

Design of Visual Language Syntax for Robot Programming Domain

Ignas Plauska and Robertas Damaševičius

¹ Kaunas University of Technology, Centre of Real Time Computer Systems,
Studentų 50, LT-51368, Kaunas, Lithuania

² Kaunas University of Technology, Software Engineering Department,
Studentų 50-415, LT-51368, Kaunas, Lithuania
ignas.plauska@ktu.lt, robertas.damasevicius@ktu.lt

Abstract. The paper discusses the development of the visual language syntax based on the application of sound methodological principles, a visual communication model, a visual syntax model, a formal description of syntax based on visual grammar metalanguage (an extension of BNF) and ontology of visual signs (graphemes). The syntax of an illustrative visual language VisuRobo for the mobile robot programming domain is presented.

Keywords: visual programming, visual language, visual communication model, visual syntax model, visual metalanguage.

1 Introduction

Software design and development involves high-level *cognitive processes* such as assimilating, constructing and sharing domain knowledge and making decisions. Cognition is based on the developer's *mental models* [1], which provide a structure for organization of domain knowledge within the developer's mind. The task of the programmer is to map a mental model of a solution of a domain-specific problem to a system of computer-readable signs, i.e. a program written in a programming language. Many types of programming languages exist. General purpose programming languages are usually domain-independent and are good for general problem solving. On the other hand, domain-specific programming languages (DSLs) are tailored towards a specific application domain, and are based only on the relevant concepts and features of that domain. A DSL allows domain experts to express high-level concepts succinctly using a notation tailored to a set of specific domain problems. Therefore, DSLs can be considered as a medium of communication that allows to bridge the gap between the mental model and the problem domain systems and, consequently, to cut the distance from ideas to products in software engineering.

The complexity of modern software engineering and its application domains such as robotics stimulated the move from one-dimensional string grammar based textual languages to visual languages which use non-linear graphical notations. The problem transcends just using graphical symbols for programming and includes *visual*

knowledge engineering [2] and *visual software engineering* [3]. Currently, visual programming languages (VPLs) are used in many different application fields such as the development of graphical user interfaces (GUIs), teaching computer science, and model-driven development [4]. In particular, Unified Modeling Language (UML) [5] is a widely known example of a visual software engineering language that is used for modelling and specifying software-intensive systems.

The main motivations for visual languages are as follows: 1) higher level of abstraction, which is closer to the user's mental model and involves manipulation of visual elements rather than machine instructions [6], 2) higher expressive power characterized by two (or more) dimensional relations between visual elements [7], and 3) higher attractiveness to non-professional or novice programmers motivated by simpler description of complex things. Other advantages of VPLs include economy of concepts required to program (i.e., smaller program size), concreteness of programming process, explicit depiction of relationships between program entities, and immediate visual feedback [8].

Visual programming allows using a conceptual model that is tailored to the mental process of the user rather than constraints of the programming language syntax or the target platform. Indeed, visual notations allow for the description and understanding of complex systems, such as concurrent and/or real-time systems, for which traditional textual descriptions are inadequate [9]. Robots are good examples of such complex systems which require having knowledge of multiple domains such as mechatronics, analogue and digital electronics, embedded software, real-time systems, kinematics, communication protocols and control algorithms, etc. The robotics domain is characterized by heterogeneity and diversity of robots and their different capabilities. Furthermore, a large variety of existing robotic platforms requires having high-level methods for general modelling and reasoning about what different robots are capable of doing. The examples of VPLs designed for the robot programming domain, are Microsoft Visual Programming Language (MVPL) and Lego NXT-G language. These languages have been criticized for lack of flexibility and usability [10]. Other challenges to these languages come from targeting users that are usually not formally trained in programming or software engineering (e.g., children, or robot hobbyists), and include unusual features that do not seem to be shared by a majority of users.

The contribution of this paper is the design of the visual language syntax for the robot programming domain based on the application of sound methodological principles including a model of visual communication, a formal description of syntax based on using visual metalanguage and an ontology of signs.

The structure of the remaining parts of the paper is as follows. Section 2 presents a Visual Communication Model as a foundation of the proposed visual programming language syntax. Section 3 describes the modelling of visual language grammar using visual meta-syntax. Section 4 provides a case study in developing syntax for the *VisuRobo* language. Finally, Section 5 presents evaluation of results and conclusions.

2 Elements and Models of Visual Communication

Using a VPL to develop a visual program (diagram) is a form of visual communication (conversation or dialogue) between a designer and a user. In general, any language is a formal system of signs described by a set of grammatical rules to communicate some meaning. In particular, a VPL is a system of *visual* signs (i.e., primitive graphical elements, graphemes or pictograms), which represent *domain-specific* concepts, processes, or physical entities, and uses *more than one dimension* of space to convey semantics [8]. Spatial arrangement of graphical elements together with a semantic interpretation provides a space of communication for users of a VPL [11]. Below we present an analysis of known models of visual communication.

2.1 Shannon-Weaver's and Berlo's Models of Communication

A model of communication proposed by Shannon and Weaver [12] consists of a sender (a source), a message, a code (a language or other set of symbols or signs that can be used to transmit a message), a channel (the path on which the message travels), noise (or interference), and a receiver. The sender selects a message and encodes it into a signal that is sent over the communication channel to the receiver. The receiver decodes the transmitted signal and interprets the message. In the process, the message may be corrupted by noise that can be psychological (arising from the receiver's attitudes, biases or beliefs), physical (coming from the environment of communication) or semantic (caused by the receiver's misinterpretation). The model has been criticized for disregarding the semantic content of message and the simplistic interpretation of concept of information. Furthermore, it implies that human communication is similar to machine communication such as sending a signal in computer systems [13].

Berlo's Model of Communication [14] extended the Shannon and Weaver's model with perceptory (hearing, seeing, touching, etc. channels), structural (content, elements, structure of message) and social (skills, culture, attitudes) elements of communication, but provided no technical details how the model could be implemented.

2.2 Semiotic Engineering

Semiotic engineering [15] interprets the communication channel as a human-computer interface (HCI). Designers (senders) send their message to users (receivers) through the interface using signs, which associate domain entities with their meaning and representation. The interface acts as a meta-message from a designer to users. This meta-message conveys the designers' interpretation of the domain problem and provides users with artefacts to support users in solving the problem. The meta-message is defined using a metalanguage, and a metalanguage consists of signs that signify relationships of interface elements to each other and to domain entities they represent. The difficulties with semiotic engineering arise from the lack of practical approaches how to deal with the interpretability problem: the designer aims to formulate a concise singular message (that can be interpreted only one way) while the users may derive multiple and often conflicting interpretations of the message.

2.3 Visual Language Model

A visual language model proposed by Hari Narayan *et al.* [16] focused on the human use of visual languages and described three objects of interest to investigation of such languages: a computational system, a cognitive system, and the visual display as a communication medium (or channel). Visual representations that encode and convey information appear on the visual display and require visual perception, comprehension and reasoning on the cognitive side, as well as visual parsing, interpretation, and program execution on the computational side. The authors themselves recognize that the model is not full as a more complete taxonomy complemented with a formal system to define semantics is required.

2.4 Theory of Visual Display and Visual Variables

Theory of Visual Display [17] elaborated on the structure of visual display as the communication channel. The main elements are: 1) objects – basic units of meaning, 2) regions – provide context for objects, and 3) relations – connect objects or regions.

Bertin [18] analysed the main elements of visual objects. These could be encoded graphically using 8 visual variables which provide a visual alphabet for constructing visual notations as follows: shape, size, orientation, pattern, colour, hue and two spatial dimensions (vertical and horizontal). Notation designers can create graphical symbols by combining the variables together in different ways.

2.5 Physics of Notations

Moody [19] proposed a framework of methodology for developing cognitively-effective visual languages. The framework defines a visual notation that consists of a set of graphical symbols (visual vocabulary), a set of compositional rules (visual grammar), and definitions of the meaning of each symbol (visual semantics). Visual vocabulary includes 1D graphic elements (lines), 2D graphic elements (areas), 3D graphic elements (volumes), textual elements (labels) and spatial relationships. A valid expression (diagram) in a visual notation consists of visual symbols (tokens) arranged according to the rules of the visual grammar. Visual vocabulary and visual grammar together form the visual syntax of the notation. The language metamodel defines mapping of visual symbols to the constructs they represent. The approach has been defined as a scientific theory that allows both understanding how and why visual notations communicate as well as improve their ability to communicate [19].

2.6 Ontological Engineering

Ontological engineering provides methodologies for building ontologies: formal representations of concepts within a domain and the relationships between concepts. Ontology defines the vocabulary of a problem domain and a set of constraints on how terms can be combined to model the domain in a declarative way [20]. The language

ontology is the description of what the primitives of a language are able to represent in terms of domain phenomena, i.e., it is the representation of a conceptualization of the domain in terms of the language’s vocabulary [21].

2.7 Proposed Model of Visual Communication

Based on the discussed models of visual communication and visual modelling, we propose our Model of Visual Communication (Fig. 1). The model has sound formal and engineering foundations. It defines the composition of language’s syntax and ensures “low noise” communication due to using shared domain and sign ontologies.

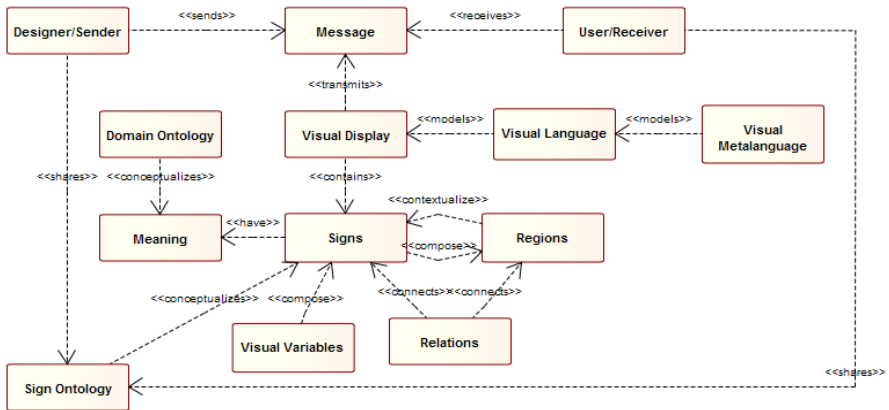


Fig. 1. Model of Visual Communication

A Visual diagram, which describes a specific domain process or a solution of domain problem, is a Message. The Message is sent by a Designer, transmitted via a Visual Display as a communication channel and received by the User(s). Visual Display is a metaphor that facilitates the rapid transfer of an effective mental model into the user’s head [22]. The elements of Visual Display are Signs (basic units of domain knowledge), Regions that provide context to Signs and Relations that connect Signs to Regions. The arrangement of Signs is defined by Visual Language that models Visual Display, whereas Visual Language is modelled by Visual Metalanguage. Signs are chosen to signify a shared meaning of domain that is conceptualized by Domain Ontology, whereas their graphical representation is conceptualized by Sign Ontology shared both by Designer and User(s). The use of ontologies as shared conceptualizations of knowledge minimize the “noise” (i.e., misunderstanding, etc.) introduced during the communication process and allows logical reasoning and inference over signs as representation of meaning. Therefore, ontologies can be employed as a tool for visual or diagrammatic reasoning [23].

3 Modelling the Syntax of Visual Language

A VPL is an artificial system of communication that uses visual elements. Any visual language can be characterized by three main elements: lexical definition (symbol vocabulary), syntactical definition (grammar), and semantic definition. VPLs differ from textual programming languages by the type of used symbols and the type of their relation to each other.

3.1 Lexical Definition

First, a set of symbols is extended from a set of characters (e.g., ASCII, Unicode) to a set of any images. Formally, a visual symbol S_v is defined as a quadruple $S_v = (I, C, M, A)$, where I is the image that is shown to the user of the language; C is the position of the symbol in the visual sentence that defines its context, i.e., the relation of the visual symbols to other symbols; M is the semantic meaning of the visual symbol; and A is the set of actions that are performed when the symbol is activated.

We propose using ontological engineering methods for modelling and specifying a vocabulary of the language. The symbol vocabulary formally can be defined as ontology of symbols (signs) $O = (D, R)$, where D is some domain, and $R \subseteq D^n$ is a set of relations defined in D . Visual language L_v can be defined as $L_v \subseteq S_v^*, S_v \in O$. The relations between symbols can be: taxonomic (the symbols belong to the same group or category of symbols), mereologic (one symbol is a part of other symbol), positional (the symbols are always used together though do not form a separate symbol).

Second, the sequencing of symbols is extended from one-dimensional to multi-dimensional. If we have a visual language L_v then a visual sentence of a language L_v is a spatial arrangement of visual symbols specified according to the syntax rules (grammar) of L_v . The definition of language grammar implies the need for a meta-grammar that defines the structural composition of the syntax grammar rules.

3.2 Syntactical Definition

In the classic definition of generative string grammars, a grammar $G(L)$ of language L is defined as $G(L) = (N, T, P, S)$, where N is a finite set of nonterminal symbols (grammar variables), none of which appear in strings formed from G , T is a finite set of terminal symbols (grammar constants), $T \cap N = \emptyset$; P is a finite set of production rules that map from one string of symbols to another in the form of $(N \cup T)^* N (N \cup T)^* \rightarrow (N \cup T)^*$, and S is the start symbol, $S \in N$. In a visual language, the ordering of visual symbols is non-linear, therefore, the visual production rules include visual relations R as follows: $((N \cup T)R)^* N (R(N \cup T))^* \rightarrow (N \cup T)^*$.

In practice, the syntax of textual languages is usually defined by meta-syntax notation such as the Extended Backus-Naur-Form (EBNF) (ISO/IEC 14977). To define a visual grammar of a visual language, the definition of string grammar must be

extended to include visual symbols and visual relations, which indicate the spatial arrangement of the elements of productions such as connection relation to connect visual symbols with adjacent regions through links or arrows, and geometric relations (containment, horizontal/vertical and left/right concatenations, etc.). In one of such extensions, Picture Description Language (PDL) [24], each terminal symbol is labelled at its head and its tail, and the coincidence operators between primitives are defined. Ledley [25] also proposed a set of topological operators as relations between terminal symbols. The summary of visual relation operators is given in Table 1.

Table 1. Visual relation operators

Operator	Interpretation	Operator	Interpretation
$S_1 + S_2$		$S_1 \rightarrow S_2$	
$S_1 \times S_2$		$S_1 \uparrow S_2$	
$S_1 - S_2$		$S_1 \odot S_2$	
$S_1 * S_2$			

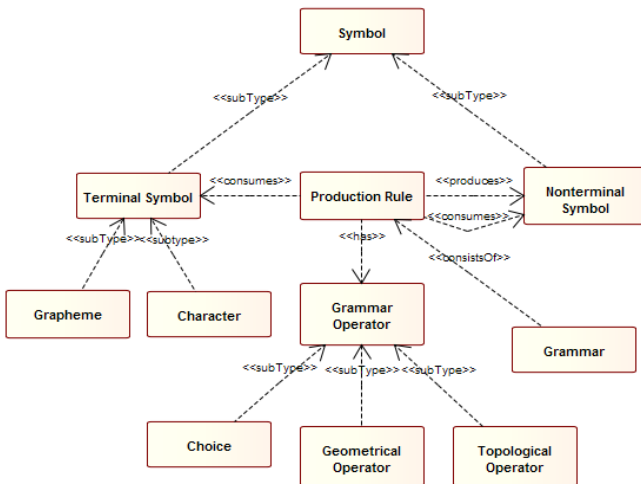


Fig. 2. Visual Syntax model

3.3 Visual Syntax Model and Meta-syntax of Visual Language

The proposed Visual Syntax Model is summarized in Fig. 2. The model defines two types of symbols: terminals (characters or graphemes) and non-terminals. Production rules describe how a sequence of terminal and nonterminal symbols can be consumed by the parser of the language to produce terminal symbols as governed by the grammar operators. Grammar operators are: *choice* operator (selection of production rules out of a set of alternatives), *geometrical* operators (2D production), *topological* operators (symbol containment/concatenation) and *abstraction* operators (symbol instantiation between different levels of abstraction beyond the 2D visual space).

Considering the above, we propose the following meta-syntax based on EBNF:

```

<syntax> ::= <rule> | <rule> <syntax>
<rule> ::= "<" <rule-name> ">" "==" <expression> <EOL>
<expression> ::= <term-list> | "(" <term-list> ")"
                | <term-list> <operator> <expression>
<operator> ::= "|" | "+" | "-" | "x" | "*" | "→" | "↑" | "⊙"
<term-list> ::= <term> | <term> <term-list>
<term> ::= <grapheme> | "<" <rule-name> ">"
<grapheme> ::= <icon> | <icon> "(" <attribute-list> ")"
<attribute-list> ::= <literal> | <literal> <attribute-list>
<literal> ::= ' " ' <text> ' " '

```

4 Case Study: Modelling Syntax of Visual Language

As a case study of the proposed approach, we analyse and present results of modelling a visual programming language *VisuRobo* for the mobile robotics domain. The designer faces two main challenges when developing a visual language: 1) selection (construction) and design of a visual vocabulary for the developed language, and 2) selection of the meta-modelling language for describing the syntax of the language.

4.1 Visual Vocabulary of the Language

When designing a vocabulary the main concerns are as follows:

1) *Understandability*. The designers want to communicate ideas using visual language with as less noise as possible. This requires that the mental models of designers must be “shareable”, i.e., the symbols used to communicate meaning must be easily recognizable and interpretable. The meaning of symbols must be known intuitively.

2) *Usability*. The symbols must be designed using a principled methodology (see, e.g., Moody’s “Physics of notations” [19]), and their usability must be thoroughly evaluated using quantitative metrics [10] and/or qualitative surveys.

For our design task we have decided to select a subset of symbols from a set of road traffic sign system rather than to design our own set of symbols. The advantages of such choice are as follows: 1) Traffic signs are a part of our everyday life and are simple, easily recognizable, legible and understandable [26]; 2) Traffic signs were designed to be usable in different contexts of use (day/night, static/dynamic

environment) and contain a minimum amount of information (or “conceptual baggage” [27]) required to communicate a message; 3) Traffic signs are an international standard (*Vienna Convention on Road Signs and Signals*); 4) Ontology of road signs is available [28]; 5) Meaning of traffic signs are easily transferrable to a mobile robotics domain using the analogy principle [29], which allows the user to metaphorically reinterpret familiar signs in another context of use based on the similarity of vehicle driving and mobile robotics domains.

The selection of signs for *VisuRobo* is based on the Robot Programming Ontology [30], which defines the main actions of a mobile robot. Ontological description allows to define their graphical representation formally and ontology tools (such as Protégé) provide capabilities for checking consistency of formal description and reasoning over the meaning of signs. An example of the description of the PARKING sign in the Road Sign Ontology [28] is given in Fig. 3. The ontology provides the classification of road signs and defines their graphical composition in terms of colour, shape and represented symbols. The description of the Parking sign states that the sign shall have blue background, white border, a rectangular shape and the “P” symbol on it.

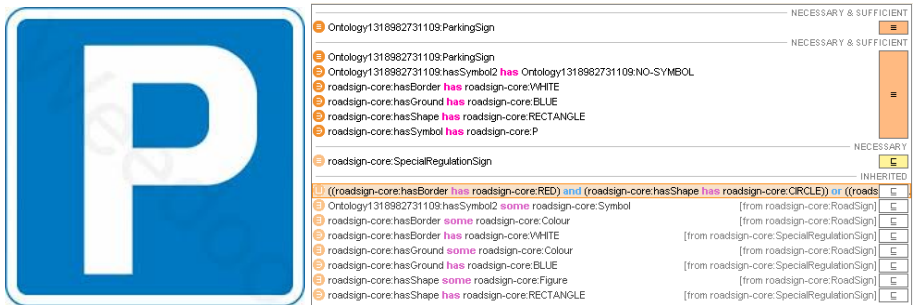







Fig. 3. Parking sign and its assertions derived from the Road Sign Ontology as seen in Protege

In Table 2, we summarize the visual graphemes of the *VisuRobo* language. We consider each sign as a metaphor; therefore the description is given following the Barr’s [31] relations (parts) of metaphor as follows: 1) *Definition*: the meaning of the metaphor itself. 2) *Designer’s Interpretation*: the meaning of the grapheme according to the designer’s mental model. 3) *Match*: how metaphor in the mental model matches to its realization, i.e., the evaluation of the designer’s interpretation. 4) *User interpretation*: the meaning of the grapheme according to the user’s mental model. 5) *Success*: how the user’s interpretation matches the definition of grapheme, i.e., the evaluation of user’s interpretation.

Visual variables [18] that are used to communicate meaning are *shape*, *colour* and *orientation*. The signs for the begin/end of a program have an octagonal shape, the signs for moving (driving) have a round shape, and the signs for temporary stopping or complex nested operation have a rectangular shape. The colours are used sparingly: *green* for starting command, *black* for road and sensors, *red* for stopping command, *yellow* for conditional execution, and *blue* for other commands. The orientation of arrows on driving signs shows the direction of movement. Textual notation is used for communicating simple meaning (“Start”, “Stop”, “Park”). Good visibility is ensured by high contrast between background, foreground and text colour.

Table 2. Graphemes (visual tokens) of language and their interpretation

Grapheme	Definition	Designer's interpretation	User's interpretation	Match/Success
	Road	Defines a path of execution for the robot	Drive, proceed to next command	The metaphor of execution is captured and transferred to user well
	(non standard sign)	Defines an entry point to the program	Start driving robot	Analogy to "STOP" sign is used (shape), but colour is changed to "prohibitive" red to "permissive" green
	Stop	Defines an end point to the program	Robot stops	Analogy between the concepts of stopping a vehicle and finishing a program is used
	Ahead only	Instructs a robot to drive forward for a set amount of time	Robot drives forward	Perfect match
	(non standard sign)	Instructs a robot to drive backward for a set amount of time	Robot drives backward	Analogy with driving forward is used to create a sign for "driving backward"
	Turn left	Instructs a robot to turn left using given the turn angle	Robot turns left	Perfect match
	Turn right	Instructs a robot to turn right using given the turn angle	Robot turns right	Perfect match
	Roundabout	Repetitive statement	Loop	Metaphor of driving roundabout matches well with repetition and looping concepts
	Recommended speed	Set a speed of driving	Robot's driving speed, in percents	Good match (though some unclarity in measurement units is noted)
	Parking	Instructs a robot to stop driving and wait for the given amount of time	Pause, parking	The meaning of sign is understood intuitively well
	Repair	Instructs a robot to execute a mission until the sensor command returns true	Robot performs some operations	An analogy between repair workshop and complexity of robot mission is observed
	Speed camera	Reads ultrasound sensor value and returns true if sensor value is higher than given threshold value	Any sensor	Metaphor of "Camera" as on observing device is extended to all kinds of sensors
	Y-intersection	Defines two paths of execution the selection thereof is defined by value returned by the sensor command	Conditional execution	Decision on taking different roads matches well with metaphor of conditional execution
	(non standard sign)	Defines an intersection of two paths of execution	Merging of execution paths	Analogy with the Y-intersection sign is used to create a sign for a merger of execution paths

4.2 Specification of Grammar Rules

The language is based on a metaphor of road traffic and follows a set of assumptions:

1. The behaviour of a robot is represented by a road metaphor.
2. Each road may consist of an unlimited number of sub-roads a robot must cover.
3. Road defines an independent path of execution.
4. Each road has its speed limitations valid until next intersection with another road.
5. Intersection means the end of incoming roads and beginning of outgoing roads.
6. There are two types of intersections. Join intersection has 2 incoming roads and one outgoing road. Split intersection has one incoming road and 2 outgoing roads.
7. Each road has only one beginning and only one end.
8. While on road the robot can execute a number of missions.
9. Each mission consists of implementing a pre-defined algorithm until specific condition is met.
10. Conditions are defined by the value returned from a sensor.

The syntax of the *VisuRobo* language is defined using EBNF-like syntax extended with visual tokens, geometrical relation primitives and topological operators (Fig. 4).

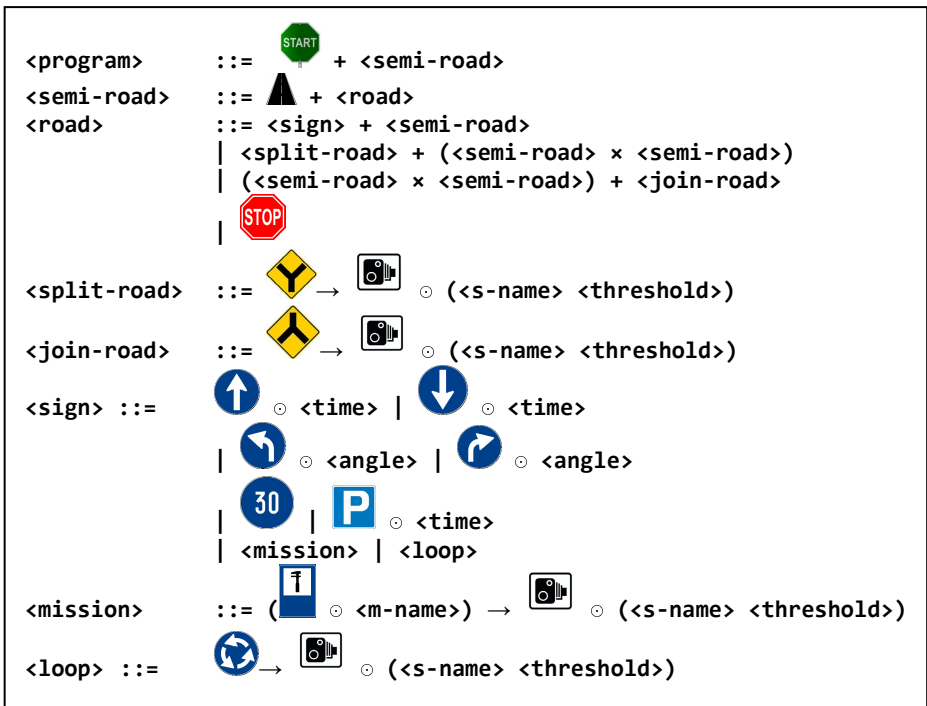


Fig. 4. Syntax description using visual extension of BNF

5 Conclusions

1. Visual Communication Model, which is based on the combination and amalgamation of ideas from [12, 14-19] as well as elements of ontological engineering. The model defines the construction of visual language's syntax and ensures "low noise" communication due to shared domain and sign ontologies.
2. Visual Syntax Model for describing grammar of VPLs and meta-syntax of visual metalanguage. The metalanguage is an extension of EBNF with visual symbols and nonlinear composition of rules that allows to define the syntax of VPL.
3. *VisuRobo*, an illustrative visual robot programming language, serves as a proof-of-concept for the proposed Visual Communication Model, Visual Syntax Model and visual metalanguage. The language uses a metaphor of road driving for a mobile robot and uses a subset of visual signs adopted from the Road Sign Ontology to facilitate transfer of meaning between language designers and users, and to deal with the sign interpretability problem.
4. The evaluation of the language syntax according to Barr [31] and Bertin [18] is given. The familiarity of visual signs and the context of their use allow to transfer the designer message to the users without significant semantic misinterpretations while the consistent use of visual variables (shape, colour and orientation) allows to create a simple, usable and attractive vocabulary of the language.

Acknowledgement. The work described in this paper has been carried out within the framework of the Operational Programme for the Development of Human Resources 2007-2013 of Lithuania „Strengthening of capacities of researchers and scientists" project VP1-3.1-ŠMM-08-K-01-018 „Research and development of Internet technologies and their infrastructure for smart environments of things and services" (2012-2015), funded by the European Social Fund (ESF).

References

1. Gentner, D., Stevens, A.L. (eds.): *Mental Models*. Lawrence Erlbaum Associates (1983)
2. Eisenstadt, M., Domingue, J., Rajan, T., Motta, E.: *Visual Knowledge Engineering*. IEEE Transactions on Software Engineering 16(10), 1164–1177 (1990)
3. Zhang, K., Kong, J., Cao, J.: *Visual Software Engineering*. Wiley Encyclopaedia of Computer Science and Engineering (2008)
4. Zhang, K.: *Visual Languages and Applications*. Springer (2007)
5. Booch, G., Rumbaugh, J., Jacobson, I.: *The Unified Modeling Language User Guide*. Addison Wesley Longman Publishing Co., Inc., Redwood City (1999)
6. Bentrad, S., Meslati, D.: *Visual Programming and Program Visualization – Towards an Ideal Visual Software Engineering System*. IIIT- ACEEE Int. Journal on Information Technology 1(3), 56–62 (2011)
7. Myers, B.A.: *Taxonomies of Visual Programming and Program Visualization*. Visual Languages and Computing 1(1) (1990)
8. Burnett, M.: *Visual Programming*. In: J. Webster (ed.), *Encyclopedia of Electrical and Electronics Engineering*. John Wiley & Sons (1999)

9. Deufemia, V.: A Grammar-based Approach to Specify and Implement Visual Languages. PhD Dissertation, University of Salerno (2002)
10. Plauska, I., Damaševičius, R.: Usability Analysis of Visual Programming Languages Using Computational Metrics. In: Proceedings of the IADIS International Conference on Interfaces and Human-Computer Interaction 2013, Prague, Czech Republic, pp. 63–70 (July 2013)
11. Lakin, F.: Visual grammars for visual languages. In: Proc. of the Sixth National Conference on Artificial Intelligence AAAI 1987, vol. 2, pp. 683–688. AAAI Press (1987)
12. Shannon, C.E., Weaver, W.: The mathematical theory of communication. University of Illinois Press, Urbana (1949)
13. Mortensen, C.D.: Communication: The Study of Human Communication. In: Communication Models, ch. 2, McGraw-Hill Book Co. (1972)
14. Berlo, D.K.: The Process of Communication. Holt, Rinehart, and Winston (1960)
15. Souza, C.S.: The Semiotic Engineering of Human-Computer Interaction. MIT Press (2005)
16. Hari Narayanan, N., Hubscher, R.: Visual language theory: Towards a human computer interaction perspective. In: Marriott, K., Meyer, B. (eds.) Visual Language Theory, pp. 87–128 (1998)
17. Tartre, M.: Theory of Visual Display (2013), <http://www.metaperture.com>
18. Bertin, J.: Semiology of Graphics: Diagrams, Networks, Maps. ESRI Press (2010)
19. Moody, D.L.: The “Physics” of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. IEEE Trans. Soft. Eng. 35(6), 756–779 (2009)
20. Devedzic, V.: Understanding Ontological Engineering. Communications of the ACM 45(4), 136–144 (2002)
21. Guizzardi, G.: On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models. In: Proc. of conference on Databases and Information Systems IV: Selected Papers from the 7th International Baltic Conference DB&IS 2006, pp. 18–39. IOS Press (2007)
22. Blackwell, A.F.: The reification of metaphor as a design tool. ACM Transactions on Computer-Human Interaction (TOCHI) 13(4), 490–530 (2006)
23. Glasgow, J., Hari Narayanan, N., Chandrasekaran, B.: Diagrammatic Reasoning: Cognitive and Computational Perspectives. MIT Press, Cambridge (1995)
24. Shaw, A.: A Formal Picture Description Scheme as a Basis for Picture Processing Systems. Inf. Control (14), 9–52 (1969)
25. Ledley, R.: Programming and Utilising Digital Computers. McGraw-Hill (1962)
26. Ng, A.W.Y., Chan, A.H.S.: Cognitive Design Features on Traffic Signs. Engineering Letters 14(1), 13–18 (2007)
27. Anderson, B., Smyth, M., Knott, R.P., Bergan, M.S., Bergan, J., Alty, J.L.: Minimising conceptual baggage: making choices about metaphor. In: Proc. of Conference on People and Computers IX (HCI 1994), pp. 179–194. Cambridge University Press (1994)
28. Pousa, M., Motto, O., Carasusán, E.: Road Sign Ontology (2011), <https://raw.githubusercontent.com/ecarasusan/roadsign/master/roadsign.owl>
29. Breitman, K.K., Barbosa, S.D.J., Casanova, M.A., Furtado, A.L.: Conceptual modeling by analogy and metaphor. In: Proc. of the 16th ACM Conference on Information and Knowledge Management, Lisbon, Portugal, pp. 865–868 (2007)
30. Plauska, I.: Ontology for Robot Programming Domain. In: IVUS, pp. 51–56 (2013)
31. Barr, P., Noble, J., Biddle, R.: A semiotic model of user-interface metaphor. In: Liu, K. (ed.) Virtual Distributed and Flexible Organisations, pp. 189–216. Kluwer Academic (2003)