Modeling language

A **modeling language** is any artificial language that can be used to express information or knowledge or systems in a structure that is defined by a consistent set of rules. The rules are used for interpretation of the meaning of components in the structure.

1 Overview

A modeling language can be graphical or textual.[1]

- *Graphical* modeling languages use a diagram technique with named symbols that represent concepts and lines that connect the symbols and represent relationships and various other graphical notation to represent constraints.
- *Textual* modeling languages may use standardized keywords accompanied by parameters or natural language terms and phrases to make computer-interpretable expressions.

An example of a graphical modeling language and a corresponding textual modeling language is EXPRESS.

Not all modeling languages are executable, and for those that are, the use of them doesn't necessarily mean that programmers are no longer required. On the contrary, executable modeling languages are intended to amplify the productivity of skilled programmers, so that they can address more challenging problems, such as parallel computing and distributed systems.

A large number of modeling languages appear in the literature.

2 Type of modeling languages

2.1 Graphical types

Example of graphical modeling languages in the field of computer science, project management and systems engineering:

- Behavior Trees are a formal, graphical modeling language used primarily in systems and software engineering. Commonly used to unambiguously represent the hundreds or even thousands of natural language requirements that are typically used to express the stakeholder needs for a large-scale software-integrated system.
- Business Process Modeling Notation (BPMN, and the XML form BPML) is an example of a Process Modeling language.
- C-K theory consists of a modeling language for design processes.
- DRAKON is a general-purpose algorithmic modeling language for specifying software-intensive systems, a schematic representation of an algorithm or a stepwise process, and a family of programming languages.
- EXPRESS and EXPRESS-G (ISO 10303-11) is an international standard general-purpose data modeling language.
- Extended Enterprise Modeling Language (EEML) is commonly used for business process modeling across a number of layers.
- Flowchart is a schematic representation of an algorithm or a stepwise process.
- Fundamental Modeling Concepts (FMC) modeling language for software-intensive systems.
- IDEF is a family of modeling languages, which include IDEF0 for functional modeling, IDEF1X for information modeling, IDEF3 for business process modeling, IDEF4 for Object-Oriented Design and IDEF5 for modeling ontologies.
- Jackson Structured Programming (JSP) is a method for structured programming based on correspondences between data stream structure and program structure.
- LePUS3 is an object-oriented visual Design Description Language and a formal specification language that is suitable primarily for modeling large object-oriented (Java, C++, C#) programs and design patterns.
- Object-Role Modeling (ORM) in the field of software engineering is a method for conceptual modeling, and can be used as a tool for information and rules analysis.
- Petri nets use variations on exactly one diagramming technique and topology, namely the bipartite graph.

The simplicity of its basic user interface easily enabled extensive tool support over the years, particularly in the areas of model checking, graphically oriented simulation, and software verification.

- Southbeach Notation is a visual modeling language used to describe situations in terms of agents that are considered useful or harmful from the modeler's perspective. The notation shows how the agents interact with each other and whether this interaction improves or worsens the situation.
- Specification and Description Language (SDL) is a specification language targeted at the unambiguous specification and description of the behavior of reactive and distributed systems.
- SysML is a Domain-Specific Modeling language for systems engineering that is defined as a UML profile (customization).
- Unified Modeling Language (UML) is a general-purpose modeling language that is an industry standard for specifying software-intensive systems. UML 2.0, the current version, supports thirteen different diagram techniques, and has widespread tool support.
- Service-oriented modeling framework (SOMF) is a holistic language for designing enterprise and application level architecture models in the space of enterprise architecture, virtualization, service-oriented architecture (SOA), cloud computing, and more.[2]
- Architecture description language (ADL) is a language used to describe and represent the systems architecture of a system.
- AADL (AADL) is a modeling language that supports early and repeated analyses of a system's architecture with respect to performance-critical properties through an exetendable notation, a tool framework, and precisely defined semantics.

Examples of graphical modeling languages in other fields of science.

- EAST-ADL is a Domain-Specific Modeling language dedicated to automotive system design.
- Energy Systems Language (ESL), a language that aims to model ecological energetics & global economics.

2.2 Textual types

Information models can also be expressed in formalized natural languages, such as Gellish.[3] Gellish has natural language variants such as Gellish Formal English and Gellish Formal Dutch (Gellish Formeel Nederlands), etc. Gellish Formal English is an information representation language or semantic modeling language that is defined in the Gellish English Dictionary-Taxonomy, which has the form of a Taxonomy-Ontology (similarly for Dutch). Gellish Formal English is not only suitable to express knowledge, requirements and dictionaries, taxonomies and ontologies, but also information about individual things. All that information is expressed in one language and therefore it can all be integrated, independent of the question whether it is stored in central or distributed or in federated databases. Information models in Gellish Formal English consists of collections of Gellish Formal English expressions, that use natural language terms and formalized phrases. For example, a geographic information model might consist of a number of Gellish Formal English expressions, such as:

- the Eiffel tower <is located in> Paris - Paris <is classified as a> city

whereas information requirements and knowledge can be expressed for example as follows:

- tower <shall be located in a> geographical area - city <is a kind of> geographical area

Such Gellish Formal English expressions use names of concepts (such as 'city') and phrases that represent relation types (such as <is located in> and <is classified as a>) that should be selected from the Gellish English Dictionary-Taxonomy (or of your own domain dictionary). The Gellish English Dictionary-Taxonomy enables the creation of semantically rich information models, because the dictionary more than 600 standard relation types and contains definitions of more than 40000 concepts. An information model in Gellish can express facts or make statements, queries and answers.

2.3 More specific types

In the field of computer science recently more specific types of modeling languages have emerged.

2.3.1 Algebraic

Algebraic Modeling Languages (AML) are high-level programming languages for describing and solving high complexity problems for large scale mathematical computation (i.e. large scale optimization type problems). One particular advantage of AMLs like AIMMS, AMPL, GAMS, LPL, MPL, OPL and OptimJ is the similarity of its syntax to the mathematical notation of optimization problems. This allows for a very concise and readable definition of problems in the domain of optimization, which is supported by certain language elements like sets, indices, algebraic expressions, powerful sparse index and data handling variables, constraints with arbitrary names. The algebraic formulation of a model does not contain

any hints how to process it.

2.3.2 Behavioral

Behavioral languages are designed to describe the observable behavior of complex systems consisting of components that execute concurrently. These languages focus on the description of key concepts such as: concurrency, nondeterminism, synchronization, and communication. The semantic foundations of Behavioral languages are process calculus or process algebra.

2.3.3 Discipline-Specific

A discipline-specific modeling (DspM) language is focused on deliverables affiliated with a specific software development life cycle stage. Therefore, such language offers a distinct vocabulary, syntax, and notation for each stage, such as discovery, analysis, design, architecture, contraction, etc. For example, for the analysis phase of a project, the modeler employs specific analysis notation to deliver an analysis proposition diagram. During the design phase, however, logical design notation is used to depict relationship between software entities. In addition, the discipline-specific modeling language best practices does not preclude practitioners from combining the various notations in a single diagram.

2.3.4 Domain-specific

Domain-specific modeling (DSM) is a software engineering methodology for designing and developing systems, most often IT systems such as computer software. It involves systematic use of a graphical domain-specific language (DSL) to represent the various facets of a system. DSM languages tend to support higher-level abstractions than General-purpose modeling languages, so they require less effort and fewer low-level details to specify a given system.

2.3.5 Framework-specific

A framework-specific modeling language (FSML) is a kind of domain-specific modeling language which is designed for an object-oriented application framework. FSMLs define framework-provided abstractions as FSML concepts and decompose the abstractions into features. The features represent implementation steps or choices.

A FSML concept can be configured by selecting features and providing values for features. Such a concept configuration represents how the concept should be implemented in the code. In other words, concept configuration describes how the framework should be completed in order to create the implementation of the concept.

2.3.6 Object-oriented

Object modeling language are modeling languages based on a standardized set of symbols and ways of arranging them to model (part of) an object oriented software design or system design.

Some organizations use them extensively in combination with a software development methodology to progress from initial specification to an implementation plan and to communicate that plan to an entire team of developers and stakeholders. Because a modeling language is visual and at a higher-level of abstraction than code, using models encourages the generation of a shared vision that may prevent problems of differing interpretation later in development. Often software modeling tools are used to construct these models, which may then be capable of automatic translation to code.

2.3.7 Virtual reality

Virtual Reality Modeling Language (VRML), before 1995 known as the Virtual Reality Markup Language is a standard file format for representing 3-dimensional (3D) interactive vector graphics, designed particularly with the World Wide Web in mind.

2.3.8 Others

- Architecture Description Language
- Face Modeling Language
- Generative Modelling Language
- Java Modeling Language
- Promela
- Rebeca Modeling Language
- Service Modeling Language
- Web Services Modeling Language
- X3D

3 Applications

Various kinds of modeling languages are applied in different disciplines, including computer science, information management, business process modeling, software engineering, and systems engineering. Modeling languages can be used to specify:

- system requirements,
- structures and
- behaviors.

Modeling languages are intended to be used to precisely specify systems so that stakeholders (e.g., customers, operators, analysts, designers) can better understand the system being modeled.

The more mature modeling languages are precise, consistent and executable. Informal diagramming techniques applied with drawing tools are expected to produce useful pictorial representations of system requirements, structures and behaviors, but not much else. Executable modeling languages applied with proper tool support, however, are expected to automate system verification and validation, simulation and code generation from the same representations.

4 Quality

A review of modelling languages is essential to be able to assign which languages are appropriate for different modelling settings. In the term settings we include stakeholders, domain and the knowledge connected. Assessing the **language quality** is a means that aims to achieve better models.

4.1 Framework for evaluation

Here language quality is stated in accordance with the SEQUAL framework for quality of models developed by Krogstie, Sindre and Lindland (2003), since this is a framework that connects the language quality to a framework for general model quality. Five areas are used in this framework to describe language quality and these are supposed to express both the conceptual as well as the visual notation of the language. We will not go into a thoroughly explanation of the underlying quality framework of models but concentrate on the areas used to explain the language quality framework.

4.1.1 Domain appropriateness

The framework states the ability to represent the domain as domain appropriateness. The statement *appropriateness* can be a bit vague, but in this particular context it means *able to express*. You should ideally only be able to express things that are in the domain but be powerful enough to include everything that is in the domain. This requirement might seem a bit strict, but the aim is to get a visually expressed model which includes everything relevant to the domain and excludes everything not appropriate for the domain. To achieve this, the language has to have a good distinction of which notations and syntaxes that are advantageous to present.

4.1.2 Participant appropriateness

To evaluate the participant appropriateness we try to identify how well the language expresses the knowledge held by the stakeholders. This involves challenges since a stakeholder's knowledge is subjective. The knowledge of the stakeholder is both tacit and explicit. Both types of knowledge are of dynamic character. In this framework only the explicit type of knowledge is taken into account. The language should to a large extent express all the explicit knowledge of the stakeholders relevant to the domain.

4.1.3 Modeller appropriateness

Last paragraph stated that knowledge of the stakeholders should be presented in a good way. In addition it is imperative that the language should be able to express all possible explicit knowledge of the stakeholders. No knowledge should be left unexpressed due to lacks in the language.

4.1.4 Comprehensibility appropriateness

Comprehensibility appropriateness makes sure that the social actors understand the model due to a consistent use of the language. To achieve this the framework includes a set of criteria. The general importance that these express is that the language should be flexible, easy to organize and easy to distinguish different parts of the language internally as well as from other languages. In addition to this, the goal should be as simple as possible and that each symbol in the language has a unique representation.

4.1.5 Tool appropriateness

To ensure that the domain actually modelled is usable for analyzing and further processing, the language has to ensure that it is possible to reason in an automatic way. To achieve this it has to include formal syntax and semantics. Another advantage by formalizing is the ability to discover errors in an early stage. It is not always that the language best fitted for the technical actors is the same as for the social actors.

4.1.6 Organizational appropriateness

The language used is appropriate for the organizational context, e.g. that the language is standardized within the organization, or that it is supported by tools that are chosen as standard in the organization.

5 See also

- Analogical models

- Anthropomorphism
- Discipline-Specific Modeling
- Metamodeling
- Model-based testing (MBT)
- Model-driven architecture
- Model-driven engineering (MDE)
- Modeling perspectives
- Scientific modeling
- Visual modeling
- Visual programming language

6 References

[1] Xiao He (2007). "A metamodel for the notation of graphical modeling languages". In: *Computer Software and Applications Conference, 2007. COMPSAC 2007 - Vol. 1. 31st Annual International*, Volume 1, Issue , 24–27 July 2007, pp 219-224.

[2] Bell, Michael (2008). "Introduction to Service-Oriented Modeling". *Service-Oriented Modeling: Service Analysis, Design, and Architecture*. Wiley & Sons. ISBN 978-0-470-14111-3.

[3]
- Andries van Renssen, Gellish, A Generic Extensible Ontological Language, Delft University of Technology, 2005.

7 External links

- Fundamental Modeling Concepts
- Software Modeling Languages Portal
- BIP -- Incremental Component-based Construction of Real-time Systems
- Gellish Formal English

8 Further reading

- John Krogstie (2003) "Evaluating UML using a generic quality framework" . SINTEF Telecom and-Informatics and IDI, NTNU, Norway
- Krogstie and Sølvsberg (2003). *Information Systems Engineering: Conceptual Modeling in a Quality Perspective*. Institute of computer and information sciences.\
- Anna Gunhild Nysetvold and John Krogstie (2005). "Assessing business processing modeling languages using a generic quality framework". Institute of computer and information sciences.

9 Text and image sources, contributors, and licenses

9.1 Text

- **Modeling language** *Source:* https://en.wikipedia.org/wiki/Modeling_language?oldid=678084550 *Contributors:* William Avery, Edward, Michael Hardy, RedWolf, Andreas Kaufmann, S.K., Bcat, Mdd, Ruud Koot, Mayumashu, XP1, Ligulem, Vonkje, Shaggyjacobs, YurikBot, Wavelength, Mjchonoles, SmackBot, Kobryn, Sholto Maud, Allan McInnes, Radagast83, Kuru, KenFehling, Laogeodritt, JoeBot, Steven Kelly, MDE, Alphachimpbot, AndriesVanRenssen, Ensign beedrill, David.t.bath, Adavidb, Mintz l, VolkovBot, Slacrampe, Jcabotsagrera, ASHPvanRenssen, BotMultichill, Jerryobject, Ivpen, Krogstie, Addbot, Sean R Fox, Maria C Mosak, Phil007, Tassedethe, Richard R White, Yobot, Sair00, Amin Hashem, Some standardized rigour, FrescoBot, Mark Renier, Mbonline, Vasywriter, Lotje, Theo10011, Crisgh, Architectchao, Sven Manguard, ClueBot NG, Widr, Carl presscott, Владимир Пароджанов, Panchang Jiang, ThomasRules, The ken evans, Phamnhatkhanh, Tentinator, Ed Gives and Anonymous: 25

9.2 Images

- **File:Commons-logo.svg** *Source:* https://upload.wikimedia.org/wikipedia/en/4/4a/Commons-logo.svg *License:* ? *Contributors:* ? *Original artist:* ?
- **File:Folder_Hexagonal_Icon.svg** *Source:* https://upload.wikimedia.org/wikipedia/en/4/48/Folder_Hexagonal_Icon.svg *License:* Cc-by-sa-3.0 *Contributors:* ? *Original artist:* ?

9.3 Content license

- Creative Commons Attribution-Share Alike 3.0