THESIS / THÈSE

MASTER IN COMPUTER SCIENCE

Towards the Operationalization of the Physics of Notations

Cherchem, Nabil

Award date:
2014

Awarding institution:
University of Namur

[Link to publication](#)

Towards the Operationalization of the Physics of Notations

Nabil CHERCHEM

Maître de stage :   Prof. Daniel Amyot

Promoteur :   _____ (Signature pour approbation du dépôt - REE art. 40)
Prof. Patrick Heymans

Co-promoteur :   Nicolas Genon

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

Abstract

Visual notations are commonly used in Information Systems (IS) to represent information by using a structured set of rules and symbols. Nowadays, visual notations are used in almost every field of IS. Over the years, and since the first modelling language of Von Neumann, visual notations drew the attention of most research in IS. However, the research efforts are focusing more on semantic issues than on visual representation issues. Indeed, most of nowadays' modelling languages do not provide any design rationale about their visual syntax, i.e., the symbols used in the visual notation. In general, modellers rely only on common sense to design the visual syntax of notations. To analyse and design notations, Moody's Physics of Notations (PoN) relies on scientific foundations from various fields such as Psychophysics, Cognitive Psychology, and Graphic Design. However, an operationalization of the PoN theory remains crucial. Indeed, without any operationalization, a theory would appear to be limited to be applied on concrete visual languages. Moreover, it appeared that the PoN was used differently from one researcher to another. This is due to the ambiguity caused by the need to refer to self-interpretation (i.e., intuition) on specific points of the PoN. An operationalization would reduce such an ambiguity. These reasons lead us to propose an operationalization of the PoN. To this end, this thesis will define a set of metrics to support two of the PoN principles. This thesis draws upon Störrle and Fish's attempt to operationalize the PoN and on various work of Genon *et al.* who applied the PoN to analyse the *cognitive effectiveness* of modelling languages (i.e., optimisation of how visual notations are processed by the human mind). This operationalization of the PoN leads us to examine the degree of scientificity of the PoN by using a meta-theory (i.e., a theory to analyse and classify other theories). This theoretical analysis reveals the PoN principles that need to be more developed in future research. Finally, we propose to apply our operationalization on a modelling language developed by Mussbacher: the Aspect-oriented User Requirements Notation (AoURN). AoURN extends ITU-T's User Requirements Notation (URN) to support encapsulation of crosscutting concerns in requirements models. Furthermore, AoURN is the only standards-based and graphical approach to bring aspect-oriented modelling (AOM) to a requirements language. Moreover, the application of the PoN to AoURN is alone a valuable contribution.

Résumé

Les notations visuelles sont généralement utilisées dans les Systèmes d'Information (SI) afin de représenter l'information en utilisant un ensemble structuré de règles et de symboles. De nos jours, les notations visuelles sont utilisées dans presque toutes les branches des systèmes d'information. Au fur et à mesure des années, et ce depuis le premier langage de modélisation de Von Neumann, les notations visuelles ont canalisé la plupart des recherches dans le domaine des SI. Cependant, ces recherches sont le plus souvent focalisées sur le traitement des problèmes sémantiques et non les aspects visuels des langages de modélisation. Il est à noter que la plupart des langages de modélisation n'expliquent pas la logique de conception de leur syntaxe visuelle (les symboles utilisés dans la notation visuelle). En général, les modélisateurs s'appuient sur le sens commun pour définir la syntaxe visuelle des notations. Pour analyser et concevoir des notations visuelles, la Physics of Notations (PoN) de Moody repose sur des bases scientifiques de plusieurs disciplines telles que la psychophysique, la psychologie cognitive et l'étude de la conception de graphiques. Cependant, l'opérationnalisation de la PoN reste primordiale. En effet, sans elle, une théorie paraîtrait limitée lorsque l'on l'applique à un langage visuel donné. De plus, il faut bien constater que la PoN a été utilisée de manière différente d'un chercheur à un autre. Ceci est lié au besoin de faire appel à l'intuition pour faire face à l'ambiguïté de certains points de la PoN. Une opérationnalisation réduirait l'ambiguïté de ces points. Ces raisons nous poussent à proposer une opérationnalisation de la PoN. Ce mémoire définira un ensemble de métriques pour renforcer deux principes de la PoN. Nous nous appuyons sur une tentative d'opérationnaliser la PoN de Störrle et Fish et sur le travail de Genon *et al*. Ces derniers ont appliqué la PoN pour analyser l'efficacité cognitive de langages de modélisation qui est l'optimisation de notations visuelles pour le cerveau humain. Cette opérationnalisation nous a mené à reconsidérer le degré de scientificité de la PoN en utilisant une meta-theorie (une théorie analysant d'autres théories). Cette analyse a fait apparaitre que la PoN devra faire l'objet de recherche dans le futur. Pour terminer, dans le but d'évaluer nos métriques, nous les appliquerons à un langage de modélisation développé par Mussbacher, l'Aspect-oriented User Requirements Notation (AoURN). AoURN est une extension du langage User Requirements Notation (URN) soutenu par l'ITU-T. AoURN est le seul langage de modélisation à supporter de manière graphique la séparation de préoccupations transversales en ingénierie des exigences. En plus, nous appliquerons les autres principes de la PoN à AoURN pour obtenir une analyse complète d'AoURN.

Acknowledgment

Table of Contents

List of Figures

List of Tables

List of Acronyms

Acronym	Definition
AoGRL	Aspect-oriented and Goal-oriented Requirement Language
AOM	Aspect-oriented Modelling
AoUCM	Aspect-oriented Use Case Maps
AoURN	Aspect-oriented User Requirements Notation
CASE tools	Computer-Aided Software Engineering tools
CDs	Cognitive Dimensions Framework
GRL	Goal-oriented Requirement Language
i*	i-star Modelling Framework
IS	Information System
PoN	Physics of Notations
UCM	Use Case Maps
UML	Unified Modelling Language
URN	User Requirements Notation

Chapter 1.   Introduction

Visual notations are commonly used in Information Systems (IS) to represent information by using a structured set of rules and symbols. Nowadays, visual notations are used in almost every field of IS. Over the years, and since the first modelling language introduced by Von Neumann, visual notations drew the attention of most research in IS.

However, the research efforts are focusing more on semantic issues than on visual representation issues (e.g., [28][35]). Indeed, most of nowadays' modelling languages do not provide any design rationale about their visual syntax, i.e., the symbols used in the visual notation. In general, modellers rely only on common sense to design the visual syntax of visual notations [40].

To analyse and design notations, Moody's Physics of Notations (PoN) [25] relies on scientific foundations from theory and empirical evidence from a various fields such as Psychophysics, Cognitive Psychology, and Graphic Design. The PoN consists of nine principles that address both evaluation and design issues of visual notations. The PoN relies on the idea of falsifiability of its principles, which states that they can be empirically testable [31].

However, an operationalization of the PoN theory remains crucial. Indeed, without any operationalization, a theory would appear to be limited to be applied on concrete visual languages. Moreover, it appeared that the PoN was used differently from one researcher to another. This is due to the ambiguity caused by the need to refer to self-interpretation (i.e., intuition) on specific points of the PoN. An operationalization would reduce such an ambiguity.

To this end, we propose to operationalize the PoN principles. Our analysis and proposals are drawn upon Störrle and Fish's attempt to operationalize the PoN [39] and on various work of Genon *et al.* who applied the PoN to analyse the *cognitive effectiveness* of modelling languages (i.e., optimisation of how visual notations are processed by the human mind) [10][11].

This operationalization of the PoN leads us to examine the degree of scientificity of the PoN by using a meta-theory (i.e., a theory to analyze and classify other theories) [17]. We chose using Gregor's taxonomy, which draws upon various resources such as philosophic literature in the fields of natural sciences and the social sciences. Gregor discusses the purposes of a theory (such as prediction and explanation) in order to propose a taxonomy that synthesizes different perspectives into a coherent meta-theory. The taxonomy defines five types of theory. In this thesis, we employ a two-step approach to apply the taxonomy: gathering of information to fill the taxonomy components and a discussion based on these components to classify each of the PoN principles.

Moody has already applied the taxonomy on the PoN; however, we think that it was done at a very coarse-grained level. Therefore, we propose a "per-principle" evaluation. This theoretical analysis reveals that some PoN principles need to be more developed in future research. In our operationalization, we voluntarily focus our research on two principles of the PoN: Semiotic Clarity and Perceptual Discriminability. Our analysis encompasses definitions of mathematic formulae, methods and advices in order to support systematic analysis approach for future applications of the PoN.

In order to validate our operationalization, we apply our metrics on the Aspect-oriented User Requirements Notation (AoURN) [27]. AoURN extends ITU-T's User Requirements Notation (URN) to support encapsulation of crosscutting concerns in requirements models. Moreover, AoURN is the only standards-based and graphical approach to bring aspect-oriented modelling (AOM) to a requirements language. As a valuable contribution, we discuss the cognitive effectiveness of AoURN by applying the PoN as Genon *et al.* We discuss several weaknesses of AoURN in terms of the PoN. Moreover, we suggest improvements at the notational level by providing a set of cognitively more efficient symbols for AoURN. Since AoURN has not yet been standardized, our suggestions may be included in the future version of AoURN. A broader goal of our analysis according to cognitive effectiveness is to encourage discussion among practitioners from the language design and standardization communities.

This thesis is divided in two parts: Part I introduces the background in Chapters 2 and 3. Chapter 2 introduces the AoURN notation and Chapter 3 introduces two theories to evaluate and design visual notations: Green's Cognitive Dimensions and the PoN. Part II gathers Chapter 4 to 6. Chapter 4 introduces Gregor's taxonomy to classify a theory and a detailed analysis of the PoN against the taxonomy. Chapter 5 operationalizes and criticizes the PoN. Chapter 6 evaluates the AoURN against the set of metrics of the PoN developed on Chapter 5. Finally, future work is discussed in Chapter 7. It includes the suggestion of a new line of research that uses CASE tool features to improve the cognitive effectiveness of notations.

Part I: Background

In Chapter 2, we present the ITU-T's User Requirements Notation (URN) and its context in the requirements engineering field. AoURN, which extends URN will then be presented. Moreover, we illustrate both URN and AoURN with an example for comprehension purpose.

In Chapter 3, we present two of the most popular theories to evaluate visual notations: the Green's Cognitive Dimensions and the Moody's Physics of Notations.

In Chapter 4, we present Gregor's Taxonomy, we did not include that introduction in this Part I to simplify the reading of the rest of Chapter 4.

Chapter 2.   Requirements Engineering and RE Modelling Languages

2.1.   Requirements Engineering

Requirements engineering is a crucial step in the software life cycle. It aims to extract, analyse, specify, validate, and manage the evolving requirements of a system's stakeholders. Often, the needs from a specific domain can be extracted and specified by modelling and organizing them with a set of diagrams. The main challenge is to extract the most relevant information of the domain, which means being precise enough to represent the domain and not gather irrelevant details.

Another challenge could be how to represent diagrams in a more cognitively efficient way. For example, a long list of requirements written in plain text is less cognitively efficient than a graphical representation of these requirements. Indeed, diagrams use a two-dimensional geometric symbolic representation of information, which is processed more efficiently by human cognition.

Over several years, research has been conducted to formalize the different activities that compose the requirements engineering process. This research led to modelling languages that have been designed to convey more effectively the extracted information. Some of them are introduced in this section.

2.2.   UML

The Unified Modelling Language (UML) is a family of modelling languages, designed as general-purpose modelling languages, that can address every step of the software development and requirements engineering processes. Typically, the requirements are modelled from high-level diagrams (overview of the domain) to lower-level diagrams (close to the implementation).

One of the main strengths of UML is that it is standardized by the Object Management Group (OMG) and also by the International Organization for Standardization (ISO). In my opinion, standardization helps providing more confidence that a modelling language can be adopted in practice by the industry.

UML is the most used software modelling language. However, UML lacks abstraction for requirements gathering because even in high-level diagrams, it requires one to handle details (e.g., synchronous / asynchronous messages or type of parameters in sequence diagrams) that are not relevant at this level.

Furthermore, UML lacks traceability capabilities for connecting high-level and low-level modelling diagrams. Even worse, there is no graphical symbol in UML to link diagrams of the different languages that compose UML (e.g., how to link a sequence diagram to a state machine diagram). Therefore, a means of connecting diagrams from different perspectives should be offered. Finally, there is no goal-oriented view in UML.

Due to its general-purpose characteristic, it seems to us that UML is not specialized for requirements engineering. Indeed, dealing with requirements engineering requires from a modelling language to provide the right degree of formality, i.e., to enable high abstractions for the high-level diagrams and more precise capabilities for the low-level diagrams.

2.3. User Requirements Notation

The User Requirements Notation (URN) [1][2][3][19] is used to model and analyse requirements with goals and scenarios. URN is composed of two sub-languages, the Goal-oriented Requirement Language (GRL), which models actors and their intentions; and the Use Case Maps (UCM) language, which lets us model scenarios in accordance with their components. One advantage of URN compared to UML is that it enables modellers to capture the rationale behind some choices that have been taken (GRL). Furthermore, UCM diagrams (i.e., UCM path) do not introduce extra details like UML sequence diagrams, and they support more workflow patterns than UML activity diagrams [26]. URN explicitly offers a means to pass from the goal-oriented modelling part to the scenario-based modelling part.

Finally, URN has been standardized by the International Telecommunication Union (ITU), an agency of the United Nations specialized in information and communication technologies. The specification of the language is described in the ITU-T Z.151 standard document; the abstract syntax and the concrete grammar syntax are precisely defined.

2.3.1 Goal modelling with URN

The Goal-oriented Requirement Language (GRL), subset of the URN language, is a visual modelling language used in the early phase of the requirement engineering process. It aims to help understand the domain without any concerns about implementation. A GRL diagram models both the stakeholders' goals and the functional and non-functional requirements. Moreover, alternatives considered and the rationale of decisions can be captured in GRL.

GRL is based on the i* framework, one of the leading goal-oriented languages, and on the Non-Functional Requirement framework, which makes it very close to these frameworks both for the abstract and the concrete grammar syntaxes. The GRL core is composed of four main categories of concepts: actors, intentional elements, indicators, and links.

A GRL graph (i.e., a GRL diagram) is composed of *Actors* (⬭) that represent stakeholders of an application domain. Those actors may contain intentional elements. An intentional element may be a *Goal* (▭), *Softgoal* (▱), *Task* (⬠), *Resource* (□) or *Belief* (◯).

The softgoal and goal concepts can be confusing: the latter does not have to be understood as "hardgoal" but as a goal that is fully quantifiable. A softgoal depicts a goal related to non-functional requirements that cannot be fully achieved in a quantifiable way (e.g., the maintainability of a system). A task is an operationalization of a goal or a softgoal. A resource is a physical or informational entity, which has to be available to complete or achieve a goal or a softgoal. Finally, a belief provides a rationale for some of the design decisions. All these elements can be measured by a level of satisfaction either qualitatively (e.g., ✔•) or quantitatively (i.e., with a number).

GRL has four kinds of links that let us express interactions between two elements. A *Decomposition link* (+——) provides the ability for an intentional element to be decomposed (AND, OR, XOR) into smaller pieces. A *Contribution* link (→) shows the desired influence of one element to another. The *Correlation link* (┈┈→) denotes the concept of an indirect influence between two elements (i.e., side effect impact). Contributions and Correlations links may have a label, which can be an icon (e.g., ╈), text or a number. The last type of link is the *Dependency link* (—▶—), which aims to model a dependency relationship between actors.

Figure 1 summarizes all the graphical symbols used in the GRL notation [19].
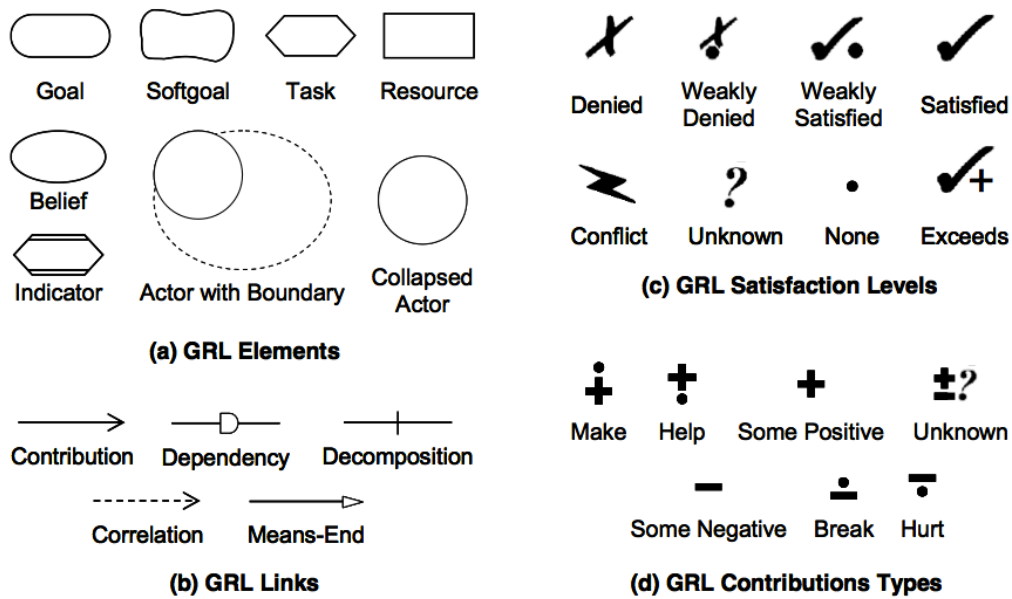
Figure 1   GRL Symbols (from ITU-T in [19], pp. 178)

2.3.2 Scenario modelling with URN

The Use Case Maps (UCM) notation, a subset of the URN language, is also a visual modelling language. UCM combines a view of behaviour and structural components that enables architectural reasoning as a first step. Afterwards, these models can be refined to more detailed scenario maps enabling refinements into the same kind of interactions than those allowed by UML sequence diagrams. However, unlike UML sequence diagrams and the activity diagrams, UCM maps offer the right abstraction level at the right moment

of the modelling process, it allows modellers to not be bothered with details such as data types in the high-level diagrams.

A UCM map is composed of paths and components; a path starts with a *start point* (●) and ends with an *end point* (▮). Along a path, *responsibilities* (✖) can be assigned which represent scenario steps or actions. UCM provides the concepts of choice and parallelism; *OR-fork* (─◁) / *OR-join* (▷─) and *AND-fork* (─╫) /*AND-join* (╫─), respectively. Due to the parallelism of the *AND-fork/AND-join*, the *AND-join* requires synchronization (unlike the *OR-join)*. Essentially, a fork node does not need to be followed by a join (i.e., diagrams do not need to be well nested), which might be confusing at first.

The concept of loop has no explicit concrete grammar syntax, although it can be modelled with a combination of *OR-fork* and *OR-join*. If the modeller needs the scenario to stop when a certain condition is triggered, the UCM notation offers two constructs: the *waiting place* (●) and the *timer* (🕘). A timer is a specialization of the waiting place. The timer handles conditions that do not occur in a certain time with a *timeout path.*

UCM *static stubs* (◇) offer a means of decomposing a complex monolithic UCM path into a vertical decomposition. To put it another way, a static stub contains one UCM sub-map, called *plug-in*. For situations where a stub contains more than one plug-ins (e.g., for enabling adaptive behaviour or architectural variation points), the UCM specification introduces the *dynamic stub* (◇). The adjective *dynamic* was chosen because at the execution of the scenario, only the plug-ins for which the precondition is true need to be followed. A complex path can be decomposed into multiple sub-paths by connecting the *end point* or empty path segment of one path to the *start point* or *waiting place* / *timer* of another path.

More advanced stubs (blocking (◇ₛ) and synchronizing(◇ₛ), with/without thresholds on output paths) also exist. Together with failure start points and aborts (for exception handling), advanced stubs enable complex workflow patterns to be captured concisely [26].

Figure 2 summarizes all the graphical symbols used in the UCM Notation [19].
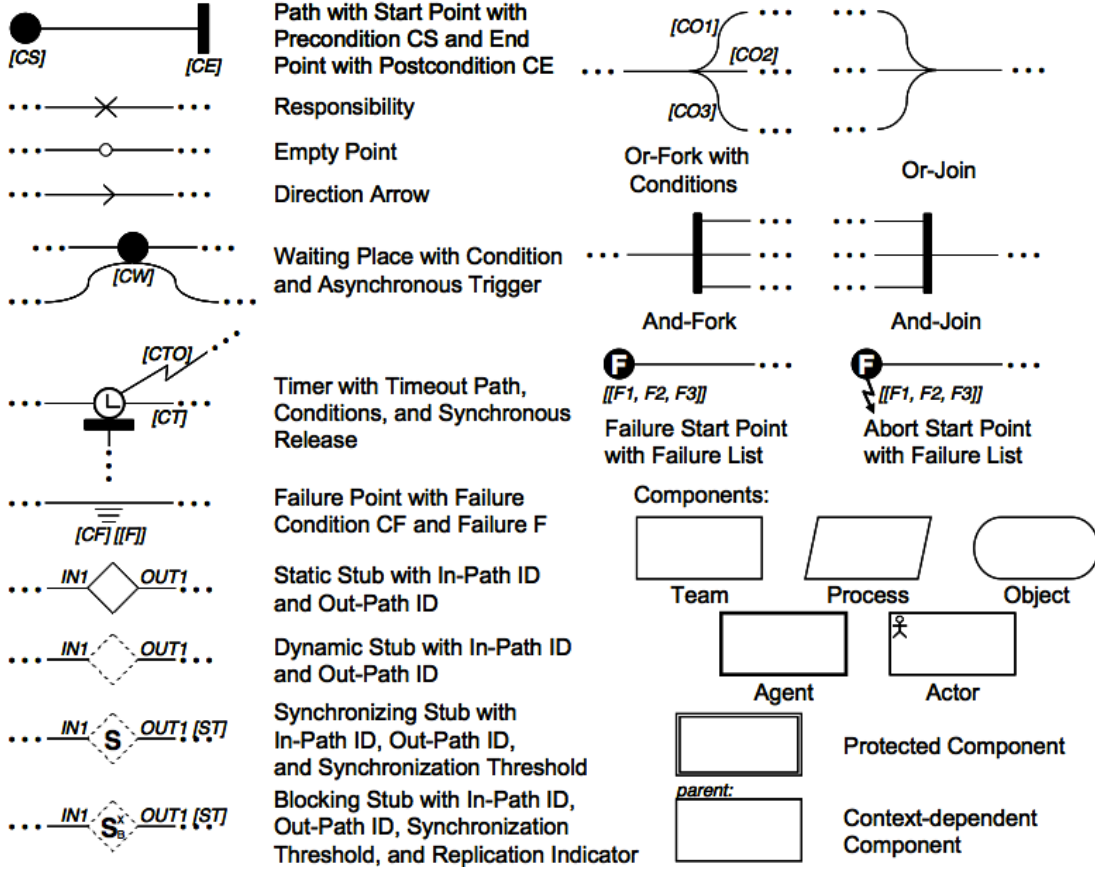
Figure 2   UCM Symbols (from ITU-T in [19], pp. 178)

2.3.3   Aspect-oriented modelling for URN

Aspect-oriented modelling (AOM) is a paradigm applied to modelling languages to en-capsulate concerns that are spread across the diagrams (i.e., crosscutting concerns). The benefit is to help focus someone's attention upon specific aspects individually. Dijsktra was the first to introduce the idea of separation of concerns [8].

The AOM paradigm is composed of three main notions; the *join point*, the *pointcut expressions* and the *composition rules*. A join point specifies at a precise loca-tion where a diagram could be modified by an aspect. A pointcut expression defines a "pattern of artefacts", matching the correspondent artefacts on a diagram. This match creates a mapping between the pointcut expression and the diagram. *Composition rules*

are the rules that attach an *aspect* (modified behaviour) to a diagram through the *pointcut expression*.

Mussbacher, in his Ph.D. thesis [27], extended GRL and UCM with aspect-oriented modelling concepts, leading to Aspect-oriented GRL (AoGRL) and Aspect-oriented UCM (AoUCM), respectively, and Aspect-oriented URN (AoURN) globally. Unlike UCM with its stubs, the GRL is a flat model, which means that it does not provide a vertical decomposition. Consequently, Mussbacher had to overload the concrete syntax grammar of GRL with the aims of designing AoGRL. For UCM, he exclusively reused the UCM notations.

Figure 3 depicts symbols of AoUCM and AoGRL, which illustrate the kind of diagrams that we may encounter in AoURN. In the following sections, we present an example to understand how to model a GRL diagram. Furthermore, we add an aspect to that diagram to show AoGRL capabilities. Finally, we connect a UCM diagram, to which, we attach an aspect to illustrate how AoUCM works.
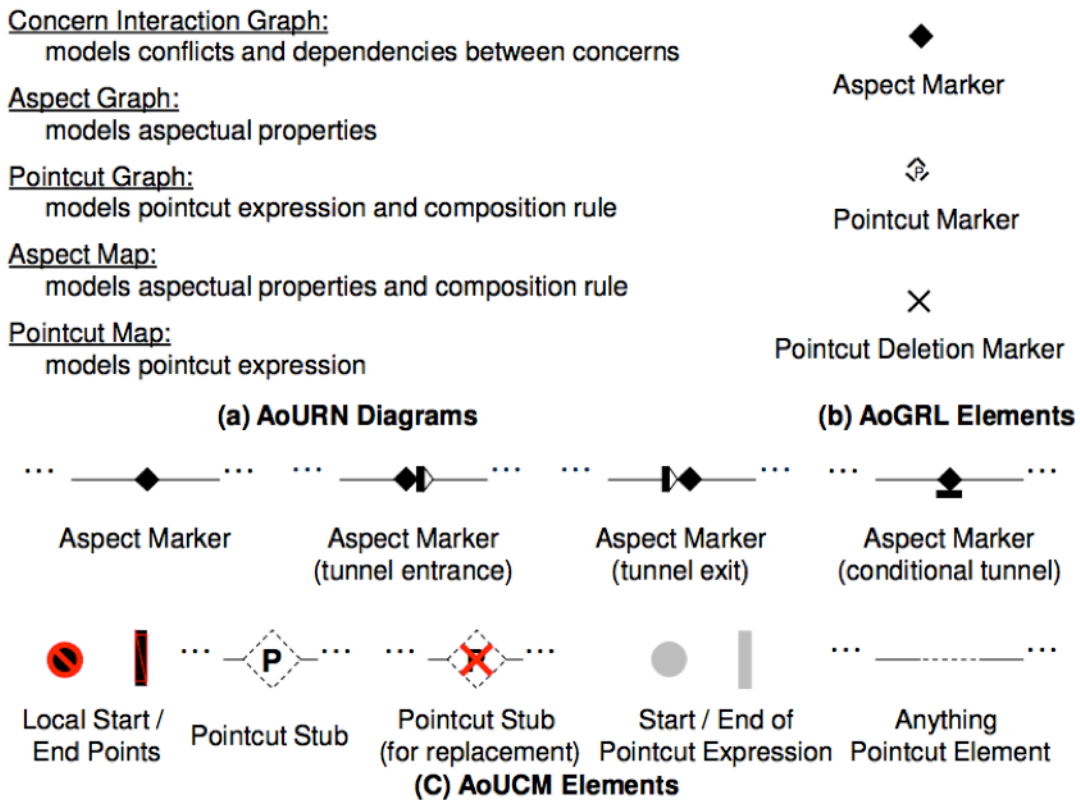
Figure 3   Summary of AoURN Elements (from Mussbacher in [27], pp. 138)

2.3.4 Example of GRL / AoGRL model

In this section, we present a simple illustrative GRL model. Figure 4 is a GRL diagram about a payment system. In order to handle the payment services of a bank institution, two choices can be taken: either relying on *open source technologies* or on *commercial technologies*. In addition, the development of one of these solutions can be supported either by *intern developers* or *extern consultants*. The GRL diagram can help support making and documenting a decision between these choices according to their cost and the available cash of the bank institution.
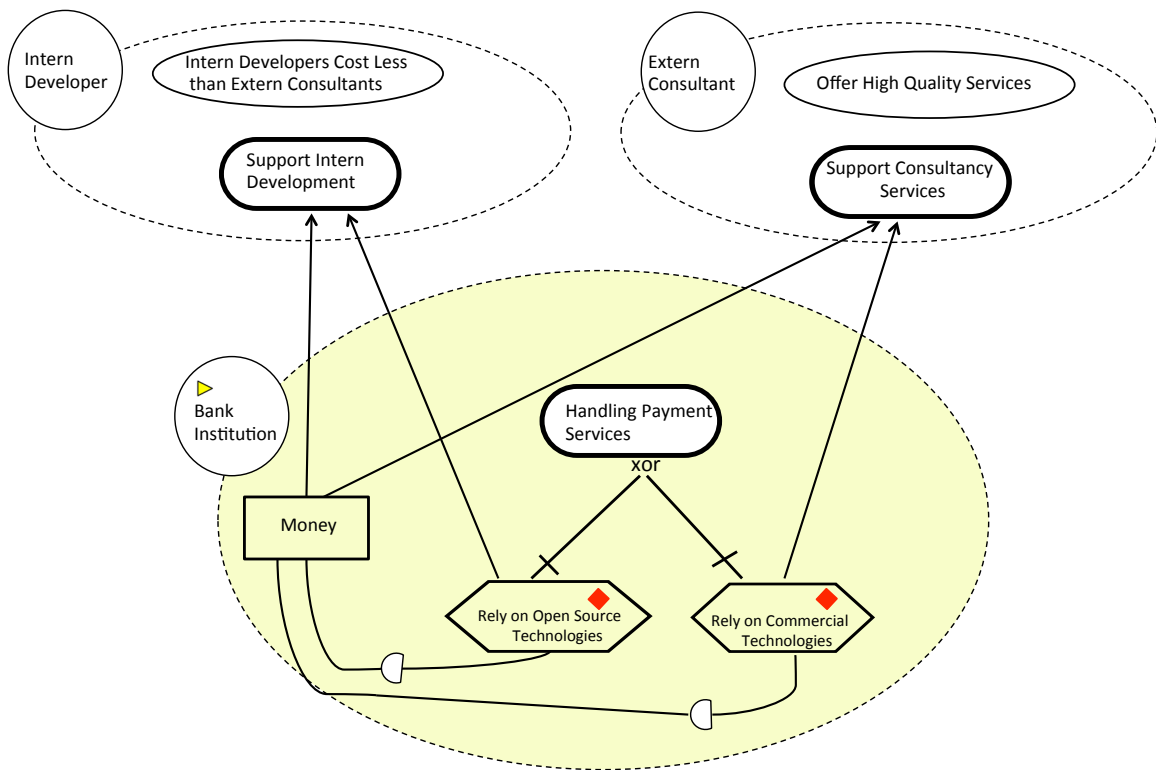
Figure 4   Payment System in GRL supporting decision reasoning

Security is crucial in the field of banking; thus, we should explicitly model it and represent it in our model. However, instead of adding it directly to the GRL model, we could use AoGRL capabilities. Assume that we want to add a security concern to all tasks decomposing the "Handling Payment Services" goal.

Firstly, we need to define elements of Figure 4 that we are interested in. To do that, AoGRL uses a pointcut graph (see Figure 5, (a)), which matches its elements tagged with pointcut markers ( ) to elements of the GRL diagram of Figure 4.

Secondly, we need to define an aspect graph, illustrated by Figure 5 (b). In this case, the aspect is attached (by a dependency link) to the tasks of the base model.

(a)   (b)
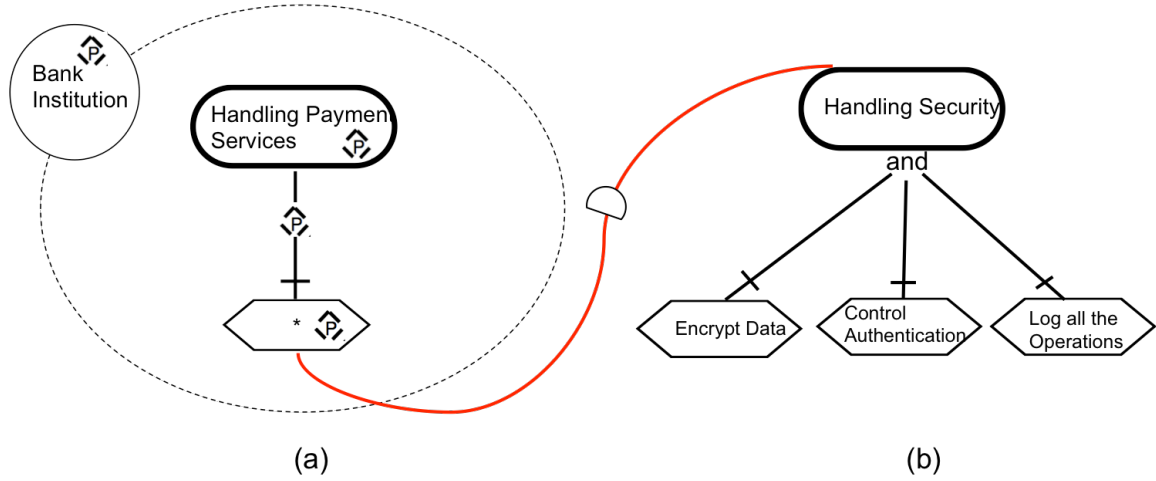
Figure 5 (a) Pointcut Graph to match elements of Figure 4 (using the symbol)
(b) Aspect Marker "Handling Security" attached to the match tasks of Figure 4

Once an aspect has been attached to an element of the GRL graph (Figure 4), that element will be tagged with an *aspect marker* (◆). We represent these directly on the base model (Figure 4). Each *aspect marker* is associated to an *AoView* as illustrated by Figure 6. To keep things simple, we represented only a part of the diagram.
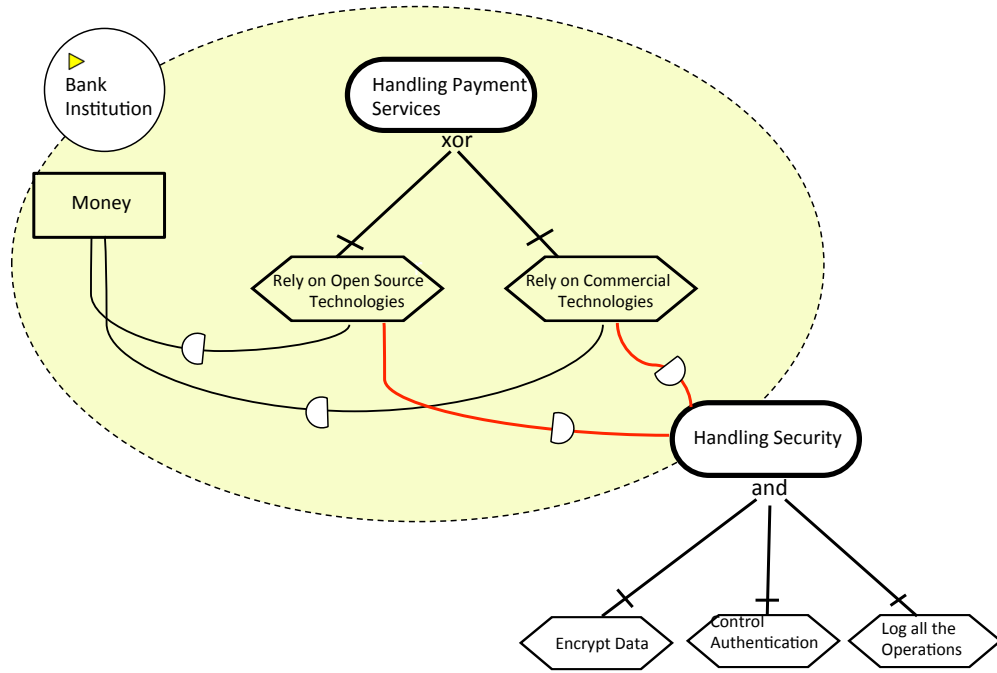
Figure 6   AoViews for tasks "*Rely on Open Sources Technologies*" and "*Rely on Commercial Technologies*"

2.3.5 Connecting GRL to UCM and Aspect to UCM

In order to show the ability of URN to connect goal and scenario models, let us assume that the goal *Handling Payment Services* is part of a more complex scenario including the actors *Client*, *Online Store*, *Delivery Service* and *Bank Institution*. Also, the UCM actor Bank Institution is traced to the GRL actor Bank Institution through a URN link (▷).

By using an editor such as jUCMNav (see Section 2.4), we can then associate either the plug-in maps presented in Figure 8 (a and b) to the dynamic stub *payment process*. The difference between the two paths (i.e., (a) and (b)) lies in the actor who processes the *do payment* responsibility. That UCM path will then influences the GRL Model (Figure 4).
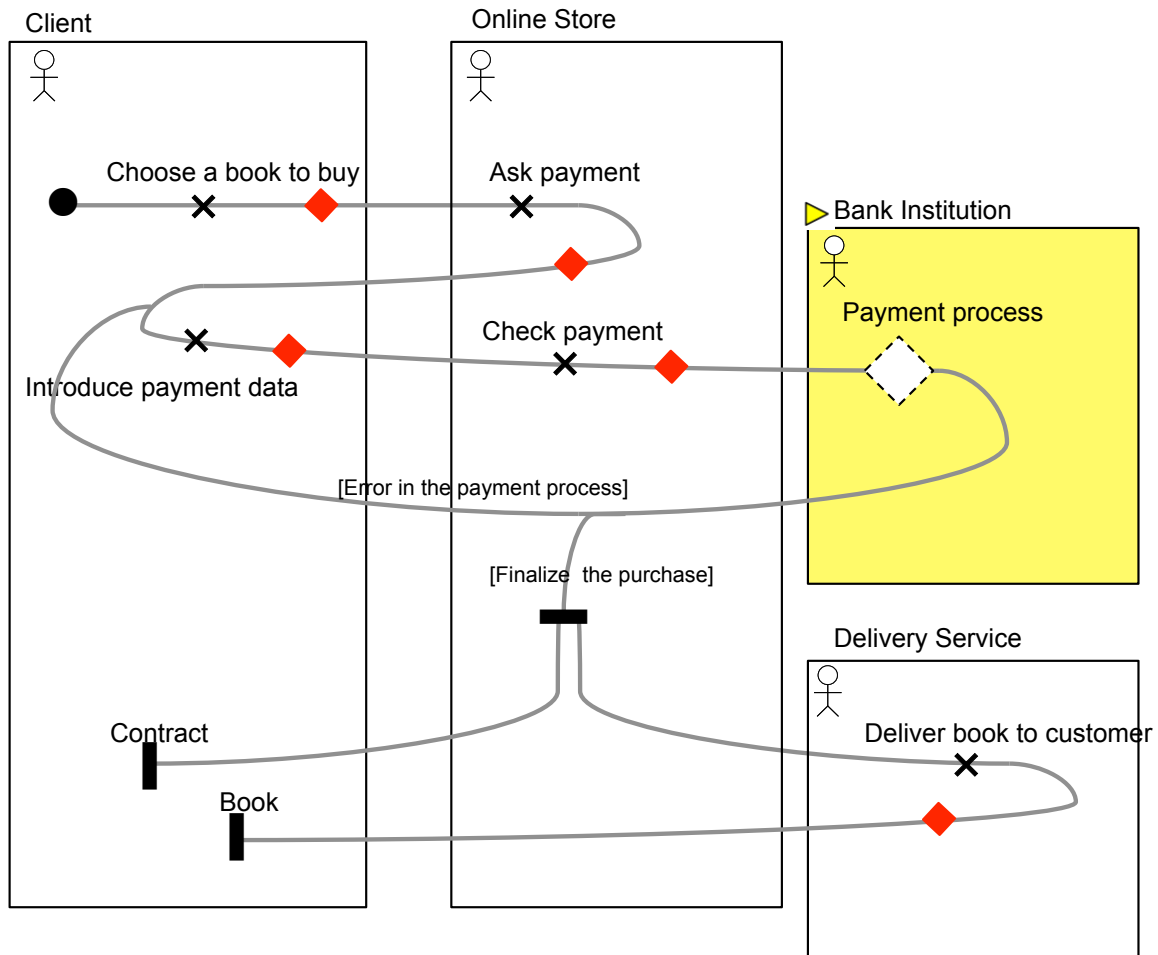
Figure 7   UCM diagram connected to GRL diagram

(a)   (b)
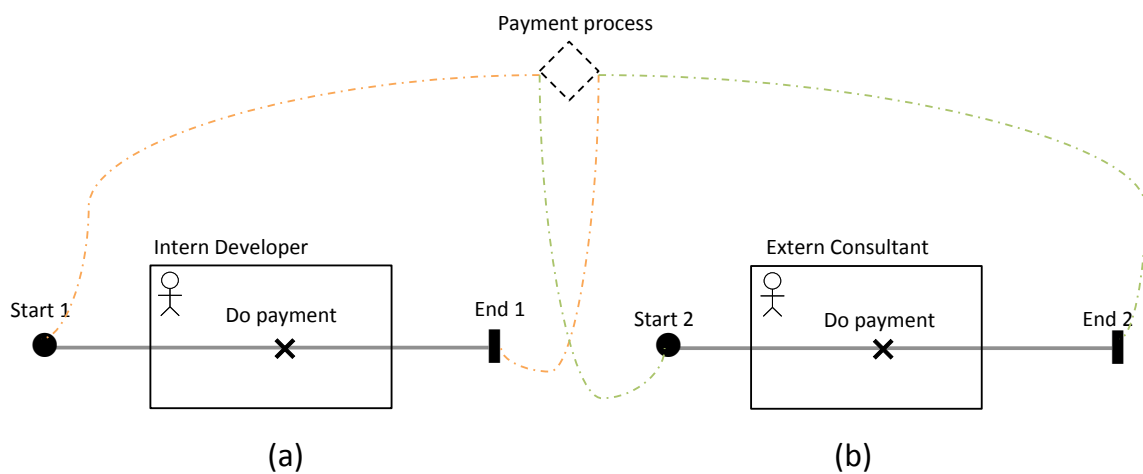
Figure 8   The payment process stub contains two UCM plug-ins (a) and (b)

Now, assume that we want to log all the responsibilities of the UCM model (see Figure 9) using AoUCM features. To log all the responsibilities, it is necessary to match them and then add a new responsibility "log" after each. The match part is represented by the pointcut map in Figure 9 (b). The special star (*) name of the responsibility means that it matches every responsibility on the base model, whatever its name. Then, the log responsibility is applied after every responsibility of the UCM diagram because it is located after the pointcut stub (⬦P).
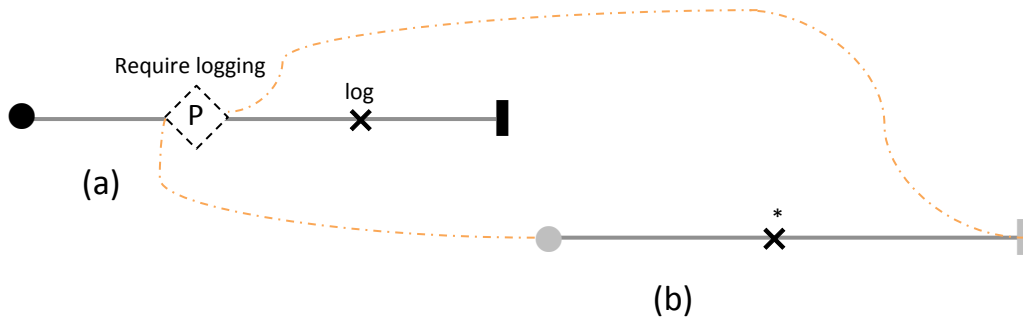
Figure 9   Logging Aspect, the path (a) is the aspect (i.e., Aspect Map) and the path (b) is the matching pattern (i.e., Pointcut Map)

Aspects are attached to a UCM path with aspect markers (◆), introduced at composition time. Each of them has an AoView that shows which aspect of an aspect map (see Figure 9, (a)) is connected to it. Figure 10 illustrates how an aspect is applied to an aspect marker by the *AoView* mechanism.
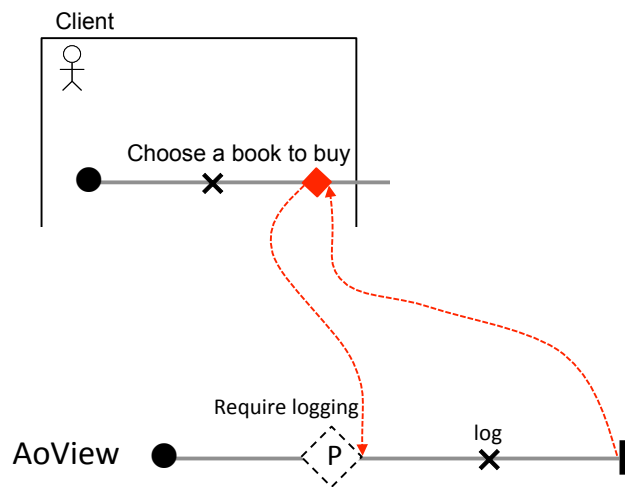
Figure 10   AoView mechanism for the first aspect marker (◆)

2.4. Tool Support

Nowadays, most of the modelling diagrams are made with Computer-Aided Software Engineering tools (CASE tools). Modelling tools enable several advantages over modelling on paper or generic diagrams tools, i.e., tools that do not check the syntax of diagrams against the metamodel of a modelling language. First, tools make modifications easier by supporting the changes or evolutions of the requirements throughout the modelling analysis process. Second, CASE tools increase coordination and handling of large projects. Third, they increase accuracy by providing ongoing checking systems that avoid language mistakes by using the metamodel of the language.

URN tool

Since 2005, jUCMNav has been developed at the University of Ottawa to support URN modelling (both GRL and UCM) [20]. The graphical editor (CASE) tool was developed before the standardization of URN. Therefore, jUCMNav's metamodel differs a bit from the standard but the newer versions tend to get closer to the URN standard.

The editor was developed as an Eclipse plugin. The Eclipse environment was chosen because of its popularity in the software development field and features offered such as generic support of metamodels or code generation facility. One of the most impressive features of jUCMNav is its ability of using Key Performance Indicators (KPI) to help the modellers make decisions using real world data.

Mussbacher has implemented the features to enable AoUCM modelling inside jUCMNav. Currently, there is no plan to support AoGRL on jUCMNav.

Chapter 3. Theories for Designing and Evaluating Visual Notations

In this Chapter, we introduce two of the most known theories to evaluate visual languages, the Green's Cognitive Dimensions of Notations and the Moody's Physics of Notations.

3.1. The Cognitive Dimensions of Notations

The Cognitive Dimensions of Notations (CDs) framework [16] developed by Thomas R.G. Green is a "*broad-brush evaluation*" technique that can be used for notational design, user interfaces, and programming languages.

Green brought, to some extent, a framework that specifies a common language to speak in a more formal way about design artefacts. CDs could be viewed as an analogy to the fundamental dimensions in physics (i.e., mass, length and time), with which every physical phenomenon can be described according by only these three dimensions. CDs are then related to human cognition processes.

Thirteen dimensions have originally been defined; most of them are related to psychology, human-computer interaction, modelling design, and software engineering.

3.1.1 Limitations of the CDs Framework

Several issues have been raised on the operationalization of the CDs framework for evaluating and designing visual notations. Here are some of the limitations. Firstly, the scope of the Green's framework covers a broad spectrum of fields, which could reduce its applicability for specific fields such as visual language assessment. Secondly, the foundation of the dimensions lies on fundamental dimensions in physics and, like them, they only aim to represent a situation and not give any clue on how bad or good a visual artefact is. Thirdly, due to the framework high-level analysis, it seems unlikely that following

it guarantees any predictive results like a global improvement of a modelling language visual notation.

Moody applied the Gregor's taxonomy to evaluate theories and he stated that the CDs are unscientific. Indeed, according to Moody, the CDs is a **Type I** theory, i.e., a theory for analysing and describing. These kinds of theories were defined by Gregor as unscientific as they lack testable propositions and cannot be *falsified*[1].

3.2. The Physics of Notations

In the following section, we present Moody's Physics of Notations theory (PoN), which is a theory to evaluate and design visual languages. Moody evaluated the PoN against the Gregor's taxonomy as a **Type V** theory: a theory for design and action.

The PoN theory consists of nine principles that address both the evaluating and designing issues of a visual notation. The principles were synthesized from theory and empirical evidence from various scientific disciplines such as cognitive and perceptual psychology, cartography, human computer interface and communication [10][25]. The Physics of Notations theory relies on the idea of falsifiability of its principles, which states that they can be empirically testable [31]. In contrast, the CDs are not testable because there are vaguely defined as noted in [10], [25] and [15].

More fundamentally, each principle is conceived to maximise the **<u>cognitive effectiveness</u>** of visual notations, i.e., optimising how visual notations are processed by the human mind. Moody defined it as the *speed*, *ease* and *accuracy* with which a representation can be processed by the human mind.

---

[1] Falsifiability is a criterion, developed by Popper, which states that a theory is scientific only if it is possible to establish that is false.

3.2.1 Background for Understanding the PoN

Vocabulary correspondence

Before going further, it is necessary to clarify the vocabulary used by Moody.

Correspondence	
Abstract grammar syntax	Metamodel of the modelling language / semantic level / ontological theory
Concrete grammar syntax	Visual syntax
Artefacts of the abstract grammar syntax	Semantic constructs
Artefacts of the concrete grammar syntax	Graphical symbols / symbols

Visual variables

The PoN theory uses the Bertin's visual variables [5] (see Figure 11), defined as "*a set of atomic building blocks that can be used to construct any visual representation*" (i.e., graphical symbol).
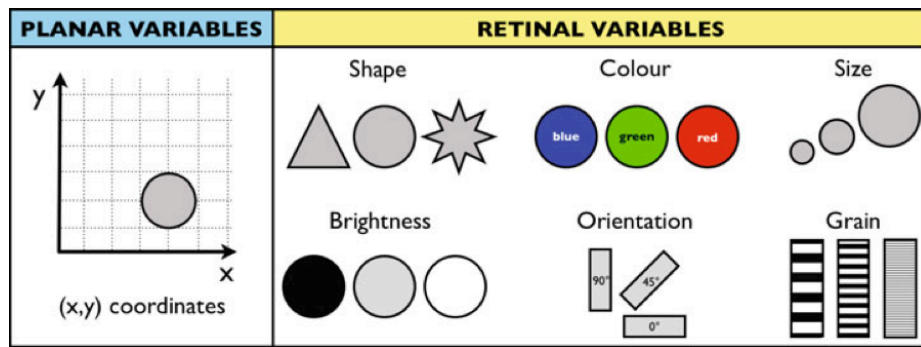
Figure 11 Bertin's Visual Variables (from Genon *et al.* in [10])

3.2.2 Principle of Semiotic Clarity

Principle formulation: "*There should be a 1:1 correspondence between semantic constructs and graphical*".

In order to evaluate the anomalies of this principle, four criteria have been proposed: symbol redundancy, symbol overload, symbol excess, and symbol deficit. These anomalies are represented in Figure 12.

Symbol redundancy: multiple graphical symbols are used for one semantic construct.

Symbol overload: multiple semantic constructs use the same graphical symbol

Symbol excess: graphical symbol that has no corresponding semantic construct.

Symbol deficit: semantic construct without any representation.
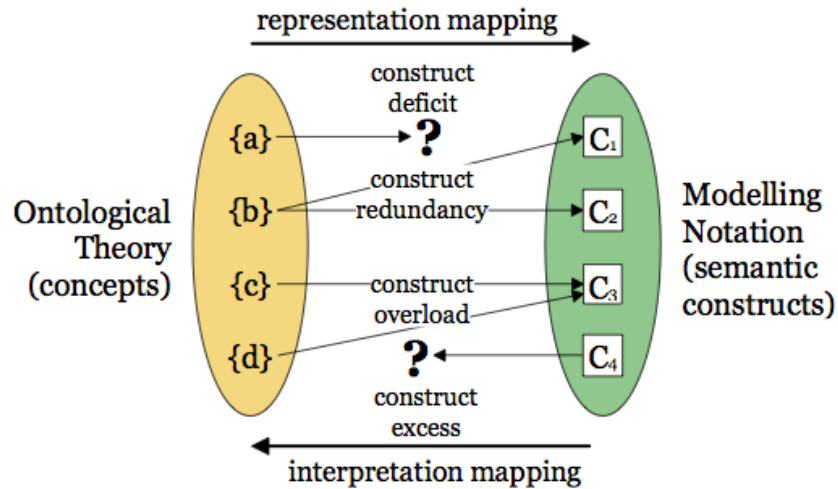
Figure 12   Mapping between the semantics constructs (left)
and the graphical symbols (right) (from Moody in [25])

3.2.3 Principle of Perceptual Discriminability

<u>Principle formulation</u>: *"Different symbols should be clearly distinguishable from each other"*.

Moody identified four criteria that increase the Perceptual Discriminability between graphical symbols: visual distance, primacy of shape, redundant coding, and perceptual popout.

Visual Distance: Moody stated that symbols are determined by their visual distances, i.e., the number of Bertin's visual variables on which they differ.

Primacy of Shape: The emphasis was made on the particular role that shapes play on the discriminability of two graphical symbols. Therefore, we should give more importance to Bertin's visual variable shape.

Redundant Coding: This criteria emphasis on using multiple visual variables to increase the visual distance between graphical symbols.

Perceptual Popout: A graphical symbol should have at least a unique value (the popout effect) on one of the Bertin's visual variables compared with the values of the other symbols, thus enabling the graphical symbol to be more easily distinguishable.

3.2.4 Principle of Semantic Transparency

Principle formulation: "*Use visual representations whose appearance suggests their meaning*".

The graphical symbols should suggest what is their semantics (i.e., meaning). This principle significantly reduces the *cognitive load* since there is no need to memorize (i.e., to learn) the graphical symbol meaning.

Moody defined a range to classify symbols according to how they suggest their meaning. Therefore, a symbol could be perceived as (see Figure 13, from left to right):

- **Semantically Perverse** if its appearance infers the opposite of its meaning.
- **Semantically Opaque** *(or conventional)* if its appearance does not give any clue about its meaning neither in a positive or in a negative way.
- **Semantically Immediate** if its appearance alone suggests its meaning
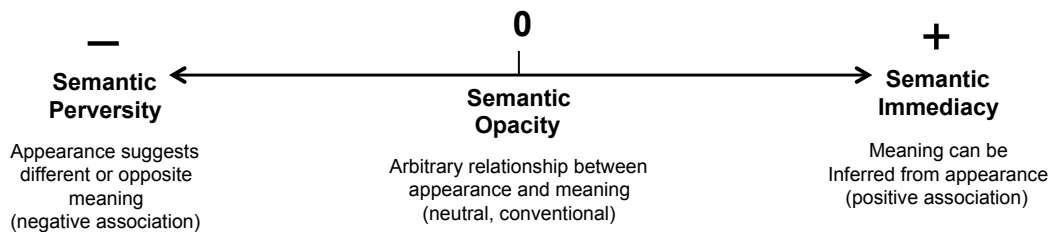
Figure 13 Different possible states of a graphical symbol according to its positive, neutral or negative impact on cognitive load. (from Moody in [25])

3.2.5 Principle of Complexity Management

Principle formulation: "*Include explicit mechanisms for dealing with complexity*".

This principle assesses complexity of the diagrams. Moody distinguishes the complexity of the graphical symbols composing the notation (see Section 3.2.9) from the diagrammatic complexity. The diagrammatic complexity is related to the number of sym-

bol instances that appear on the diagram. For the latter purpose, two mechanisms are illustrated: modularization and hierarchy. Both should be viewed as guidelines to improve the level of complexity management.

Modularization: The overall complexity could be reduced if the visual notation of the modelling language allows us to divide a complex system into smaller parts. By doing so, we would improve the speed and accuracy of understanding of the diagrams.

Hierarchy (levels of abstraction): Complexity could be handled by decomposing a complex diagram into a hierarchy of diagrams at different levels of abstraction. Indeed, hierarchical decomposition is recognized as the most effective way to organise complexity for human comprehension.

3.2.6 Principle of Cognitive Integration

Principle formulation: "*Include explicit mechanisms to support integration of information from different diagrams*".

While the previous principle divided a monolithic system into multiple diagrams to reduce the complexity, this principle aims to give a solution to integrate these spread diagrams into one cognitive representation of the initial system.

Conceptual integration: The visual notation should offer a means to summarize the diagrams in one integrated diagram. Also, the environment of the modelling language should offer the feature of representing the rest of the system according to our current focus of the subsystem (Figure 14).
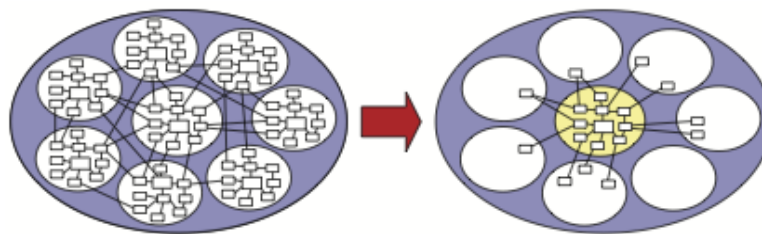
Figure 14   Contextualization Applied to System (in yellow)   (from Moody in [25])

Perceptual integration: The environment of the visual notation should support features that could help for navigating through diagrams.
The environment should provide:

- *Identification*, a means of identifying the diagrams
- *Level numbering*, if the system is decomposed into hierarchy abstraction
- *Navigation cues*, helping the user to know where he currently is in a complex system
- *Navigation map of the diagrams*, allowing easier transitions between diagrams.

3.2.7 Principle of Visual Expressiveness

Principle formulation: "*Use the full range and capacities of visual variables*".

This principle suggests that a visual notation should use a maximum of the Bertin's visual variables to encode information. Conveying information through different ways is cognitively more efficient. Therefore, the more visual variables are used, the better.

Most of visual notations convey information only though one visual variable: the shape. Even though, the graphical form of a shape is one of the most recognizable (see Section 3.2.3), its expressiveness is one of the least powerful. Moody is referring to most modelling languages using shapes as boxes where the meaning is written textually (see Figure 15) without any meaning of the shape itself. Other visual variables such as colour should be used in these visual notations.
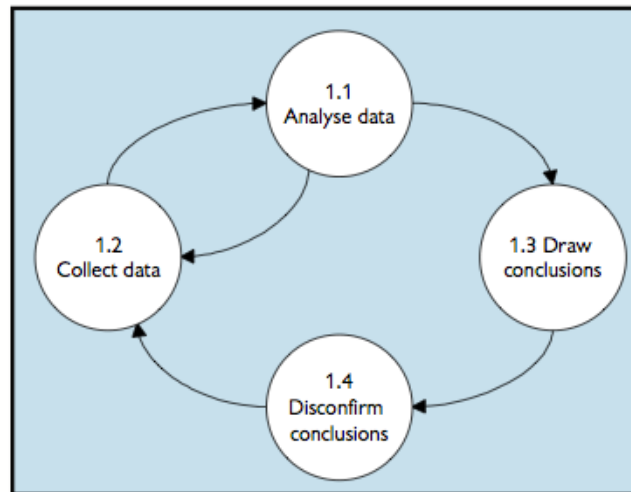
Figure 15 Only the visual viable shape is used in DFDs (from Moody in [25])

Moody has proposed a simple and efficient measurement for this principle by comparing the used and the not-used visual variables (called the free variables). The focus is put on new ways to convey information (i.e., the free variables). Figure 16 illustrates at what extent visual variables are used in a visual notation, from zero to eight. In the case of Figure 16, the visual notation uses three visual variable.
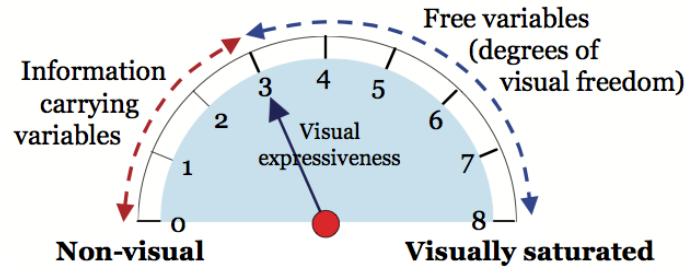
Figure 16 Odometer for used and not-used visual variables in for a visual variable (from Moody in [25])

Moreover, Figure 17 gives for each visual variable the kind of information that can be encoded (interval, ordinal and nominal) and its range of perceptive values. For instance, the visual variable "Texture" is nominal (it represents fixed states) with a range 2-5 (a maximum of 5 textures could be set in a visual notation).

Variable	Power	Capacity
Horizontal position (x)	Interval	10-15
Vertical position (y)	Interval	10-15
Size	Interval	20
Brightness	Ordinal	6-7
Colour	Nominal	7-10
Texture	Nominal	2-5
Shape	Nominal	Unlimited
Orientation	Nominal	4

Figure 17   Visual variables in accordance with their perceptive range values. (from Moody in [25])

3.2.8 Principle of Dual Coding

Principle formulation: "*Use text to complement graphics*".

Information conveying is more effective by using visual variables instead of text but it has been demonstrated that using both at the same time increase even more the effectiveness of information conveying.

PoN suggests using two techniques to support Dual Coding: annotations and hybrid symbols.

Annotations: The visual notation should include an annotation construct, which allows modellers to write textual explanations directly on diagrams instead.

Hybrid (Graphics + Text) Symbols: Text could be used on symbols to reinforce and expand their meaning.

3.2.9 Principle of Graphic Economy

Principle formulation: "*The number of different graphical symbols should be cognitively manageable*".

Over time and due to modelling language evolutions, the visual notations tend to increase their graphical symbols because of the growth of the semantic constructs, which is normal according to the Semiotic Clarity principle (Section 3.2.2). Nevertheless, it is necessary for each evolution of a visual notation to gauge the impacts against human cognitive limitations. Perhaps certain graphical symbols are not important and can be removed from the visual notation.

3.2.10 Principle of Cognitive Fit

Principle formulation: "*Use different visual dialects for different tasks and audiences*".

The SE notations do not consider the different audiences that take place in the modelling process and the result is that the graphical symbols made for experts have to be understood by novice audiences. The cognitive fit addresses this issue. The theory encourages using at least two sets of graphical symbols: one for the experts and the other for the novices.

In addition, the cognitive fit could be applied when different representation media are used, such as whiteboard modelling or computer-based modelling.

3.2.11 Interactions Among Principles

The physics of notations principles are not mutually exclusive. Thus, trying to improve the visual notation against one principle can lead to negative impacts in another principle. On the positive side, some principles have a positive impact to one another. Figure 18 illustrates such positive, neutral, negative, and undefined interactions.

	Semiotic Clarity	Perceptual Discriminability	Semantic Transparency	Complexity Management	Cognitive Integration	Visual Expressiveness	Dual Coding	Graphic Economy	Cognitive Fit
Semiotic Clarity								±	
Perceptual Discriminability				+				+	
Semantic Transparency	+							±	
Complexity Management							−	+	
Cognitive Integration	−			+			−		
Visual Expressiveness		+					+	±	
Dual Coding								+	
Graphic Economy		+	+			−		+	
Cognitive Fit									

Figure 18   Interactions among the Physics of Notations principles
(from Moody in [25])

Part II: Problem Statement & Contributions

A. Problem Statement

The goal of the thesis is to operationalize the PoN. In Section 3.2, we saw that the PoN was analysed by Moody as a Type V theory according to Gregor's Taxonomy. Based on her taxonomy, a Type V theory should provide to the PoN the power to be fully operationalizable (see Section 4.2.1). Yet, Störrle and Fish [39] argue that the PoN, in its current form, is neither precise nor comprehensive enough to be objectively applied to visual notations.

Nevertheless, Genon *et al.* applied the PoN to several visual notations (e.g., [10][11]). Therefore, we propose to evaluate the PoN according to Gregor's Taxonomy. Although Moody has already assessed its theory, we think that it was done in a very coarse-grained level. In Chapter 4, we propose an fine-grained evaluation relying on the evaluation of every of the nine principles. This evaluation will validate whether the PoN is a Type V theory.

In Chapter 5, we assess how Störrle and Fish have operationalized the PoN by comparing their suggestions to those made by Genon *et al.* The main purposes are to (1) operationalize the PoN and (2) to give a constructive critic about how the PoN should be applied. Indeed, Störrle and Fish comprehended and applied the PoN differently from Genon *et al.* In addition, we propose our own versions of metrics to support the PoN. However, we restrict that analysis to two principles of the PoN: Semiotic Clarity and Perceptual Discriminability.

Finally, in Chapter 6, we evaluate our metrics by applying them to AoURN.

B. Contributions

This section gives the contributions of this thesis tagged as either *major* or *minor* according to how they help operationalize the PoN.

Chapters	Contributions

Chapter 4. Critics of the Physics of Notations	• Major
	    o A detailed analysis of the PoN against Gregor's taxonomy
	• Minor
	    o A method to assess the PoN with Gregor's taxonomy

Chapter 5. Operationalization and Critics of two PoN Principles	• Major
	    o A constructive critic of the operationalization of the PoN (Semiotic Clarity + Perceptual Discriminability) as proposed by Störrle and Fish in [39]
	    o A constructive critic of the Genon *et al.* operationalization of the PoN [10][11]
	    o A discussion about how to improve the operationalization of PoN

Chapter 6. Evaluation: Analysis of AoURN	• Major o An evaluation of our suggestions to operationalize the PoN on the Semiotic Clarity and the Perceptual Discriminability principles (applied on AoURN)

	• Minor
	o An evaluation of the *cognitive effectiveness* of AoURN. We applied both Störrle and Fish, and our metrics for the Semiotic Clarity and the Perceptual Discriminability. For the other PoN principles, we applied the Genon *et al*. metrics
Chapter 7. Future Work	• Minor
	o A discussion and a proposal to extend the Bertin's visual variable to animations
	o A set of examples that use the visual variable animation in CASE tools to support some PoN principles
	o A implementation (Java Software) of the Semiotic Clarity and the Perceptual Discriminability metrics to automatize the evaluation of a visual notation

Chapter 4.   Critics of the Physics of Notations

In this chapter, we apply Gregor's taxonomy to the PoN. As stated by Moody, the PoN is (should be) a Type V theory, i.e., a theory for *design and action*. Although Moody has already applied the taxonomy on the PoN, we think that it was done at a very coarse-grained level. Therefore, we propose an exhaustive evaluation relying on the evaluation of every of the nine principles. The goal pursued is to evaluate the scientific degree of the PoN and its operationalizability.

4.1.   Falsifiability of the Physics of Notations

A meta-theory, a very high level of abstraction theory, is required to evaluate the level of the PoN. Evaluation of theories was done for centuries by philosophy and in particular by epistemology. However, our thesis is related to SE modelling languages and is thus in the Information Systems (IS) field. This consideration drives us to find a meta-theory related to IS which is easily applicable and enables scientific evaluation of a theory. Moreover, Moody has already used a meta-theory to evaluate the PoN. Therefore, if we select the meta-theory that was previously chosen and that we apply it in a more detailed way, it should give us the scientific degree of the PoN. Moreover, it will indicate how the meta-theory behaves under detailed and high-level analysis. As a result of all these considerations, Shirley Gregor's "*The Nature of Theory in Information Systems*" is the most suitable candidate meta-theory.

4.2.   The Nature of Theory in Information Systems

The Nature of Theory in Information Systems (IS) [17] is a taxonomy that classifies information systems theories into one of the five interrelated types of theory: (1) theory for

analysing, (2) theory for explaining, (3) theory for predicting, (4) theory for explaining and predicting and (5) theory for design and action.

The taxonomy draws upon various fields of science including epistemology. Some philosophers, such as Karl Popper ([30], [31]), significantly contributed to the philosophy of science. More importantly, Gregor's work is also significant because it synthesizes different perspectives into one coherent meta-theory.

The core of the taxonomy relies on theory issues of causality, explanation, prediction and generalization. Indeed, a theory should have a certain degree of generalization and abstraction. Moreover, philosophers illustrated that it is important for an explanation to include a notion of causality. Finally, a theory can be tested when predictions are made possible in the theory statements.

Considering these issues, Gregor spotlights four primary goals of a theory: **(a)** analysis and description, **(b)** explanation, **(c)** prediction and **(d)** prescription. From theses goals, five types of theory emerged; (Type I) *Theory for Analyzing* from goal **(a)**, (Type II) *Theory for Explaining* from goal **(b)**, (Type III) *Theory for Predicting* from goal **(c)**, (Type IV) Theory for Explaining and Predicting from goals **(c)** and **(d)**, and finally, (Type V) Theory for Design and Action from goal **(d)**. Figure 19 illustrates how the theory types were modelled.
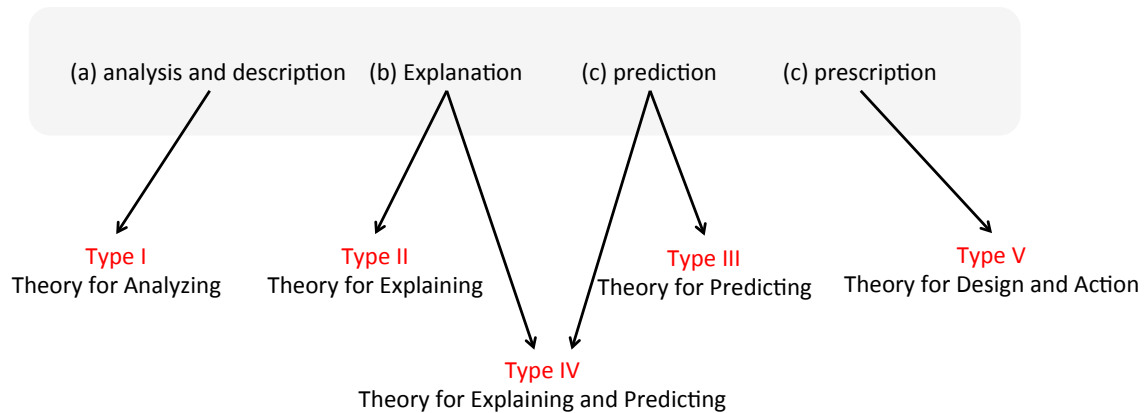
Figure 19   From the Theory Goals to the Theory Types

Gregor insisted that a theory of a certain type is not more valuable than a theory of another type. The misunderstanding is related to choosing ordered numbers to name the theory (Type I, Type II, etc.). Common sense suggests that Type V is a lot more valuable then Type I but as stated by Gregor: "each class of theory can provide important and valuable contributions".

4.2.1 Gregor's Theory Types

Type I: Theory for Analyzing

Theories of Type I focuses on questions of type "what is". These theories include classification schema, frameworks, and taxonomies. They provide neither causal effects to their core statements, nor any predictive statement. So, theories of Type I cannot be tested empirically, which is a true weakness of this type of theory.

But then, what is the benefit of this type? Gregor stated that theories of type I are valuable when little is known about some phenomena. Type I can be viewed as a foundation to any scientific theory.

Type II: Theory for Explaining

Type II analyses the "how" and "why" a phenomenon occurs. A theory of Type II may contain testable statements, however, testability is not the main concern of this theory type. Gregor labelled this type as "theory for understanding": the focus is put on explaining phenomena. Theories that fall into this category include, amongst others, case studies, surveys, phenomenological and hermeneutic[2] theories. Gregor states that Type II theories should be new and interesting, or imperfectly understood beforehand.

Type III: Theory for Predicting

Unlike the Type II theories, the Type III theories focusing on prediction making (i.e., what will be). They do not address the "why". These theories make predictions without explaining in a detailed way every factor that participates in the prediction. For instance, we could predict that it will rain because the sky goes darker. The true reasons depend on

---

[2] Hermeneutics is a theory of understanding and interpretation of linguistic and non-linguistic expressions

more complicated factors than only the "colour" of the sky. However, what is important is the fact that it will rain. The complicated factors are not part of that concern.

The main benefit of the Type III theories is that they are pragmatic theories that can predict effects without considering the detailed factors. The more precise the predications, the better.

Various field extensively use this theory type including finance. Another example in the IS is Moore's Law. However, the limitations of this type of theory should be known because even though two variables are perceived as correlated, it does not imply a causal relationship between them. In other words, the correlation between the two variables is coincidental.

Type IV: Theory for Explaining and Predicting

Type IV is the preconceived vision of a scientific theory. This type encompasses questions such as "what is", "why", "when", and "what". Theories intersecting the explaining and predicting theory types allow to predict and explain the underlying causes. Therefore, the statements of the theory are fully testable, which guarantee a certain level of assessment, and hence, the falsifiability of the theory. The theory of information described by Shannon [34] falls into this category.

Type V: Theory for Design and Action

Type V focuses on "how" to do something. This kind of Type V theories are extensively used in IS and can be viewed as implementation of processes. It includes artefacts such as methods, functions, and proof of concept. This type contributes to knowledge by providing these artefacts.

However, in the context of IS, every piece of software is composed of this kind of artefacts. In order to not consider each IS theory as a Type V theory, Gregor adds extra criteria to distinguish a theory from a "simple" piece of software. The criteria are, for instance, the novelty, simplicity, completeness, and consistency of the artefacts used by the theory.

The IS includes a lot of Type V theory, such as the Codd's theory of relational database [9]. In that particular case, the theory is considered belonging to Type V because

its underlying principles are operationnalizable and, therefore, it explains how to put the theory of relational database into practice.

4.2.2 Interrelationships among Theory Types

The different types of theory exposed by Gregor have interrelationship dependencies. As presented in Figure 18, the Type I is the foundation to all the others types. Type II and Type III contribute to form Type IV or Type V theories. Finally, Type IV and V are strongly interrelated because, on the one hand, we need "scientific" theories (Type IV) to produce interesting artefacts, and, on the other hand, some Type IV theories can be validated by implementing them (type V).
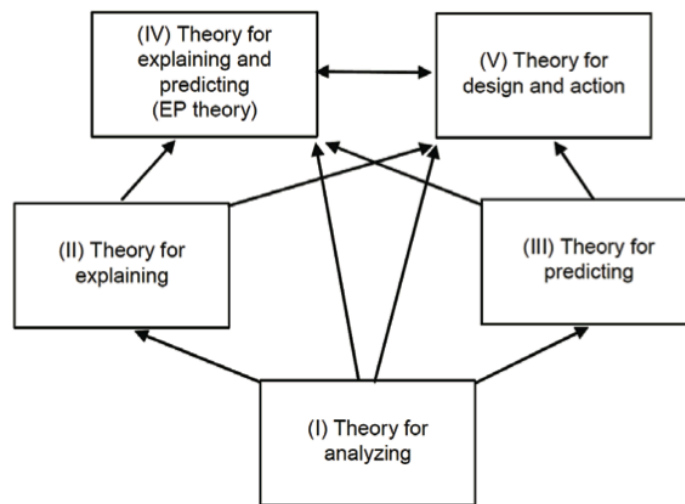
Figure 20   Interrelationships among Theory Types (from Gregor in [17])

4.3. Assessment of the Physics of Notations

Moody referred twice to the Gregor terminology to analyse the PoN. He analysed the foundations of its theory (titled "how visual notations communicate") and the PoN itself.

In the following sections, we will describe and criticize the foundations of the PoN. Then, we analyse the physics of notations in the same way that Moody did but we do that in a detailed way, i.e., principle by principle.

4.3.1 Assessment of how visual notations communicate

Moody designed the foundations of the PoN theory by applying (specifying) Shannon's theory and Weaver's theory of communication to the modelling languages. Moody used diagram as the main artefact of the communication theory. Furthermore, the encoding and decoding processes (of the communication theory) have been replaced by the notions of *design space* and *solution space*, respectively.

Design Space (Encoding Side)

Moody identified a means of encoding information by using Bertin's visual variables (see Figure 11). With these visual variables, he conceived the notion of *primary notation* (i.e., the graphical symbols and their meaning defined by the visual notation) and *secondary notation* (i.e., the graphical symbols and their meaning, which are not explicitly defined by the visual notation). Also, the noise artefact of the communication theory has been specified as "*unintentional use of random variation in visual variables that conflicts with or distorts the intended message*".

Solution Space (Decoding Side)

Moody divided the decoding side into two phases: perceptual processing and cognitive processing, which distinguish the seeing part from the understanding part. Figure 21 represents both perceptual and cognitive processing.
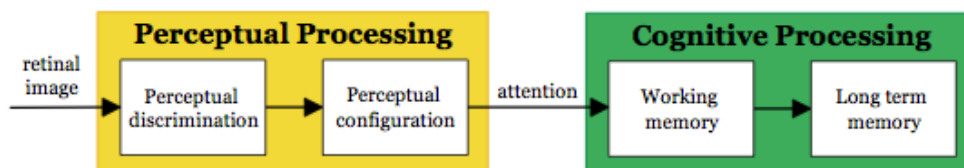
Figure 21   Human information processing (from Moody in [25])

Critics on the Assessment

First of all, we should distinguish the Information Theory developed by Shannon, which falls in Gregor's Type IV theory from the Communication Theory, which was not been assessed by Gregor. The Information Theory explains information in terms of physical laws; it is a mathematical quantification of information. In "The Mathematical Theory of Communication", Shannon and Weaver simplified the Information Theory to design the communication theory. Nowadays, what was a mathematical theory has become a theory more related to human science than physics.

The communication theory as explained in PoN theory falls in the human science field and not in the mathematical or physics field. Therefore, it is difficult to draw any firm conclusion from it.

If we consider that the information theory and the communication theory are strongly related to each other, we could then categorize the communication theory as a Type IV theory as well.

The design space and the solution space are a specialization of the artefacts of the communication theory to the world of visual notations. Therefore, the design and solution spaces do not conflict with the scope of the communication theory. We could then conclude that the theory of "how visual notations communicate" is a Type IV theory.

4.3.2 Moody's claim about the Physics of Notations

Moody assessed his theory as a Type V theory: a theory for design and action. Moreover, repeatedly, Moody insisted on the falsifiability of each of the nine PoN principles. Noticeably, he emphasized the scientific value and the operationalization of the PoN principles.

4.4. Detailed Analysis of PoN according to Gregor's Taxonomy

In this section, we assess the level of the PoN against the Gregor's taxonomy. In order to analyse at a fine-grained level, we consider each principle as an independent theory. We proceed with a two-step analysis: filling the *structural components of theory* and discussing the theory type.

The first step allows us to identify the composition of the theory. While the second step is a discussion of the theory type based on the components of the structural components of theory.

The components of the first step, defined by Gregor are as follows:

Components Common to All Theory

- **Means of representation**: Physical representation of the theory (e.g., words, mathematical terms, diagrams)
- **Constructs**: phenomena of interest in the theory
- **Statements of relationship**: relationship among the constructs
- **Scope**: the degree of generality of the *statements of relationships*

Components Contingent on Theory Purpose

- ➢ *Causal explanations*: *statements of relationships* among phenomena that show causal reasoning
- ➢ *Testable propositions*: *statements of relationships* are stated in such a form that they can be tested empirically.
- ➢ *Prescriptive statements*: statements, which specify how *statements of relationship* can accomplish something in practice.

The "*Components Common to All Theory"* are mandatory while the "*Components Contingent on Theory Purpose"* are not. The latter is the most important group of components because we can derive the theory type from it.

We will classify the different principles in accordance to —Table 1.

Table 1 Theory Type in Accordance with the three Components.

	Type I	Type II	Type III	Type IV	Type V
Causal relationships	No	Partly	Partly	Yes	No
Testable propositions	No	May Contain[3]	Yes	Yes	No
Prescriptive statements	No	No	No	No	Yes

Gregor considered only the "*Components Contingent on Theory Purpose"* to classify a theory. Furthermore, the scope of a theory is not taken as a *primary classification*; however, Gregor suggested that the level of generality could be subjected to *secondary classification* to distinguish between the theory types.

The assessment of the level of generality depends on *our appreciation* and *our expertise* acquired after analysing how Gregor assessed the scope of each theory. As this appreciation is not taken into consideration for the type level, there are few consequences on giving a subjective appreciation.

4.4.1 Principle of Semiotic Clarity

Structural Components of Theory	
Theory Component (Components Common to All Theory)	**Instantiation**
Means of representation	Words, Figure
Constructs	Semantic constructs, Graphical symbols
Statements of relationship	There should be a 1:1 correspondence between semantic constructs and graphical symbols (bidirectional relationship)
Scope	High level of generality, potentially applicable to any language though better suited to formal lan-

---

[3] A Type II theory may contain testable propositions, but it is not a primary concern

	guages. (Subjective appreciation)
Theory Component (Components Contingent on Theory Purpose)	
Causal explanations	Yes. Goodman's theory of symbols [14][25], which states "for a notation to satisfy the requirements of a notational system there must be a one-to-one correspondence between symbols and their referent concepts".
Testable propositions (hypotheses)	Yes. Empirical Study: we could test two visual notations once that respect perfectly the Semiotic Clarity principle (languageA) and another one (languageB) that does not. Then, we ask for two groups to model a scenario. The first group will use languageA and the second group the languageB. Finally, we ask for a third and fourth group to analyse what they actually understand of the diagram base on languageA (3rd group) and language (4th group). That way we could easily compare the benefits of the Semiotic Clarity
Prescriptive statements	Partially Addressed. Provide four metrics to assess anomalies when the 1:1 correspondence is not respected.

Level Theory Analysis

This theory contains all the characteristics of a Type IV theory. Its generality level is high because it can be used in almost any formal language. In our case, almost any formal language needs to be consistent by avoiding synonyms and homonyms and the 1:1 correspondence answers to these issues.

More importantly, this theory is testable with empirical studies (see example above). While the theory gives metrics to assess the 1:1 correspondence principle, it does not precisely define its constructs: semantic constructs and semantic constructs. This issue is related to how to concretely accomplish the theory. Questions as how to capture semantic constructs from meta-language diagrams are not covered.

To wrap up, the theory responds to the how and why the phenomenon occurs. It is testable with empirical studies. The "what will be" is also clearly represented. But the "how" needs further research. Therefore, It seems to us that the Semiotic Clarity principle falls into the **Type IV** category.

4.4.2 Principle of Perceptual Discriminability

Structural Components of the Theory	
Theory Component (Components Common to All Theory)	Instantiation
Means of representation	Words, Figure
Constructs	Symbols
Statements of relationship	Different symbols should be clearly distinguishable from each other.
Scope	High level of generality. This principle is suited to any field, from SI notations to everyday life such as road traffic signs
Theory Component (Components Contingent on Theory Purpose)	
Causal explanations	Yes. "In general, the greater the visual distance between symbols, the faster and more accurately theory will be recognised."

Testable propositions (hypotheses)	Yes.
	We could assess how the human cognition can distinguish between symbols (setA) which are close (i.e., a small visual distance) and symbols that appear radically different (setB) (i.e., a greater visual distance). A simple and effective way, is to give two sets of symbols and see at what speed they are learnt. A first group has to learn the setA and another group has to learn the setB. Then, we can compare the speed (i.e., when errors are inferior to a certain threshold) that the two sets are considered as learnt.
Prescriptive statements	Partially Addressed.
	Primacy of shape: what kind of shape are more suited to a particular situation? => Requires human expertise.
	Redundant Coding: well-defined, and easily applicable in practice.
	Perceptual Pop-out: well-defined, and easily applicable in practice.
	Textual Differentiation: well-defined, and easily applicable in practice.

Level Theory Analysis

This theory contains all the characteristics of a Type IV theory, "Theory for explaining and predicting". This theory has causal explanations, which come from research in psychophysics [36][38][5].

The theory relies on solid basis causal explanations. Moreover, it is testable through empirical studies. The prescriptive statements are partially addressed. However, high-level definitions are sufficient to provide guidelines but not for evaluating analysis. Therefore, we will give more mathematical metrics in the following chapter to answer to that issue.

The theory responds to the "what", the "how" and the "why" issues. Thus, by following this theory, the "what will be" is clearly taken into account. Indeed, the purpose is to improve discriminability by following the guidelines. The Perceptual Discriminability principle falls into the **Type IV** category and not the Type V because of the issue about evaluating purpose.

4.4.3 Principle of Semantic Transparency

Structural Components of Theory	
Theory Component (Components Common to All Theory)	Instantiation
Means of representation	Words, Figure (e.g., Semantic Transparency scale)
Constructs	Graphical symbols
Statements of relationship	Use visual representations whose appearance suggests their meaning (provide cues to their meaning).
Scope	High level of generality, potentially applicable to any visual language.
Theory Component (Components Contingent on Theory Purpose)	
Causal explanations	Partially Addressed. Graphical symbols that provide cues to their meaning (form implies content) reduce cognitive load because their meaning can be either perceived directly or easily learnt.
Testable propositions (hypotheses)	Not Main Concern. This statement can be tested through empirical studies though testability does not seem to be the main concern of the theory.

	The main problem comes from the relative subjectivity to evaluate a graphical symbols in one of the three states given by Moody (i.e., semantically immediate, semantically opaque and semantically perverse)
Prescriptive statements	No. Only high level advices are given as "using physical analogies, visual metaphors and cultural associations to design objects"

Level Theory Analysis

This Theory contains all the characteristics of a Type II. This theory deals with human perception. Therefore, all the artefacts must be empirically tested. In this theory, the empirical studies are crucial because the improvements are quite different from an audience (e.g., experts) to another (e.g., novices). However, it appears to us that testability is not the main concern of this theory.

The theory relies on social science experiments, which is totally normal considering that human perception is the main concern of this theory. This purpose of this theory is more related to the guidelines' purpose than to the evaluating the purpose of a visual language.

It seems to us that the theory fails to give clues on how to use it in practice. Indeed, defining a scale of Semantic Transparency (semantically immediate, opaque and perverse) is needless when there is no clue on how to evaluate a graphical symbol. Once again, we understood that due to complexity of human perception, it is really hard to give a systematic way to evaluate or to propose a semantically more transparent graphical symbol. However, by giving no clue, the theory aims to explain the phenomena of Semantic Transparency. Even if the "what will be" issue can be answered through empirical studies (testable proposition component), it seems inappropriate to put the theory on the

Type IV class, which is related to more scientific theory. Nevertheless, a lot of intuition is necessary to process this theory; it falls into the **Type II** category.

4.4.4 Principle of Complexity Management

Structural Components of Theory	
Theory Component (Components Common to All Theory)	**Instantiation**
Means of representation	Words, illustrated with figures
Constructs	Diagrams, Diagrammatic complexity (i.e., measured by the number of elements on a diagram)
Statements of relationship	Include explicit mechanisms for dealing with complexity (diagrammatic complexity)
Scope	The complexity is restricted to only diagrammatic complexity. Therefore, it seems to us that the scope is limited because the theory copes with a certain type of complexity, which concern symbol instances on diagrams.
Theory Component (Components Contingent on Theory Purpose)	
Causal explanations	Yes. Perceptual limits: The ability to discriminate between diagram elements increases with diagram size. Cognitive limits: The number of diagram elements that can be comprehended at a time is limited by working-memory capacity (i.e., in-

	creasing the number of diagram will overwhelm working-memory capacity)
Testable propositions (hypotheses)	Yes.[4] Introduction of a means of reducing diagrammatic complexity can easily be tested. Combined with empirical studies.
Prescriptive statements	Yes. Two techniques of reducing diagrammatic complexity have been given: modularisation and hierarchy abstraction. CASE tools features could support such techniques (see Section 7.1.5).

Level Theory Analysis

This theory contains all the characteristics of a Type V. It deals with a restricted form of complexity, the diagrammatic complexity, which is related to the number of elements that compose a diagram. The causal explanations rely on empirical studies, which showed that humans have limits to comprehend a certain amount of information at a time. These causal explanations have been validated for a long time.

The main concern here is about the scope of the theory, which responds to only a small part of complexity, i.e., the diagrammatic one. However, the taxonomy does not take the level of generality as a primary characteristic. We will then not take it into consideration.

The prescriptive statements are precisely defined in the perspective of how to reduce Complexity Management. This is why, this theory falls into the **Type V** category. Indeed, it answer not only the "what is", "how", "why" and the "what will be" but also gives a way to concretely reduce the diagrammatic complexity.

---

[4] Only because we consider diagrammatic complexity

4.4.5 Principle of Cognitive Integration

Structural Components of Theory	
Theory Component (Components Common to All Theory)	Instantiation
Means of representation	Words, Illustrated with figures
Constructs	Diagrams, Cognitive Integration (i.e., mental representation of a system that has been spread over different diagrams)
Statements of relationship	Include explicit mechanisms to support integration of information from different diagrams
Scope	Highly generalizable. The theory is applicable to any information spread into multiple diagrams / documents.
Theory Component (Components Contingent on Theory Purpose)	
Causal explanations	Yes. Rely on Kim *et al*'s [18][21] cognitive integration of diagrams that state: "multi-diagram representations to be cognitively effective must include mechanisms to support **c**onceptual integration and *perceptual integration*". Conceptual integration: mechanism to assemble information from separate diagrams Perceptual integration: perceptual cues to simplify navigation and transitions between diagrams.
Testable propositions (hypotheses)	Yes. We could ask questions that could be answered only by integrating information spread over several diagrams. Two groups will take part

	in an empirical study. The first group will have at its disposal the visual notation enhanced with the principle of Cognitive Integration while the second group has to do the same work only with the regular visual notation.
Prescriptive statements	<mark>Yes.</mark> Two techniques are available: conceptual integration and perceptual integration. CASE tools features could support such techniques (see Section 7.1.6).

Level Theory Analysis

This theory has a high level of generality because it is almost applicable to any visual language that deals with diagrams.

The causal explanations rely on the *cognitive integration of diagrams* theory [18][21], which gives solid basis. Furthermore, the prescriptive statements are defined in an operationalized way by giving hints.

In this theory, many prescriptive statements are given but some of them are more suitable to be implemented as CASE tools features. Empirical studies could be used to test the theory statements.

The theory answers the "what is", "how", "why" and the "what will be" questions, and it gives features that have to be implemented to manage the Cognitive Integration. Therefore, this theory is integrally part of the **Type V** theory type: theory for design and action.

4.4.6 Principle of Visual Expressiveness

Structural Components of Theory	
Theory Component (Components Common to All Theory)	**Instantiation**
Means of representation	Words, Illustrated with a figure
Constructs	Bertin's visual variables, visual expressiveness

	(i.e., number of visual variables used in a notation)
Statements of relationship	Use the full range and capacities of visual variables
Scope	Applicable to any visual notations.
Theory Component (Components Contingent on Theory Purpose)	
Causal explanations	Yes. "Using a range of visual variables results in a perceptually enriched representation that exploits multiple visual communication channels and maximises computational offloading"
Testable propositions (hypotheses)	Yes. The statement is fully testable upon the visual variables are precisely defined.
Prescriptive statements	No. Ranges for each visual variable has been given but there are no clues on what should we preferably use to encode a certain type of information.

Level Theory Analysis

This theory has causal explanations and its propositions can be tested through empirical studies, and is applicable to any visual language.

However, the theory only says that the more visual variables are used, the better. It fails to give weighted values to visual variables.

From my point of view, the essential idea is not to use all the visual variables but to use the most relevant ones for a given situation. Further, that knowledge can then be applied when designing visual languages. We think that it is also useful to criticise the visual var-

iables used in a visual notation than just focusing on those that are not used yet (i.e., free variables).

That being said, this theory answers to all the **Type IV** theory type.

4.4.7 Principle of Dual Coding

Structural Components of Theory	
Theory Component (Components Common to All Theory)	**Instantiation**
Means of representation	Words, Illustrated with a figure
Constructs	Graphical symbols, Cognitive management
Statements of relationship	Use text to complement graphics.
Scope	Applicable to any visual notations.
Theory Component (Components Contingent on Theory Purpose)	
Causal explanations	Yes. "According to *dual coding theory* [29], using text and graphics together to convey information is more effective than using either on their own" "The textual encoding is most effective when it is used in a supporting role: to supplement rather than to substitute"
Testable propositions (hypotheses)	Yes. Example of an Empirical Study: we could assess how explanations included on diagrams (annotation) are more effective than explanations on separated documents. Two groups could take part to assess this criterion, one that will have the information directly on the diagrams and another

	one that has to see them on another document.
Prescriptive statements	Yes. Adding comments to diagrams Using text as a supplement way (not alone) to encode information on graphical symbols

Level Theory Analysis

This theory can be applied to any visual language. Besides this general applicability, the propositions can be assessed with empirical studies. It also answers to concerns about how to concretely apply it. These are some reasons why we classify the Dual Coding in the **Type V**.

4.4.8 Principle of Graphic Economy

Structural Components of Theory	
Theory Component (Components Common to All Theory)	**Instantiation**
Means of representation	Words, Illustrated with a figure
Constructs	Graphical symbols, Cognitive management, Graphic complexity (i.e., number of graphical symbols in a notation, notational level and not instantiations)
Statements of relationship	The number of different graphical symbols should be cognitively manageable.
Scope	Reduced to formal notational language.
Theory Component (Components Contingent on Theory Purpose)	

Causal explanations	Yes.
	"The human ability to discriminate between perceptually distinct alternatives is around 6 categories"
	"Empirical studies show that graphic complexity significantly reduces understanding of SE diagrams by novices"
Testable propositions (hypotheses)	Yes.
	It is obvious that reducing the visual vocabulary (i.e., number of graphical symbols in a notation) benefits for discriminability and maintaining meanings of symbols in working memory.
Prescriptive statements	Partially Addressed.
	Three strategies for the "how": • Reduce (or partition) semantic complexity • Introduce symbol deficit (i.e., choosing not to show some constructs graphically) • Increase Visual Expressiveness (Instead of reducing the number of symbols, we increase the human discriminability ability)

Level Theory Analysis

This theory deals with the increase of the visual vocabulary of a visual language, and relies on empirical studies in social science. Furthermore, the theory was written to be testable. In this state, the purpose of Graphic Economy is applicable as a guideline theory and as an evaluating theory. However, the theory does not give any clue on the artefacts that, for example, **should** be removed.

It seems to us that the number of artefacts of a visual language is not critical. Most modelling languages have many graphical symbols but very few of them are used.

From my point of view, the theory should give a checklist that modellers can use to assess the usefulness of a graphical symbol.

These considerations let us think that the Graphic Economy principle falls into the **Type IV** theory since it has causal explanations and testable propositions.

4.4.9 Principle of Cognitive Fit

Structural Components of Theory	
Theory Component (Components Common to All Theory)	Instantiation
Means of representation	Words, Illustrated with a figure.
Constructs	Cognitive Fit, Visual dialects (i.e., use multiple versions of a visual notation to fit different audiences or different representational media)
Statements of relationship	Use different visual dialects for different tasks and audiences.
Scope	Highly generalizable.
Theory Component (Components Contingent on Theory Purpose)	
Causal explanations	Yes. "Different representations of information are suitable for different tasks and different audiences", validated empirically SE notations use frequently one sets of graphical symbols regardless of tasks and/or audiences Two reason to create multiple visual dialects; expert-novice differences and representational medium

Testable propositions (hypotheses)	Yes. We could assess at what extent a group composed of novices suffer from difficulty to discriminate and remember symbols, and also, how they are affected by complexity compared with a group of experts. The comparison should be made with the regular visual notation and the visual notation that fits more the group of novices.
Prescriptive statements	No. Not present.

Level Theory Analysis

Causal explanations are present in this theory, which was designed in a way that it can be tested with empirical studies. However, no process has been given to transform a graphical symbol depending on another audience or another representational medium.

Type IV theory type is the most suitable type for this theory. Indeed, the theory focuses on explaining why SE notations are not well suited to different audiences or different representational mediums. The only solution that has been given is to use multiple visual dialects of a visual notation. It seems that further research is appropriate for this theory.

4.5. Overall assessment

The previous section compared the PoN principles against Gregor's taxonomy; Table 2 provides a summary of the analysis described previously. None of the principles of the PoN has been categorized as a Type I or a Type III. Indeed, all the principles rely on solid causal explanations. Moreover, none of the principles has been designed to make precise predictions. Most of them rely on the idea of improvement between the "before and after" application of the principle. Such improvements have to be proved by empirical studies.

Table 2   Assessment of the PoN principles Against Gregor's Taxonomy

Type I	
Type II	• Semantic Transparency
Type III	
Type IV	• Semiotic Clarity • Perceptual Discriminability • Visual Expressiveness • Graphic Economy • Cognitive Fit
Type V	• Complexity Management • Cognitive Integration • Dual Coding

Moody compared the PoN theory according to the cognitive effectiveness. However, due to the interactions among the principles, we cannot predict an overall improvement of the cognitive effectiveness, i.e., the *speed*, *ease* and *accuracy* with which a representation can be processed by the human mind. This is related to the well-known pattern in IS of the local versus global optimization.

In Chapter 5, we will gather definitions and metrics that will make the PoN *more* operationalized. Nevertheless, these precisions should be viewed as *one* way (and not the only) to operationalize the PoN.

Chapter 5.   Operationalization and Critics of two PoN Principles

In this chapter, we provide consistent definitions, metrics, and different ways to interpret the PoN for the modelling languages. However, we restrict that analysis to two principles of the PoN: Semiotic Clarity and Perceptual Discriminability.

The following work is drawn upon an operationalization of these principles described by Störrle and Fish in [39]. As well as literature published by Genon *et al.* in [10] [13]. We also focus on comparing how these two principles were operationalized by Störrle and Fish, and Genon *et al*. Furthermore, we propose our own versions of some metrics.

5.1.   Preliminary Definitions

The PoN theory uses the concept of graphical and semantic constructs. Due to the scope of the PoN, Moody opted for very general (i.e., applicable to any visual notations) definitions, which are good to cover a larger field of visual languages. However, when we apply the theory, we have to cope with issues about how to interpret such concepts. The challenge is to give a means to collect all the semantic constructs from the metamodel language. Also, we need to collect the graphical symbols from the concrete syntax of the modelling language.

5.1.1   Semantic Constructs: one metaclass - one construct

Moody stated that semantic constructs should be found in the metamodel. A naïve methodology could be to associate a metaclass directly to a semantic construct. However, we would either lose some semantic constructs in the process (i.e., false negative error, not detected a characteristic) or declare metaclasses as semantic constructs that should be considered as semantic constructs (i.e., false positive error, not corresponding to reality).

Most modelling language metamodels use the UML Class diagrams to represent the language concepts. Figure 22 represents a particular metamodel of GRL. This metamodel is interesting because three of its classes (highlighted in blue) encompass more than one semantic symbol. For instance, the class `IntentionalElementType` is composed of the semantic constructs: `Softgoal`, `Goal`, `Task`, `Resource` and `Belief`.

Figure 22   A metamodel diagram of GRL

Therefore, we should take into account not only the class definition but also its attributes. Genon *et al.* have encountered such issues when analysing UCM according to the PoN [13]. They have proposed a way to classify the UCM metaclasses according to six categories. We believe that their classification is general enough to be applied in a variety of metalanguages. The first process of PoN is, therefore, to collect and to sort the metaclasses into these six categories. According to the classification of Table 3, the three examples of Figure 22 (e.g., *ContributionType*, `IntentionalElementType` and `DecompositionType`) fall in the *"Collection"* category. Therefore, they should be taken into account to further extract their semantic constructs.

We then consider only the metaclasses that fall into the *"To consider"* and *"Collection"* categories. These metaclasses will be subject to an analysis to extract the semantic constructs.

The classification of Table 3 was conceived for the analysis of UCM ([13]) but it does not appear in the paper itself but in the technical report [12]. Therefore, it fits to the URN metamodel specified in [19]. It appears to use that the scope of this classification is larger and can be applied to most of metamodels. However, the classification should not be regarded as the end point for classifying metaclasses but a starting point to develop classification groups that could be applied to any metamodel. Therefore, we suggest taking these groups as a basis that we could enhance and/or overload by other groups.

Table 3   Classification of metaclasses of a metamodel (from Genon *et al.* in [12])

To consider	"Metaclasses that denotes (a part of) a semantic construct and hence, should have a visual representation" (from [12])
Abstract	"Abstract metaclasses, even if they refer to semantic construct, are usually not represented on diagrams. These semantic constructs are refined in children metaclasses that are mapped to visual symbols." (from [12])
Structural	"Structural metaclasses are metaclasses that exist for meta-modelling matter and that do not correspond to any semantic construct. There is no rationale for associating symbols to them." (from [12])
Collection	"Collection is a generic term that denotes enumeration metaclasses. We suggest to do not associate concrete syntax to this kind of metaclass. However, the values of the collection may be represented." (from [12])
Graphical	"Graphical metaclasses are metaclasses which purpose is to store concrete graphical information such as default line thickness, spatial location of symbols and default font. These metaclasses do not denote any semantic construct of the modelling language and they should not be represented as notational element." (from [12])
Out of scope	"This category gathers all metaclasses that are "by nature" or

	considered as out of scope for the notation analysis." (from [12])

5.1.2 Graphical Symbols: collect the symbols with a non-combinatory way

Although it is difficult to precisely define a graphical symbol, we would like to draw attention to the problem of collecting the graphical symbols. If proper care is not taken, the number of symbols could dramatically increase. The problem occurs because most modelling languages combine simple graphical artefacts to build more complex ones. Thus, the visual vocabulary is composed of a small set of graphical symbols.

For instance, the URN standard explains that we can attach a `ContributionType` (Make, Help, SomePositive, Unknown, SomeNegative, Hurt, Break) to a `Contribution`. The `ContributionType` is represented by an icon. Likewise, we also can attach a *text* to name the *contribution* and a *number* to give a value to the *Contribution*. The specification gives five ways to combine these elements:

(a) Contribution + an icon,

(b) Contribution + a text,

(c) Contribution + a icon + a text,

(d) Contribution + a number,

(e) Contribution + an icon + number.

Furthermore, the `ContributionType` are also associated to the *correlation* links. The number of graphical symbols by combining *contribution link* and *correlation link* with the `ContributionType`, the text and the number equals to:

$$2* 7 + 2 + 2*7 + 2 + 2*7 = \textbf{46}.$$

Instead, we could collect these graphical symbols: a *Contribution* link alone, a *Correlation* link alone, 7 graphical symbols (icons) for the `ContributionType` and a number and a text alone, which equals to **11** graphical symbols. This ***non-combinatory*** way of collecting graphical symbols is more ***scalable***. The careful reader may have noticed that the **non-combinatory** way of collecting symbols contradicts the way Moody suggested to collect symbols. However, this proposal is relevant, for instance, for the principle of Graphic Economy. That principle evaluates the graphical complexity as the

number of symbols of the visual notation, but that number could be too high. Indeed, as we stated, a visual notation chunks of symbols to build more sophisticate symbols. Therefore, a modeller has only to remember these simple chunks of symbols and the rules to combine them.

Another reason to consider the non-combinatory way to collect symbols is that even if we want to gain in precision, sometimes, we could simply not take all the combinations. For instance, in the previous example we considered a number as one instance but in reality, there are an infinity of values for a number.

Further, over time, a modelling language tends to add more symbols to its concrete syntax. This could add other multiplicative factors.

Background Definitions

In [39], Störrle and Fish defined the PoN principles with mathematic functions. As the following two first sections rely partly on their ideas, we will introduce some of their concepts to help for the overall understanding.

In their paper, they call the semantic constructs "Concepts C" and the graphical symbols "Graphems G". *C* is the set of all the semantic concepts and *G* is the set of all the graphical symbols (see Figure 23). A function σ takes a concept in argument and gives its set of related graphems. The inverse function σ^{-1} takes a graphem in argument and gives its set of related concepts.
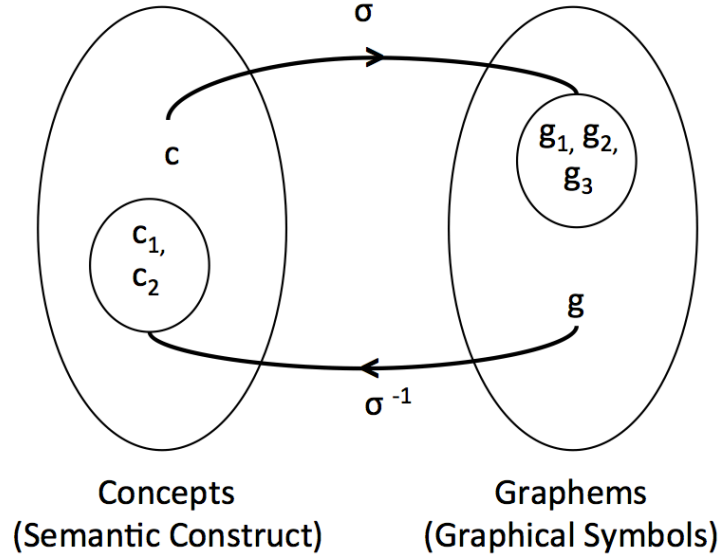
Figure 23 The σ and the σ^{-1} functions

Additionally, they defined a set of concepts that are represented graphically, called visualized(C). Also, the set of graphems that are related to concepts is called meaningful(G).

$$visualized(C) := \{c \in C \,|\, \sigma(c) \neq \emptyset\}$$
$$meaningful(G) := \{g \in G \,|\, \sigma^{-1}(g) \neq \emptyset\}$$

5.2. Semantic Clarity

Moody has given four metrics to assess the potential anomalies for the semantic clarity principle. In this section, we will expose and compare the metrics proposed by Genon *et al*. and Störrle and Fish.

5.2.1 Symbol Redundancy

A direct assessment of the symbol redundancy can be a ratio between (a) the semantic constructs represented by multiple graphical symbols and (b) the cardinality of the set "*To consider*" (i.e., the number of semantic constructs).

Therefore, the ratio a/b represents the percentage of the symbol redundancy anomaly. In other words, the more it tends to 0, the better.

Störrle and Fish proposed the formulae shown in Figure 24. The $SR(N)$ (N for visual **N**otation) function returns (applied to whole semantic notation) a value v between 0 (better) and 1 (worse), more precisely $v \in [0, 1[$. $Redundancy(c)$ tends to 1 when more graphical symbols are associated to a semantic construct. For instance if the semantic construct c has three related graphical symbols, redundancy(c) is $1 - \frac{1}{3} = 0.666 \dots 6 \dots$

$$SR(N) := \frac{1}{|visualized(C_N)|} \sum_{c \in C_N} redundancy(c)$$

$$redundancy(c) := \begin{cases} 0 & \text{if } \sigma(c) = \emptyset, \\ 1 - \frac{1}{|\sigma(c)|} & \text{otherwise.} \end{cases}$$

Figure 24   Processing of Symbol Redundancy (from Störrle *et al.*)

From my point of view, the metric proposed by Störrle and Fish is more precise because it takes into consideration the number of graphical symbols associated to each semantic construct. Genon's formula does not differentiate whether two or more graphical symbols are associated to the same semantic construct (i.e., without any deteriorating factor). However, Störrle's formula could be improved by applying an exponential function (see Figure 25) instead of the linear function associated to $|\sigma(c)|$. In this way, the function will more closely respect the 1:1 correspondence principle (between semantic constructs and graphical symbols).

$$redundancy'(c) := \begin{cases} 0 & \text{if } |\sigma(c)| < 2 \\ 1 - \frac{1}{exp(|\sigma(c)|)} & \text{if } |\sigma(c)| \geq 2 \end{cases}$$

Figure 25   An improved version of the function *redundancy*

5.2.2 Symbol Overload

Genon *et al.* assessed this criteria with a ratio between (a) the graphical symbols, which represent multiple semantic constructs and (b) the number of graphical symbols.

Störrle and Fish proposed the metric of Figure 26. The $SO(N)$ function returns a value v between 0 (better) and 1 (worse), more precisely $v \in [0,1[$. The overload(g) function tends to 1 when more semantic constructs are associated to a certain graphical symbol.

$$SO(N) := \frac{1}{|meaningful(G_N)|} \sum_{g \in G_N} overload(g)$$

$$overload(g) := \begin{cases} 0 & \text{if } \sigma^{-1}(g) = \emptyset, \\ 1 - \frac{1}{|\sigma^{-1}(g)|} & \text{otherwise.} \end{cases}$$

Figure 26   Processing of Symbol Overload (Störrle *et al.*)

In my opinion, Störrle and Fish's function is more precise than the metric proposed by Genon *et al.* Indeed, there is no deterioration factor in the Genon's formula.

We would recommend to apply an exponential function to the $|\sigma^{-1}(g)|$ term for the same reasons that we exposed in the previous section.

$$overload'(g) := \begin{cases} 0 & \text{if } |\sigma^{-1}(g)| < 2 \\ 1 - \frac{1}{exp(|\sigma^{-1}(g)|)} & \text{if } |\sigma^{-1}(g)| \geq 2 \end{cases}$$

Figure 27   An improved version of the function *overload(g)*

5.2.3 Symbol Excess

Genon *et al.* evaluate symbol excess with a ratio between (a) the graphical symbols are not related to any semantic construct and (b) the number of graphical symbols of the language. Therefore, a/b is the percentage of symbol excess.

Störrle and Fish also proposed a ratio (see Figure 29) between (i) the meaningful graphical symbols (i.e., the graphical symbols that have corresponding semantic concepts, see Figure 28) and (ii) the number of graphical symbols of the whole language. The formula returns a value v between 0 (better) and 1 (worse), more precisely $v \in [0,1]$.
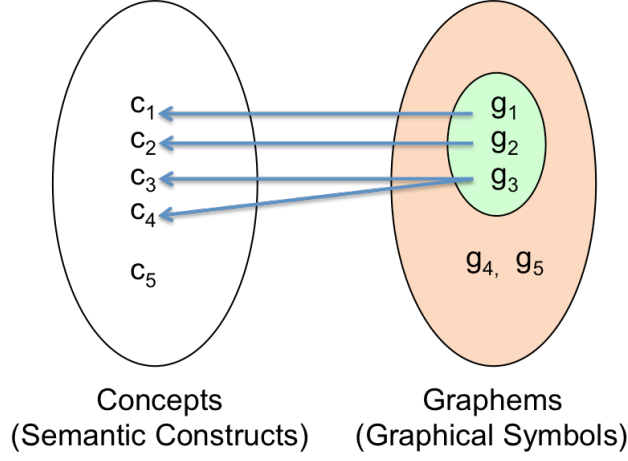
Figure 28 The $meaningfull(G_N)$ set and the $meaningless(G_N)$, set.

$$SE(N) := 1 - \frac{|meaningful(G_N)|}{|G_N|}$$

Figure 29 Processing of Symbol Excess (from Störrle and Fish)

Although the Genon and Störrle formulae appear different, they are mathematically equivalent.

Proof

Let us define a new set: $meaningless(G_N)$, composed of the graphical symbols that have no corresponding semantic concepts (e.g., g_4 et g_5 of the Figure 28).

Let us assume that the set G_N (i.e., the set of the graphical symbols of the notation N) is composed of the $meaningful$ and $meaningless$ sets and $meaningful(G_N) \cap meaningless(G_N) = \emptyset$. So additionally, $|meaningful(G_N)| + |meaningless(G_N| = |G_N|$.

Then the Genon's metric of Symbol Excess can be rewritten like Figure 30.

$$SE_{genon}(N) := \frac{|meaningless(G_N)|}{|G_N|}$$

Figure 30 Rewritten of the Genon's metric of Symbol Excess

The Figure 31 shows that Störrle's formula of symbol excess (see Figure 29) mathematically equals to Genon's formula of symbol excess (see Figure 30).

$$
\begin{aligned}
SE(N) \quad &:= \quad 1 - \frac{|meaningful(G_N)|}{|G_N|} \\
&= \quad \frac{|G_N|}{|G_N|} - \frac{|meaningful(G_N)|}{|G_N|} \\
&= \quad \frac{|G_N| - |meaningful(G_N)|}{|G_N|} \\
&= \quad \frac{|G_N| - (|G_N| - |meaningless(G_N)|)}{|G_N|} \\
&= \quad \frac{|G_N| - |G_N| + |meaningless(G_N)|}{|G_N|} \\
&= \quad \frac{|meaningless(G_N)|}{|G_N|} \\
&:= \quad SE_{genon}(N)
\end{aligned}
$$

Figure 31   Development of the Störrle and Fish formula

5.2.4 Symbol Deficit

Genon *et al.* assessed the anomalies related to symbol deficit with a ratio between (a) the semantic constructs that are not represented by any graphical symbol and (b) the cardinality of the set "*To consider*" (i.e., the number of semantic constructs).

Störrle and Fish also proposed a ratio (see Figure 32) but between (i) the visualized semantic concepts (i.e., the semantics concept which are represented graphically by graphical symbols) and (ii) the number of semantic concepts of the language. Like their previous formulae, they ensured that this formula returns a value v between 0 (better) and 1 (worse), more precisely $v \in [0, 1]$.

$$
SD(N) := 1 - \frac{|visualized(C_N)|}{|C_N|}
$$

Figure 32   Processing of Symbol Deficit (from Störrle and Fish)

Störrle and Fish's formula is mathematically equivalent to the Genon *et al.*'s formula.

Proof

Let us define a new set: $none\text{-}visualized(C_N)$, composed of the semantic concepts that are not represented graphically (e.g., c_5 of Figure 33).

Let us assume that the set C_N (i.e., the set of the semantic constructs of the notation N) is composed of the *visualized* and *none-visualized* sets and $visualized(C_N) \cap none\text{-}visualized(C_N) = \emptyset$. As well, $|visualized(C_N)| + |none\text{-}visualized(C_N)| = |C_N|$.

Than, Genon *et al.*'s metric can be rewritten as Figure 33.

$$SD_{genon}(N) := \frac{|none\text{-}visualized(C_N)|}{|C_N|}$$

Figure 33   Rewrite of the Genon's metric of Symbol Deficit

Concepts
(Semantic Constructs)

Graphems
(Graphical Symbols)
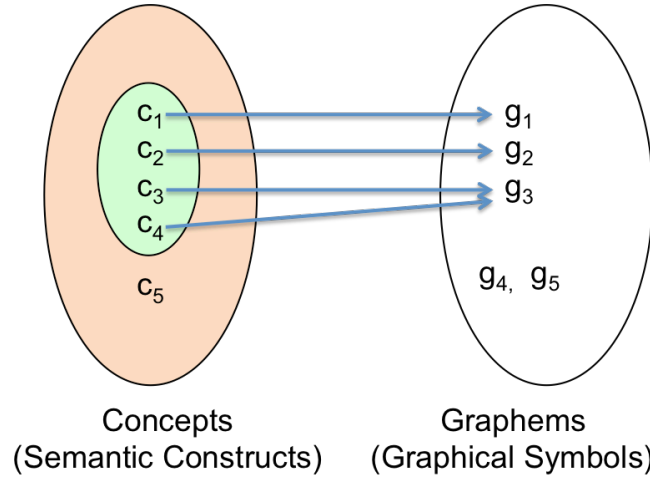
Figure 34   The $visualized(C_N)$ (in green) and the $none\text{-}visualized(C_N)$ (in brown)

The Figure 35 shows that the Störrle and Fish formula of symbol deficit (see Figure 32) mathematically equals to the Genon *et al.*'s formula of symbol deficit (see Figure 33).

$$
\begin{aligned}
SD(N) \quad &:= \quad 1 - \frac{|visualized(C_N)|}{|C_N|} \\
&= \quad \frac{|C_N|}{|C_N|} - \frac{|visualized(C_N)|}{|C_N|} \\
&= \quad \frac{|C_N| - |visualized(C_N)|}{|C_N|} \\
&= \quad \frac{|C_N| - (|C_N| - |none\text{-}visualized(C_N)|)}{|C_N|} \\
&= \quad \frac{|C_N| - |C_N| + |none\text{-}visualized(C_N)|}{|C_N|} \\
&= \quad \frac{|none\text{-}visualized(C_N)|}{|C_N|} \\
&:= \quad SD_{genon}(N)
\end{aligned}
$$

Figure 35   Development of the Störrle's formula

5.2.5 Critics of the Semantic Clarity Operationalization

In previous sections, we compared each criterion of the Semantic Clarity according to its implementation by Genon *et al.* and by Störrle and Fish. We suggested improvements for the symbol redundancy and for the symbol overload criteria. Besides these improvements, we proved that the symbol excess and symbol deficit criteria of Genon *et al.* are mathematically equal to the Störrle and Fish versions.

However, these formulae are only one aspect of the assessment of the Semantic Clarity. Indeed, these formulae are useless if it does not imply concrete decisions for the modelling language. In other words, assessment by mathematic formulae is useful only if we point out the problems among the semantic concepts or the graphical symbols. And then, it pushes us to take decision such as reducing/increasing the number of graphical symbols.

Both Genon *et al.* and Störrle and Fish rely on the four metrics to discuss the Semantic Clarity. However, they clearly differentiate to one another about how they interpret these results and whether they give suggestions to actually improve the concerned modelling language.

Störrle and Fish in [39] keep focusing on discussing the flaws of PoN. For them, because PoN lacks on giving thresholds to compare their results, they could not interpret these results. I do agree that PoN or future research should be conducted to answer that issue. Although, it would be complicated to give general thresholds because each visual notation is unique (e.g., depends on properties such as the targeted audience). In any case, Störrle and Fish could give some hints about the semantic constructs or graphical symbols that cause issues. Indeed, instead of assessing each criterion on the notational level, they could assess each criterion in a detailed way, i.e., applying the metrics to a symbol or a semantic construct at a time. This results in a lack of relevant suggestions on how to improve UCD in accordance with the Semantic Clarity, which is normally the aim of PoN.

On the other hand, Genon *et al.* in [13][10] use the metrics to give an overview about how the modelling language behaves against each criterion. However, all the rest of the discussion is focused firstly on pointing out the "guilty" semantic construct or symbol (or group of semantic constructs or symbols) and, secondly, on ways to improve that issue. For instance, what symbols should be removed or what semantic construct is important to appear in the visual notation.

The Genon *et al.* assessment of the Semantic Clarity appears to be more *constructive*. Even if their metrics are less precise than those developed by Störrle and Fish, they rely on expertise to stand out. The output between the two assessments is clearly more useful on the Genon *et al.* side due to the several recommendations that they give on this principle.

5.3. Perceptual Discriminability

5.3.1 Primacy of Shape

Genon *et al.* adopted a methodology based on qualitative assessment to be able to give some advice for the visual language, while Störrle and Fish worked more on quantitative assessment. However, they both assessed the visual variable shape in the same way: considering a certain number of shape families. Each graphical symbol of the visual language has to belong to one of the shape families. These shape families have to be defined on a case-by-case basis according to the analysed visual language. Thus, no generalization of the shape families can be determined.

In their operationalization of PoN, Störrle and Fish have defined three families of shape: *Line*, *Icon,* and *Region* (i.e., component). The latter is then decomposed as *Simple* and *Complex*. These shape families are presented as applicable to any visual language. However, it seems to us that this is an ad-hoc classification, which is defined to suit to their analysis of the UML Use Case Diagrams (UCD).

In order to show that shape families are not generic enough, let us discuss about the GRL language. In GRL, unlike in UCD diagrams, lines (e.g., *contribution links, correlation links*) are not simple because we can attach icons (e.g., *contributionType* as Make) to them. It would therefore be logical to distinguish the "simple" lines from the lines with attached icons because they could be perceived differently (i.e., the extra icon changes the perception of the line).

Furthermore, the way Störrle and Fish computed the distance between the different shape families is limited, perhaps even erroneous. They claimed that two graphical symbols belonging to two different shape families are more discriminable than two symbols taken from the same family (Figure 36). That claim is not always true. For instance, let us assume that we want to compute the visual distance between two icons that both look **very** different. The icons will be put in the Icon shape family, and, according to the Figure 36, the visual distance equals 0.5, which is not the optimum. Although, according to the PoN, the two shapes perfectly satisfy the shape criterion of the Perceptual Discriminability.

$$vvd(a,b) := \begin{cases} 0 & \text{if } a = b \text{ or both are undefined} \\ \frac{|a-b|}{c} & \text{if } a \neq b \text{ on ordinal scale with capacity } c \\ 0.5 & \text{if } a \neq b \text{ are shapes in same basic group} \\ 0.75 & \text{if } a, b \text{ are shapes in different subgroups} \\ 1 & \text{if } a, b \text{ are shapes in different main groups} \\ & \text{or } a \neq b \text{ on a nominal scale (except shape)} \\ & \text{or exactly one of } a \text{ or } b \text{ is undefined} \end{cases}$$

Figure 36   Visual Distance between the graphical symbols according to their shape families.

Another limitation of such classification is the difficulty encountered to classify certain graphical symbols according to a set of shape families. For instance, how do we to classify the *time path* (i.e., the zigzag path of Figure 37) of a UCM timer. The *timeout path* could be classified either as a Line or as an Icon (i.e., the zigzag path was designed to remind the icon of the thunder).
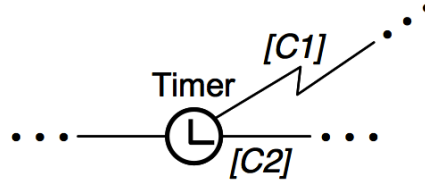
Figure 37   UCM timer with its timeout path (zigzag path)

Shape Comparison

These different limitations prompt us to seek another way to assess the shape discriminability. The idea is to measure the difference between shapes as the Levenshtein distance[5] does with strings. Belongie *et al.* [4] described a means to compare shapes between them not in accordance to a family class but by considering the *shape context* of feature points, i.e., a description of points of a shape compared to the other points of the shape. Furthermore, their technique offers interesting properties, e.g., invariance to common deformations, invariance to scale (i.e., two shapes can be compared no matter their sizes), and invariance to rotations. All of these properties are interesting for the Perceptual Discriminability.

---

[5] Mesures the difference between two strings.

The interested reader may refer to Appendix [XX], which popularizes the shape context by showing how concretely two shapes can be compared to one another.

Tool-based Shape Comparison

The previous technique makes it possible to develop a tool-based shape comparison. That tool would give warning and suggestions when two graphical symbols have two shapes, which are visually close. The suggestions can be generated from best practices in modelling languages. Furthermore, the technique can be applied on complex graphical symbols such as components, which are composed of other components and of all kinds of icons. However, that kinds of tools would be limited since nothing can replace the human expertise. In my opinion, a tool can only suggest the use of a different shape when it estimates that the used shapes are visually too close, but it cannot advice the use of a shape based on its meaning (Semantic Transparency).

Pattern-based Matching

Simpler than the tool-based shape, we propose to rely on graphical patterns instead of shape families. The idea is to associate each graphical symbol to a more generic pattern. Ling and Jacobs mathematically developed that matching technique in [22].

Figure 38 shows how the pattern (⟶◆⟶) is associated to four graphical symbols of UCM: static stub, dynamic stub, synchronizing stub and blocking stub. The pattern is obtained by filling the shape with a black colour. That way, the focus is put on the shape of the graphical symbol and not on none-relevant details (for the visual variable shape) such as the texture of the border of the graphical symbol.
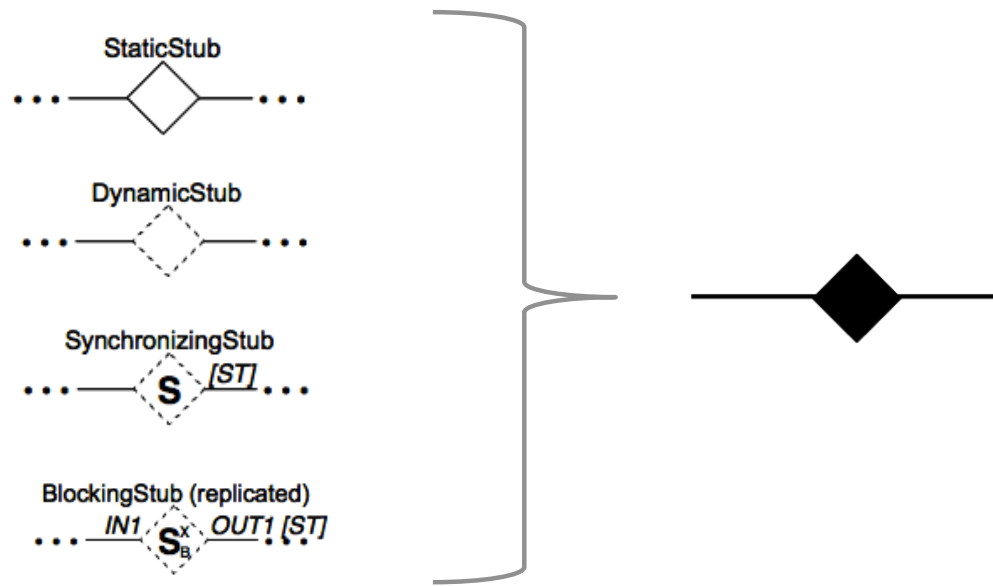
Figure 38 Association of a pattern (right) to graphical symbols (left)

Another example, the graphical symbol of UCM timer (see Figure 37) is associated to the

pattern ( ).

5.3.2 Visual Distance

Störrle and Fish defined a vector composed of Bertin's visual variables; each graphical symbol is associated to such a vector. The authors then proposed to differentiate symbols that have the same values for all visual variables by their textual annotations.

The purpose of that textual differentiation is to assess the visual notation against the anomaly of using only text to distinguish between graphical symbols. PoN mentioned the anomaly but did not give a means to evaluate it.

The textual differentiation is quite simple: it is a ratio between (a) the graphical symbols that can only be distinguished by considering their textual captions and (b) the number of graphical symbols (see Figure 39).

$$TD(N) := \frac{|\{g \in G_N \mid \exists g' \in G_N : g \text{ and } g' \text{ differ only by text}\}|}{|G_N|}$$

Figure 39   The textual differentiation metric

In my opinion, the metric is useful only to point out the graphical symbols that lack the ability to differentiate from each other otherwise than by text. However, that kind of comparison is impossible with the $TD(N)$ formula since it is applied to the whole visual language (N).

We propose to assess the textual differentiation for individual graphical symbols against the set of graphical symbols (see Figure 39).

$$TD'(G,g) := \begin{cases} 1 & \text{if } \exists g\prime \in G : g \text{ and } g' \text{ differ only by text} \\ 0 & \text{otherwise.} \end{cases}$$

Figure 40   Textual differentiation applicable to a specific graphical symbol

5.3.3 Redundant Coding

In PoN, Moody explained the need to use multiple visual variables to increase the visual distance between graphical symbols. Although Genon *et al.* did not use a formula to assess this principle, they applied the principle in the same way that Störrle and Fish operationalized it in their metric. Yet, Störrle and Fish's formula is more precise because it could show not only pair (i.e., 2-tuple) of value of visual variables but also the other combinations (i.e., n-tuple).

Two kinds of evaluation are made possible with the Störrle and Fish formulae, either evaluating a graphical symbol against the others like Genon *et al.* did (see Figure 41), or evaluating the entire visual notation (see Figure 42). In my opinion, in the absence of empirical studies, the second evaluation has less value because of the difficulty of interpreting the results. However, to some extent, it could be used to evaluate the evolution of a visual notation.

Figure 41 evaluates two graphical symbols, g and h on the proportion of distinct values of visual variables compared with the number of visual variables of the language (i.e., dimension d). The $[\phi]$ denotes 1 if the predicate ϕ is true, and 0 otherwise. At least $vr(g,h) \geq 2/d$ (i.e., g and h differentiate from at least two visual variables) and it should be the case for all the other graphical symbols $\forall g, h \in G \,|\, g \neq h$: $vr(g,h) \geq 2/d$.

$$vr(g,h) := \frac{1}{d} \sum_{i=1}^{d} [v_i(g) \neq v_i(h)]$$

Figure 41   Visual redundancy of two graphical symbols g and h (Störrle and Fish)

For the redundant coding evaluation at the notational level, Störrle and Fish applied the visual redundancy formula for the entire set of the graphical symbols (see Figure 42). We would like to draw the attention on the denominator $|G_N|^2$ (see Figure 42) which is not correct.

Given g_1, g_2 and g_3. The sum $\sum_{g,h \in G_N} vr(g,h)$ is composed of A_3^2 elements of $vr(g,h)$. Therefore in our case, we would have six different elements of visual redundancy formula.

$$vr(g_1, g_2) + vr(g_1, g_3) + vr(g_2, g_1) + vr(g_2, g_3) + vr(g_3, g_1) + vr(g_3, g_2)$$

If we assume that g_1, g_2 and g_3 have unique values for each of their visual variables, we would have $RC(N) = \frac{1}{3^2} * (1 + 1 + 1 + 1 + 1 + 1) = \frac{6}{9} = \frac{2}{3}$. The result is not correct; it should be equal to 1 since the graphical symbols satisfy perfectly the redundancy criterion. The solution is to use the denominator $|G_N| * (|G_N| - 1)$ instead of $|G_N|^2$ denominator. That way, we would have $\frac{6}{3*2} = 1$. The Figure 43 takes that issue into account.

$$RC(N) := \frac{1}{|G_N|^2} \sum_{g,h \in G_N} vr(g,h)$$

Figure 42   Redundant coding applied to the notational level

$$RC'(N) := \frac{1}{|G_N| * (|G_N| - 1)} \sum_{g,h \in G_N} vr(g,h)$$

Figure 43   A correct version of the redundant coding formula

5.3.4 Perceptual Pop-out

In this principle, the focus is on finding a unique value among the visual variables, which allows discriminating a graphical symbol against the others. Both Genon, and Störrle and Fish assessed this criterion in the same way.

Let us expose and explain the perceptual pop-out formula of Störrle and Fish.

Figure 44 translates in mathematics the following intuitive formulation: "A graphical symbol g has the perceptual pop-out characteristic if compared with the others graphical symbols, g has at least a unique value of one of its visual variables".

Then, that criterion is applied to the entire set of the graphical symbols (see Figure 44).

$$ppo(G, g) := \begin{cases} 1 & \text{if } \exists 0 < i \le d \text{ such that } \forall h \in G \\ & \quad g \ne h \implies v_i(g) \ne v_i(h) \\ 0 & \text{otherwise.} \end{cases}$$

Figure 44   Pop-out criterion for a graphical symbol g

$$PPO(N) := 1 - \frac{1}{|G_N|} \sum_{g \in G_N} ppo(G_N, g).$$

Figure 45   Pop-out principle applied to the entire visual notation

5.3.5 Critics of the Perceptual Discriminability Operationalization

Exactly as for the Semiotic Clarity (see Section 5.2.5), Störrle and Fish in [39] give only the raw results without relevant explanations of their assessment. A reader that is interested to concretely improve the discriminability of UCD elements cannot find any recommendation to comply with the principle of Perceptual Discriminability. The main problem is not how to assess the different criteria but how to explain them and even more importantly to give suggestions that can be applied on visual notations.

On the other hand, Genon *et al.* discuss each visual variable on whether it is used, and how it should be used in a specific modelling language (e.g., UCM in [13] and BPMN in [10]). By following the recommendations given by Genon.,we could *act* on visual notations instead of try to comprehend a bunch of meaningless numbers.

5.4. Developed Software

Assessment of the previous formulae could be processed either manually or semi-automatically. We have chosen the semi-automatic option because it represents a valuable contribution, not only for this thesis but also for the PoN community.

There are essentially two major ways to semi-automatically process the metrics of PoN: either by using a spreadsheet application (e.g., Microsoft Excel), or by using a programming language. The former option is the most accessible since it requires neither programming skills, nor knowledge of any programming language. However, it seems to us that using a programming language is more maintainable and powerful than spreadsheets.

Instead of choosing a programming language vice a spreadsheet processor, we take into account both. The data of the analysis will be gathered in Excel spreadsheets while all the assessment will be processed by a Java program.
We developed a piece of software that automatically applies the metrics mentioned in the previous sections. In Section 7.2.1, we present the different components of that software.

Chapter 6.   Evaluation: Analysis of AoURN

In this chapter, our first goal is to evaluate the metrics introduced in Chapter 5. We assess our proposed metrics for the principles Semiotic Clarity and Perceptual Discriminability by applying them on AoURN. A secondary goal is also to analyse the cognitive effectiveness of AoURN by applying all of the PoN principles. Since AoURN extends URN, we need to take into account URN and hence, both GRL and UCM. Previous research have analysed UCM according the PoN [11] and the i* modelling language, which is closely related to GRL, in [23]. Considering these evaluations, we focus our analysis on the cognitive effectiveness of the elements introduced to support AOM in URN. Therefore, in the following sections, when we refer to AoURN (or respectively AoUCM), we mean the newly introduced elements to support AOM in URN (and respectively UCM).

Since the interpretation of the results of applying the PoN is strongly related to the kind of audience and the medium used to design the diagrams, we firstly consider the principle of Cognitive Fit. Semiotic Clarity and Perceptual Discriminability are discussed in Section 6.2.2 and 6.3.4, respectively.

6.1. Cognitive Fit

Principle formulation: "*Use different visual dialects for different tasks and audiences*".

URN is mainly used by experts such as requirements engineers or academics. In [10], Genon *et al.* considered that the focus should be put on defining a comprehensive set of symbols with clear semantics (Semiotic Clarity), to be able to represent any required extra detail through text (Dual Coding) and to structure large models through Complexity Management and Cognitive Integration.

In our assessment, we consider an audience of experts, who essentially interact with CASE tools to design diagrams. However, both GRL and UCM, and hence AoGRL and AoUCM diagrams, are often modelled by sketching on physical medium like white-boards and sheets of paper. Therefore, we avoid hardly sketchable symbols by taking a *degraded* version of the symbols (e.g., swapping the background colour and the icon of a symbol).

6.2. Semiotic Clarity

Principle formulation: "*There should be a 1:1 correspondence between semantic constructs and graphical*".

In order to assess the Semiotic Clarity, we rely on the 2012 version of the ITU-T Z.151 specification document [19]. The semantic constructs of GRL and UCM were sorted into the six groups defined in section 5.1.1. Previously, the semantic constructs of AoURN introduced to support AOM in URN were collected in [27]. Once again, we only consider the incremental addition to AoURN, i.e., symbols that are in AoURN and not in URN.

In this section we present the synthesized data. The complete analysis can be found in appendix A.

Table 4   Number of semantic constructs (SC) metaclasses by categories

Category	Description	#SC
To Consider	Elements that should be mapped to symbols	66
Abstract	Abstract metaclasses that are not mapped to symbols. Although, their specialization could be mapped to symbols	11
Structural	Metaclasses aim to structure the metamodel. Therefore, no symbols are mapped to them.	14
Collection	Enumeration metaclasses from which we should consider combination of values. These could be associated to elements of the "To consider" set.	12

| Graphical | Elements that refer to technical information such as the type of font or special location. | 16 |
| Out of Scope | Metaclasses that intentionally do not have graphical representation | 8 |

We analysed in detail the combination of the "*Collection*" set and the "*To Consider*" set since the combination may introduce new semantic constructs. The final set of semantic constructs counts 103 elements. We collected 70 graphical symbols from [19] and [27]. We paid attention to the combinatory effect as explained in Section 5.1.2.

Figure 46 depicts the assessment of the Semiotic Clarity using the Störrle and Fish formulae. We tagged each semantic construct and each graphical symbol according to the following conventions:

UCM tag: element that is part of the UCM notation;

GRL tag: element that is part of GRL notation;
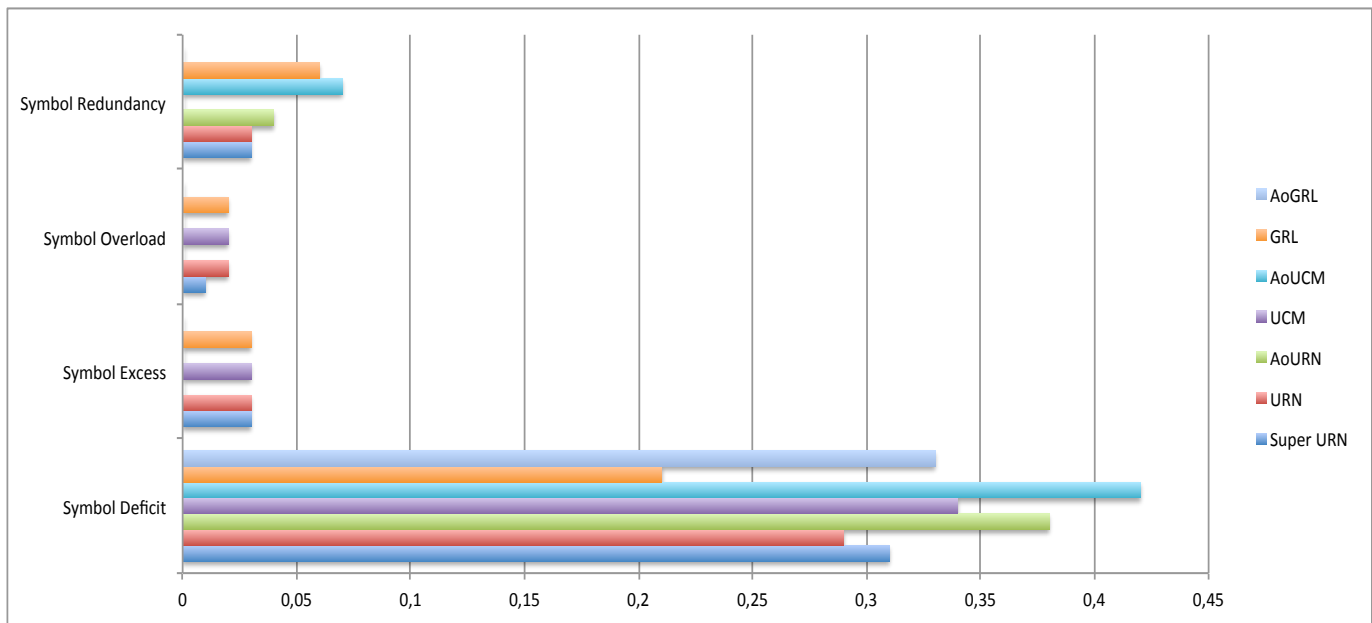
URN tag: elements that is part of UCM or GRL notation;

AoUCM tag: element introduced by Mussbacher to support AOM for UCM;

AoGRL tag: element introduced by Mussbacher to support AOM GRL;

AoURN tag: element introduced by Mussbacher to support AOM for both GRL and UCM, i.e., element that belongs to GRL or UCM set;

SuperURN: element that is part of either URN or AoURN set.

Figure 46   Assessment of the Semiotic Clarity.
Each criterion ranges between 0 (better) and 1 (worse)

Figure 46 shows that most problems are related to the symbol deficit criterion. Indeed, symbol redundancy, symbol overload, and symbol excess remain negligible (less than 0.07).

The UCM notation lacks of graphical representations for its semantic constructs (scores 0.34). As noted by Genon *et al.* in [13], the UCM notation has no graphical representation to support performance analysis. We agree with Genon *et al.* who stated that we should not represent graphically such concepts. Furthermore, the UCM notation does not graphically represent UCM plug-ins (see section 8.3 in [19]). Also, The GRL notation suffers from symbol deficit (scores 0.21). For instance, the evaluation strategies are not graphically represented. Again, we could represent such strategies textually or by using CASE tools features.

Table 5 shows that the concepts introduced by Mussbacher to support AOM (column AoURN) increases symbol deficit around 10% compared to the current notation (column URN). Moreover, we assessed that AoGRL degrades slightly more the results of symbol deficit than AoUCM. Indeed, AoUCM and AoGRL increase symbol deficit (compared to UCM / GRL) by 8% and 12%, respectively.

Table 5   Assessment of URN and AoURN (AoUCM + AoGRL)

	URN	AoURN	AoUCM	AoGRL
Symbol Overload	0.02	0.00	0.00	0.00
Symbol Deficit	**0.29**	**0.38**	0.42 (0,34 in UCM)	0.33 (0,21 in GRL)
Symbol Excess	0.03	0.00	0.00	0.00
Symbol Redundancy	0.03	0.04	0.07	0.00

Mussbacher chose to visually represent AoGRL concepts such as Aspect Graph or Pointcut Graph. Furthermore, in AoUCM, he did the same for the Pointcut Map and Aspect Map. However, these four representations exist for comprehension purposes and are not intended to be in the AoURN notation. If we consider that each of them could be represented textually, we would reduce the Symbol Deficit for AoUCM and AoGRL. In the following section, especially in Complexity Management, we will see whether these concepts should be represented graphically.

6.2.1 Issues at the semantic level

We discovered two issues when we completed the AoUCM and AoGRL semantic construct analysis. The first one concerns the AoUCM aspect marker. The second one is related to how an aspect graph is associated to the pointcut graph.

The Aspect Marker Issue (AoUCM)

Mussbacher stated in [27]: "*Conditional aspect markers* (♠) *indicate that the* [UCM] *scenario may not continue past the aspect marker, whereas "regular" aspect markers* (◆) *guarantee that the scenario continues past them*".

However, we found an example that clearly shows that this assumption is not correct. Figure 47 illustrates that the aspect view (represented by using --➤) of a regular marker (M1) could be composed of a conditional marker (M2).

In the following, we give an example depicting that the aspect marker does not respect Mussbacher's statement. Either we choose to not represent it graphically by introducing symbol deficit or we would need to change its meaning at semantic level, which is out-of-scope of this thesis.

Example

The scenario executes R1, then the aspect related to (M1) is executed, which contains (M2) and R3. The aspect related to (M2) is then executed. Then, R2 is executed and depending on the condition *[cond]*, the scenario terminates or executes R3 and finally executes R4.

This example proves that an aspect marker could not guarantee the none interruption of a scenario. Therefore, a CASE tool like jUCMNav should have a continuous check of what a regular aspect marker is composed of. It has to check whether one of the aspect marker path leads to the termination of the scenario.
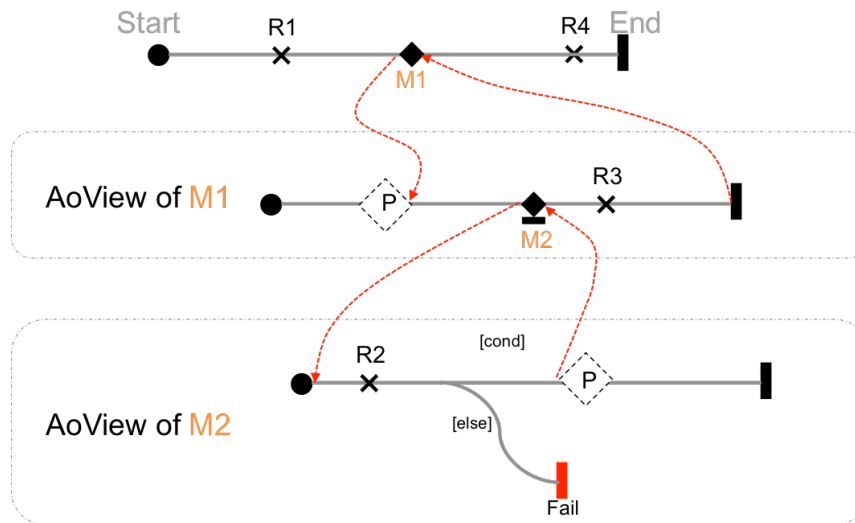
Figure 47   The aspect marker M1 (◆) does not guarantee
the none-interruption of a scenario

**Association of Aspect Graphs (AoGRL)**

In AoGRL, a pointcut marker is associated to only the intentional elements of the base model. However, it seems that if an advice in not linked (e.g., via a contribution link) to an intentional element in a pointcut graph, we could not represent that an advice is attached to the base model with an aspect marker. Figure 48 illustrates this issue with the

pointcut graph *A* and ***B***. When the goal advice is linked through a contribution arrow to an intentional element (e.g., Goal1), the intentional element of the **base model** is tagged with a pointcut marker (◆).

However, if the goal advice (in pointcut graph B) is not linked to any intentional element, the base model (base model B) does not indicate that an aspect is actually associated the model, i.e., there is no pointcut marker (◆) on the model.

A solution could be to associate a pointcut marker to GRL actors. In our case, a pointcut marker would be associated to the Actor of the base model B.
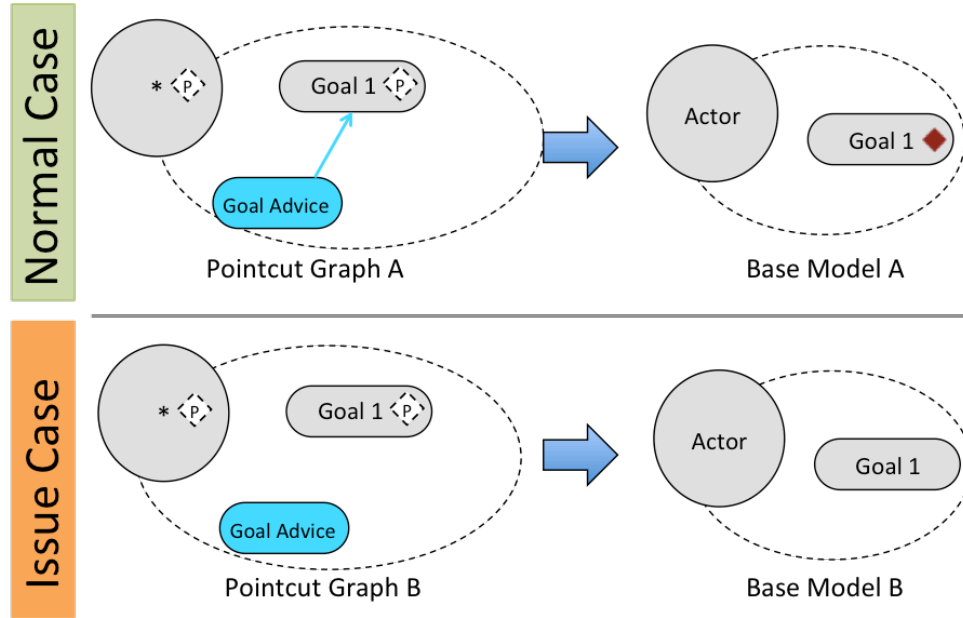
Figure 48   Goal Advice of the Issue Case is not represented in the Base Model

6.2.2 Validation of the metrics

In Sections 5.2.3 and 5.2.4, we proved that the symbol excess and the symbol deficit metrics of Störrle and Fish are mathematically equal to Genon's metrics. In Figure 49, we evaluated Symbol Excess and Symbol Deficit for UCM by using Störrle and Fish, Genon and our metrics. It appears that the Störrle and Fish metrics are really equal to Genon's metrics at a precision of 10^{-9}. This validates what we mathematically proved in Sections 5.2.3 and 5.2.4. Furthermore, our results confirm the trend that Genon *et al.* observed for UCM in [11]: all the criteria are negligible except Symbol Deficit. However, the results are not strictly the same between the assessment of Genon *et al.* and our assessment as

we made different choices, e.g., an UCM empty point has only one (our assessment) graphical representation and not two (Genon *et al.*'s assessment).

We associated some symbols differently from Genon *et al.*'s assessment. Indeed, we used the Patter-based Matching technique (see Section 5.3.1), which may vary the results from the technique used by Genon *et al.* For instance, unlike Genon *et al.,* we did not associate the same graphical symbol for the UCM waiting place and the start point. Indeed, the waiting place is associated to the pattern (━━━●━━━) while the UCM start point is associated to the pattern (●━━━━).
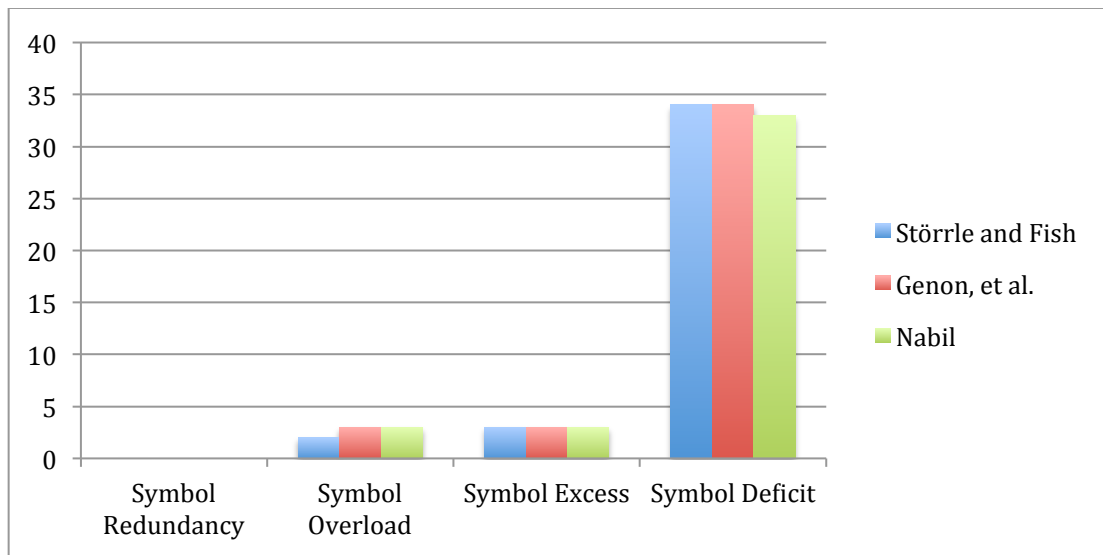
Figure 49   Assessment of UCM Semiotic Clarity

In the remainder of this section, we consider the assessment of the metrics related to symbol redundancy and symbol overload. We have three sets of metrics: the Störrle and Fish metrics, Genon's metrics and our metrics, which is an improvement of Störrle and Fish metrics. Table 6 shows an important result: Genon's metrics of symbol redundancy and symbol overload give almost (accuracy of two percent) the same results as the Störrle and Fish metrics (see how the columns A and B of each language are close to another).

Table 6 Modelling languages assessment for
symbol redundancy (S. R.) and symbol overload (S. O.)
with Störrle and Fish (A), Genon *et al.* (B), and mine metrics (C) in percentage.

	AoGRL			GRL			AoUCM			UCM			AoURN			URN			SuperURN		
	A	B	C	A	B	C	A	B	C	A	B	C	A	B	C	A	B	C	A	B	C
S. R.	0	0	0	6	9	10	7	8	**12**	0	0	0	4	5	7	3	4	4	3	4	5
S. O.	0	0	0	2	3	3	0	0	0	2	3	3	0	0	0	2	3	3	1	3	2

We could argue that our metrics are not useful since, most of the time, they provide re-sults closely related to the two others. However, most of the graphical symbols of "SuperURN" are not associated to more than one construct. In fact, only 2 symbols over 76 are associated to two constructs. The same phenomenon occurs for the constructs: only 4 constructs are associated to two symbols over a set of 104 constructs. Therefore, these two phenomena reduce the difference between our metrics and the Störrle and Fish metrics. However, our metrics are more efficient since they degrade the value of AoUCM according to criterion of symbol redundancy. Indeed, since AoUCM has only 12 con-structs (small set of constructs), even if there is only one problematic construct (i.e., a construct linked to more than one symbol), our metrics stand out more that problem. In comparison, the other modelling languages we assessed have more constructs or symbols. There are not enough problematic symbols or constructs in order to see a significant dif-ference between our metrics and the two other.

6.3. Perceptual Discriminability

<u>Principle formulation</u>: "*Different symbols should be clearly distinguishable from each other*".

In this section we focus on the discriminability of AoURN. In [13], Genon *et al.* presented improvements to increase the perceptual discriminability of UCM. In [23], Moody and Heymans introduced improvements that could be applied to GRL.

6.3.1 AoUCM

It seems important to increase the <u>*Foreground Differentiation*</u> between the UCM path and the elements (e.g., responsibilities) on it. Indeed, unlike other modelling languages, UCM is not composed of elements connected to one another by links. Instead, the elements are dispatched on the UCM path. Therefore, we propose to change the colour of the UCM path to highlight the elements that are on the path, such as the responsibilities. Figure 50 shows the difference when the UCM path is depicted in grey, which highlights the responsibilities R1 and R2. An empirical study could be conducted to assess to which extent this modification improves the discriminability.
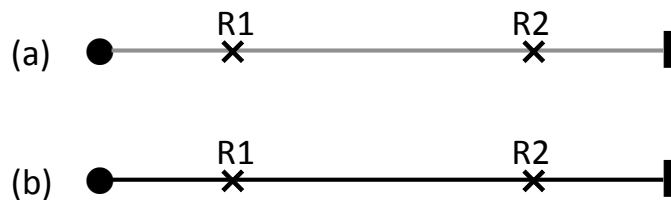

Figure 50 Foreground differentiation between UCM paths and the Responsibilities. The responsibilities are highlighted in (a) and the current version is depicted in (b).

The visual variable <u>*Shape*</u> is one of the most important to differentiate graphical symbols. However, AoURN overuses the pattern (⸺◆⸺). The differentiation is made possible only by referring to textual differentiation (see Figure 51, (e) and (f)) or by modifying the visual variable <u>*Size*</u> (e.g., compare elements (a) and (e) in Figure 51). Furthermore, the UCM notation already extensively uses this pattern (e.g., stub, synchronization stub). We propose to change the <u>*Shape*</u> of the graphical symbols (a), (b), (c) and (d) to differentiate

symbols appearing on a base model from symbols appearing on a pointcut map (i.e., symbols (e) and (f)). Therefore, we introduce an icon for aspectual view (👁) and a new icon for the replacement pointcut stub (⟳). We switch the background *Colour* for the pointcut stub and the replacement stub to increase discriminability. However, if these symbols have to be sketched (e.g., whiteboard), we could simply change back the background colour and the icon.
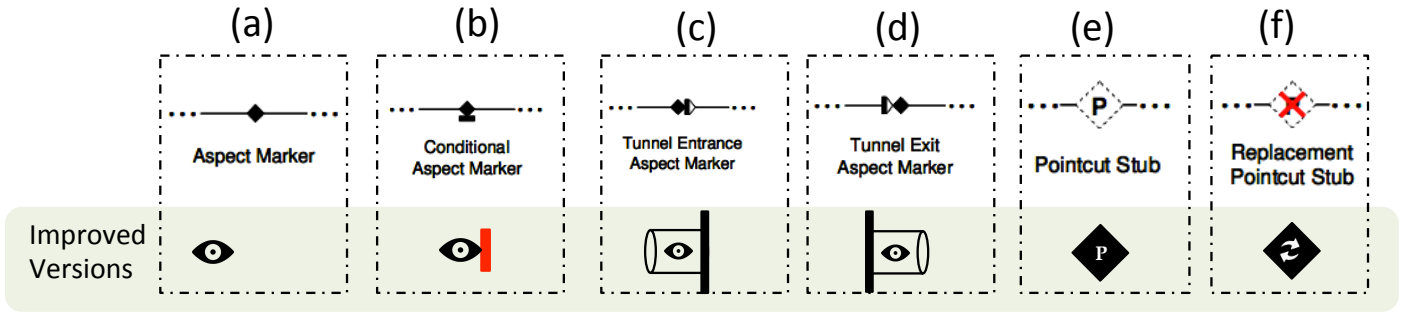

Figure 51   Graphical symbols using the pattern (─◆─)
The symbols in the green background are the improved versions

The visual variable *Texture* is used only on the edges of dynamic stubs (dashed edge) and on protected components (double line of the edges). We agree with Bertin's claim [5] that *Texture* should be used as a means to fill in the shapes (i.e., areas) of the graphical symbols.

According to [19], dynamic and statistic stubs share the same role of indicating the presence of hierarchically-structured UCM maps. The difference between them is that a dynamic stub may contain multiple maps instead of only one map for a static stub. Therefore, a dynamic stub can be used as a static stub but not in vice versa [19].

We suggest keeping the notion of dynamic stub and remove the notion of static stub. Moreover, since we do not need to distinguish between them anymore, instead of a dashed edge, we propose to use the plain edge for the dynamic stub. Figure 52 represents the old versions (a) and (b) of stubs and the new one (c).
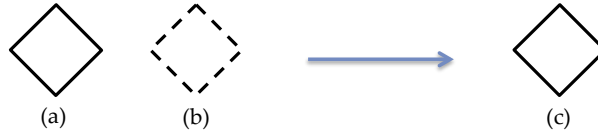
Figure 52　(a) The old static stub,
(b) the old dynamic stub and (c) the new dynamic stub

Depending on the graphical layout of a UCM path, the Or-Fork and Or-Join are especially hard to distinguish because they do not differ from the UCM path. The UCM standard [19] does not give a special representation of the paths of the Or-Fork and Or-Join.

We propose to either force the output paths to respect a certain figure such as the one used in [19] to describe the Or-Fork (Figure 53, (a)) or use a shape to represent the Or-Fork/Or-Join (Figure 53, (b)). Concerning the Or-Join, we do not change its representation. Using the symbol depicted in Figure 53 (b) would also reduce the confusion when the scenario presented in Figure 54 (b) occurs). The current solution is to use an arrow (see Figure 54 (a)).
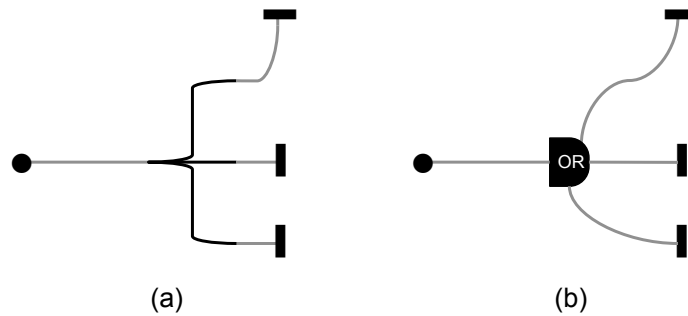
Figure 53　Proposed versions of OR-Fork
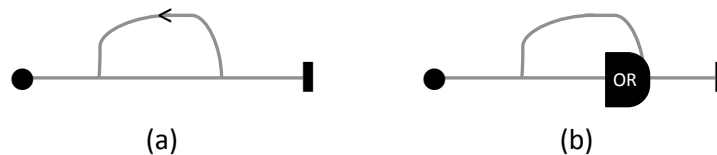
Figure 54　Confusion between the Or-Fork and Or-Join

The visual variable _Colour_ is used in AoUCM for understanding purposes only. It seems that there is no intention to officially use _Colour_ on AoUCM. We suggest keeping _Colour_ in the official visual notation of AoUCM.
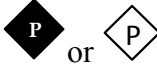
Finally, _Brightness_ and _Orientation_ are neither used on UCM nor on AoUCM. These visual variables do not seem to be relevant for UCM.

6.3.2 AoGRL

GRL graphs, are _flat models and do not support hierarchies._ In [27], Mussbacher added the vertical decomposition concern to the GRL notation to be able to encapsulate aspectual concerns outside a GRL graph.

In this section, we restrict the analysis to how to improve the distinction between the graphical symbols of AoGRL. Table 7 (a) shows the graphical symbols referring to aspect markers, i.e., a marker that indicates on a base model graph that an aspect is applied at that spot. Table 7 (b) shows the pointcut marker, i.e., marker that identifies an element of the base model in the pointcut graph. We propose to use the same symbol for the aspect marker than the one used for the UCM aspect marker (⊙) in order to increase the discriminability between the GRL aspect marker and the GRL pointcut symbol. For the GRL pointcut maker, we suggest to use the same symbol like the UCM pointcut stub.

Table 7   Aspect marker (a) and pointcut marker (b), current versions (above) and newer versions (below)

(a)	(b)
◆	⬦P⬦
◉	◆P or ◇P

Furthermore, we could use the visual variable _Colour_ to improve the discriminability in the pointcut graph. We should choose colours that do not interfere with the colours (e.g., shades of green, red and yellow) used in GRL diagrams by jUCMNAv to evaluate a GRL graph [20]. Therefore, let us use blue to represent the aspectual elements and the composition rule.

The other elements of the pointcut graph will then be represented in white (see Figure 55, (Solution A)) or grey (see Figure 55, (Solution B)). Depending on the background colour, we chose the version of the pointcut marker that increases more the contrast. We do not fill the actor boundary because it would decrease the contrast with its elements.

Solution A   Solution B
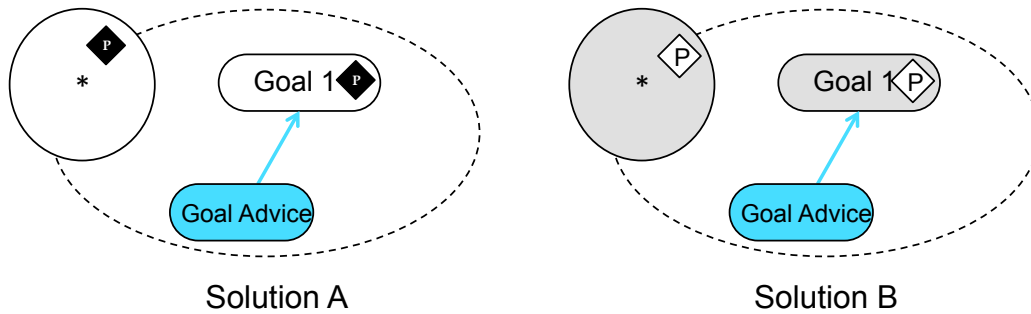
Figure 55   Improved pointcut graph: the aspectual advice is represented in blue.

Furthermore, we no longer need to differentiate links and arrows (e.g., dependency, contribution) by applying pointcut markers (marker). Indeed, we would simply colour an arrow either in blue if it is part of the "aspect graph" or in black if it is part of the pointcut elements (i.e., for matching purpose).

6.3.3 Colouring AoURN elements

We could also apply the blue colour to highlight the aspect in AoUCM that is in an AoView. Figure 56 highlights in blue the aspect of the AoView related to the aspect marker M1.
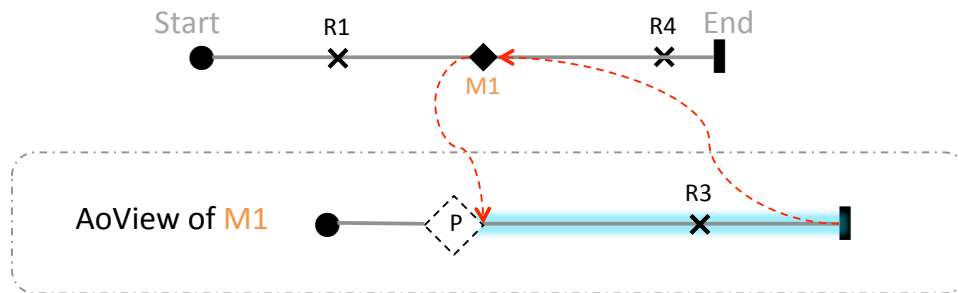
Figure 56   Highlighted Aspect of the AoView of Aspect Marker M1

6.3.4 Validation of the metrics

In Section 5.3.1, we demonstrated that Störrle and Fish assessment of the primacy of *Shape* is not general as it was intended to evaluate only the UML Use Case Diagrams. We also illustrated the issues that we could encounter when we use the Genon *et al.* method to evaluate the visual variable *Shape*. Therefore, we decided to use our proposal to assess *Shape*: the Pattern-based Matching (see Section 5.3.1). Nevertheless, we should recognize that our technique could be simulated by the technique used by Genon *et al.* Indeed, Genon's technique uses labelled text to characterise a *Shape* family. Therefore, we could simply use more detailed tags for each *Shape* family. For instance, instead a "Diamond" *Shape* family (◆), we could tag the *Shape* family as "Diamond attached to in and out paths". However, our Pattern-based Matching techniques seem to be more practical since it is visual.

We applied the metrics of Störrle and Fish to assess the visual distance, the redundant coding, the perceptual pop-out, and the textual differentiation criteria. For the visual distance, we evaluate whether two shapes share the same pattern instead of using the metric of Störrle and Fish, which is once again not correct. Figure 57 shows the results we obtained for each of the four criteria. However, as stated by Störrle and Fish in [39], empirical studies are necessary to interpret these results. Even though the Störrle and Fish metrics were designed with adjustable weights to accommodate future empirical findings, their current states cannot be used to assess the Perceptual Discriminability. Finally, we discussed issues related to how metrics should be used to assess the Perceptual Discriminability in Section 5.3.5.

Therefore, we state that the assessment of the Perceptual Discriminability is more valuable when we actually identify and resolve issues as Genon *et al.* did in [10].
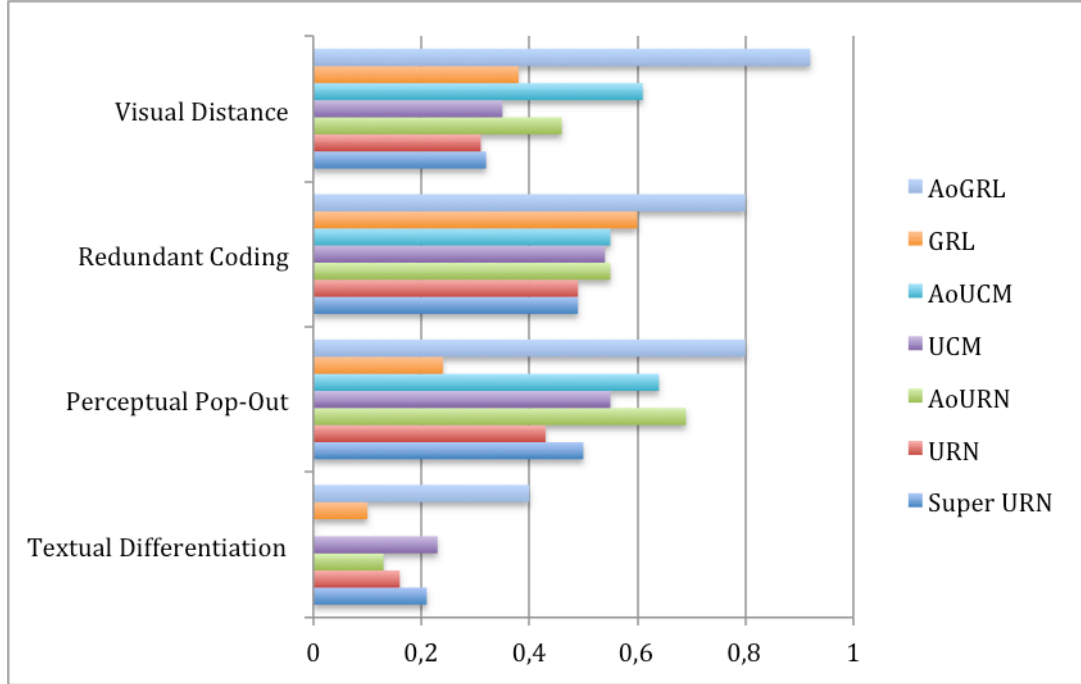
Figure 57   Assessment of Perceptual Discriminability
with Störrle and Fish metrics

6.4. Semantic Transparency

Principle formulation: "*Different symbols should be clearly distinguishable from each other*".

6.4.1 AoUCM

Instead of using a particular composition language, Mussbacher relies on the expressive power of UCM itself to support AOM. Therefore, AoUCM inherits from the problems identified in [13]. Indeed, AoUCM uses conventional shapes such as the (⎯◆⎯) pattern, which does not convey any particular meaning.

In Section 6.3.1, we used an "eye" icon to increase discriminability. The rationale behind that icon is that it highlights the word "view" of the notion of aspect views. In Table 8, we use that icon for four symbols. In (a) we use it as a symbol for the *regular aspect marker*. In (b), we use it for the *conditional aspect marker*, combined with a red bar expressing that the scenario may not continue past that marker (exactly as an end

point). Finally, we use the "eye" icon for (c) *entrance* and (d) *exit tunnel aspect markers*. The symbol for tunnel is proposed by Genon *et al.* in [11].

Table 8   An "eye" icon used to refer to symbols of aspect views

(a)	(b)	(c)	(d)
👁	👁▮	👁	👁

For the *pointcut stub* and the *replacement pointcut stub*, Mussbacher used the UCM dynamic stub as a primary shape on which he added either the letter P or a crossed P, respectively. On the one hand, the *pointcut stub* (⬦P) suggests its meaning once we learn that "P" stands for pointcut. On the other hand, the *replacement pointcut stub* (⬦X) suggests the opposite of its meaning (semantic perversity). Indeed, the cross often suggests the notion of deletion and not replacement. This is why we suggest using a different icon (such as ⇄) that refers to the idea of change.

AoUCM has an *anything pointcut* (**………**) symbol that allows a pattern to be matched where a portion of the pattern is open to variations in the match [27]. Figure 58 shows two UCM paths one that contains the anything pointcut and a regular UCM path. The pattern matches the responsibilities A – D – Z and not the responsibilities A – B – C – Z since the anything pointcut matches the shortest candidate for a match.
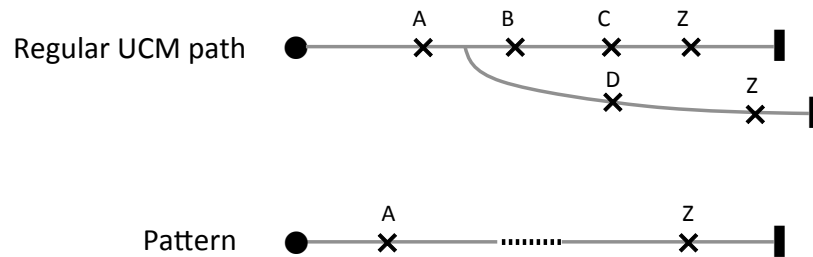
Figure 58   Example of an anything pointcut

This symbol is semantically immediate once it is put on a UCM path. Indeed, it matches any form of a UCM path since the dots do not exist in the UCM notation. Additionally, the dots are a kind of abstraction of a path.

When we discussed the Perceptual Discriminability principle, we saw that the visual variable *Size* is only used to distinguish between symbols. We could use it to give a *meaning* to a symbol such as the UCM stub. For instance, the size of the stub symbol could be related to the path contained in the stub.

6.4.2 AoGRL

Mussbacher reused two symbols of AoUCM for the aspect marker and pointcut marker. We propose to use the "eye" symbol for the aspect marker (◉) and the diamond symbol (◆ or ◇) for the pointcut marker. The rationale supporting these symbols has already been explained in Section 6.3.2.

AoGRL defined a *pointcut deletion marker* (✕), which can be used to remove matched elements of the composition rule. This symbol is also semantically immediate because the notion of "delete" is often represented with a cross symbol.

When applying the Perceptual Discriminability principle, we suggested to colour aspects in blue. That gives a visual clue about the elements that could be deleted (i.e., elements that are not part of the advice, in black) from the ones that could not be deleted (i.e., elements part of the advice, in blue).

AoGRL uses a tag <<anytype>> on the intentional elements (e.g., a goal) of a pointcut graph to match any intentional element of a base model. The problem is that the shape influences the perception. In Figure 59, even if the goal (a) is tagged with <<anytype>>, it is difficult to imagine that this element could represent any intentional element. We propose to introduce a new symbol (Figure 59, (b)) designed to look different from all GRL symbols. This symbol could match any type of intentional element. In fact its shape convey the idea of anything because it has not a conventional geometrical-base structure (e.g., circle, rectangle).

(a)   (b)

Figure 59   AoGRL Anytype symbol

6.5. Complexity Management

<u>Principle formulation</u>: "*Include explicit mechanisms for dealing with complexity*".

AoUCM did not extended UCM capabilities in terms of decomposition since AoUCM reuses the UCM vertical decomposition.

Unlike UCM, GRL is a flat model with neither vertical nor horizontal decomposition. Therefore, AoGRL introduced a means to decompose GRL diagrams according to vertical decomposition. Once the issue related to GRL aspect marker (see Section 6.2.1) has been resolved, AoGRL capabilities are enough powerful to encapsulate any GRL element.

UCM already supports horizontal decomposition (see the metaclass `UCMMap` in [19]) but it gives no symbol to show whether a UCM path is connected to another path. It could be interesting to visually represent that a *start point* is connected to an *end point* and an *end point* is connected to a *start point*. Figure 60 shows how we represent that E1 is followed by another path starting by S2. Visually, it is obvious that E1 fits into S2.

Every time an *end point* in connected to a *start point*, we suggest to use the end point and start point symbols used for E1 and S2 instead of the conventional one.

Figure 60   Visual cues to connect maps of UCMMAP

Unlike UCM, GRL does not support any horizontal decomposition. Further research is needed to support such decomposition.

6.6. Cognitive Integration

Principle formulation: "*Include explicit mechanisms to support integration of information from different diagrams*".

The previous section reduced the diagrammatic complexity by decomposing diagrams. This section aims to provide means to mentally integrate the information spread among these (sub-)diagrams.

Moody suggested to include mechanisms at the notational level to help the reader assemble information (*conceptual integration*) and to give cues to simplify navigation across diagrams (*perceptual integration*).

6.6.1 AoUCM

In order to mentally apply an aspect to a *base model*, we need to understand an *aspect map*, composed of *advices* and *pointcut map*. Then we need to understand how the advice is mapped to the base model. Mussbacher defined the notion of AoViews to simplify that composition. Instead of composing all advices to the base model in one step, an AoView highlights one advice of the *aspect map* that is applied to a precise location of a *base model* marked by an *aspect marker*. However, AoUCM lacks on specifying symbols (except the start / end of pointcut expression, represented in grey) to differentiate the three types of maps: *base model*, *aspect map* and *pointcut map*. Therefore, we propose to identify these types of diagrams with an icon. Figure 61 depicts icons, whose shapes are in-

spired by the wind rose. For aspect map, we propose the icon (A), coloured in blue (same colour as UCM AoViews and GRL aspectual elements). We suggest either the icon (B.1) or the (B.2) to represent the pointcut map. For both, we fill the shape with a pattern *Texture* to refer to the role of a pointcut map. Furthermore, the second icon (B.2) is named "Pattern" to convey more effectively (Dual Coding) the role of the pointcut map. We did not apply that pattern to the pointcut stub of an *aspect map* to avoid any confusion.
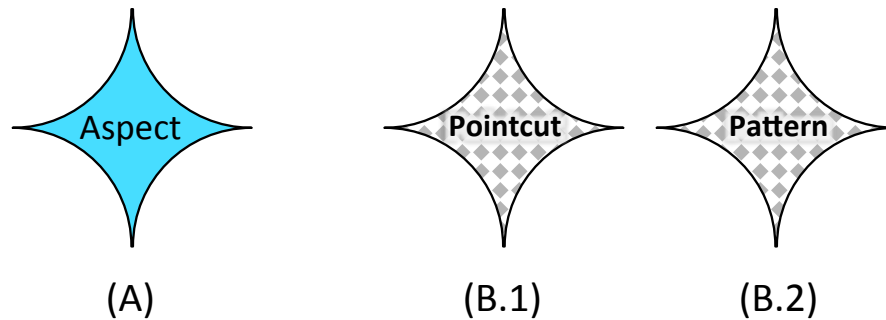
Figure 61   Icons to identify aspect map (A) and pointcut map (B.1 / B.2)

It appears also important to show the three types of maps at the same time to help the reader compose the aspect to the base model (see Figure 62).
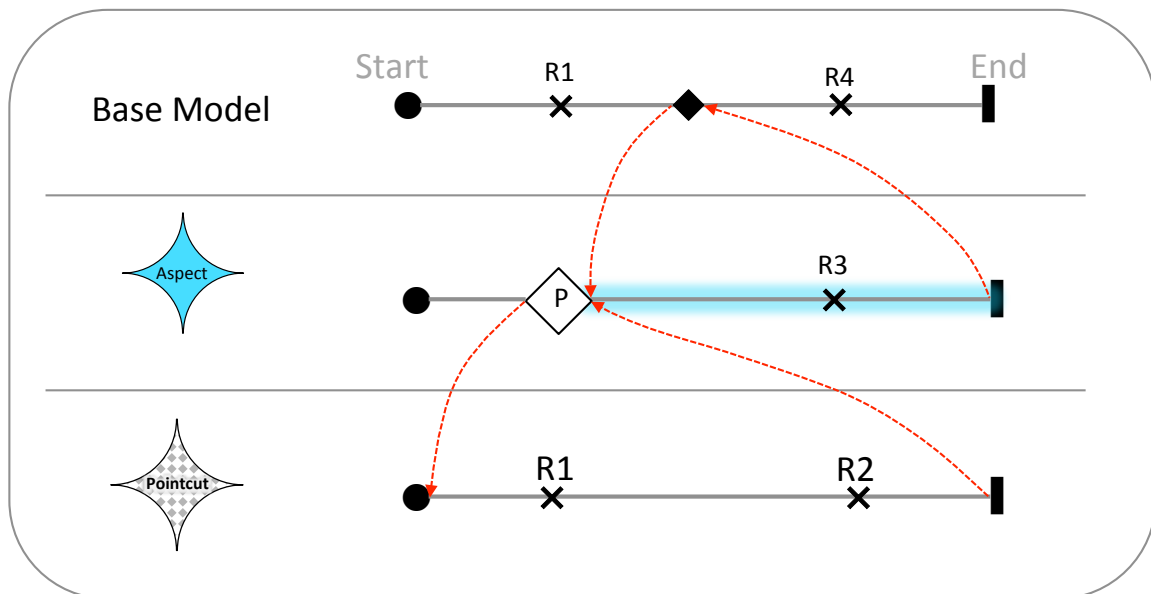
Figure 62   The three types of maps: *Base Model*, *Aspect Map*, and *Pointcut Map*

6.6.2 AoGRL

AoGRL introduced a vertical decomposition for which we improved the visual notation in Perceptual Discriminability (see Section 6.3.2) and Semantic Transparency (see Section 6.4.2).

We suggest using the contextualisation technique to easily navigate through large diagrams and reduce the complexity shown at a time. Figure 63 illustrates how the contextualisation technique could be applied for a specific actor (e.g., *Actor A*) and what a hover event on a GRL link (e.g., a correlation link) could show. In Figure 63 (left), we handle the diagrammatic complexity by intentionally hiding the elements of all actors except the focused Actor A. We also show the GRL links, which go outside Actor A, but we consider that a link goes from a GRL intentional element of Actor A to another Actor.

Furthermore, in Figure 63 (right), we illustrate a hover mouse event on a correlation link (in green). This triggers a CASE tool feature, which unfold all the elements of Actor C. Moreover, if we hover an element of Actor C, we will display all the links that go outside of Actor C boundary.
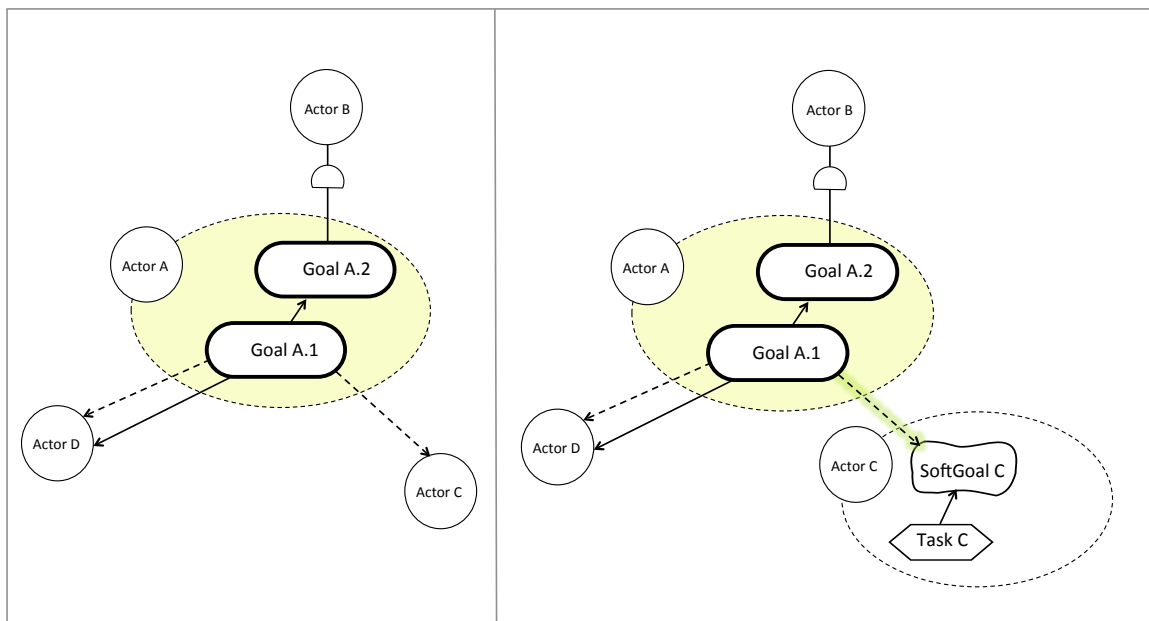
Figure 63   Contextualisation technique applied on GRL actor A.
A hover on correlation link (highlighted in green) shows all the Actor C elements

6.7.   Visual Expressiveness

Principle formulation: "*Use the full range and capacities of visual variables*".

In this section we measure how the graphic *design space* — composed of visual variables and their respective range of values — is used in AoURN.

It is difficult to distinguish values of visual variables used in AoUCM since some of them were chosen for comprehension purposes. For instance, the aspect marker is sometimes coloured in red and sometimes in black. In our analysis, we consider all these variations to be part of AoUCM. However, we discuss the different cases to suggest what should be part of the primary and secondary notations.

In Table 9, we measured the elements introduced to support AOM in UCM.

Table 9   Design space covered by AoUCM

	Power	Capacity	AoUCM values	Saturation *
Location (x,y)	Interval	10 – 15	/	0%
Shape	Nominal	Unlimited	(conditional aspect marker) (aspect marker tunnel entrance) (aspect marker tunnel exit) .... (anything pointcut element) (pointcut stub + aspect marker) (start of pointcut expression + local start point) (end of pointcut expression; local end	/

			point)	
Colour	Nominal	7 – 10	Black&white, grey, red	30%
Texture	Nominal	2 – 5	Dashed line, filled	40%

AoUCM uses almost the same design space as UCM. Therefore, it does neither better, nor worse than UCM except for *Colour*, which is now used in AoUCM. *Texture* could be used more extensively in AoUCM but instead of applying it to edges, the notation should use it to fill shapes (see Figure 61 (B.1) and (B.2)).

AoGRL introduced new symbols to support AOM. However, their appearance are close to UCM or AoUCM concepts. It is not a drawback: even if it decreases the Perceptual Discriminability, it improves the understanding of the concepts. Indeed, symbols that are closely related to GRL (i.e., appearance and meaning) benefit to expert as they are already used to the language. In addition to these concepts, symbols that are close to AoUCM (e.g., AoGRL aspect marker and AoUCM aspect marker) increase understanding, as they have to be learnt only once.

Colour in AoGRL is even less used than in AoUCM, which gives a range of potential values to use in further development of AoGRL. However, we should pay attention to the saturation value of 20% of *Colour,* which only takes into account the AoGRL elements. Indeed, if we gather GRL, AoGRL, and the version of GRL used in jUCMNav, that percentage would increase up to 40% (Black&White + shades of green + red + yellow).

Table 10   Design space of AoGRL

	Power	Capacity	AoUCM values	Saturation *
Location (x,y)	Interval	10 – 15	/	0%
Shape	Nominal	Unlimited	◆ (aspect marker) (pointcut marker) ✕ (Pointcut Deletion Marker)	/
Colour	Nominal	7 – 10	Black&white, red	20%
Texture	Nominal	2 – 5	Dashed line (e.g., pointcut marker), filled (e.g., aspect marker)	40%

6.8. Principle of Dual Coding

Principle formulation: "*Use text to complement graphics*".

AoUCM and AoGRL rely on the metaclass `Comment` [19] to store information. Genon *et al.* held that the `Comment` metaclass should not be considered as an example of "*Annotation*". As we agree with them, we will not develop any further.

 AoUCM uses *hybrid symbols* to encode the pointcut stub. Both the diamond shape and the text "P" are needed to represent a pointcut stub. Indeed, without the text "P" (see Figure 64, (a)), we would confuse the pointcut stub with an UCM pointcut (see Figure 64, (b)).

(a)   (b)
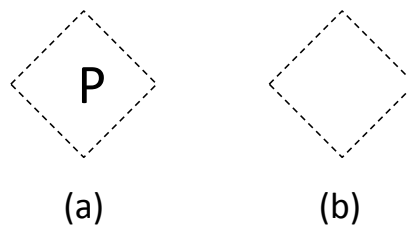
Figure 64   AoUCM Pointcut stub (a) / UCM stub (b)

We do not consider that the AoGRL pointcut stub ( ) is case of *hybrid symbol* since it has dashed edges and it is not filled unlike the AoGRL aspect marker (◆). Therefore, even without its text "P", we could recognize a pointcut stub.

That being said, we consider that there is no case of hybrid symbol in the AoGRL notation.

Text is used in AoUCM and AoGRL to encode more sophisticated concepts such as a textual matching mechanism. It allows matching on names of UCM and GRL elements. However, this mechanism takes place at the diagram level.

6.9. Graphic Economy

Principle formulation: *"The number of different graphical symbols should be cognitively manageable"*.

AoUCM

In [13], Genon *et al.* evaluated the graphic complexity to 28, to which we should add 11 elements brought by AoUCM. A total of 39 symbols can reveal not manageable. However, we should take into account that some of the AoUCM symbols are similar to UCM symbols. Moreover, UCM and even more AoUCM target an audience of experts who can deal with a larger visual vocabulary, i.e., the number of symbols in the notation.

Instead of reducing the visual vocabulary, we suggest to increase it in order to solve the symbol deficit that we assessed with Semiotic Clarity. We proposed two new symbols: one for the aspect map ( ) and another for the pointcut map ( ).

If the visual vocabulary of AoUCM is an issue, we could replace the regular marker and the conditional aspect marker by the tunnelling aspect markers (entrance + exit). Indeed, the tunnelling aspect marker is more general than the two other kinds of aspect markers. The others concepts of AoUCM are essential to support AOM in UCM. We suggest to keep them in the visual notation.

AoGRL

In [23], Moody and Heymans evaluated the graphical complexity of i* to 16. However, we cannot take that complexity for GRL because the GRL notation is more general than i*. We evaluate the complexity of GRL to 29 graphical symbols.

To this number, we add a graphical complexity of 3 introduced by AoGRL. The graphical complexity of AoGRL is low because several concepts are not graphically represented (e.g., anytype element). Unlike UCM that needs to be read according to a direction (i.e., from a start point to and end point), GRL does not include such a direction of reading. Therefore, there is no need to create symbols according to a direction such as the entrance/exit aspect marker of AoUCM. The three AoGRL symbols are mandatory to support AOM. In consequence, we could not remove any of them.

Chapter 7.   Future Work

This chapter includes the suggestion of a new line of research that uses CASE tool features to improve the cognitive effectiveness of notations. Moreover, we propose a piece of software to automatize evaluation of visual notations.

7.1. Supporting the PoN with CASE tools

Some principles of the theory (e.g., principle of Cognitive Integration) refer to features of modelling languages that have to be handled by CASE tools. However, it seems to us that the PoN does not explicitly consider such tools. Yet, for instance, in the Cognitive Fit principle, criticism is made about modelling languages that are designed for "*pre-computer era*" which "*make little use of the powerful capabilities of modern graphics software*". These elements lead us to consider features enabled by CASE tools.

7.1.1 Need for CASE Tools in Modelling Languages

CASE tools provide specific features that could support some principles of the PoN. The PoN theory itself refers to the need for designing modelling languages by taking into account the capabilities of modern graphics software ([25], pp. 772).

7.1.2 Extending Bertin's Visual Variables

Several principles of the PoN rely on the Bertin's visual variables (see Figure 65), which where published before the development of CASE tools software. The visual variables consider only *static* characteristics and not *dynamics* one that can be introduced by modern graphics software such as CASE tools.
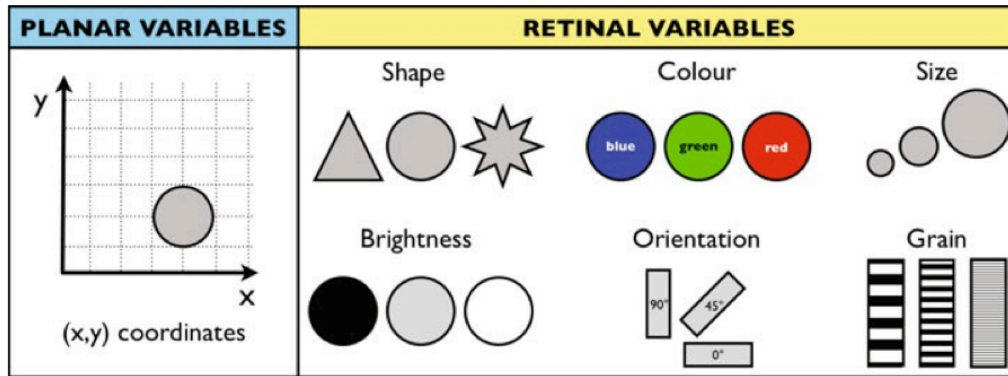
Figure 65   Bertin's visual variables (from Genon *et al.* in [10])

These reasons drive us to propose a new visual variable (inspired by Rebah *et al.* in [32] and Carpendale in [7]) called "*Animation*". This visual variable is not restricted to the idea of movement from a point A to a point B. It could be generalized as any significant change in one or more of the current visual variables. Nevertheless, all kinds of animations can be encompassed by this definition. Indeed, CASE tools offer capabilities that cannot be described in terms of these visual variables such as a deformation or a focus (e.g., highlighting the edge of a shape with a flashy colour for a short period of time) on a graphical symbol. These features could modify how to shape looks but it would do that for a short period of time

Animations can be seen according to two perspectives: either as a *technical variable* (i.e., applied to a single graphical symbol to highlight a phenomenon) or as a *modelling variable,* which is applied to a group of graphical symbols. Furthermore, the modelling variable perspective can be used to show temporal changes across a diagram to improve the comprehension of a diagram. Indeed, we could use modelling variable as a means for explaining diagrams in a *step-by-step* way instead of showing only the last versions.

Animation as a technical variable

The visual variable *animation* is composed of several properties. The properties of *Animation* are: **animation type**, **trigger** of the animation, **duration** of the animation, and **frequency** of the animation. A graphical symbol could have one to many *animations*. Each *Animation* is characterized by its properties. An *Animation type* could be either a signifi-

cant change in one/multiple of its visual variable(s) **or** animation features offered by a CASE tool. Each *Animation* has a *trigger*, which defines the start of the animation while the length of the animation is defined by the *duration* property. Moreover, a frequency could be defined, which will execute the animation repeatedly by defining the number of repetitions and interval between them. The UML state diagram depicted in Figure 66 illustrates a graphical symbol regarding its static and animation states. It shows that a graphical symbol never lasts forever in the animation state.
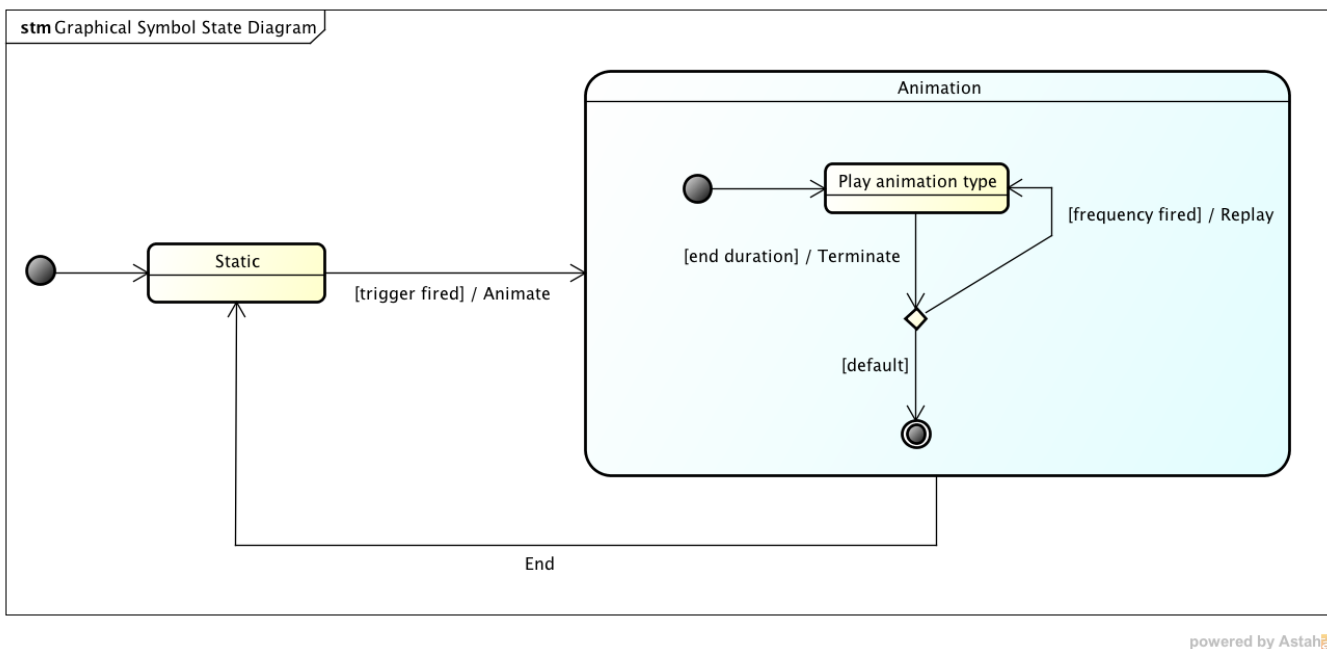
Figure 66   State Diagram of a Graphical Symbol
according to its static and animation states

Bertin's visual variables have to be chosen in accordance with the Bertin's list of visual characteristics (e.g., associative, quantitative). Indeed, some visual variables are more appropriate to encode associative than qualitative information (e.g., *Colour*).

According to Carpendale [7], animations could be used to encode: selective, associative, quantitative, or ordered characteristics. Moreover, its capacity range (i.e., number of perceptible steps for a characteristic) is theoretically infinite.

Animation as a modelling variable

For centuries, cartographers convey information through graphical representation. Nowadays, the computer era enables cartography to not only see a phenomenon but also make visual comparisons, relationships/causes of anomalies, and ability to show trends for long-term analysis of a phenomenon.

This kind of use is not related to highlighting a graphical symbol (i.e., animation as a technical variable). It can be used to facilitate understanding of a phenomenon by using animation on a group of symbols. At the diagram level, we could define a *scenario* of animations that involves one to several graphical symbols. It includes animations applied to individual graphical symbols and those applied to a group of graphical symbols.

The difference between the animation as a ***technical variable*** and ***modelling variable*** rely on the scope and the purpose of the animation. The technical variable is related to a single graphical symbol and has the purpose to point out that symbol. In contrary, animations as modelling variable have a larger scope and purpose, they are applied to several graphical symbols (i.e., a group of graphical symbols), and they bring useful features such handling complexity of a diagram.

Static and Dynamic Visual Variables

Bertin's visual variables are static variables. With the visual variable *animation*, we add dynamic capabilities to graphical symbols. However, both worlds coexist because static graphical symbols could be represented by omitting the visual variable *Animation*. Furthermore, for animations like deformation, the graphical symbol is changed only for a fixed period of time and at the end of the animation, it is reset to its initial state.

7.1.3 Animation and the PoN's Principles

Animations, as technical or modelling variable, could further enhance every principles of the PoN. In the following sections, we will restrict the implication of the visual variable animation to three principles: Perceptual Discriminability, Complexity Management, and Cognitive Integration. These principles are essential for visual notations that are used by experts (e.g., AoURN) since the diagrammatic complexity (i.e., number of symbols on a diagram) and the number of diagrams tend to increase over the modelling process. The

Complexity Management goes hand-in-hand with the Cognitive Integration (i.e., on decompose and the other one integrate). Beyond these two key points, we know that Perceptual Discriminability is also important for experts because of the increase of diagrammatic complexity. Indeed, the more symbols that appear on a diagram (i.e., symbol instances, or tokens), the more we should consider a larger visual distance between these symbols (i.e., symbol types).

7.1.4 Principle of Perceptual Discriminability

The visual distance between two graphical symbols could be highly increased by applying *Animation*, which according to empirical studies is the most cognitively efficient way of focusing human attention to a phenomenon. Moody stated that shapes play a special role in discriminating between symbols (in [25], "primacy of shape"). We state that animations play an even more important role.

There are multiple ways to discriminate graphical symbols with animation as a technical variable. Symbols could be distinguished by applying different type of animations. We could differentiate two graphical symbols by animating one and not the other (***animation vs none-animation***, see Figure 67, (a)). In addition to this variance of basic animation, we could differentiate two graphical symbols that are animated by changing their **animation types** (see Figure 67, (b)), by modifying their **frequency** property both the *number of repetitions* and the *interval* between two animations (Figure 67, (c)) or by manipulating their **duration** time of animation (see Figure 67, (d)). Figure 67 illustrates the four properties of *Animation*. The movement is represented by the tail of degradation colour.

| (a) | (b) | (c) | (d) |
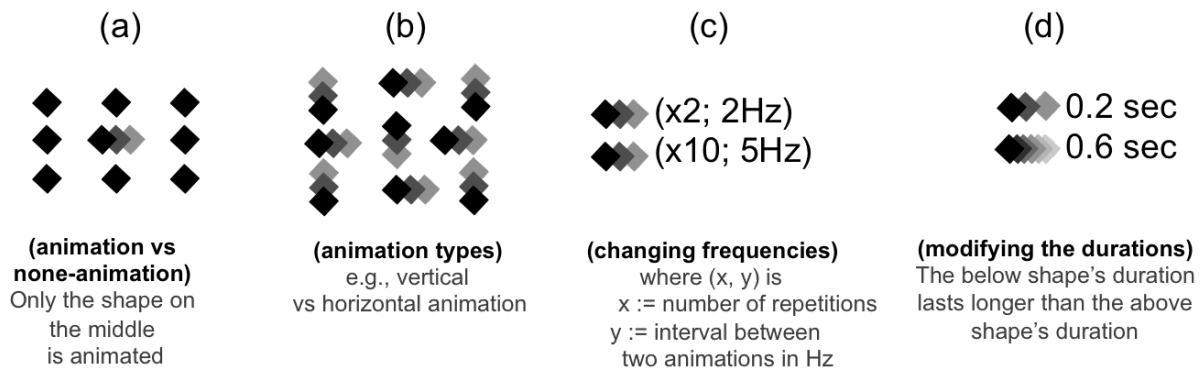
Figure 67   Differentiation between symbols by using one of the animation property

We could increase the discriminability of symbols by changing multiple animation properties (e.g., animation vs none-animation + changing frequencies). In the point of view of the discriminability principle, the more properties are different, the better.

Animations have to be supported by CASE tools, which could offer limitless features to support the Perceptual Discriminability. The difficulty is then more about finding a suitable animation for a specific situation.

7.1.5 Principle of Complexity Management

The diagrammatic complexity, as explained by Moody, is measured by the number of elements on a diagram. Moody gave notation-level features to reduce it. However, some CASE tools offer features to handle this kind of complexity, which is exactly the concern of this section.

Complexity is related to a group of graphical symbols and not to individual graphical symbols. For this reason, the animations as a modelling variable are more suited in this section.

Case tools features

Feature 1: The online maps give the ability to **zoom in** and **zoom out**. That process increase/decrease the data displayed on a map. The same idea can be applied to flat model (i.e., a modelling language that does not support vertical hierarchy) such as GRL. These

models do not have any modularization feature (i.e., vertical decomposition) in their notational level. Moreover, it is not appropriate to add such features in the notation-level for several reasons like standardization issues[6] or simply habits taken by the modellers. Therefore, we can add a feature that will automatically (i.e., in accordance with algorithms) add/remove artefacts to the diagram in according with a zoom in/zoom out button.

Feature 2: Books describe stories in a linear way, which improve the overall complexity because the reader has had the time to understand the different elements of the story. Furthermore, complexity has been added step by step. By contrast, diagrams show the final result, it is like showing the beginning to the end of a story all at once. Therefore, it seems to us that an ability to design and play *scenarios* for diagram comprehension is an interesting asset. In this way, a typical user will learn from simple to complex concepts at its own pace. Unlike books, CASE tools may handle both forward and rewind feature. The rewind feature is important to comprehend misunderstood steps of the scenario. Figure 68 represents a **step-by-step diagram visualization feature**. The user can visualize the different steps either in chronological (S1 » S2 » S3) or in non-chronological order (S3 » S2 » S1).
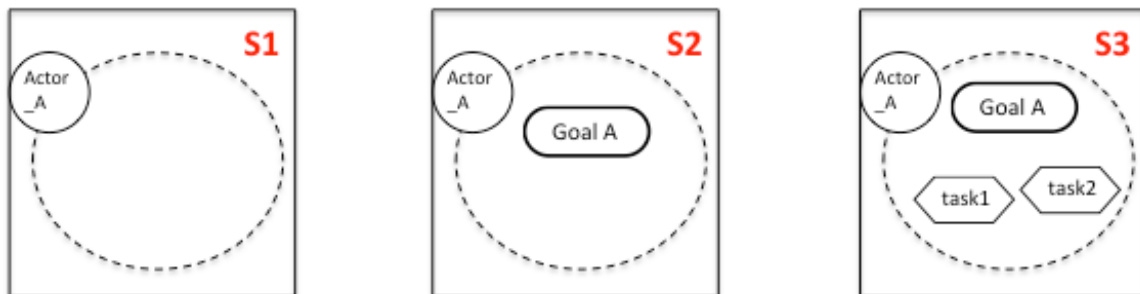
Figure 68   Step-by-Step Diagram Visualization.

---

[6] It seems very unlikely such modularization features can be added after the modelling language has been standardized. Indeed, such features alter how the modelling language is perceived.

7.1.6 Principle of Cognitive Integration

Among all the PoN principles, the Cognitive Integration is the one that will benefit more of CASE tools support.

In the conceptual integration, Moody stated that the contextualisation technique could be used in diagrammatic context. We think that such contextualisation should be used in CASE tools level because otherwise it will increase in the number of diagrams. Indeed, if each specific context has to be represented, it will lead to a significant increase in the number of diagrams. Figure 69 is an example given by Moody to represent contextualization. It is clear that such a feature is more *scalable* as a CASE tools feature than as a notation-level feature. A feature in a CASE tool will automatically show the context depending on the user's focus. If the focus process has to be done to all the systems of the Figure 69 (left), we will need eight focused diagrams (like the one on the right of Figure 69). Here, only one interactive diagram is necessary at the software level.
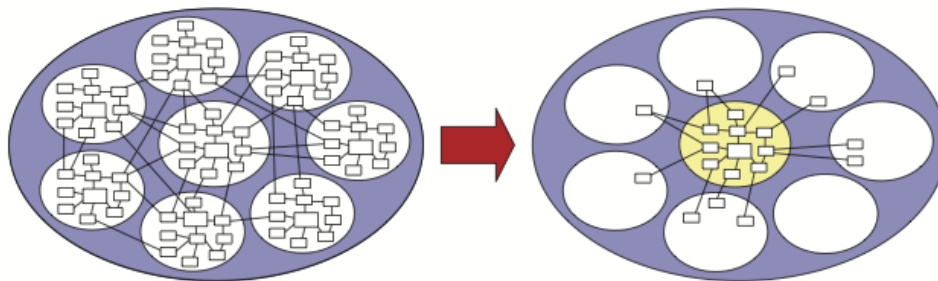
Figure 69   Initial Diagram with eight system (left)
Contextualization applied to one system (in yellow, right)
(from Moody in [25])

In addition to *contextualization*, the *perceptual integration* could also benefit from CASE tools features. All CASE tools have the ability to *identify diagrams*. CASE tools like jUCMNav have the *ability to navigate* through the hierarchical decomposition (e.g., jUCMNav supports navigation through UCM stubs). Finally, level numbering is an easy feature to develop in CASE tools.

Key feature: From my point of view, the most important feature for the Cognitive Integration is the navigational map. Most CASE tools offer a mini-map view of the current selected diagram. The idea of Moody is more ambitious; the navigational map will not only represent the current diagram but all of the diagram of the model. The links be-

tween diagrams should also be represented. We think that such features require the use of CASE tools. Nevertheless, the notational level could give a standard to represent a summary version of a diagram. Several kinds of links should be defined such as *hierarchy link*, which will allow navigation from a parent to a child, or the opposite. Besides the hierarchy, it would be useful to represent *subsystems links*, which will allow navigation between chunks of the system.

Moreover, users could define their own links depending on business related diagrams. A user-defined link could also be useful when defining a *scenario* for learning purposes (see feature 2 of Complexity Management). One of the steps in the scenario can be a link that will automatically open another diagram at a defined step. Figure 70 shows an example on how a *goTo link* may be used to pass from the GRL diagram to the step 128 of the UCM diagram (right). The reverse way can also be modelled, i.e., from the UCM diagram to the GRL diagram. The *goTo link* is activated either by the user (to go to the next or previous step of the **step-by-step visualization feature** of Complexity Management) or by the CASE tool itself.
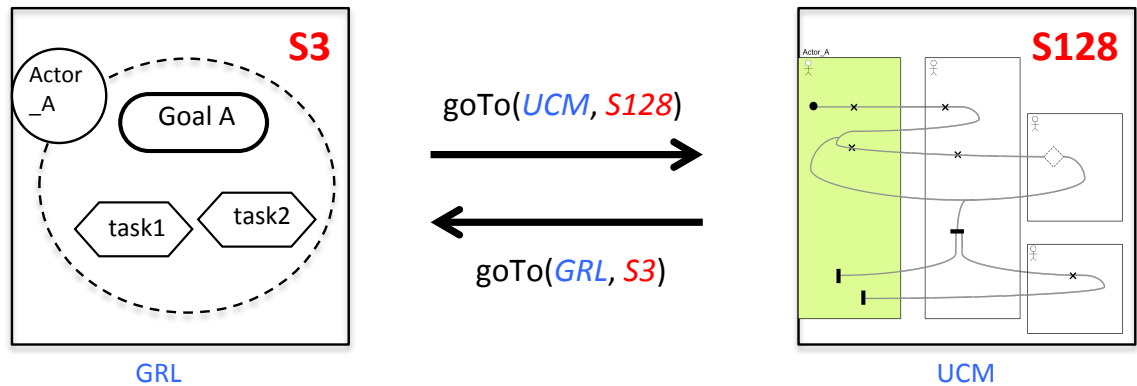
Figure 70   Navigation from a diagram to another using the user-defined *goTo link*.

7.2. Towards an Automatization of the Evaluation of Visual Notations

The PoN Assessment Software is composed of two major components: the Java software and the specifications of spreadsheet files.

The Figure 71 represents the main components of the software. Firstly, we explain the scope and the interactions among the sub-components of the Java program (i.e., application) and then we will define the different spreadsheet files.
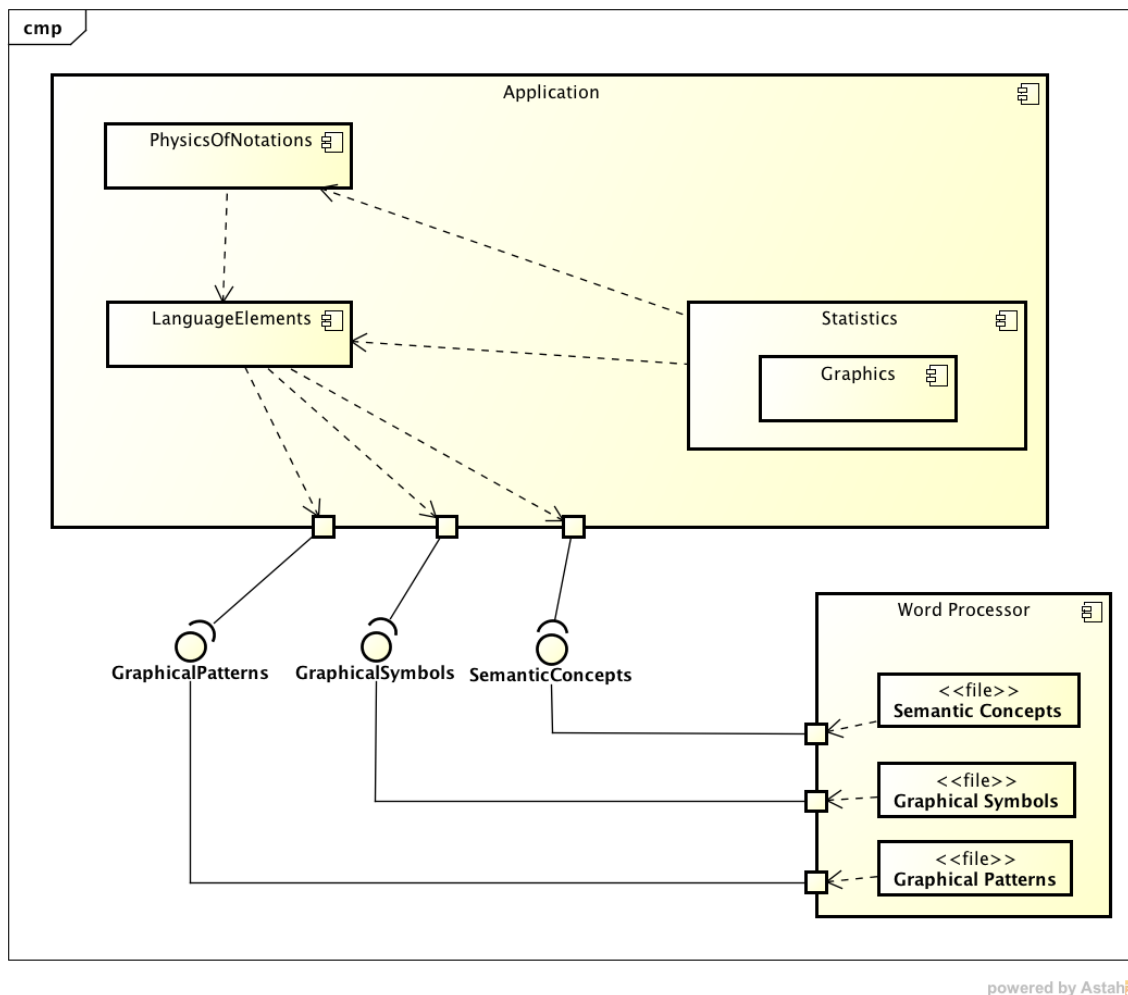
Figure 71   UML Component Diagram of the PoN Assessment Software

7.2.1 Java Program

The program is composed of 3 components: `LanguageElements`, `PhysicsOfNotations` and `Statistics`. The `LanguageElements` component has the responsibility to read and gather information from the spreadsheet files. The information included these spreadsheets such as the symbol pictures or graphical symbol values is casted into data classes (e.g., `Graphem, SetConcepts`) that are used across the program. Moreover, the program enables model checking of the data contained in the spreadsheet files. Basically, it alerts the user about wrong values present in the spreadsheet files such as values that are not included in a certain range. Technically, we use Apache POI (i.e., API developed by the Apache Software Foundation) to manipulate the Microsoft Excel files with our Java program. The `PhysicsOfNotations` component contains the formulae to compute the Semiotic Clarity and Perceptual Discriminability. In order to process the visual variable shape, we chose use the pattern-based matching technique developed in Section 5.3.1. The *Statistics* component is responsible for generating a PDF report that contains the list of semantic constructs, the graphical symbols, their binding and the metrics. The report is generated by using the iText API, an open source library that allows the creation and the manipulation of PDF documents. The *Graphics* sub-component is responsible for generating graphics of the statistics by using the jFreeChart API, a Java library to create graphic charts.

7.2.2 Specifications of Spreadsheet Files

There are three spreadsheet files: Semantic Concepts file, Graphical Symbols file and the Graphical Patterns file. These files are the main data that are required by the Java application.

Semantic Concepts File

Let us take a line i of the spreadsheet.

	A	B	C	D	E
	Ai	Bi	Ci	Di	Ei

The different values must respect these specifications:

- Ai contains the identifier of the semantic concept. (mandatory)
- Bi contains a list of identifiers separated by a comma. These identifiers are the graphical symbols associated to Ai. (optional[7])
- Ci contains a definition or the name of the semantic concept (optional)
- Di contains the page reference or the figure where the semantic concept has been taken such as from a certain page of a specification document. (optional)
- Ei contains a group tag of assessment allowing more specific assessments depending on the group tag. (optional)

The Ei tag allows us to define groups separated by commas, to which a semantic concept belongs. The assessment could then target a certain group of semantic concepts. This technique allows us to see for instance the evolution of a modelling language against the PoN by defining different tags for the newer semantic concepts.

---

[7] If there is no value, the cell must contain the slash character (/)

Graphical Symbols File

The values of the line *i* must respect these specifications:

- A*i* contains the identifier of the graphical symbol. (mandatory)
- B*i* contains a list of identifiers separated by a comma. Each of these identifiers are semantic constructs which correspond to the graphical symbol A*i*. (optional)
- C*i* contains an image of the graphical symbol. (optional)
- D*i* contains a reference to a pattern. (optional)
- E*i*, F*i*, G*i* are the shape categories which have been defined by Störrle and Fish. (Line, Icon, Shape) (optional)
- H*i* is the texture applied to the border of a graphical symbol (e.g., dashed). (optional)
- I*i* defines whether the graphical symbol is filled with a colour. (optional)
- J*i* contains the text that is written near the figure. It allows us to capture the textual differentiation defined by Störrle and Fish. (optional)
- K*i* contains the size of the graphical symbol (XS, S, M, L). (mandatory)
- L*i* contains a group tag of assessment allowing more specific assessments depending on the group tag. (optional)

Patterns File

The values of the line *i* must respect these specifications:

- A*i* the identifier of the pattern. (mandatory)
- B*i* the image of the pattern. (optional)

Chapter 8.   Conclusion

The Physics of Notations (PoN) is a theory meant to design and assess visual notations. While the theoretical foundations of the theory have been quite largely discussed in the literature, only few research exists on its operationalization. In this thesis, we undertake that task. More precisely, we study the falsifiability of the PoN to determine the level of operationalization that the theory should be able to provide. The analysis reveals that most of the principles of the theory belong to the group of Type IV theory: theories for explaining and predicting. To be operationalized, a theory needs to reach Type V. Based on these observations, we decided to focus our work on Semiotic Clarity and Perceptual Discriminability, two of the nine principles that compose the PoN.

We based our work on previous research where the PoN was once applied in an ad-hoc (and informal) manner and another one where the authors turned the informal metrics into mathematical formulae. We noticed that the mathematical formulae allow automatic computation but did not provide results significantly better than the informal analysis. Regarding Semiotic Clarity, we conclude that the metrics obtained with mathematical formulae are either mathematically equivalent or not significantly different. The analysis of Perceptual Discriminability does not benefit from mathematical formulae due to the lack of empirical evaluation. For these two principles, we introduced new metrics that provide more accurate results when applied to a larger set of notations.

To evaluate our proposal, we analyze the cognitive effectiveness of the Aspect-oriented User Requirements Notation (AoURN) according to the PoN by using the three approaches: the informal manner, the original mathematical formulae and the improved ones. AoURN extends ITU-T's User Requirements Notation (URN) to support encapsulation of cross-cutting concerns in requirements models. Most of the issues found in AoURN are related to the complexity brought by Aspect-oriented Modelling. Therefore, special attention has been drawn to the principles of Complexity Management and Cognitive Effectiveness. We propose improvements at the notational level by providing a set of

cognitively more efficient symbol for AoURN. We showed that our improved metrics provide more accurate results than the current operationalization of the theory. We also discuss how CASE tools could integrate principles of the PoN.

We obtained a more surprising observation: the improved formulae provide results that are almost identical to those obtained in an informal way. This observation leads us to the conclusion that regardless of how the PoN is applied, the greatest interest in its application relies in the interpretation of the results and the improvements that can be envisioned. One of the major challenges that still has to be addressed relates to a fine-grained understanding of the interactions between the principles of the PoN. It would in order to prioritize depending on the context, task and audience. Research on this aspect should definitely be undertaken.

References

[1]   Amyot, D. and Mussbacher, G.: "User Requirements Notation: The First Ten Years, The Next Ten Years". Invited paper, *Journal of Software (JSW)*, Academy Publisher, Vol. 6, No. 5, pp. 747–768 (May 2011)

[2]   Amyot, D., Mussbacher, G., and Mansurov, N.: "Understanding Existing Software with Use Case Map Scenarios". *3rd SDL and MSC Workshop (SAM02)*, Aberystwyth, Wales, UK, Springer, LNCS 2599, pp. 124–140 (June 2002)

[3]   Amyot, D.: "Introduction to the User Requirements Notation: Learning by Example". *Computer Networks*, Elsevier, Vol. 42(3), pp. 285–301 (21 June 2003)

[4]   Belongie, S., Malik, J. Puzicha, J.: "Shape matching and object recognition using shape contexts." *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* 24(4), pp. 509–522

[5]   Bertin, J.: "Semiology of Graphics: Diagrams, Networks, Maps" Madison, Wisconsin, USA: University of Wisconsin Press (1983)

[6]   Bookstein, F. L.: "Principal Warps: Thin-Plate-Splines and Decomposition of Deformations" IEEE Trans. Pattern Anal. Mach. Intell., 11(6):567–585 (1989)

[7]   Capendale, M. S. T.: "Considering Visual Variables as a Basis for Information Visualisation" University of Calgary, Department of Computer Science. 2001-693 vol. 16 (2003)

[8]   Dijkstra, E. W.: "On the role of scientific thought" In: *Selected writings on Computing: A Personal Perspective*. New York, NY, USA: Springer-Verlag. Pp. 60-66. ISBN 0-387-90652-5 (1982)

[9]   E.F. Codd.: "Relational database: a practical foundation for productivity" In: Communications of the ACM 25,2, pp. 109–117 (February 1982)

[10]   Genon, N., Amyot, D., Heymans, P.: "Analysing the Cognitive Effectiveness of the BPMN 2.0 Visual Notation". In: Malloy, B. Staab, S., van den Brand, M. (eds.) SLE 2010. LNCS, vol. 6563, pp. 377–394. Springer, Heidelberg (2011)

[11]   Genon, N., Amyot, D., Heymans, P.: "Analysing the Cognitive Effectiveness of the UCM Visual Notation." Presentation Document SAM (2010)

[12]   Genon, N., Amyot, D., Heymans, P.: "Applying PoN to URN – UCM (draft)" Technical Report. University of Namur (2010)

[13]   Genon, N., Amyot, D., Heymans, P.: "Analysing the Cognitive Effectiveness of the UCM Visual Notation". In: Kraemer, F.A., Herrmann, P. (eds.) SAM 2010. LNCS, vol. 6598, pp. 221–240. Springer, Heidelberg (2011)

[14]   Goodman, N.: "Languages of Art: An approach to a Theory of Symbols". Indianapolis: Bobbs-Merrill Co. (1968)

[15]   Green, T., Blandford, A., Church, L., Roast, C. Clarke, S.: "Cognitive Dimensions: Achievements, New Directions, and Open Questions. Journal of Visual Languages and Computing 17, 328–365 (2006)

[16]   Green, T.R.G.: "Cognitive Dimensions of Notations." In A. Sutcliffe and L. Macaulay (Eds.) *People and Computers* V. Cambridge, UK: Cambridge University Press, pp. 443–460 (1989)

[17]   Gregor, S.: "The Nature of Theory in Information Systems." In: *MIS Quarterly* 30(3), 611-642 (2006)

[18]   Hahn, J., Kim, J.: "Why are Some Diagrams Easier to Work With? : Effects of Diagrammatic Representation on the Cognitive Integration Process of Systems Analysis and Design." In: ACM Transactions on Computer-Human Interaction. 6(3). pp. 181–213 (1999)

[19]   ITU-T, Recommendation Z.151 (10/12) *User Requirements Notation (URN) – Language definition*, Geneva, Switzerland (2012)

[20]   *jUCMNav*, version 5.5.0, University of Ottawa. Canada, http://softwareengineering.ca/jucmnav (accessed August 2014)

[21]   Kim, J., Hahn, J. Hahn, H.: "How Do We Understand a System with (So) Many Diagrams? Cognitive Integration Processes in Diagrammatic Reasoning." Information Systems Research. 11(3), pp. 284–303 (2000)

[22]   Ling, H., Jacobs, D. W.: "Shape classification using the inner-distance" In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* 29(2), 286-299 (2007)

[23]   Moody, D. L., Heymans, P., Matulevicius, R.,: "Visual syntax does matter: improving the cognitive effectiveness of the i* visual notation." *Requirements Engineering* 15(2) pp. 141–175 (2010)

[24]   Moody, D., van Hillegersberg, J.: "Evaluating the Visual Syntax of UML: An Analysis of the Cognitive Effectiveness of the UML Family of Diagrams." In: Gašević, D., Lämmel, R., Van Wyk, E. (eds.) SLE 2008. LNCS, vol. 5452, pp. 16–34. Springer, Heidelberg (2009)

[25]   Moody, D.L.: "The Physics of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering." *IEEE Trans. Software Engineering* 35(6), pp. 756–779 (2009)

[26]   Mussbacher, G. and Amyot, D. (2008) "Assessing the Applicability of Use Case Maps for Business Process and Workflow Description." In: *3rd Int. MCeTech Conference on eTechnologies*, Montréal, Canada, January 2008. IEEE Computer Society, 219-222.

[27]   Mussbacher. G.: "Aspect-oriented User Requirements Notation." Ph.D. thesis, SITE, University of Ottawa, Canada (November 2010)

[28]   Opdahl, A.L., Henderson-Sellers, B.: "Ontological Evaluation of the UML Using the Bunge-Wand-Weber Model" In: Software and Systems Modelling. 1(1). Pp.43-67. (2002)

[29]   Paivio, A.: "Mental Representations: A Dual Coding Approach." Oxford, England: Oxford University Press. (1986)

[30]   Popper, K. The logic of Scientific Discovery, Unwin Hyman, London (1980)

[31]   Popper, K.R.: "Science as Falsification." In: Routledge, Keagan, P. (eds.) Conjectures and Refutations, London, pp. 30–39 (1963)

[32]   Rebah, B., Zanin, C.: "Rethinking dynamic visual variables: towards a framework of dynamic semiology." *GeoViz: Linking Geovisualization with Spatial Analysis and Modeling*, pp. 10-11. (March 2011)

[33]   Shannon, C.E., Weaver, W.: "The Mathematical Theory of Communication." In: Illini Books ed. Illini books. University of Illinois Press. (1963)

[34]   Shannon, C.E.: "A Mathematical Theory of Communication.", Bell System Technical Journal, 27, pp. 379–423 & pp. 623–656 (1948)

[35]   Siau, K., Cao, Q.: "Unified Modeling Language: A Complexity Analysis" In: Journal of Database Management. 12(1) pp. 26-34. (2001)

[36]   Spence, I.: "Visual psychophysics of simple graphical elements." Journal of Experimental Psychology: Human Perception and Performance. 16(4) pp. 683–692 (1990)

[37]   Stanford Encyclopedia of Philosophy, University of Stanford. USA, http://plato.stanford.edu/entries/hermeneutics (accessed August 2014)

[38]   Stevens, S.S.: "Psychophysics." New York, USA: John Wiley & Sons. (1975)

[39]   Störrle, H., Fish, A.: "Towards an Operationalization of the Physics of Notations for the Analysis of Visual Languages." In: Model-Driven Engineering Languages and Systems, pp. 104–120 (October 2013)

[40]   Wheildon, C.: "Type and Layout: Are You Communicating or Just Making Pretty Shapes?" Hastings, Victoria, Australia: Worsley Press (2005)

Appendix A: Shape Comparison

In this appendix, we illustrate how two shapes can be compared to one another by using the shape context defined by Belongie *et al.* [4].

Let us assume that we want to compare the two shapes (a) and (b) of the Figure 1. The edge elements of these shapes must be converted to a set of N feature points. These points are obtained by an algorithm that transforms a shape into a set of points (see Figure 2). We should now consider a set of vectors originating from a feature point to the N-1 remaining other feature points. For instance, a set of vectors could be the vectors that took their origin from a feature point (⊙). Obviously, this set of vectors is a rich descriptor of the shape according to the feature point (⊙) because the description depends on the configuration of the other points compared to the feature point. Therefore, we obtain a precise description for each feature point. However, the set of vectors could not be used directly since shapes and their sampled representation vary from one instance to another.

Instead of considering the set of vectors, we will consider the distribution over relative positions (i.e., points which belong to defined areas). Belongie *et al.* used a *log-polar coordinate system*, i.e., a 2-D coordinate system where each point is described by a distance and the angle formed from a certain point (e.g., the origin, a feature point). This coordinate system is more sensitive to the nearby points than the more distant ones relative to a feature point (e.g., the feature point (⊙)). They split a two-dimensional surface around a feature point in 12 equally spaced angle bins and five equally spaced log-radius bins. Figure 3 shows the diagram of log-polar histogram bins used to compute the shape contexts. The shape context is now represented by a 12 x 5 matrix M. Each element of M represents the number of points (represented by a shade of grey, dark=large value of points), which lies on an area, formed by the angle and distance bin. The 0 shows graphically how the points are linked to M. The matrix M represents the *shape context* of the feature point (⊙).
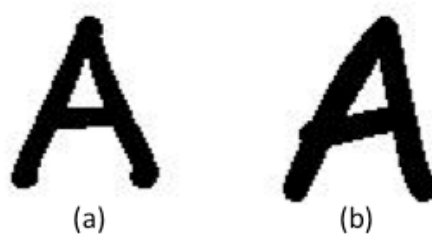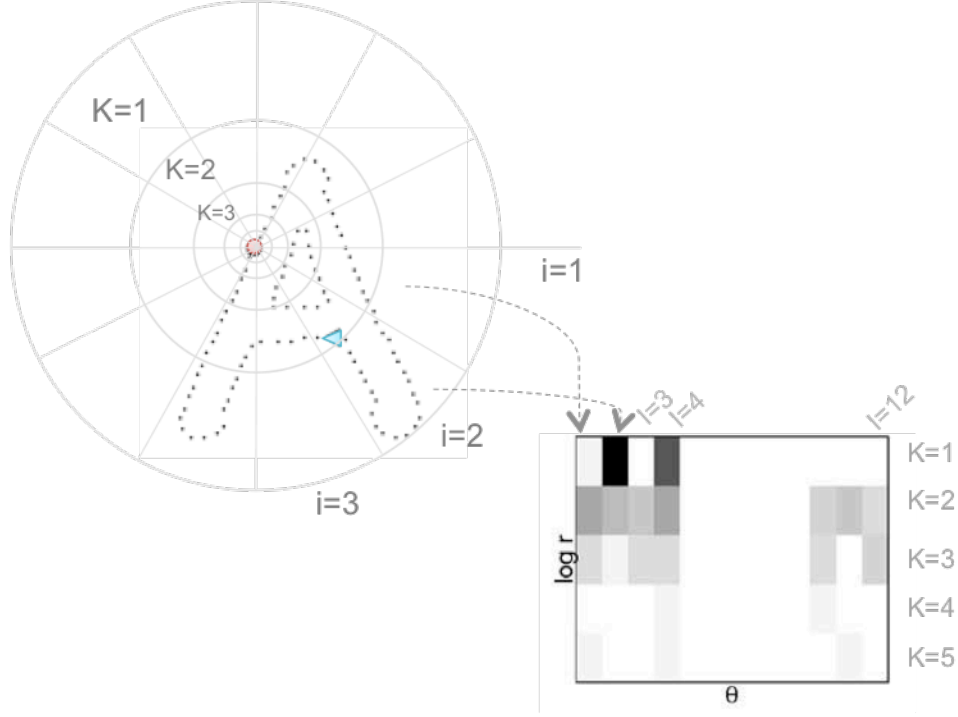
Figure 1   Original shapes
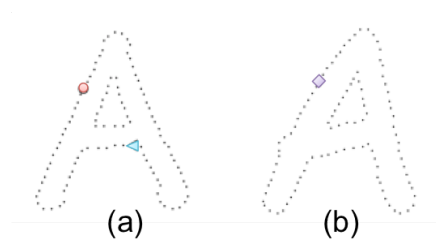
Figure 2   Sampled edge points of shape (a) and shape (b) of Figure 1. Notice the feature points (●), (◁) and (◇).
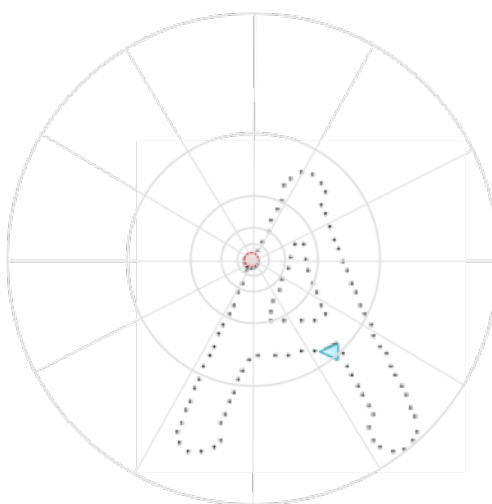

Figure 3   Diagram of log-polar for the feature point (●).

Figure 4   Building the shape context histogram (e.g., the two-dimensional matrix) of the feature point (⬤). The more points lies on an area $< i, k >$ the darker $M(i, k)$ becomes.

Let us assume that we also computed the shape context of the feature points (◁) and (◇). The shape contexts of the Figure 5 (a) and (c) look very similar to each other. Mathematically, the distance between two shape context histograms is measured with a Chi-squared test.
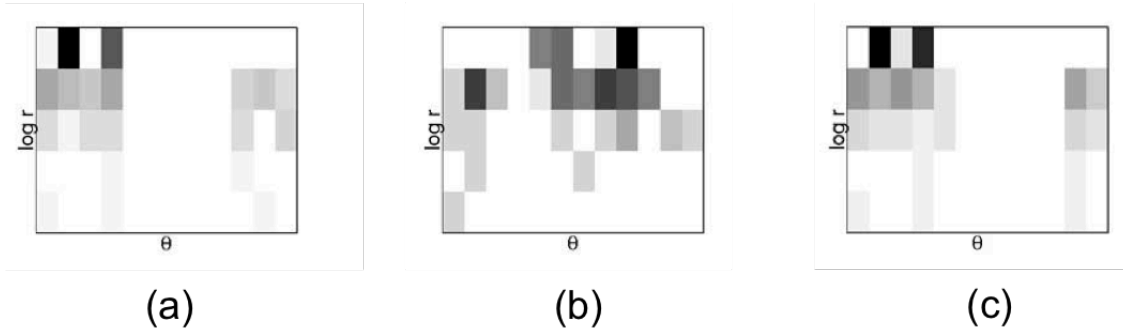
Figure 5   Shape context of the feature points (⬤) = (a) , (◁) = (b) and (◇) = (c).

For every match like (a) and (c) (Figure 5), Belongie *et al.* estimated the transformation between them using iteratively the Thin Plate Sline (TPS) technique [6].

Figure 6 represents the transformation between the two shapes.
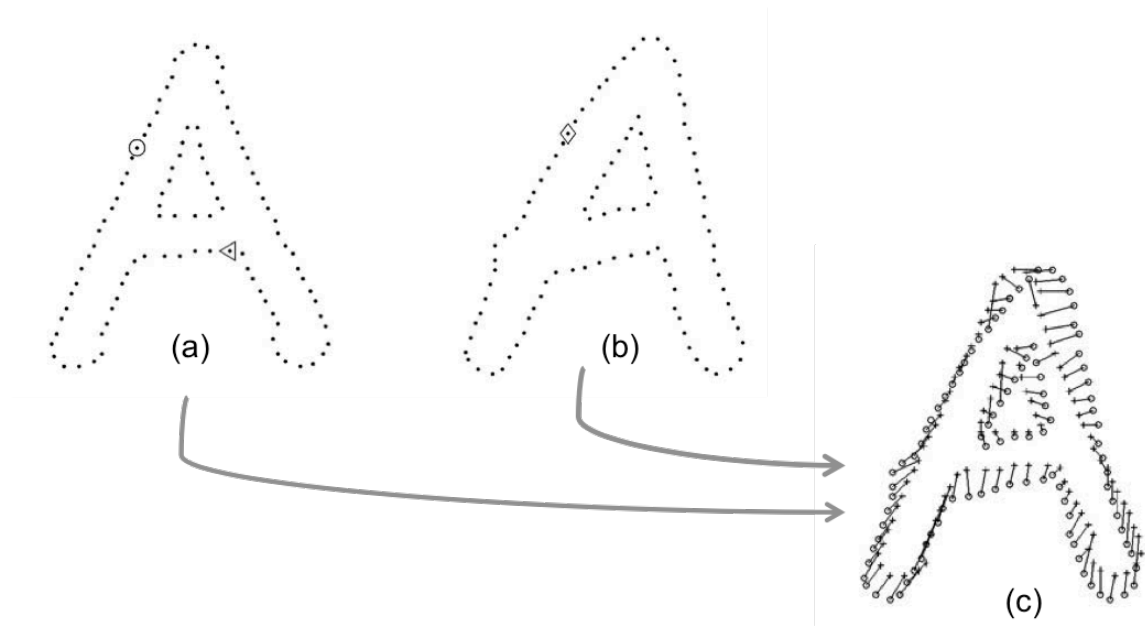
Figure 6   Transformation between shapes

Appendix B: Gathering of Semantic Constructs and Graphical Symbols of URN and AoURN

In this appendix, we present the metaclasses (either the semantic constructs or the graphical symbols) of AoURN according to the six categories defined in Table 3. The URN elements, AoGRL and AoUCM are coloured in back, blue and orange, respectively.

To Consider	
1. Z151.F4 UCM_Condition	33. Z151.F10 GRL_Actor - F31 ActorRef
2. Z151.F60 UCM_NodeConnection	34. Z151.F10 GRL_IntentionalElement – F35 IntentionalElementRef
3. Z151.F60 UCM_Responsibility	35. Z151.F10 GRL_Contribution
4. Z151.F60 UCM_RespRef	36. Z151.F10 GRL_Decomposition
5. Z151.F60 UCM_StartPoint	37. Z151.F10 GRL_Dependency
6. Z151.F60 UCM_EndPoint	38. Z151.F24 GRL_EvaluationStrategy
7. Z151.F60 UCM_OrFork	39. Z151.F24 GRL_Evaluation
8. Z151.F60 UCM_OrJoin	40. Z151.F26 GRL_Indicator
9. Z151.F60 UCM_AndFork	41. Z151.F26 GRL_IndicatorEvaluation
10. Z151.F60 UCM_AndJoin	42. Z151.F26 GRL_LinearConversion
11. Z151.F60 UCM_EmptyPoint	43. Z151.F26 GRL_QualToQMapping
12. Z151.F60 UCM_WaitingPlace	44. Z151.F29 GRL_ContributionContext
13. Z151.F60 UCM_Timer	45. Z151.F29 GRL_ContributionChange
14. Z151.F60 UCM_FailurePoint	46. Z151.F30 GRL_CollapsedActorRef
15. Z151.F60 UCM_Connect	47. AoGRL_Pointcut Graph
16. Z151.F60 UCM_Stub	48. AoGRL_Pointcut Marker
17. Z151.F77 UCM_PluginBinding	49. AoGRL_Pointcut Deletion Marker
18. Z151.F77 UCM_InBinding	50. AoGRL_Composition Rule
19. Z151.F77 UCM_OutBinding	51. AoGRL_Aspect Marker
20. Z151.F85 UCM_ComponentRef	52. AoGRL_AoView
21. Z151.F85 UCM_Component	53. AoGRL_Anytype Pointcut
22. Z151.F85 UCM_ComponentType	54. AoUCM_Conditional Marker
23. Z151.F85 UCM_ComponentBinding	55. AoUCM_Tunnel Aspect Marker
24. Z151.F94 UCM_OWPoisson	56. AoUCM_Concern Interaction Graph
25. Z151.F94 UCM_OWPeriodic	57. AoUCM_Conflict
26. Z151.F94 OWUniform	58. AoUCM_Aspect Map
27. Z151.F94 UCM_OWPhaseType	59. AoUCM_Pointcut Stub
28. Z151.F94 UCM_ClosedWorkload	60. AoUCM_Replacement Pointcut Stub
29. Z151.F94 UCM_PassiveRessource	61. AoUCM_Pointcut Map
30. Z151.F94 UCM_ProcessingRessource	62. AoUCM_Local Start Point
31. Z151.F94 UCM_ExternalOperation	63. AoUCM_Local End Point
32. Z151.F94 UCM_Demand	64. AoUCM_AoView
	65. AoUCM_Pointcut Variable
	66. AoUCM_Anything Pointcut

Figure 7 Elements of URN and AoURN that lies in the *To consider* set (see Table 3)

Abstract	Structural
1. Z151.F3 UCMmodelElement	1. Z151.F3 URNspec
2. Z151.F60 PathNode	2. Z151.F3 UCMspec
3. Z151.F94 Workload	3. Z151.F3 URNlink
4. Z151.F94 OpenWorkload	4. Z151.F3 URNmodelElement
5. Z151.F94 GeneralRessource	5. Z151.F3 Concern
6. Z151.F94 ActiveRessource	6. Z151.F3 Metadata
7. Z151.F3 GRLmodelElement	7. Z151.F60 UCMmap
8. Z151.F9 ElementLink	8. Z151.F3 GRLSpec
9. Z151.F29 ContributionContext	9. Z151.F9 GRLLinkableElement
10. Z151.F30 GRLNode	10. Z151.F9 GRLContainableElement
11. Z151.F26 IndicatorConversion	11. Z151.F9 GRLmodelElement
	12. Z151.F24 StrategiesGroup
	13. Z151.F26 QualToQMappings
	14. Z151.F29 ContributionContextGroup

Figure 8   Elements of URN and AoURN that lies into either the *Abstract* or the *Structural* set

Collection	Graphical
1. Z151.F60 FailureKind	1. Z151.F5 ConcreteURNspec
2. Z151.F60 WaitKind	2. Z151.F6 Description
3. Z151.F85 ComponentKind	3. Z151.F7 ConcreteCondition
4. Z151.F94 TimeUnit	4. Z151.F95 DirectionArrow
5. Z151.F94 DeviceKind	5. Z151.I.10 Label
6. Z151.F9 ImportanceType	6. Z151.I.10 Position
7. Z151.F10 IntentionalElementType	7. Z151.I.10 Size
8. Z151.F10 ContributionType	8. Z151.I.10 Comment
9. Z151.F10 DecompositionType	9. Z151.I.10 ConcreteStyle
10. Z151.F24 QualitativeLabel	10. Z151.F30 ConcreteGRLspec
11. AoUCM_F73 UCM AspectKind	11. Z151.F30 GRLGraph
12. AoUCM_F73 (Ao) UCM PointcutKind	12. Z151.F52 Label
	13. Z151.F30 LinkRefBendPoint
	14. Z151.F53 Position
	15. Z151.F54 Size
	16. Z151.F55 ConcreteStyle

Figure 9   Elements of URN and AoURN that lies into either the *Collection* or the *Graphical* set

Out of Scope
1. Z151.F3 GRLspec
2. Z151.F3 GRLmodelElement
3. Z151.F92 ScenarioDef
4. Z151.F92 ScenarioGroup
5. Z151.F92 Initialization
6. Z151.F92 Variable
7. Z151.F92 EnumerationType
8. Z151.F92 DatatypeKind

Figure 10   Elements of URN that are not considered.