands the person participated and albums he/she composed solo. There is no way to visualize patterns between the relationships, such as chains of relationships, as they are presented centered on an individual in a single level of depth.


MusicBrainz
search
Artist
Search
About Blog Products Search Documentation Contact Us
Log In Create Account
Overview Releases Recordings Works Relationships Aliases Tags Details Edit
« ‹ 1 2 3 › »

Michael Jackson ("King of Pop")
~ Person

Annotation

Cover versions should link to the first release of a song.
I believe this is the first release of *Billie Jean*.

Read more...

Annotation last modified on 2009-06-25 22:53 UTC.

Page 1 of 3

Album

Year	Title	Artist	Rating	Releases
1972	Got to Be There	Michael Jackson	☆☆☆☆☆	2
1972	Ben	Michael Jackson	☆☆☆☆☆	2
1973	Music & Me	Michael Jackson	☆☆☆☆☆	1
1975	Forever, Michael	Michael Jackson	☆☆☆☆☆	2
1979	Off the Wall	Michael Jackson	★★★★☆	9
1982	Thriller	Michael Jackson	★★★★☆	18
1984	Farewell My Summer Love	Michael Jackson	☆☆☆☆☆	1
1986	Got to Be There / Ben	Michael Jackson	☆☆☆☆☆	1
1987	Bad	Michael Jackson	★★★★☆	7
1991	Dangerous	Michael Jackson	★★★☆☆	7
2001	Invincible	Michael Jackson	★★★★☆	4
2010	Michael	Michael Jackson	☆☆☆☆☆	2

Single

Year	Title	Artist	Rating	Releases
1979	Don't Stop 'Til You Get Enough	Michael Jackson	☆☆☆☆☆	2
1979	Rock With You	Michael Jackson	☆☆☆☆☆	2

Artist information
Sort name: Jackson, Michael
Type: Person
Gender: Male
Born: 1958-08-29 – 2009-06-25 (50 years)
Country: United States

Rating
★★★★☆ (see all ratings)

Tags
pop, american, américain, chanteur, classic pop and rock, more...

Editing
- Log in to edit

External links
Official homepage
Official homepage
BBC Music
Discogs
IMDb
MusicMoz
michaeljackson
michaeljackson
en: Michael Jackson
michaeljackson
Last updated on 2011-05-27 10:00 UTC

Figure 6-11 MusicBrainz data web interface

7 Conclusion

“Questions about whether design is necessary or affordable are quite beside the point: design is inevitable. The alternative to good design is bad design, not no design at all.”
Book Design: A Practical Introduction by Douglas Martin

Elaborating conceptual models and communicating with them is a difficult task, which should not be taken lightly. We should use whatever means possible for us to overcome our limited human capacity in transferring information and create an out of mind representation of abstractions. One of these approaches is conceptual modeling, which facilitates the creation of precise specifications of abstractions (models), for the purpose of understanding and communication (MYLOPOULOS, 1992). To create precise models and communicate effectively we have assumed that the modeling language should be founded on upper-level ontologies which provide a conceptualization to create domain abstractions, based on the work of Guizzardi (2005). Models created with the OntoUML modeling language can represent domain abstractions that conform to the Unified Foundational Ontology.

To increase confidence in the quality of the model produced, model assessment practices enable both the verification of the model’s adherence to the modeling language rules as well as validating its adequacy in representing the intended domain abstraction. Our approach to model validation is based on the simulation of models and the visual assessment of state-of-affairs at the instance level. In our earlier work, visual assessment was based on automatically generated node-link diagrams (ref) with an ad hoc notation and a layout scheme that is built in the Alloy Analyzer. We identified an opportunity for improving diagrams in the work of Moody (2010), which details human graphical information processing and provides design principles that can explore the qualities in the way humans assess graphical information to create cognitive effective diagrams. Further explorations in the work of Colin Ware (2004) revealed a vast realm of possibilities to improve visualizations.

We have taken one approach to improving the instance diagram visualization by investigating how to design the symbols used. Semiotics and human graphical information perception mechanisms were used as reference to design symbols that can be easily distinguished and preattentively processed. A domain-specific design is required to assign symbols of distinctive shapes to indicate different identity criteria for model instances and shape decoration to indicate class instantiation by the individuals. Trade-off decisions were identified: emphasizing one type of classification may require omitting another type, therefore a careful account to the purpose of the diagram should be considered before designing a symbol set.

Besides the individual symbol design, the concern of how the symbols interact was extremely important in the results. Gestalt theory describes how visual percept is grouped together into meaningful units i.e. the types of visual patterns that emerge. These patterns are inevitable, an integral part of our perception and cognition. Better than to avoid them (which would be extremely difficult, if not impossible), is to take advantage of them to convey meaning. Naturally, this meaning should reflect a conceptual relationship between the symbols that constitute the pattern.

There is more to visualizations than meets the eye. While it is important to design symbols to be naturally distinguished and easily remembered, the choice of visual elements in itself does not entail symbol meaning understanding, contributing mostly to perceiving symbol difference and similarity. Diagrams have a purpose and the designer should consider the diagram's cognitive processing and the tasks that the user will perform while using it. "What information visualization is really about is external cognition, that is, how resources outside the mind can be used to boost the cognitive capabilities of the mind." (WARE, 2004, p. xvii) Diagrams can boost human cognitive capabilities by working as memory extensions and supporting visual thinking e.g. the way a road map helps planning travelling routes. Working with diagrams allows a vast amount of data to be accessed rapidly, as quick as an eye movement. Visual queries are acts of attention which pull patterns from the display to perform visual thinking; their design is largely related to deciding the purpose of the diagram. Symbol design should be carefully harmonized with visual query design to support visual assessment tasks.

All cognitive processing of diagrams is bottlenecked by working memory limits. Humans possess a limited working memory and can only hold a few "chucks" of information at a time. "Indeed, knowledge about working memory limitations suggest humans are particularly poor at complex reasoning unless most of the elements with which we reason have previously been stored in long-term memory." (SWELLER et al., 1998, p. 254) We explored the concept of mental schemas as a guiding notion to design a strategy to overcome working memory limitations and promote long-term memory formation. "According to schema theory, it is through the building of increasing numbers of ever more complex schemas by combining elements consisting of lower level schemas into higher level schemas that skilled performance develops." (SWELLER et al., 1998, p. 255) These higher level schemas can be easily held in working memory, taking as much resources as a single "chunk" of information would. As skilled performance develops, more information can be held at a time in working memory, facilitating reasoning.

To reduce working memory loads we discussed complexity management techniques that segment information in modules, effectively reducing the amount of information to be dealt with at a time and allowing coherent cognitive processing to occur. To promote long-term memory formation, sequences of modules should gradually introduce domain concepts, each subsequent one elaborating on the concepts presented on a previous module. This presentation strategy for domain concepts partially attends to the problem that novices lack a strategy to process diagrams (MOODY, D. L., 2010), since the user handles only a fraction of the domain at a time and is not distracted by extraneous information.

We advocate that there is no "one best way" to display information in a diagram: each design decision involves some *trade-off*s. The body of knowledge we studied in the information visualization literature allowed us to forecast which decisions are good but only corpus of empirical data can determine the real effects of our decisions and determine directions for future work.

Considering that the criteria we adopt for the design of visualizations are (mostly) based on research on human physiology (as opposed to culture), the emergent perceptual properties should positively affect anyone. However, it has been argued that diagrams cannot be optimized both to novices and expert user. We have deliberately prioritized novices, considering that conceptual modelers are usually novices in the domain of discourse and domain experts are usually novices in modeling notations. We hope this can contribute to reducing the semantic gap between the people involved in the analysis of domains and in their specification in the form of OntoUML conceptual models. This has practical relevance to Computer Science as the quality of the conceptual models improves the quality of software produced based on them.

8 Future Work

8.1 Empirical validation

We have reviewed some theory relevant for information visualization design and experimented by designing instance diagrams for the MusicBrainz ontology. However, we still must conduct empirical validation to evaluate the benefits and assess directions for improvements. Conducting usability research, both qualitative (understanding of human behavior and the reasons that govern such behavior) and quantitative (systematic empirical investigation of quantitative properties and phenomena and their relationships) would improve our understanding of how efficient the diagrams are, how easy they are to use and how satisfying it is to use them. We have identified some research questions, such as:

- Can the simulations improve understanding of the domain?
- What is the impact of applying these design principles regarding model validation? Does it improve the amount of errors found? What about the speed in finding them?
- We have argued that each module should be subject to symbol design to optimize the visual assessment of the fraction of domain that is emphasized in the module. Is it harmful to have multiple representations of the same individuals, depending on the module? Is it better to have the same representation across modules? What are the tradeoffs involved?
- How much can diagrams designed for novices facilitate skill development? Is it possible to make a smooth transition to diagrams designed for experts?
- What is the learning curve for instance diagrams in a given domain? What about the learning curve to be domain independent?

We have identified relevant theory for designing diagrams and explored the design of diagrams for visualizing information according to the MusicBrainz ontology. This provided us with some insight into formulating these questions for further investigation. We consider the work presented here as the first step towards a prescriptive theory to designing instance diagrams, which should then be validated thoroughly.

8.2 Diagram design for the domain expert

The work we present here is primarily concerned with the design of instance diagrams to novices. Although this facilitates communication on the instance level between conceptual modelers and domain experts, it does not focus on facilitating communication on the model level. Conceptual modeling notations for novices in the conceptual modeling language would need to be designed as well. Additionally, instance diagram visualizations for the domain expert can be designed to be more expressive, since their knowledge on the domain diminishes the negative effects of visual complexity. These would help them assess more information at a time, making the diagrams more efficient.

8.3 Layout algorithm

Spatial positioning is one of the main reasons why the Alloy Analyzer's native visualization was considered unfit for our purposes; it applies a slightly modified version of the classical Sugiyama hierarchical layout algorithm (SUGIYAMA et al., 1981 *apud* CHANG, 2009) individually to each snapshot of the instance diagram. The resulting layouts thus do not account for keeping individuals in place between multiple snapshots. As a consequence of applying the algorithm to each state separately, a change in the set of nodes requires a serial search in the labels of all nodes to identify and track the changes in individuals. A graph layout algorithm that considers multiple snapshots would need to be investigated.

8.4 Behavioral specifications

OntoUML language supports the definition of structural models; it is not meant to specify behavioral aspects of a domain. Unfortunately, without the behavioral specification, even the simplest of models may exhibit undesired behavior, interfering largely with the process of validation. The simulation is expected to provide snapshots (or sequences thereof) that reflect accurately the domain of abstraction. If an invalid state-of-affairs is met, it may be difficult for inexperienced users of the language to determine the nature of the inconsistency, whether it is structural or not. In this sense, we believe that there shouldn’t be a struggle to differentiate between the two; rather, the effort should be directed at finding and removing model inconsistencies, regardless of their nature.

Since there is no ontologically well founded language for defining behavior to complement the structural models, an OntoUML model is incomplete in terms of constraining possibilities for sequences of state-of-affairs. The only available way to constraint the simulation accordingly is to add constraints in the Alloy language, after the model transformation. This procedure is inadequate for many different reasons: (i) altering the behavior means the simulation no longer represents the reference model; (ii) changes in the structural model may outdate the hand-written Alloy rules, requiring refactoring; (iii) it requires knowledge in the Alloy language; (iv) it requires knowledge in the transformation procedure, including optimization variations, which are largely intertwined with knowledge in the details of OntoUML language.

Work towards the definition of a modeling language for behavioral models that comply with the UFO is required. For example, state-charts could model transition rules between phases in a partition set and thus complement the current OntoUML class diagrams. Additionally, OntoUML2Alloy would need to be modified to incorporate the behavioral constraints described by the behavioral model. This would allow us to adapt the visualization scheme considering implied ordering of state changes and that could influence the layout of sequences of snapshots.

9 References

ANASTASAKIS, K.; BORDBAR, B. UML2Alloy: A tool for lightweight modelling of Discrete Event Systems. **IADIS International Conference in Applied Computing**, v. 1, n. 1999, p. 209–216. doi: 10.1590/S0011-52581999000100002, 2005.

BASILI, V. R.; BOEHM, B. Software Defect Reduction Top 10 List. **Computer**, v. 34, n. 1, p. 135-137, 2001.

BENEVIDES, A. ontouml - An editor for ontologically well-founded conceptual modeling - Google Project Hosting. . Retrieved June 16, 2011, from http://code.google.com/p/ontouml/, 2011.

BENEVIDES, A. B. **A Model-based Graphical Editor for Supporting the Creation, Verification and Validation of OntoUML Conceptual Models**. Vitória, E.S., Brazil: Federal University of Espírito Santo (UFES). Retrieved from http://code.google.com/p/ontouml/downloads/detail?name=MSc_thesis_Alessander_Botti_Benevides.pdf, 2010, February.

BENEVIDES, A. B.; GUIZZARDI, G. A Model-Based Tool for Conceptual Modeling and Domain Ontology Engineering in OntoUML. In: W. Aalst; J. Mylopoulos; N. M. Sadeh; et al. (Eds.); **Enterprise Information Systems**, Lecture Notes in Business Information Processing.. v. 24, p.528-538. Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-01347-8_44, 2009.

BENEVIDES, A. GUIZZARDI, G. BRAGA, B.; ALMEIDA, J. Assessing Modal Aspects of OntoUML Conceptual Models in Alloy. **Advances in Conceptual Modeling - Challenging Perspectives**, Lecture Notes in Computer Science.. v. 5833, p.55-64. Springer Berlin / Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-04947-7_8, 2009.

BERTIN, J. **Semiology of graphics**. University of Wisconsin Press, 1983.

BRAGA, B. ALMEIDA, J. GUIZZARDI, G.; BENEVIDES, A. Transforming OntoUML into Alloy: towards conceptual model validation using a lightweight formal method. **Innovations in Systems and Software Engineering**, v. 6, n. 1, p. 55-63, 2010.

BROOKS, F. P. No Silver Bullet Essence and Accidents of Software Engineering. **Computer**, v. 20, n. 4, p. 10-19, 1987.

CARRARETO, R. **A modeling infrastructure for OntoUML**. UFES, 2010, July.

CHANG, F. Re: Visualization lines. **Alloy Forum**. Retrieved January 27, 2011, from http://alloy.mit.edu/community/node/527, 2009, February 6.

DEPARTAMENT OF DEFENSE. Directive 5000.59. ,2007, August 8.

DIJKSTRA, E. W. The humble programmer. **Communications of the ACM**, v. 15, n. 10, p. 859-866, 1972.

GREGORY, R. L. (RICHARD L. **The intelligent eye / R.L. Gregory**. New York :: McGraw-Hill, 1970.

GUARINO, N. Formal Ontology and Information Systems. , p. 3--15, 1998.

GUARINO, N.; WELTY, C. Ontological Analysis of Taxonomic Relationships. **DATA AND KNOWLEDGE ENGINEERING**, v. 39, p. 51--74. Retrieved February 21, 2011, , 2000.

GUIZZARDI, G. **Ontological foundations for structural conceptual models**. PhD Thesis, Enschede: CTIT, Centre for Telematics and Information Technology. Retrieved from http://doc.utwente.nl/50826/, 2005.

GUIZZARDI, G. Agent Roles, Qua Individuals and the Counting Problem. In: A. Garcia; R. Choren; C. Lucena; et al. (Eds.); **Software Engineering for Multi-Agent Systems IV**.

v. 3914, p.143-160. Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved June 16, 2011, from http://www.springerlink.com/content/484066204t135u31/, 2006.

GUIZZARDI, G. WAGNER, G. GUARINO, N.; SINDEREN, M. VAN. An Ontologically Well-Founded Profile for UML Conceptual Models. CAiSE. **Anais...** p.112-126, 2004.

JACKSON, D. **Software Abstractions: Logic, Language, and Analysis**. The MIT Press. Retrieved December 16, 2010, from http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/0262101149, 2006.

MAYER, R.; MORENO, R. Nine Ways to Reduce Cognitive Load in Multimedia Learning. **Educational Psychologist**, v. 38, n. 1, p. 43-52. Retrieved May 31, 2011, , 2003.

METABRAINZ. Artist - MusicBrainz. . Retrieved May 13, 2011a, from http://musicbrainz.org/doc/Artist, 2011.

METABRAINZ. Artist Intent - MusicBrainz. . Retrieved February 15, 2011b, from http://musicbrainz.org/doc/Artist_Intent, 2011, February 15.

METABRAINZ. Style Principle - MusicBrainz. . Retrieved February 15, 2011c, from http://musicbrainz.org/doc/Style_Principle, 2011, February 15.

METABRAINZ. What Defines A Unique Release - MusicBrainz. . Retrieved February 16, 2011d, from http://musicbrainz.org/doc/What_Defines_A_Unique_Release, 2011, February 15.

METABRAINZ. How To Contribute - MusicBrainz. . Retrieved February 17, 2011e, from http://musicbrainz.org/doc/How_To_Contribute, 2011, February 17.

MILLER, G. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. **The Psychological Review**, v. 63, p. 81-97. Retrieved February 11, 2011, , 1956.

MOODY, D. What Makes a Good Diagram? Improving the Cognitive Effectiveness of Diagrams in IS Development. **Advances in Information Systems Development**. p.481-492. Springer US. Retrieved from http://dx.doi.org/10.1007/978-0-387-70802-7_40, 2007.

MOODY, D. L. The "physics" of notations: a scientific approach to designing visual notations in software engineering. Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2. **Anais...** , ICSE '10.. p.485–486. New York, NY, USA: ACM. doi: http://doi.acm.org/10.1145/1810295.1810442, 2010.

MYLOPOULOS. Conceptual modelling and Telos. p.49-68, 1992.

POSTMA, A. IZENDOORN, R.; EDWARD. Sex Differences in Object Location Memory. **Brain and Cognition**, v. 36, p. 334-345, 1998.

SACKS, O. **The Man Who Mistook His Wife For A Hat: And Other Clinical Tales**. touchstone Trade Paper Edition, 12th Pri ed. Touchstone, 1998.

SUGIYAMA, K. TAGAWA, S.; TODA, M. Methods for Visual Understanding of Hierarchical System Structures. **Systems, Man and Cybernetics, IEEE Transactions on**, v. 11, n. 2, p. 109-125. doi: 10.1109/TSMC.1981.4308636, 1981.

SWELLER, J. VAN MERRIENBOER, J.; PAAS, F. Cognitive Architecture and Instructional Design. **Educational Psychology Review**, v. 10, p. 251-296, 1998.

TUFTE, E. The magical number seven, plus or minus two: Not relevant for design. **edwardtufte.com**. Retrieved February 11, 2011, from http://www.edwardtufte.com/bboard/q-and-a-fetch-msg?msg_id=0000U6&topic_id=1, 2000, September 23.

VANDERBURG, G. Real Software Engineering. . Retrieved June 22, 2011, from http://vimeo.com/16287115, 2010, October 28.

WARE, C. **Information Visualization, Second Edition: Perception for Design**. 2nd ed. Morgan Kaufmann, 2004.
WIKIPEDIA CONTRIBUTORS. Apophenia - Wikipedia, the free encyclopedia. . Retrieved June 17, 2011, from http://en.wikipedia.org/wiki/Apophenia, 2011.
XU, F.; CAREY, S. Infants' Metaphysics: The Case of Numerical Identity. . Retrieved from http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.59.1268, 1996.
ZAMBORLINI, V. **Estudo de Alternativas de Mapeamento de Ontologias da Linguagem OntoUML para OWL: Abordagens para Representação de Informação Temporal**. UFES, 2011.
s