Design Models and Design Languages 2

In this chapter, we discuss our interpretation of the basic design notions, terminology and meta-concepts that are frequently used in design and design specification approaches. We present precise definitions for those notions that underlie our design methodology, namely *modelling*, *abstraction* and *refinement*, *design concepts* and *design language*.

These basic design notions play a crucial role in the clear and unambiguous understanding of our basic design concepts, design methodology and specification techniques. We explain the role of model as a representation of an object for a certain purpose and the notion of abstraction as a means to focus only on those properties of an object that are considered relevant for a model at a certain abstraction level. We explicate design refinement as a means to add relevant design details at a lower abstraction level. We explain the role of a design language as an artificial means to symbolically represent an object, and as a means to communicate and store those representations amongst humans and machines. We finally conclude that our basic design concepts are elementary building bricks to conceive models of an object.

2.1 Design Model

Section 1.5 argues that designing a system is a creative process that can only take place in the mind of a designer. A designer has to imagine and conceive a system in his mind before further steps can be taken. This principle, however, holds only for simple systems that are easy to imagine and conceive. As soon as a system grows beyond elementary complexity it quickly becomes difficult to completely survey and grasp it. The process of imagining and conceiving such systems only in the mind of the designer then gets obscure and error prone, let alone repeating this conceiving process at successive levels of decomposition. Therefore we need a tool

C.A. Vissers et al., *Architectural Design*, DOI 10.1007/978-3-319-43298-4_2

to embody our imagination and conceptions and that help us survey and master the complexity of systems and their design.

One of the first tools we need is a mechanism to shape, hang on and remember our mental images of the design at different composition levels. In other words, we need a tool that represents and stands for our conceptions so that we can repeatedly look at them, refresh our memory, analyse them and change them if appropriate. Such a tool is provided by the notion of model.

Definition 2.1 Model
A *model* of an object represents certain properties of that object and is made for one or more specific purposes.

This implies that a model of an object is not the object itself, but is a *representation* of that object.

If the purpose of a model is to represent a design, we speak of a *design model* or simply of a *design*. A design model is created for the purpose of representing the properties of an object, analysing these properties and evaluating alternative properties of the object before the object is possibly made to exist.

Using a design model, the designer can refer to it as a more tangible representation of his design rather than only referring to his memory when creating it and working on it. This implies in the concurrent existence of two related representations: (1) the more volatile imagination in the mind of the designer, and (2) the more tangible model. The latter can only be derived from the designer's imagination.

Whereas models are indispensable in a design process, the reader should be aware that a model is only a passive means for representation. The creative process, the imagination and conception of a system, remains the responsibility of the designer for which he still must (and only can) use the intellectual capabilities of his mind. This responsibility can never be overtaken by any model. Moreover, the creation and modification of the design model is the very responsibility of the designer.

One can build a miniature airplane of some material to represent the shape proportions of a new airplane under design for the purpose of measuring and testing its aerodynamic properties in a wind tunnel. This representation is used as a stand-in for the real airplane because testing the real airplane in free air is way too expensive or too risky, certainly if the designer is not sure of his case and wants to test various alternatives before making a choice.

Models are used in many fields of engineering. In [23–26] we find ICT models used for designing Geo-information Services, and distributed applications in general.

2.1.1 Alternative Models

An object can be represented by different, alternative models to serve the same purpose.

An alternative model for checking the aerodynamic properties of an airplane is a mathematical description of the shape proportions whereas the wind tunnel is replaced by a simulation program that can run on a computer. In this case, both the miniature airplane and the mathematical description represent the same shape proportions. However, the form of representation in both models is quite different. The miniature airplane represents the shape proportions by proportionally imitating the real shape of the airplane in some constructive material, which makes it suitable for wind tunnel testing. The mathematical description represents the shape proportions by variable settings in mathematical formulary's, eventually expressed in data, which are suitable to be processed by the simulation program.

An object can also be represented by different, alternative models to serve different purposes.

A system environment can be diverse and have different sorts of users, each one having a different view on the system and requiring a different model. This means that a system can have multiple alternative models, one for each of the perspectives of these different sorts of users.

A TV set may be modelled from the perspective of the spectators, the manufacturers, the salesmen and the repairmen. These sorts of users normally have different expectations with respect to the system. For the manufacturers a TV set is a product that has to be cheaply and efficiently assembled, for the salesmen a TV set must appeal to its potential buyers, and for the repairmen a TV set must be easy to diagnose and repair.
Taking the construction of a new house as an example, a scale replica and a construction drawing (blueprint) are different models that represent the same house, but they can be used for different purposes. The scale replica may be used by the real estate agent to give potential buyers a fair impression of how the house is going to look like and to attract their interest. The blueprint is used by the builders to understand how the house should be constructed.

In this text we normally focus on the primary purpose of a system when identifying a model for its users.

For a TV set, we normally consider the spectator as the system user because showing TV programs is the main purpose of a TV set.

2.1.2 Model Requirements

A model can only be effective for its purpose if it fulfils the following related requirements:

- *Preciseness*: it exactly reflects the system properties necessary for the purpose of the model.
- *Parsimony*: it only reflects the system properties necessary for the purpose of the model.
- *Unambiguity*: the properties that are reflected in the model can be interpreted in only one way.
- *Clarity*: the representation of the properties can be easily understood, and appeal to the intuitive understanding of those who have to use it.

Fulfilling these requirements also allows the model to be used to communicate properties of the object effectively amongst the people interested in it, e.g. designers and users.

2.1.3 Purposes of Modelling

In day-to-day system design, a model may serve several purposes:

- Help conceive and imagine system properties.
- Represent system properties by expressing them in some way.
- Preserve and remember system properties, e.g. by building or documenting the model.
- Communicate a system design amongst designers by exchanging the model.
- Analyse and validate system properties against criteria such as, e.g. correctness, effectiveness, performance, ease of use and costs.
- Starting point for improving the system design, i.e. by redesigning the model.
- Starting point for continuing the design process by extending the model or by creating a decomposition.
- Basis for building a real system.
- Basis for testing the properties of the real system against the properties conceived in the design.

Related to these purposes, terms like test model, validation model, simulation model and blueprint are used.

2.2 Abstraction

In order to be useful for a certain purpose, a model must reflect exactly the properties of the object that are essential for that purpose. This implies that it may ignore all other properties that are irrelevant for that purpose, and in practice it better does so, because including irrelevant properties only creates confusion and chance of errors. Here we touch the essence of the notion of abstraction.

Definition 2.2 Abstraction

An *abstraction* of an object reflects only aspects (or properties) of that object that are considered essential for certain purposes while ignoring, or discarding, aspects that are considered irrelevant for those purposes.

This definition implies that abstraction is a very precise notion that is made operational when the aspects of an object that are considered essential are explicitly identified. When used in this way, abstraction is not a vague and intangible notion as some people may think.

Abstraction is a technique that underlies and imbues our approach.

The shape proportions of an airplane are essential properties for its aerodynamic behaviour and must be accurately represented in the miniature aeroplane if we want to use the latter for checking this behaviour. The real size of the airplane, the power of the engines, the internal construction, etc. are also relevant properties of the real airplane. However, they are irrelevant for the purpose of checking the aerodynamic behaviour. This allows one to abstract from such properties and build a relatively cheap miniature.
In the construction of a new house, when building a miniature we are forced to use some suitable construction material. However, this choice of material is further irrelevant to the shape of the house represented by the miniature.

We conclude that a model of an object is by definition an abstraction of this object because it represents only those properties of the object that are necessary for a certain purpose. Rather than saying a *model of an object with a certain abstraction*, we often abbreviate this to an *abstraction*, an *abstract model* or simply a *model* of that object.

When designing or analysing a system on basis of a model, one should be explicitly aware of the aspects that are essential for the purpose of reaching the specific goals in this design or analysis activity. Consequently, one should ignore all other aspects that are irrelevant for that purpose.

The first design step discussed in Sect. 1.5 aims at designing the Service of the system. The model used for representing the Service, if expressed at the right abstraction level, should be capable of only expressing the externally observable behaviour of the system we want to dispose of, and should ignore how this behaviour is provided by some internal construction of components.

If our goal is to analyse a company in terms of the functions of its departments and their relationships, we should ignore, i.e. *abstract from,* the role of specific individual employees, machines or procedures that function within a department.

Abstraction is both a very powerful and a very dangerous technique, since it tells what to do, i.e. consider certain aspects and ignore others, but not how to do it, i.e. which aspects to consider and which aspects to ignore. This means that the decision on what to abstract from remains a designer's responsibility. In practice it appears that many design projects suffer from poor understanding or inconsistent application of abstraction.

2.2.1 Equivalent Abstractions

When different models are used to represent the same essential properties of an object, we speak of *equivalent abstractions* or *equivalent abstract models*.

The miniature airplane and the mathematical shape description can be considered as equivalent abstract models to represent the shape proportions that are considered essential for the aerodynamic properties of the real airplane.

2.2.2 Viewpoints, Perspectives or Projections

When different models are used to represent different essential properties of an object, we often speak of *viewpoints*, *perspectives* or *projections*.

The scale replica and the drawing blueprint of a new house are different perspectives of the new house. One is used to show the layout and proportions of the house to potential buyers, the other is used to explain the construction details of the house to the builder.

Achieving and maintaining consistency of different viewpoints of the same object is an important and non-trivial design concern, as illustrated in [27–29].

2.2.3 Abstraction and Refinement

Definition 2.3 Refinement

We say that an abstraction A_2 of a system *refines* another abstraction A_1 of the same system if the essential aspects of the system that have been considered in A_1 are fully preserved in A_2, while A_2 adds some more aspects, often called *details*, to these preserved aspects. We say *A_2 is a refinement of A_1*, and we also say that *A_2 implements A_1*.

Refinement is the opposite of abstraction, since if an abstraction A_2 refines another abstraction A_1, we, reversely, can abstract from the details introduced in A_2 and then we obtain again A_1. Thus if A_2 is a refinement of A_1, then A_1 is an abstraction of A_2. Figure 2.1 shows two abstractions and their relationships.

2.2.4 Abstraction Levels

Definition 2.4 Consecutive abstraction levels

A sequence of abstractions A_1, A_2, A_3, …, A_n, where A_2 refines A_1, and A_3 refines A_2, etc. are said to be at *consecutive abstraction levels*.

Figure 2.2 shows an example of four abstraction levels and their relationships.

The first, second and third design steps discussed in Sect. 1.5 produce three models at consecutive abstraction levels, if we assume that each design resulting from these steps is represented by a model and the essential aspects represented in model A_1 (respectively A_2) are preserved in model A_2 (respectively A_3).

The notion of abstraction levels, in particular consecutive abstraction levels, is quite important in design methodologies. It enables the designer to define the abstraction levels along the design process, and their corresponding essential aspects, according to well-defined, related shapes purposes, possibly formulated in concrete guidelines for designing certain systems.

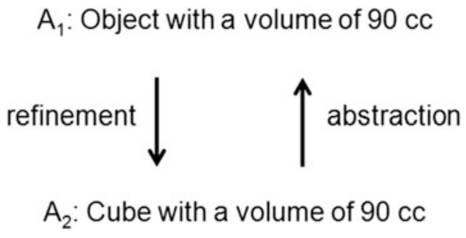
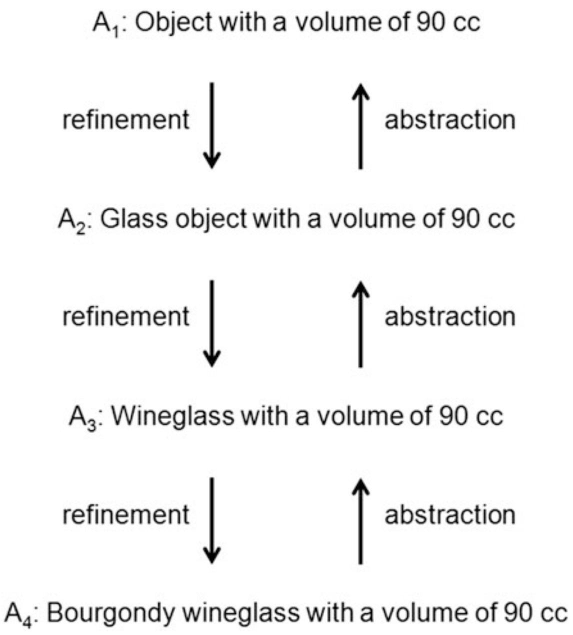

A1: Object with a volume of 90 cc
refinement
abstraction
A2: Cube with a volume of 90 cc

Fig. 2.1 Two related abstractions, forming two consecutive abstraction levels



A_1: Object with a volume of 90 cc
refinement
abstraction
A_2: Glass object with a volume of 90 cc
refinement
abstraction
A_3: Wineglass with a volume of 90 cc
refinement
abstraction
A_4: Bourgondy wineglass with a volume of 90 cc

Fig. 2.2 Four related abstraction levels A_1, A_2, A_3 and A_4

In general, considering a system and its internal composition of system parts, one can zoom in on the internal system structure in consecutive steps, by starting at a coarse composition of high-level components and successively identifying more fine-grained components.

In the design of a bicycle, we can start with relatively big parts, such as a frame, wheels, saddle, handle bars, etc. and in successive abstraction levels reach smaller parts, such as tyres, rims, spokes and hubs.

The application of successive abstraction levels induces *levels of granularity* or *levels of detail*, in which a high abstraction level corresponds to a coarse level of granularity (or a low level of detail), and a low abstraction level corresponds to a fine level of granularity (or a high level of detail).

Examples of the application of abstraction and refinement, leading to a sequence of abstractions at consecutive abstractions level, can be found in, e.g. [30, 31].

2.2.5 Common Properties

Certain objects may have a number of properties in common while they otherwise differ in other properties. Abstraction can then be used to capture the common properties.

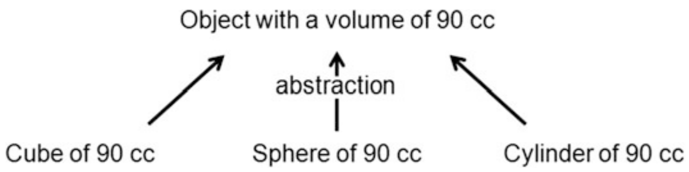


Fig. 2.3 Object with a volume of 90 cc as an abstraction of three concrete objects of 90 cc but with different shapes

Figure 2.3 shows an example of an object with a volume of 90 cc that can be considered as a common abstraction of three different objects when considering the volume of each object as the common property and ignoring the property in which the objects differ, in this case the shape property.

2.2.6 Service as Common Property of Different Implementations

When designing a system, one may start with defining the Service and then consider different alternative implementations that all provide this Service. In this approach, the Service is a common abstraction of the different implementations, whereas the latter are different refinements of the same Service. This particular form of abstraction constitutes a cornerstone of our design methodology, since it allows a systematic selection and assessment of alternative implementations.

2.3 Design Language

To be able to *represent* properties of an object a model must always imply two essential but quite different characteristics: the model must *contain* these properties, and these properties must be *expressed* in some way. The latter is necessary to allow the user, e.g. a designer, a wind tunnel or a computer, to interpret the model for the purposes it is intended for. The mechanism to express something is generally called a *language*. Thus the properties of an object represented by a model are always expressed in some language.

Definition 2.5 Language

A *language* is an expressive means used to formulate expressions about certain items, such that these expressions can be conveyed to and interpreted by a person or a machine that understands this language.

The use of languages lies at the very basis of our society, and its cultural and artificial activities. Hence an expression in a language may have a large variety of names, e.g. a *statement*, a *declaration*, a *composition*, a *sentence*, etc. Correspondingly the items that are the subjects of the expression can be an enormous variety of things, such as a *message*, a *design*, the *smell of a flower*, the *behaviour of an animal*, a *solar system*, a *piece of music*, a *law*, a *doctor's prescription*, an *emotion*, etc.

In case we want to express a design, we use a *design language* to formulate an expression that we often call a (*design*) *description*, or a (*design*) *specification*, or in certain environments a *blueprint*.

In case we design an object on basis of a model, we express in the model the properties of the object in the design language that comes along with the model. This implies that the interpreters of a model, usually humans or special machines, must always look at the language expression to understand the properties of the object we have designed. The language expression acts as an intermediary, like a 'window', through which the interpreters virtually have to look at the object to understand its properties. This is illustrated in Fig. 2.4.

2.3.1 A Property and Its Expression Are Different Notions

The reader should be strongly aware that a property is quite a different notion than a language expression of the property.

Violets have the property that they can smell (nicely) and we can physically experience this property of smell, and even remember it in our minds, without the necessity to even express it in some language. This is quite different from the French expression 'l'odeur des violettes', which is just a collection of words consisting of letters, which do not smell.
The notion of a flat plane as a property that can be given to some object (e.g. a table), is quite different from the notion of the English expression 'flat plane'.
In a ballet, a certain body movement can be considered as a property of that ballet. This movement can be graphically expressed in a language specific for ballet movements. This graphical expression is quite different from the movement itself.
Tones and rhythms are properties in a piece of music. These are quite different from the musical notation that can express them. In fact, there are large numbers of people who can play music without mastering this musical notation.

Consequently, to retain a design of some pursued object that is conceived in the mind of a designer in terms of a set of properties requires in principle just the designer's mind. We simply do not know how properties are represented in a designer's mind and what happens if the designer thinks of a property; we simply say

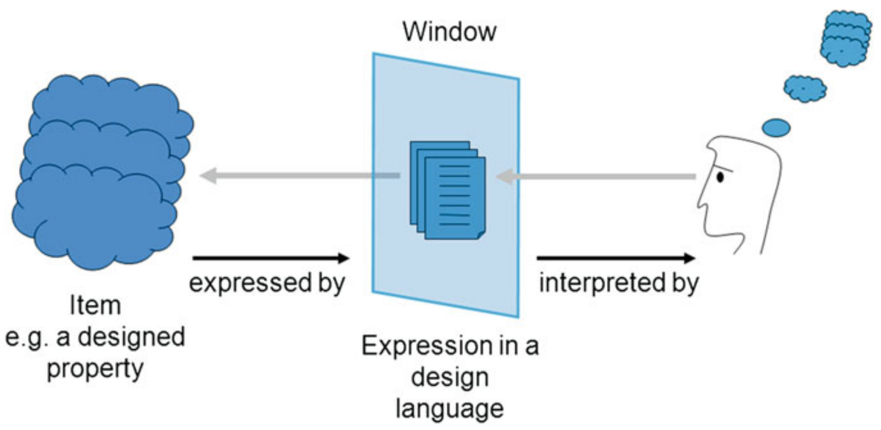


Fig. 2.4 A property and a language expression representing this property as two different notions

that he makes a mental image of the property. This mental image, however, cannot be directly communicated to users, nor to other designers or machines. Communication requires a more tangible form of representing the design, namely by expressing the properties in some language and modulating this expression on some medium or material. This tangible form may assume various appearances such as a *physical expression*, or a *symbolic expression* or even combinations thereof.

A scale replica of a house, or an airplane, expresses shape proportions physically by the form of the surface of the material used.
A message, written on paper, expresses something symbolically by the combination of letter symbols.
Flag signalling between ships combines physical and symbolic expression.

Normally, an expression can be conveyed by different media, such as the air (e.g. for voice), paper or some magnetic medium. The expression cannot be retained if the modulation is *volatile*, such as a sound through the air. To retain the expression, the modulation has to be *fixed*, i.e. stored or recorded on some medium. This allows one to *document* a design for the purpose of memorising, communicating or analysing it.

The communication of sound (e.g. the spoken word) and flag signals between ships is volatile because they are modulated on the air and the ether, respectively.

The communication of the blueprint of a house, when documented on paper or stored on a hard disc, can be retained because its modulation is fixed.

This means that the language expression can be documented. The property itself, though, cannot be documented, but is either the human interpretation (mental imagination) of the language expression when dealing with the model, or the human exposure to the real property when dealing with the real-world system.

The expression '*l'odeur des violettes*' can be documented and we can imagine, e.g. create a mental image of, the smell. However, the real physical smell can only be experienced by smelling real violets.

2.3.2 Language Alternatives

Properties of an object can be expressed in different languages.

The physical language used by the scale replica and the symbolic mathematical language used by the mathematical expression are two alternative ways to express the shape proportions of an airplane.

2.3.3 Natural and Artificial Languages

Languages should be suitable for the items they have to express. We therefore distinguish between *natural languages* and *artificial languages*.

Definition 2.6 Natural language

A *natural language* is a language that has grown in a community of natural creatures to support their communication.

Examples of natural languages are English, German, Dutch, the communication sounds of whales, the communication patterns of bees and the gesticulation language used by deaf and dumb.

Natural languages produced by humans are used for a large variety of purposes. Even designs are frequently described in a natural language, although such designs normally suffer a great deal of ambiguity. To avoid such ambiguities, artificial languages have been developed for specific purposes.

Definition 2.7 Artificial language
An *artificial language* is a language that is intentionally developed for and is thus particularly suitable for a specific category of purposes.

This implies that an artificial language is generally only suitable for this specific category of purposes.

Examples of artificial languages are Morse, the flag and light signalling languages between ships and the telegraph signals used by Napoleon.

When an artificial language is specifically developed to express designs we speak of a *design language*.

Examples of artificial design languages are the drawing language in Mechanical Engineering, chemical formulas, the ballet dancing language, Petri nets and the causality relations language treated in this book.

2.4 Design Model and Design Language Relationship

Our reasoning thus far leads us to the conclusion that if we want to design and build a more than elementary system, we better use a design model in which we represent the relevant properties of the system and express them in a suitable design language. If the system is complex, we do this at consecutive abstraction levels.

This implies that we have to assume (establish) an explicit relationship between the properties represented by the model and the language used to express these properties, so that we or others can understand these properties later again by unambiguously interpreting the expression. This leads us to two questions: What are the properties we want to express? How should we express them? We deal with these questions one at a time.

2.4.1 Design Concepts

In a design model we want to represent properties that we want to give to real-world objects. Properties of real-world objects, though, can come in many varieties.

In case of the airplane model, we want to represent the shape proportions of the wings, the body, the tail fin and the horizontal stabiliser of the planned real airplane.
In case of a house, we want to represent the shape and relative proportions and locations of rooms, walls, roof(s), windows, doors, etc.

This implies that the properties represented in a design model cannot be chosen arbitrarily, but have to be chosen such that they reflect the essential properties of the real-world systems, and the relationships between them, in a restricted and well-defined area of concern that forms the application domain of the design model.

The observation above implies that design models for airplanes and houses differ significantly in the properties they represent.

In imagining and conceiving a design model, designers generally think in terms of elementary properties from which they can compose the model. This implies also that more sophisticated properties can be represented by a composition of more elementary properties. Such composable elementary properties thus should embody essential elementary properties of real-world objects (or processes) in the application domain. These composable elementary properties, must be imaginable and conceivable by humans, and can therefore be considered as conceptual building blocks (bricks) for a model. We henceforth call them *elementary design concepts*, the smallest building blocks of designs.

The shape of a real-world object can be conceived in terms of a composition of specific surfaces, such as flat planes, cylinders, spheres, hyperboloids, cones, cubes, etc. These specific surfaces can be easily imagined by the designer and used as elementary design concepts.
A real-world (discrete) process can be conceived in terms of individual actions and their relationships, where in each action something is established. Actions and their relationships can be used as elementary design concepts.

We want such design concepts also to be *general purpose*, such that they can be used to represent essential properties, and their relationships, that are broadly and frequently found in real systems in the application domain [26, 31–35].

Definition 2.8 Elementary design concept

An *elementary design concept* is a general purpose concept derived from and capable of representing an essential property of real-world systems in the area of concern, which a designer can use as a basic building brick in imagining, conceiving and constructing a design model.

Alternatively we use the terms *architectural concepts* and *implementation concepts* to denote general purpose concepts that can be used to compose architectures and implementations, respectively.

We further want to dispose of a *complete set* of consistently related elementary design concepts, so that we can represent, either directly or as a composition, any essential property that we want to give to real-world systems in our application domain.

In the case of a house, we compose a design from mental images of walls, floors, doors, windows, etc. These mental images are our elementary design concepts with which we can design all kinds of different houses by linking walls to floors, windows and doors to walls, etc.

A design model is an application of elementary design concepts by selecting them and putting them together in a certain composition. Consequently, a design in its pure mental form, i.e. when it is not concerned with the way in which the design concepts are expressed in a language, is a conceptual model or mental image of a system. This conceptual model defines the properties of the system that are relevant to the designer, but does not express these properties in any symbolic or physical form.

Interpretations of the term 'model'

The reader should be aware that the term model is used in the literature in different ways, such as:

1. To denote a (complete) set of elementary design concepts, which we also call a *metamodel.*
2. To denote an application of a set of elementary design concepts, such as used by us to compose a design model, and which we call a design model.
3. To denote the set of all possible design models (the set of all possible applications of a set of elementary design concepts). This is often used in formal theories.

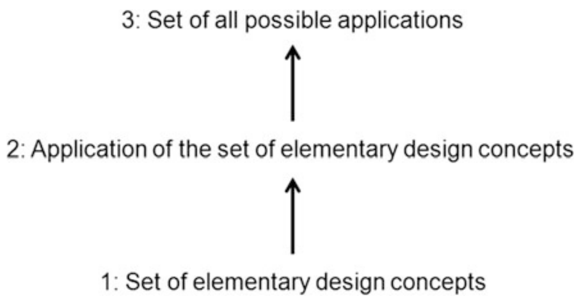

3: Set of all possible applications
2: Application of the set of elementary design concepts
1: Set of elementary design concepts

Fig. 2.5 Three different but related interpretations of the term 'model'

These interpretations of the term model form a hierarchy, as shown in Fig. 2.5.

2.4.2 Broad Spectrum Elementary Design Concepts

Complex systems generally are conceived at different abstraction levels. This implies that the design concepts that are generally applicable within an application domain should in particular be generally applicable at the different abstraction levels in the design process, i.e. they should be applicable in multiple subsequent design steps.

Consider the business process (re-)engineering application domain, in which business processes consist of activities that are performed by business organisations. These activities can be administrative, manufacturing or logistic processes. Furthermore, activities can be identified at different levels of granularity. For example, a logistic process as a whole can be considered as a single activity, or it can be considered as a collection of many related (sub-)activities, such as accepting orders, assigning carriages to trucks, calculating route tables, and preparing waybills. An elementary design concept suitable for the modelling of all these activities can be considered as a general purpose elementary design concept in the area of business process (re-)design. The action concept introduced in Chap. 3 fulfils this requirement.

The applicability of a limited set of general purpose elementary design concepts over a large part of the design process gives these concepts a broad scope. In this case we speak of a set of *broad spectrum elementary design concepts*.

The motivation for using broad spectrum elementary design concepts is to improve the efficiency and clarity of the design process through repeated applicability of the same elementary concepts to the functional definition of a system and system parts at various abstraction levels. Further, the number of concepts needed to model a system is smaller when using broad spectrum elementary concepts than when using specific concepts.

The use of broad spectrum elementary design concepts also facilitates the comparison, and eventually the conformance proof of designs at different abstraction levels, since these designs can be constructed as compositions of the same elementary design concepts.

Compositions of broad spectrum elementary concepts can be used to model many system properties, including specific ones, through proper combination, instantiation and parameterisation. Specific concepts, though, can only be used to model a few specific properties. The application of a limited number of broad spectrum elementary design concepts that can be used throughout a large part of the design process, finally, may facilitate their use, and reduce the learning effort.

In practice though, design concepts have a limited scope since they are often suitable for only a restricted number of consecutive design steps. When going beyond this scope, either going to finer or coarser levels of granularity, concepts may lose their significance. This implies that we often have to use multiple sets of elementary design concepts.

At the Service level, the concepts should preferably be capable of representing properties that are close to the user's perception of the system. At the level of the final implementation, though, the concepts should preferably be capable of representing properties of the physical or logical components that eventually implement and realise the system.

A programming language allows a programmer to define a final implementation in terms of programming language statements, ultimately meant to the interpreted and executed by a (virtual) machine. Understanding the externally observable functions of a system in terms of a program, however, is quite difficult and inappropriate for many end users. For this purpose, one needs a high-level specification language.

2.4.3 Language Elements for Design Concepts

We mentioned earlier that a design has to be expressed in some way by modulating it on some medium if we want to communicate it, document it or work on it with tools. This implies that we need a language to express our design concepts.

In its most simple form, a design language is a set of notational elements (symbols), one for each elementary design concept. The expression of a design is then the composition of notational elements that reflect the composition of elementary design concepts.

Figure 2.6 shows the relationships between design concepts, design, design language and design specification.

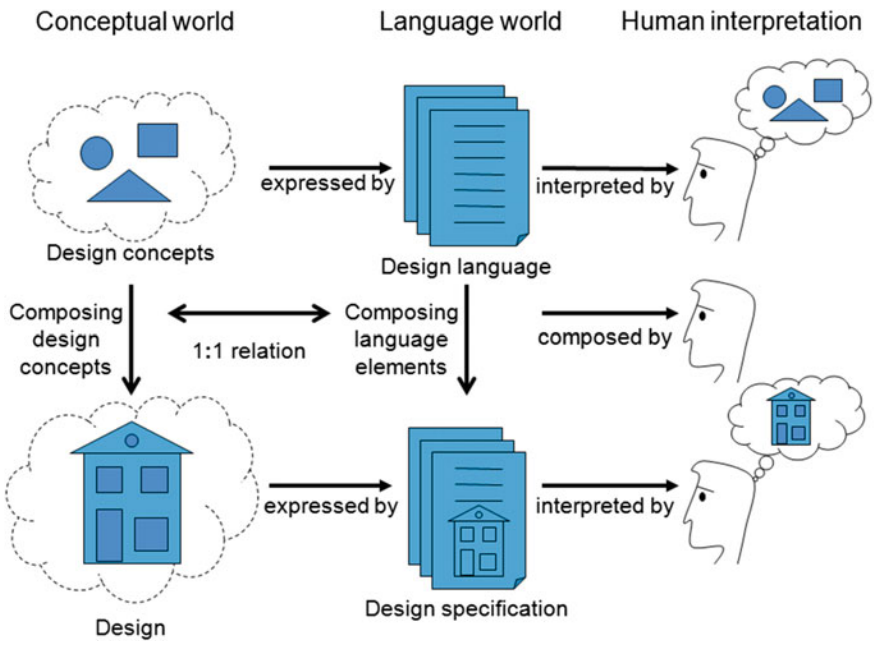


Fig. 2.6 Relationships between design concepts, design, design language and design specification

The design of a house is generally expressed as a paper document elaborated according to drawing conventions. This enables the architect, the builder and the future owner of the house to understand where the walls, the doors and windows are going to be placed. Alternatively, the design of a house may be expressed as a scale replica using a physical expression, in which walls, doors and windows are expressed by means of, e.g. Lego-like building blocks.

2.4.4 Characteristics of Design Languages

Characteristics of a design language, such as its *syntax* (symbols and combination rules) and *semantics* (meaning), must be derived from the relevant general purpose design concepts of the application domain and their relationships. This makes a design language general purpose in that domain.

Mental images of walls, doors and windows are elementary design concepts that can be expressed as drawing symbols. These symbols can be combined in the drawing of different houses and hence form a general purpose drawing language for houses and buildings. We suppose that the materials used to build walls, doors and windows, the colour of the walls, etc. have been ignored to make the concepts generally applicable, and thus are not expressed in this language.

We call an expression in a design language a *design specification* or simply a *specification*. Consistently, we call a design language also a *specification language*. A design language is used to express design models with a *prescriptive* character, i.e. the real-world object has to be constructed as *prescribed* by the language expression. In this context, instead of the term specification we may also use the term *prescription*.

The construction drawing of a house forms a specification, since it prescribes to the builder which properties the house should have.

The suitability of a specification to express a design depends on how faithfully the specification language is capable of expressing the design concepts used to develop the design [36]. The semantics of all language constructs and operators, and their possible compositions, should be defined precisely in terms of the design concepts they express. This should guarantee the unambiguous interpretation of a specification, i.e. only one interpretation of the specification should be possible. Unambiguous interpretation, i.e. precise semantics, enables the construction of automated (software) tools that support design and specification.

In the past, we have worked intensively on several software tools that support design and specification expressed in a formal language. We mention the developments related to LOTOS [37–39], AMBER [40], ISDL [41–44] and COSMO [45–47].

The development of software support tools, though, is a large field and deserves attention of its own. For these reasons we restrict ourselves in this book to design and specification issues and do not address tool support.

We often use the term *architectural semantics* to denote the unambiguous interpretation of language elements in terms of architectural design concepts [48–50].

The degree of precision or unambiguity of a specification language can sometimes be enhanced by defining mappings between this language and mathematical constructs or formula. Languages that have these mappings are usually called *formal specification languages* or *Formal Description Techniques* (*FDTs*).

The formal model of an aeroplane can be defined using mathematical formulas to express the shape proportions. In this way it is possible to evaluate the aerodynamic properties of the aeroplane by executing a program that evaluates these formulas.

The past has shown a rich history of approaches to develop adequate FDTs for the abstract implementation-independent representation of designs. Prominent examples are (Extended) Finite State Machines [(E)FSMs], Petri Nets, System Description Language (SDL) [51], Communicating Sequential Processes (CSP) [52], Temporal Logic [53], Calculus of Communicating Systems (CCS) [54], Language for Temporal Ordering Specification (LOTOS) [55], Estelle [56] and (to some extent) Unified Modelling Language (UML) [57]. In many cases, these approaches were dominated by the desire to underpin the language by a formal, mathematical based theory, combined with a lack of insight in proper design concepts. This often caused a choice for language constructs that indeed suit the theory, but that have no or only marginal significance for practical engineering. Consequently, most of these FDTs have found no or only marginal application in real practice. For an in depth treatment of specific FDTs or formal modelling of specific aspects or types of distributed systems we refer to [58–62].

2.4.5 Specification Versus Description

Instead of the term specification we often also find the term *description*. This may cause confusion since the term description is used associated to the representation of something that has been *observed*. For example, one can give an eyewitness description of an accident, and we would not be inclined to call this an eyewitness specification, let alone that we do not want to prescribe an accident.

Suppose you want to observe an existing system, i.e. a system that has been built, for example, for the purpose of testing its behaviour. In this case you want to describe the observed behaviour of the system and compare it with the prescribed behaviour according to the design specification. If the design specification has been implemented and built faithfully, the observed behaviour is *included* in the prescribed behaviour since the latter defines (prescribes) all possible behaviour.

A system generally does not exhibit all its possible behaviour when observed, even if the observation time is very long. So, observed behaviour is generally not sufficient to exhibit all possible behaviour. For example, after repeatedly observing the sequence of the colours red, orange and green of a traffic light, one can never be sure that this is the only possible sequence, and the system cannot suddenly start blinking orange under some specific circumstances.

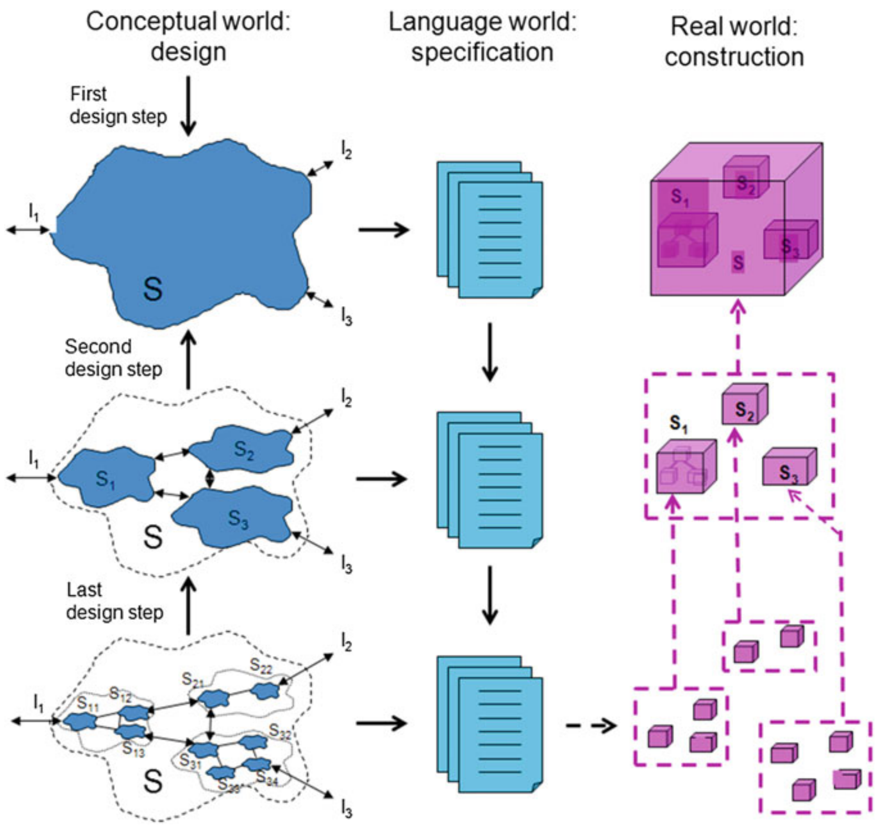


Fig. 2.7 The conceptual world of the (top-down) design of a system, the language world to document this design, and the real world of the (bottom-up) construction of the system

Therefore, description and specification are different notions and we will use the term description only to denote the documentation of something that is observed and not for documentation of a design.

2.5 System Design

Continuing the discussion in Sect. 1.7, we may conclude from Chap. 1 and this chapter that to perform the design process effectively, the designer should consider three worlds at the same time: the world of the real object that has to be designed and constructed, the world of the conception of this real object and the world of the language in which this conception has to be formulated. This is illustrated in Fig. 2.7, which reflects also the top-down design process and the bottom-up design construction process that was shown in Fig. 1.11, but now including the design documentation (in a language) process.

To be successful, the designer has to meet a number of challenges

- Understand the essential properties of the real-world object that has to be designed.
- Understand the constraints that have to be obeyed by the design process.
- Choose the right abstraction levels at which a system should be designed.
- Choose the right set of design concepts (the design model) to be used at each abstraction level.
- Choose the right structure (composition of parts) at each abstraction level.
- Choose the right functionality (Service) of each part at each abstraction level.
- Choose the right relationships between parts at each abstraction level.
- Choose the right specification language at each abstraction level.
- Express (specify) these functions and relationships precisely, unambiguously and clearly.
- Establish the conformance relationships between designs at various abstraction levels.

A *design methodology* can help a designer meet such challenges by providing appropriate methods and techniques.

2.6 General Purpose Languages and UML

The reader may wonder at this point why in this book we introduce a set of design concepts and a language to represent them, instead of using an available general purpose language, from which UML (Unified Modelling Language [57]) is the most popular representative.

UML has been originally developed to allow the abstract representation of different aspects of software systems (static structures, behaviours, functions, etc.), but its application has been stretched along the years to also cover higher level design (i.e. conceptual modelling). Although it has not been originally developed for this purpose, UML is therefore currently being used as a *broad spectrum language*.

UML has brought quite a lot of benefits in a time when each software designer had his own particular notation to represent software systems, by introducing a (more or less) generally understood notation for this purpose. However, the use of UML beyond its original intent has exposed some of its limitations, like the lack of formal semantics, its limited expressiveness and its multitude of poorly related diagrams (model representations), to name just a few [29, 36, 63, 64].

These limitations have made UML unsuitable for the purpose of this book. If we had used UML to represent our design concepts in this book, most of the time we would have to explain that the UML specifications do not exactly represent what we wanted to represent, and how these UML specifications should be interpreted. However, we understand that a designer may devise a way to cope with UML's limitations, and that he may decide to use UML as communication vehicle for designs conceived using our design concepts (at his own risk).