Accepted version of the paper: https://doi.org/10.1002/sys.21547

A System Concept Representation Framework and its Testing on Patents, Urban Architectural Patterns, and Software Patterns

Yaroslav Menshenin (corresponding author)
Skoltech Space Center, Skolkovo Institute of Science and Technology, Bolshoy Boulevard 30, bld. 1, Moscow, Russia 121205, y.menshenin@skoltech.ru, +7 985 311 5294

Edward Crawley
Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 77 Massachusetts Avenue 33-409, Cambridge, MA, USA 02139, crawley@mit.edu, +1 617 230 6604

Abstract

The development of a concept for a system is a key step towards creating the system's architecture. Most previous concept development approaches focus on the procedures for the conceptual design activity - the sequence of activities and tasks. Our work is motivated by the desire to elaborate in details the notional content of a system concept and to provide the means of encoding and analyzing it in a digital environment. The objective of this work is to develop a *system concept representation framework* that can systematically represent the concept's constituents, their definitions and interconnections. In order to demonstrate the utility of this framework, we have conducted three studies: mapped eight selected US patents, nine selected urban architectural patterns, and three selected software patterns to the framework. Patents, urban architectural patterns, and software patterns each contain a rich body of knowledge about the system they describe, and therefore they must logically contain a description of the concepts underlying them. We show that the main features of proposed framework can be found in patents, urban architectural patterns, and software patterns. The major utility of the framework is that it provides the means to encode existing system concepts and to inform the conceptual design of new systems, contributing to the INCOSE Model-Based Conceptual Design initiative.

Keywords: model-based conceptual design, system concept, concept representation, knowledge management

1. Introduction

Conceptual design is one of the most challenging and crucial stages of the product development process. The importance of this phase has been widely highlighted in literature. For example, Kroll et al. mentioned that the conceptual design "significantly affects the product novelty, performance, robustness, development time, value, and cost".[1] Farid and Suh mention that in the conceptual design stage "…design decisions have the greatest impact on the project outcomes with respect to functionality, performance, appearance, costs and sustainability".[2] The objective of our work is to develop a system concept representation framework that represents the concept's constituents, their definitions and interconnections. Such a framework includes the key constituents of concept, and can be represented digitally, facilitating the conceptual design process. Such a model is representative of the concept and has potential utility. But like any model, is not "complete"

in that there is always a next layer of information that can be added. However, we believe there is clear interconnections of the information specified at conceptual design phase with information required at later stages of design process, thus the concept knowledge is not lost, but reused during the architecture development.

In this paper we describe the framework and map eight patents, nine urban architectural patterns, and three software patterns, representing a broad spectrum of engineering and urban problems, to the framework. Patents have been chosen as the unit of knowledge, as these are internationally recognized ways to protect an invention. Urban architectural patterns represent the knowledge about one of the basic human needs: how to build a town for citizens or a home of your own. Software patterns inform us about the fundamentals of coding approach. The selection of patents, urban patterns and software patterns was made because they represent three consistently structured, well documented, commonly used repositories of conceptual information that are fully open in the public domain.

The patents, urban architectural patterns and software patterns must logically contain a definition of the underlying concept included within them. Success in demonstrating that all these sources of knowledge can be mapped to the concept framework is therefore a necessary condition to demonstrating the utility of the framework.

The proposed framework has several forms of utility. First it lends structure. Often the information in documents specifying system concept is presented in an unstructured way, as a set of textual and graphical information. The proposed framework provides the means to *encode* this unstructured data into a structured set of both texts and models, which are consistent with each other. This would allow systems engineer to have a concept classification scheme and searchable database, documenting core information about a concept. This approach could be used within the INCOSE's Model-Based Conceptual Design initiative[3].

Another utility is that the proposed framework supports concept generation at early phases of the design process in a model-based environment. For example, it allows systematic combination of entries to produce new concepts. The proposed framework contains 28 entries and a modeling ontology. By extending and re-combining them, the systems engineer can develop novel concept, revealing alternative and previously unexplored concepts in a concise way. This utility could also be utilized within the INCOSE's Model-Based Conceptual Design initiative[3].

In addition to the two above-mentioned forms of utility, which we return to below, there is additional utility in the framework. The framework provides a methodology for the system engineer to measure the distance between concepts. This allows the identification of similarity between alternative concepts.

Finally, another additional form of utility is that using the framework leads seamlessly to architecture development. The approach leads to the definition of a concept as a subset of the information that will eventually be used to describe the architecture. Thus, the utility of the framework is that the concept knowledge is reused in later stages of the design process – during the architecture development.

Note that in this work the word ontology implies the entity that "defines the terms used to describe and represent an area of knowledge... Ontologies include computer-usable definitions of basic concepts in the domain and the relationships among them... They encode knowledge in a domain and also knowledge that spans domains. In this way, they make that knowledge reusable".[4]

1.1 Background and research opportunity

Concept is not a precisely defined idea. According to Crawley et al., concept is "a product or system vision, idea, notion, or mental image that maps function to form. It is a scheme for the system and how it works. It embodies a sense of how the system will function and an abstraction of the system form. It is a simplification of the system architecture that allows for high-level reasoning".[5] Another definition, proposed by Ulrich and Eppinger, states that a concept is "an approximate description of the technology, working principles, and form of the product. It is a concise description of how the product will satisfy the customer needs".[6] These two definitions have a number of features in common. We observe that the *concept* has a form, function, and initial hints about how the system works – the concept of operations. The form is the physical embodiment of the system, while the function outlines what processes the system is performing. These observations are a key starting point for this research, as they create an appropriate level of abstraction for the system concept representation framework. The definition of concept and related models by other authors will be discussed below.

According to INCOSE System Engineering Handbook, the purposes of the concept stage are to identify stakeholders' needs, explore concepts, and propose viable solutions.[7] The focus of the handbook is on what should be accomplished during this life cycle stage, as opposed to the methods and tools to be used. The proposed framework is a tool that might be considered by the INCOSE's Model-Based Conceptual Design (MBCD) working group[3] in their deliberation. Their focus is on the important task of the "... application of model-based systems engineering (MBSE) methodologies to the Exploratory Research and Concept stages of systems engineering".[3] Morris et al. conducted a survey for people involved in MBCD aiming to identify the issues associated with conceptual design phase. Their finding was that one of the core issues is the "lack of MBCD best practice examples".[8] In our work we propose to fulfill this need by applying an MBCD framework to patents, urban architectural patterns, and software patterns.

Concept is often considered within the context of procedures and steps of the design process. For example, Pahl and Beitz[9] proposed the steps in the planning and design process. According to them, conceptual design is performed after the planning and task clarification step. Pahl and Beitz noted that the role of the conceptual design phase is to define the principle solution. Ulrich and Eppinger have also focused on the presentation of sequence of activities and tasks in their generic product development process.[6] Andreasen et al. proposed the Concept Synthesis Model,[10] which consists of three major steps: goal formulation, ideation, and evaluation and choice. Although the output from these steps is a concept that is further used in the product synthesis, the nature of the description is still procedural. This creates a research opportunity to explore the notion of the content of concept itself. In this paper, we propose a framework that provides an explanation of what is inside the concept, what are its elements and intrinsic parts, and what are their formal and functional relationships. Additionally, we demonstrate how all this information can be supported by the model-based approach.

Hubka formulated the Theory of Technical Systems,[11] which has been extended by Andreasen into the Domain Theory.[12] This theory attracts our attention, as Andreasen proposed strict domains that are required for the design process. Each of these domains has its own "language", allowing the designer to "spell a product in different ways". Packing this theory into the Chromosome model[13-15] reveals the interconnectivity between entities of different domains. C-K theory is an approach aiming at explanation of concept emergence.[16] In this theory Hatchuel and Weil distinguish concept space and knowledge space. The

authors of C-K theory define a design as “the process by which a concept generates other concepts or is transformed into knowledge, i.e. propositions in the knowledge space”. These theories aim at specifying the concept more concretely, however they lack of consistent model-based approach that would allow prescribing not only "what" should be defined within the notion of concept, but also "how" this could be done.[17,18] Thus, there is a research opportunity to develop the model-based instruments that would allow encoding the existing concepts, and supporting a concepts generation process in early phase of systems engineering. Such instrument aimed at supporting systems engineers during the conceptual design is proposed in our paper.

Selva et al. formulated seven tasks of the system architect in regards to the decision-making problems (See Figure 1).[19] The framework proposed in our paper has a lot of common features with these seven tasks. Our framework specifies not only the information that should be entered into the framework, but also provides the modeling ontology. In addition it provides criteria (see Appendix A), which define the information that is to be included. Thus, the proposed framework could support the systems engineer in the decision-making process at the conceptual design phase.

Task	Consists of
1. **Decomposing**/Aggregating Form	Choosing a system decomposition, i.e. clustering elements of form.
2. **Mapping** Function to Form	Assigning elements of function to elements of form, or goals to functions
3. **Specializing** Form and Function	Choosing one among several alternatives for a certain element of Form or Function (e.g. going from solution-neutral to solution-specific)
4. **Characterizing** Form and Function	Choosing one among several alternatives for the attributes of an element of Form or Function
5. **Connecting** Form and Function	Defining system topology and interfaces
6. **Selecting** Goals and Functions	Defining scope by choosing among a set of candidate goals or functions
7. **Planning** system deployment and operations	Defining the sequence in which technologies will be infused into the project, system components will be deployed, or major operations will be conducted

Figure 1. Seven tasks of system architect[19]

Raz et al. proposed a system architecting and design space characterization process that is based on four artifacts of system architecting (see Figure 2).[20] They introduced a taxonomy that "emphasizes the dependence between the constituent elements of the system architecture process".[20] The five propositions of our framework contain the essential features of all four artifacts of system architecting process proposed by Raz et al. Our paper focuses not only on processes that should be performed by systems engineer during a conceptual design phase, but also on the modeling tools allowing to encode the information about concepts.

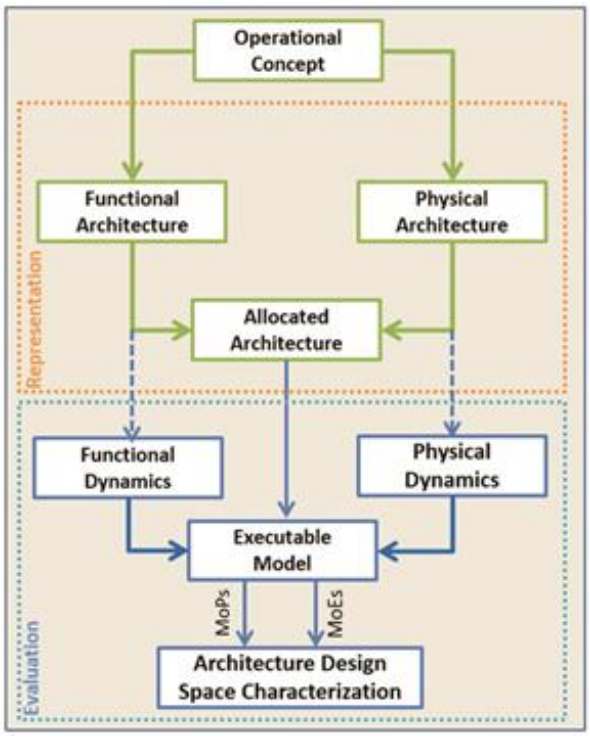


Figure 2. System architecting and design space characterization process[20]

The Web Ontology Language has been widely used in computer science to express a language suitable for application in the Semantic Web. OWL enables capabilities to "describe the semantics of knowledge in a machine-accessible way."[21] Our paper describes the elements of system concept representation framework, demonstrating a concept classification scheme and criteria for element's inclusion into the concept framework. The information is populated through a human reasoning, so there is a clear area of future work to engage a machine-accessible analysis enabled by OWL.[22] It is desirable that a concept framework be represented by one of several common modeling languages. For instance, the system modeling language (SysML) has a grammar of nine types of diagrams that support the specification, analysis, design, verification, and validation of systems. The Object-Process Methodology[23] (OPM) is a modeling approach that is able to represent the system graphically in a significantly smaller number of diagrams than SysML. Another advantage of OPM is that it has both linguistic and graphical representations, one of which can be generated from the other in OPCAT. [24] OPM is now specified in ISO 19450.[25] Although both conceptual modeling languages (SysML and OPM) could be used for the system concept representation framework, in our paper we will adopt OPM as the modeling language, because of the single diagram feature. We leave the implementation in SysML to future work.

1.2 Objective and overview

The specific objective of our paper is to develop a framework by defining the *core idea* of concept as form and function in the solution-specific environment (the solution statement). This *core idea* is surrounded upstream by information about the stakeholders and their needs, which then flow, still upstream, to the solution-neutral problem formulation (a functional requirement). Conceptual design is then the specialization of the solution-neutral to solution-specific domains. In addition to the stakeholder needs, solution-neutral problem formulation and solution-specific solution statements, the concept framework may also contain more detailed information about internal form and function, called the

integrated concept, as well as contextual elements and concept of operation. It is an objective of this research to demonstrate that the proposed framework could successfully be applied to represent conceptual information for a wide range of different types of systems.

Another objective of the paper is to validate the framework by mapping eight patents, nine urban architectural patterns, and three software patterns broadly representing the engineering and societal problems to the framework. The ability to map these sources of knowledge to the proposed framework is a necessary condition to show its utility for a broad spectrum of socio-technical systems. If successful, such a framework could be used for variety of purposes: encoding the core information about concepts; and generating concepts at early phases of the design process in a model-based environment, thus contributing to INCOSE Model-Based Conceptual Design initiative.[3] The additional value of the proposed framework is that it helps in measuring the distance between concepts, which allows the formal analysis, such as identification of similarity between alternative concepts. Another utility of the framework is that the concept knowledge is reused in later stages of the design process – during the architecture development.

In the following section we present the five propositions of the proposed framework, as well as introduce the framework, defining concept and integrated concept and their constituents. We present the idea of the solution-neutral and solution-specific environments and demonstrate how the framework spans both environments. After a brief discussion of the utility and structure of patents, urban architectural patterns, and software patterns, we show the methodology for mapping these sources of knowledge to the framework. After this we demonstrate, through a small-N analysis, that there is a common scheme for how information is presented in patents, urban architectural patterns, and software patterns, and that the framework can capture the conceptual information in them. Appendix A contains the core definitions of the concept framework, as well as the criteria for including an entry. Appendix B provides the detailed explanation of how the analysis has been performed based on an example patent, and demonstrating the model-based representation of patent. Appendix C demonstrates an example of urban architectural pattern, and Appendix D – an example of software pattern.

2. System concept representation framework

The proposed system concept representation framework is built upon five propositions that are shown in Figure 3. They will eventually lead to the 28 entries of the framework (Figure 6).

In this section we will introduce the canonical representation of a system, and will use it to represent conceptual design and concept framework. The framework contains information potentially necessary to define concept of a system.

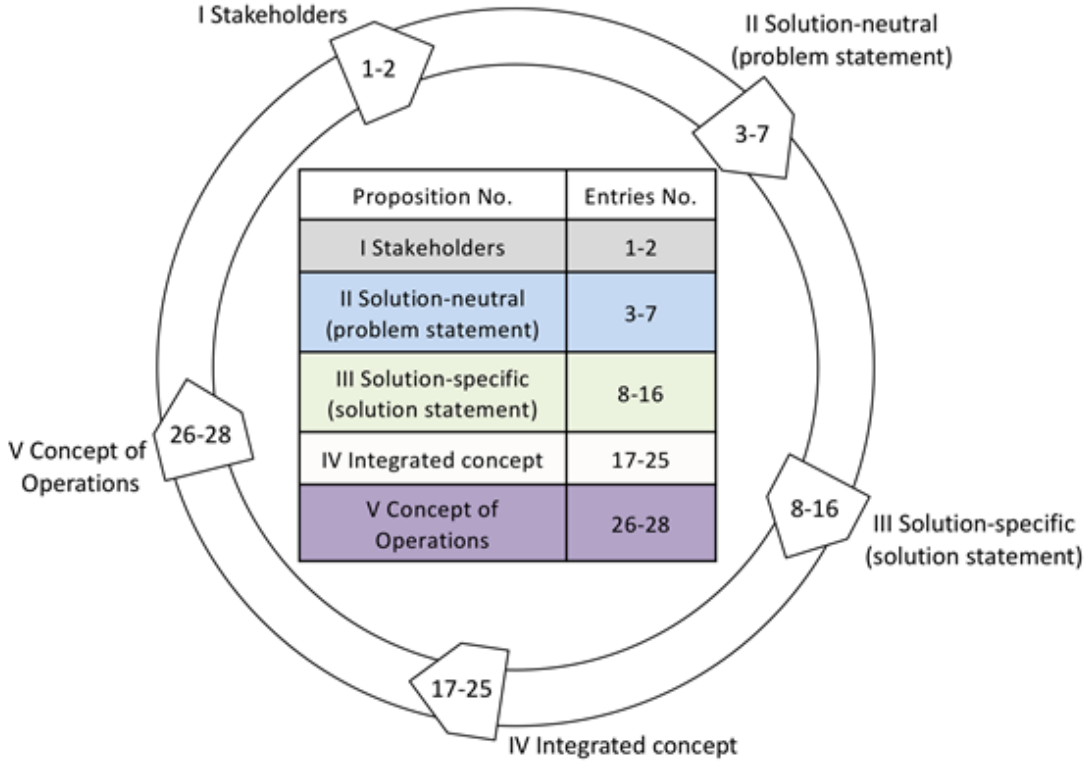


Figure 3. Five propositions of system concept representation framework

2.1 Representation of conceptual design

Any system can be described by means of three main entities: the *form* (the instrument object), a *process* (a transformation), and an *operand* (an object that is changed by the process). Together process and operand are the *function* of the system. The canonical representation of a system is shown in Figure 4A using objects and processes, as the name Object-Process Methodology implies.[26,27] Note that relationship between the operand and process is shown by a closed-headed arrow, while the instrumental relationship between form and process is represented by a circular-headed "arrow". These arrows are inherited from OPM notation.[23]

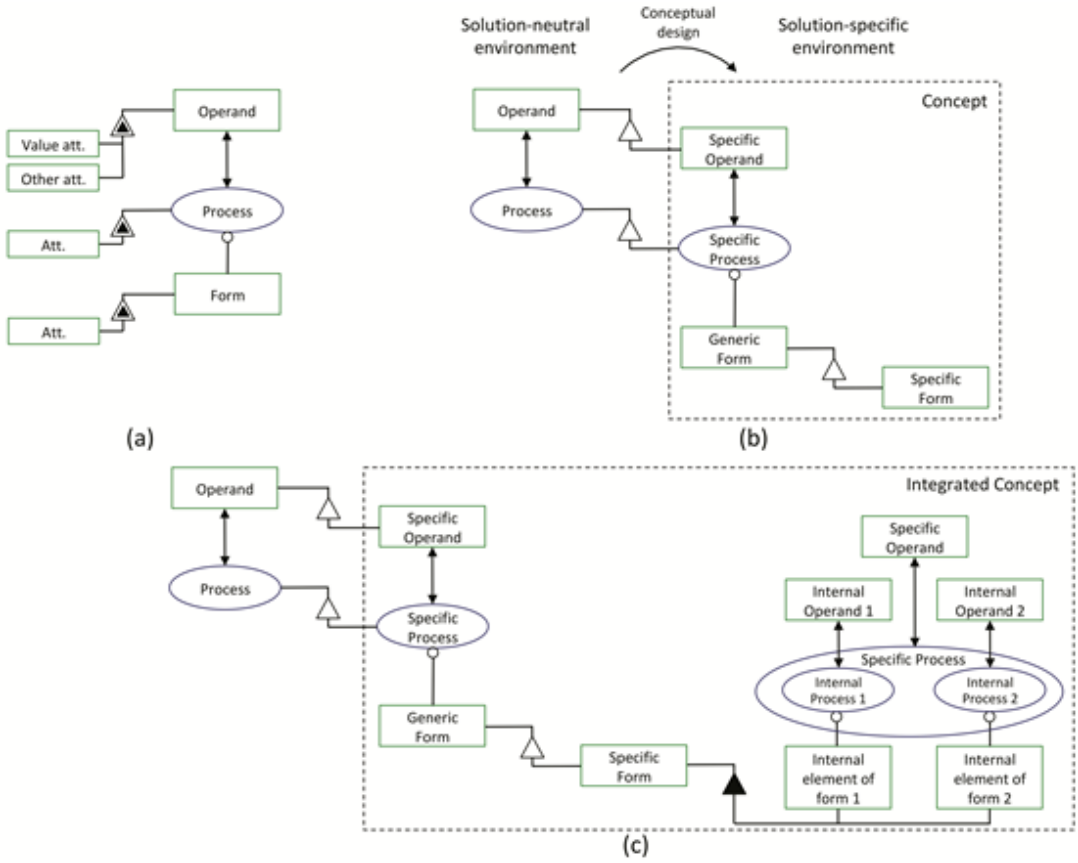


Figure 4(a). Canonical representation of system showing objects, processes, links and attributes; 4(b). Representations of conceptual design; 4(c). Conceptual design extended to include the integrated concept

The operand might have value-related *attributes* or other attributes, which describe the object, and are indicated by the partially filled triangle. Generally, value is created when the value-related attribute of the operand is changed. We will use this canonical model to explore concept.

Figure 4B represents a definition of the *concept* of a system[5]. The concept contains a specific operand, specific process and specific form. It exists in the solution-specific domain and conveys the solution. (Note that on Figures 4B and 4C the attributes have been suppressed for clarity). In the solution-neutral domain on the left of Figure 4B, we also express the solution-neutral problem function (or functional requirement) by means of an operand and process.

Conceptual design then involves a specialization from solution-neutral environment to solution-specific environment, as it is denoted in Figure 4. This relationship implies that all or some of the entries of solution-neutral environment take on new values in the solution-specific environment. The most obvious option is that the solution-neutral operand specializes to the solution-specific operand; and likewise for the processes. as is illustrated in Figure 4. However, sometimes only one entry is specialized (an operand or a process). Sometimes only an attribute of an operand or process is altered in specialization. In other words, all entries do not necessarily specialize during the conceptual design process, but some of them are altered. We will return to this subtle point below.

Figure 4C shows an extension of concept by one level of decomposition to reveal an *integrated concept*, which inherits all the features of the concept for the system. This provides for the possibility that essential information about the concept might be hidden one

level of decomposition[28] lower than the concept itself. The form is decomposed into the internal elements of form – decomposition is indicated by the filled triangle. These elements work as instruments of internal processes potentially acting on the internal operands. Reading Figure 4(c) in reverse demonstrates *emergence* – the internal processes act on the internal operands in such a way as the specific process emerges working on the specific operand.

2.2 An example of conceptual design

In order to demonstrate the process of conceptual design, we will use the example of an air transportation service. The first step in using the proposed concept framework is actually upstream of Figure 4C - the identification of stakeholders and their needs. Let us assume the stakeholders are members of the public, and their need is to visit family in a distant location. Once the stakeholders and their needs are defined, these needs should be translated into functional goals in the solution-neutral environment. This environment contains the solution-neutral process and solution-neutral operand, but does not contain the instrument of execution.

Applying the template of Figure 4B, we would obtain "traveler" as the solution-neutral operand, "moving" as the solution-neutral process, and therefore "moving traveler" as the solution-neutral function. Together with the attributes, the solution-neutral function is "safely moving a traveler of mass 70 kg from one to another location" as shown in Figure 5.

Our next step is to fill in the integrated concept of Figure 4C. In this case, the specific operand is the "passenger" specialized from the solution-neutral one – the traveler. A traveler could be specialized to other specific operands which imply a more active role, such as "walker", or "biker". The solution-neutral process "moving" could also be specialized to a variety of specific processes, such as "flying", "rolling", or "floating". Applying a rigorous approach to alternative concepts generation the systems engineer can keep track of different alternative operands/processes that satisfy the stakeholder's need. Ultimately, these different operands/processes would lead to different system concepts. Thus, the value of the approach is in the ability to keep track of and to encode multiple concepts that can be used to meet the abstract goal that is getting more concrete as the design process matures to concept.

We chose the "flying" specific process as indicated in Figure 5. Next the generic and specific forms are specified respectively as "a flying device" and a "jet aircraft". Other options for specific form might include rockets, helicopters and blimps.

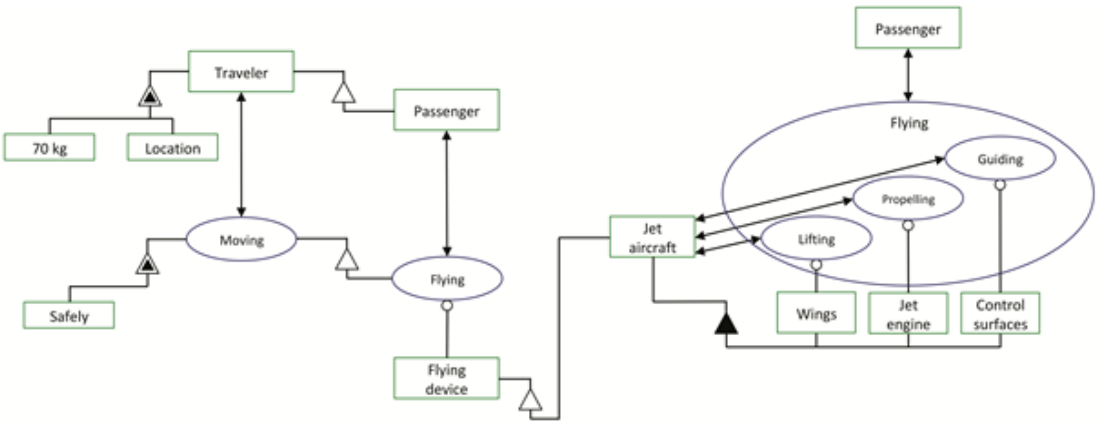


Figure 5. Integrated concept framework for air transportation

By specializing the operand from “traveler” to “passenger”, specializing the process from “moving” to “flying”, we are executing conceptual design. In other words, we are crossing that imaginary barrier between solution-neutral and solution-specific environments. The outcome of such specialization is that once we know the solution-specific function, we are able to assign the generic form to it. There is also a specialization link at the level of forms: “flying device” could be specialized to a number of specific forms, and “jet aircraft” is one of them. This example illustrates conceptual design as the specialization from solution-neutral to solution-specific environments, and the extension of concept by one level to reveal something about internal working of the system.

“Flying” is a rich concept, and in order to explain this concept, it might be necessary to reveal three internal processes: lifting, propelling, and guiding. The instruments of these processes are wings, jet engine, and control surfaces, correspondingly. Note that in this particular case the specific form is the jet aircraft itself. But the emergent function, which encompasses all three internal processes, is still “flying the passenger”.

We deliberately presented the obvious case of the operand and process each individually and directly taking on new values during specialization. There are more subtle options as well (which should not be a surprise to anyone knowledgeable in system design). Other options include:

- The process changes but the operand doesn't change so we have “traveler moving” specializing to “traveler flying”, which might lead to an active role for the traveler, such as the traveler piloting an aircraft. It’s also possible that the operand changes but the process doesn't;
- The attributes of the operand (or process) are all that change – a “traveler” specializes to an “injured traveler” which might lead to an air ambulance;
- The relationship between the solution-neutral operand plus process and solution-specific operand plus process is one of specialization, but in a collective and not in a one-to-one way. We might have “protecting a city” specialize to “building a wall”. Protecting does not directly specialize to building, and a city does not specialize to a wall, but protecting a city as a whole does specialize to building a wall.

It is possible that there are a comprehensive set of sub-frameworks that would capture all of these cases, but this is beyond the scope of the current work.

2.3 The entire system concept representation framework

It is our observation that the framework needed to specify a concept includes: the information about the stakeholders and their needs (upstream); all of the information on Figure 4C: the solution-neutral problem statement of function, plus the solution-specific integrated concept; structure and interactions; and the concept of operations. In addition, it implicitly contains the attributes of the entities that are operands, processes, and forms.

Figure 6 shows the 28 entries spread among 5 propositions of the framework. We observe that these 28 entries contain information that is commonly used to describe a concept. Note that the detailed definitions of each entry of the proposed framework are contained in Appendix A.

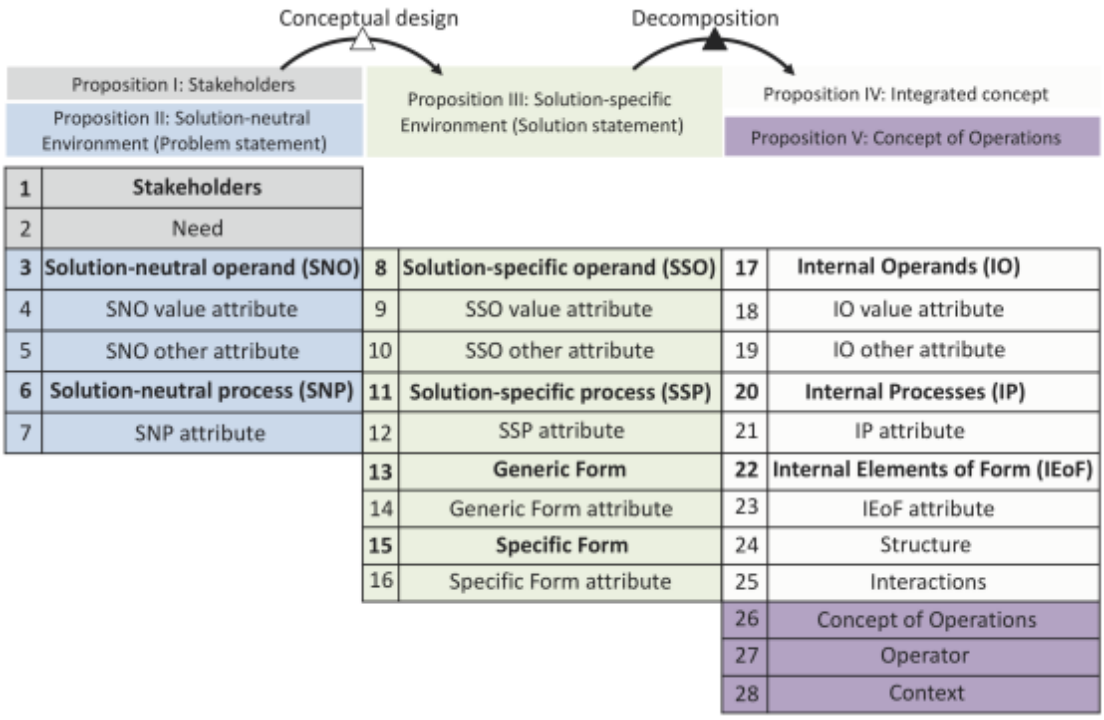


Figure 6. The 28 entries in system concept representation framework

3. Patents, Urban Architectural Patterns, and Software Patterns

In this section we provide a brief overview of patents (sub-section 3.1), urban architectural patterns (sub-section 3.2), and software patterns (sub-section 3.3). A short introduction to each one of these sources of knowledge is provided and the structures of them are explained. This overview informs us that each one of these sources has a rich body of knowledge contained in them, and therefore they must logically contain a description of the concepts underlying them.

3.1 Patents

Patents are one of the most common ways to protect the rights of an inventor. According to the United States Patent and Trademark Office (USPTO), a *patent* is "a property right granted by the Government of the United States of America to an inventor 'to exclude others from making, using, offering for sale, or selling the invention throughout the United States or importing the invention into the United States' for a limited time, in exchange for public disclosure of the invention when the patent is granted".[29] Note that the patent doesn't provide a right to make, use, offer for sale, sell or import, but provides the right to exclude others from making, using, offering for sale, selling or importing the invention.

Patents also have utility to other users. In accordance with the World Intellectual Property Organization (WIPO), "it is estimated that some 70% of the information disclosed in patent documents have never been published anywhere else".[30]

From the perspective of this study, patents represent a large database of publically accessible documents, describing new inventions in some detail. The patents must logically describe the concept underlying or contained within the invention. Success in demonstrating that the patent can be mapped to the concept framework is therefore a necessary condition to demonstrating the utility of the framework.

US patents are structured according to internationally agreed standards. The information in a patent is presented as a combination of structured and unstructured data. The structured data includes the template of a patent while the unstructured data includes the text.

From the formal point of view, we may highlight that the patents consist of three main parts: the abstract in the front page, the description of the invention, and the claims. The roles of these parts are:

- The abstract at the first page of the patent briefly explains the core of the invention and the elements of form;
- The background of the invention outlines the technical field of the invention. It also contains a detailed description of the invention, including its form and function; it also provides information about the concept of operations of the system;
- The claims describe the scope of the invention and the technical features of it. The claims identify the features of the invention that are distinct from all previous inventions. The claims section is the part of the patent that has a legal importance.

Patents can be issued for systems (machines), methods (processes, acts), and the composition of matter (chemical compounds, chemical compositions).

3.2 Urban Architectural Patterns

In his book that became widely recognized, Christopher Alexander had explored the pattern language of the architecture.[31] Alexander provided the following description of the pattern: "Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing in the same way twice". Alexander developed the set of 253 patterns each one of which belongs to one of the three types of urban architectural patterns: towns (94 patterns), buildings (110 patterns), and construction (49 patterns). He claimed that the design methods should have the impact on real world, rather than serve the theoretical purposes.[32] The merit of Alexander's work is that he observed existing architecture looking for patterns in it and formulating the recommendations how to build your own architecture. The general elements of the urban architectural pattern are presented in Table 1.

Table 1. The general elements of urban architectural pattern (in chronological order as they appear while reading a pattern)

Element of pattern	Description of the element
Name of the pattern	A short name of the architectural pattern
Picture	Shows an archetypal example of that pattern
Introductory paragraph	Sets the context for the pattern
Essence of the problem	One-two sentences explaining the essence of the problem
Body of the problem	A longer section elaborating on the body of the problem
Solution	The solution accompanied by the diagram contains the proposed means to solve a problem
Conclusion	A paragraph that ties that pattern to all subsequent patterns in the language

The example of urban architectural pattern is demonstrated in Figure 7. Note that the presented pattern contains 6 pages, and we only present one page in Figure 7. The presented page covers the following elements from Table 1: introductory paragraph (a paragraph at

the top of Figure 7), the essence of the problem (a sentence in bold), and the beginning of the body of the problem (the text after the essence of the problem).

. . . within an urban area, the density of building fluctuates. It will, in general, be rather higher toward the center and lower toward the edges—CITY COUNTRY FINGERS (3), LACE OF COUNTRY STREETS (5), MAGIC OF THE CITY (10). However, throughout the city, even at its densest points, there are strong human reasons to subject all buildings to height restrictions.

✣ ✣ ✣

There is abundant evidence to show that high buildings make people crazy.

High buildings have no genuine advantages, except in speculative gains for banks and land owners. They are not cheaper, they do not help create open space, they destroy the townscape, they destroy social life, they promote crime, they make life difficult for children, they are expensive to maintain, they wreck the open spaces near them, and they damage light and air and view. But quite apart from all of this, which shows that they aren't very sensible, empirical evidence shows that they can actually damage people's minds and feelings.

"The Ministry of Truth—Minitrue, in Newspeak—was startlingly different from any other object in sight. It was an enormous pyramidal structure of glittering white concrete, soaring up terrace after terrace 300 metres in the air." (*George Orwell, 1984*)

Figure 7. Example of urban architectural pattern (Excerpt from №21 "Four-Story Limit"[31] pattern)

Urban architectural patterns have utility both for amateurs and professionals. Using the knowledge and observations gathered by Alexander and summarized in his patterns, the one can, for example, build his own house or to improve the environment in his or her community.

Thus, in our study we consider the urban architectural patterns as the relevant and rich source of knowledge that should describe the architecture in some detail. Christopher Alexander's patterns should depict the concept underlying the principles of our work on the issues related to the towns, buildings, and construction.

In sub-sections 2.1 and 2.2 we mentioned that the specialization process is closely associated with conceptual design. For urban architectural patterns, architect Christopher Alexander had developed highly abstract patterns that deal with problems occurring in

urban or rural environment. Taking into account the nature of explored issues, it should be noted that in some cases (as it is presented in Appendix C) the solution-neutral environment contains a problem that should be solved (for example, a problem is "destroying townscape"). It order to solve this problem, he proposes such solution in solution-specific domain as "keeping the building's limit stories in urban area". This is an example of a case where the operand is altered ("townscape" to "urban area") and the process is altered (from "destroying" to "keeping a limit"). As was discussed above, this change in the process is not one-to-one. But collectively a problem of destroying a landscape does specialize to a solution of limiting height. We discuss this case in details in Appendix C.

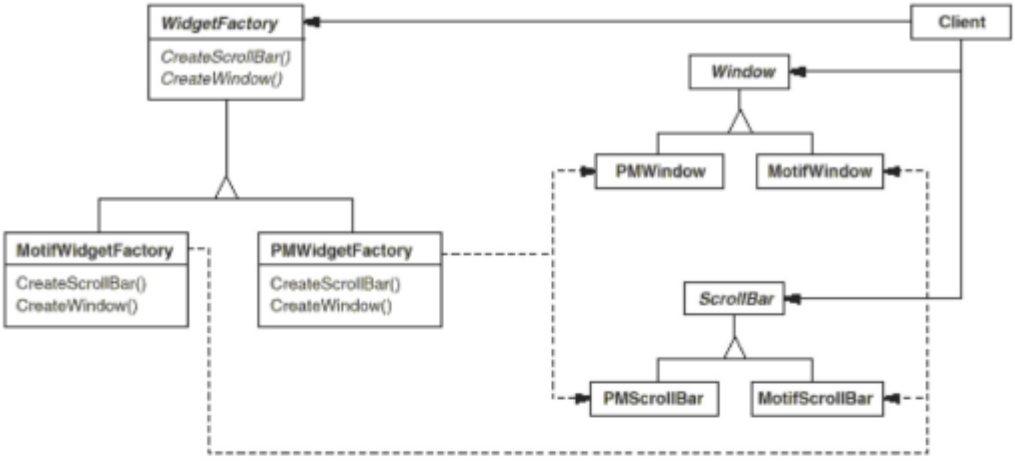
3.3 Software Patterns

In the book entitled "Design Patterns: Elements of Reusable Object-Oriented Software" the 23 patterns are discussed.[33] The authors of the book have taken the Alexander's idea to define a universal pattern language and have applied it to software. These patterns represent the solutions to specific problems in the object-oriented software design. According to Gamma et al., the software patterns "… are descriptions of communication objects and classes that are customized to solve a general design problem in a particular context".[33] Each one of the patterns is related to one of the three different types, namely, the creational patterns (5 patterns), the structural patterns (7 patterns), and the behavioral patterns (11 patterns). The creational patterns are related to the process of object creation. The structural patterns deal with the composition of classes or objects. The behavioral patterns "characterize the ways in which classes or objects interact and distribute responsibility".[33]

Table 2. The general elements of software pattern (in chronological order as they appear while reading a pattern)

Element of pattern	Description of the element
Pattern name	Describes a design problem, its solutions, and consequences in a word or two.
Problem	Describes when to apply the pattern. It explains the problem and its context.
Solution	Describes the elements that make up the design, and their relationships. The solution doesn't describe a particular concrete design, because a pattern is a template that can be applied in many different situations.
Consequences	Supports in evaluating design alternatives and to understand the costs and benefits of applying the pattern

The example of software pattern is demonstrated in Figure 8. Note that the presented pattern contains 11 pages, and we only present one page in Figure 8. The presented page covers the element "problem" from Table 2.



There is a concrete subclass of WidgetFactory for each look-and-feel standard. Each subclass implements the operations to create the appropriate widget for the look and feel. For example, the CreateScrollBar operation on the MotifWidgetFactory instantiates and returns a Motif scroll bar, while the corresponding operation on the PMWidgetFactory returns a scroll bar for Presentation Manager. Clients create widgets solely through the WidgetFactory interface and have no knowledge of the classes that implement widgets for a particular look and feel. In other words, clients only have to commit to an interface defined by an abstract class, not a particular concrete class.

A WidgetFactory also enforces dependencies between the concrete widget classes. A Motif scroll bar should be used with a Motif button and a Motif text editor, and that constraint is enforced automatically as a consequence of using a MotifWidgetFactory.

Applicability

Use the Abstract Factory pattern when

- a system should be independent of how its products are created, composed, and represented.
- a system should be configured with one of multiple families of products.
- a family of related product objects is designed to be used together, and you need to enforce this constraint.
- you want to provide a class library of products, and you want to reveal just their interfaces, not their implementations.

Figure 8. Example of software pattern (Excerpt from “Abstract Factory”[33] pattern)

The inclusion of software patterns into our study is relevant, as they represent the samples that have both features: the text of the patterns serves the purpose of facilitating the function's execution in a digital environment. Thus, there should be underlying principles that are highly related to conceptual framework proposed in this work.

4. Methodology of mapping patents, urban architectural patterns, and software patterns to the system concept representation framework

4.1 Sampling of patents, urban architectural patterns, and software patterns

This section describes the methodology of mapping patents, urban architectural patterns, and software patterns to proposed framework. The first step is to construct the set of samples.

For the purpose of selecting patents, we were guided by sampling techniques for "small N" qualitative studies.[34] By "small N" we imply the number of samples that is relatively small comparing to a whole population of samples (in this case, the entire amount of patents). We listed the "independent" variables appropriate for the purposes of our study. These "independent" variables focused on the types of systems and methods that are represented in patents. In order to test for broad applicability, we chose four quite different types of patents: biological, thermodynamic, electro-mechanical, and software. In addition, we wanted the set to include methods patents, systems patents, and some that were both method and system patents.

This yielded eight patents in the four types:

1. Vehicle mounted traffic light and system[35];
2. Traffic signal device for driver/pedestrian/cyclist advisory message screen at signalized intersections[36];
3. System and method for launching a browser in a safe mode[37];
4. System and methods for detection of fraudulent online transactions[38];
5. Method to generate novel bioactive molecules[39];
6. Methods and compositions for enhanced delivery of bioactive molecules[40];
7. Heat exchanger arrangement for turbine engine[41];
8. Heat engine and heat to electricity systems and methods.[42]

For the urban architectural patterns by "independent" variables, we applied the classification proposed by Alexander: the patterns are related to towns, buildings, and construction. As a result we randomly selected the following nine urban architectural patterns for small-N analysis:

1. Pattern №11: Local transport areas
2. Pattern №21: Four-story limit
3. Pattern №22: Nine per cent parking
4. Pattern №110: Main entrance
5. Pattern №124: Activity pockets
6. Pattern №127: Intimacy gradient
7. Pattern №221: Natural doors and windows
8. Pattern №224: Seat spots
9. Pattern №242: Front door bench

Similarly with the urban architectural patterns, in case of software patterns there are three types of them: creational patterns, structural patterns, and behavioral patterns. This resulted in the following three urban architectural patterns for the software patterns:

1. Abstract Factory
2. Adapter
3. Strategy

4.2 Mapping patents, urban architectural patterns, and software patterns to system concept representation framework

In order to introduce a rigorous process for mapping patents, urban architectural patterns, or software patterns to proposed framework, we assigned a number to each one of the 28 entries of the framework that is presented in Figure 6. Entries 1 and 2 are related to stakeholders and their needs (proposition I in Figure 3); 3-7 deal with solution-neutral environment (proposition II in Figure 3); 8-16 are related to solution-specific environment or concept (proposition III in Figure 3); entries 17-25 are concerned with integrated concept, including structure and interactions (proposition IV in Figure 3); and entries 26-28 deal with concept of operations (proposition V in Figure 3), respectively. Note that if only the information suggested in Figure 4B (solution-neutral problem and core solution-specific concept) were represented, entries would only be found in patent, urban architectural pattern, or software pattern under entries 3-16.

The text of each of the eight patents, nine urban architectural patterns, and three software patterns was analyzed, and the outcome summarized as the answer to the following question: “Is this specific entry used to map patent, urban architectural pattern, or software pattern to the framework?” If the answer is yes then we have put "V" against the corresponding entry and, consequently, we counted this entry as present at patent, urban architectural pattern, or software pattern. This allows us to see how the information on the 28 entries is spread throughout the study. Appendix A provides the definitions of the terms and the criteria applied during the analysis in order to define whether the specific information in the text corresponds to the term of concept framework entry.

5. Results of the experiment

5.1 Mapping patents to the system concept representation framework

The results of mapping for small-N sample of patents is shown in Figure 9, which summarizes the outcomes for the question “Is this specific entry used to map patent to the framework?” The criteria based on which the text was analyzed and the specific entry was counted or not counted for inclusion into the framework could be found in Appendix A. The axis X of the plot of Figure 9 contains all 28 entries of the concept framework, while the axis Y represents the frequency of references in percentage. For example, if one sees 100% frequency of references against a specific entry, it means that this entry is present in all patents that were analyzed during the study. Bar at top of Figure 9 helps with identification of frequency of references for each entry and the allocation of entries on propositions I-V.

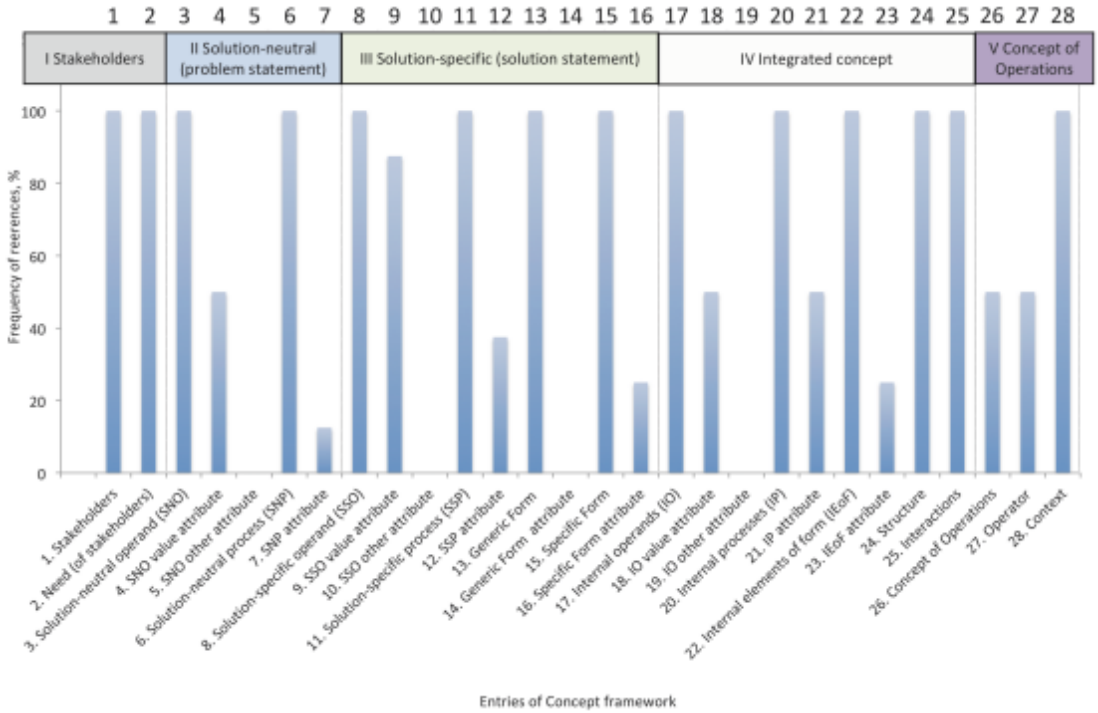


Figure 9. Occurrence of concept framework's entries in patents

The first important conclusion is that the broad concept framework is needed for mapping a patent, and not just the information at the core of the concept (entries 3-16 indicated in Figure 6). In particular, the stakeholder information (entries 1-2) is vital, as is the integrated concept (entries 17-25), including the relationship information (structure, interactions), and concept of operations – entries 26-28. Using the more restrictive definition of concept in Figure 4B would not be sufficient, as it only partly describes the concept. As it can be seen in Figure 9, such entries of framework as solution-neutral operand and process (entries 3 and 6), solution-specific operand and process (entries 8 and 11), generic form (entry 13), and specific form (entry 15) are always present in patents.

An especially interesting result is that the claims, the intrinsic parts of any patent, are primarily reflected in the first level decomposition of the integrated concept (17-25). Every patent maps claims to the internal operands, processes and elements. There is clearly an underlying force at work here. For a patent to describe the invention in a legally defensible way, the patent must decompose the invention into a number of pieces and explain what each one of the pieces does. Each one of such pieces composed of internal element that is used to execute the internal function (internal process plus internal operand in the notation of the proposed framework). It is also an opportunity to engage the model-based conceptual design to represent the integrated concept.

Another observation is that the "other attributes" are never used in the mapping (entries 5, 10, 19 indicated in Figure 6), while "attributes" are seldom used to characterize the corresponding operand, process, or form (entries 7, 12, 14, 16, 21, 23 indicated in Figure 6) – the average appearance is 25%. The value-related attributes are appearing more often – in 62.5% cases (entries 4, 9, 18 indicated in Figure 6). There is some sense to this. Since the person applying for a patent wants the coverage to be as broad as possible, it is in their interest to not restrict or constrain it by qualifications implied by these entries. It is likely that for patent analysis the nine entries related to the "other attributes" and "attributes" could be omitted without loss of important information.

5.2 Mapping urban architectural patterns to the system concept representation framework

The small-N analysis for urban architectural patterns is presented in Figure 10, during which the same question was asked - “Is this specific entry used to map urban architectural pattern to the framework?” The 28 entries of the concept framework are contained in the axis X of the plot of Figure 10, while the axis Y represents the frequency of references in percentage. The criteria based on which the decision on inclusion of specific information from the text of urban architectural patterns was made could be found in Appendix A.

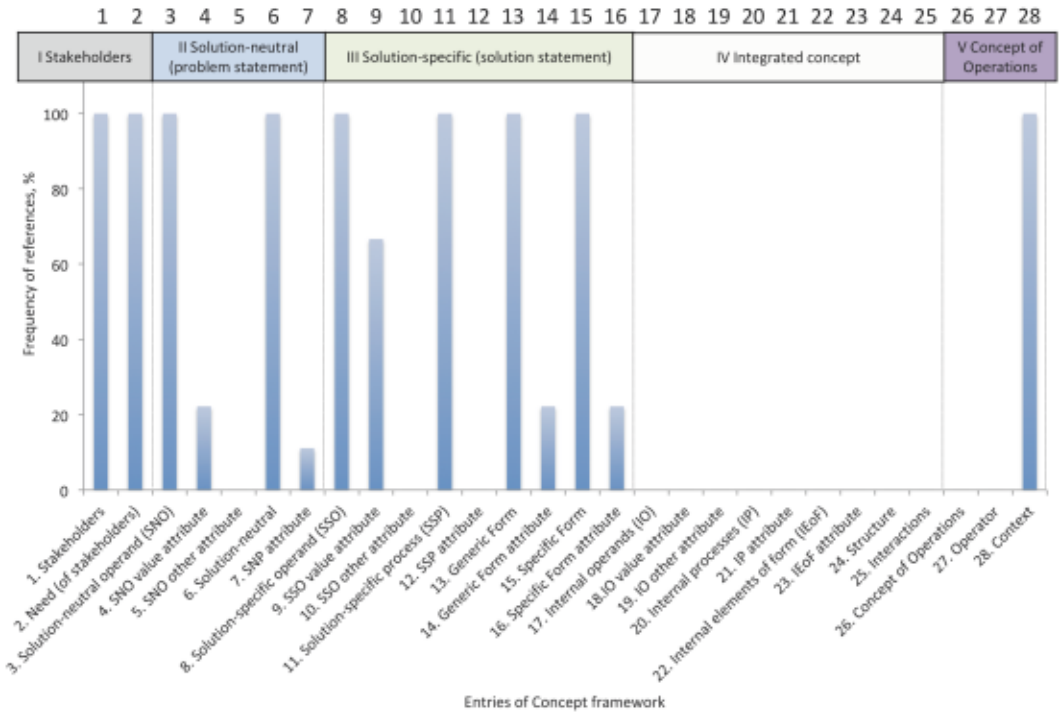


Figure 10. Occurrence of concept framework’s entries in urban architectural patterns

From Figure 10 we may notice that urban architectural patterns are mainly focusing on the first propositions of the concept framework: stakeholders (entries 1-2), solution-neutral problem statement (entries 3-7), and solution-specific solution statement (entries 8-16). In particular, such entries as stakeholders and stakeholders’ need (entries 1-2), solution-neutral’s operand and process (entries 3 and 6), solution-specific’s operand and process (entries 8 and 11), as well as generic form and specific form (entries 13 and 15) are always present in urban architectural patterns. Additionally, in Alexander’s work we always found the context (entry 28).

A pertinent observation is that comparing with the outcomes of patents analysis, the core difference of urban architectural patterns analysis is that there is no information about integrated concept. Partly this can be explained by highly abstracted nature of Christopher Alexander's work: his goal was to identify the problem and propose the solution - both (problem and solution) are highly interwoven with societal structures. In urban architectural patterns there is no such part as claims, as we find in patents. Claims require very clear evidence, since they are legally significant. Another reason of the absence of the integrated concept in the texts of urban architectural patterns is that its elements sometimes appear in the accompanying figures. However, the scope of our study was to only analyze the

text. Of course the patents also contain figures, and we consistently did not use them in the analysis above.

Another outcome of urban architectural patterns analysis is that the context (entry 28) is always present in them. There is some explanation to this. Alexander's work is closely related to the issues appearing in human's daily life – either while constructing the personal houses, or during the searching for the solution to improve the communities where people live. Thus, it is important to provide the context under which urban architectural pattern is considered.

5.3 Mapping software patterns to the system concept representation framework

The small-N analysis for software patterns is presented in Figure 11. Similarly to the previous studies, the same question was tested: "Is this specific entry used to map software pattern to the framework?" The criteria and key definitions of the concept framework are summarized in Appendix A.

The analysis of small-N study results reveals that the first proposition of proposed framework (stakeholders) is always present in software patterns; the second proposition (solution-neutral problem statement) is presented in its core entries - process and operand; the third proposition (solution-specific solution statement) is also present in its core entries - process, operand, and form. The forth proposition (integrated concept), including structure and interactions, is always contained in software patterns; and finally the fifth proposition (concept of operations) can always be found in software patterns.

Another important result is that the attributes related to the two propositions of the concept framework (solution-neutral environment, and solution-specific environment) regularly appear in software patterns - in 54% of cases. The attributes related to the fourth proposition of concept framework (integrated concept) appear much more seldom - in 17% of the cases. This result could be explained by the nature of software patterns: they tend to propose the solution in solution-specific domain (third proposition of the proposed framework). The information contained in the decomposed elements (integrated concept) aims to support the solution at a higher level. Thus, there is no need to define the attributes of each one of the subclasses - this should be done at the later stages of the design process. These results are illustrated in Figure 11.

Comparing to the results of the previous studies we may conclude that every proposition of proposed concept framework is present in software patterns.

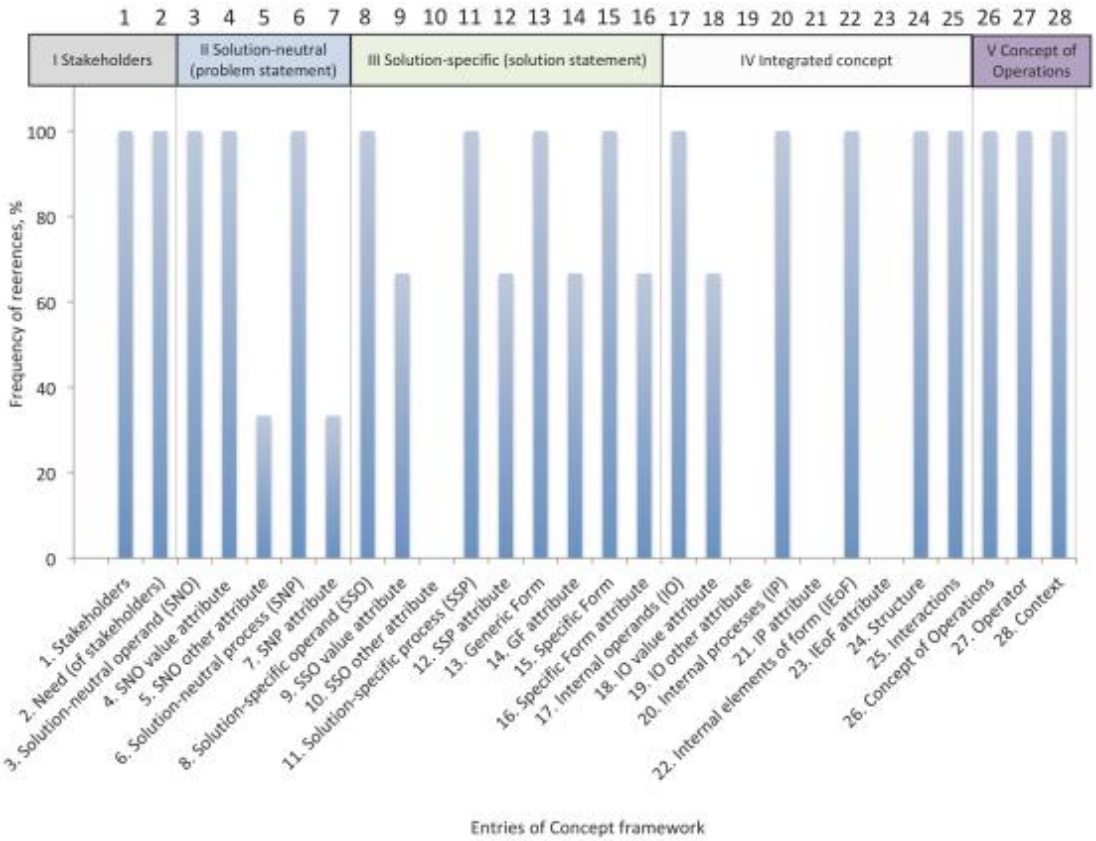


Figure 11. Occurrence of concept framework's entries in software patterns

5.4 Cross cutting results for the system concept representation framework

The small N analysis of patents, urban patterns, and software patterns revealed another important outcome – that they can all be successfully mapped to the framework. The entries in the framework largely describe the concept represented in patent, urban pattern, or software pattern (see Figure 12). Therefore, mapping the information from these sources of knowledge to the concept framework encodes valuable information about their concepts.

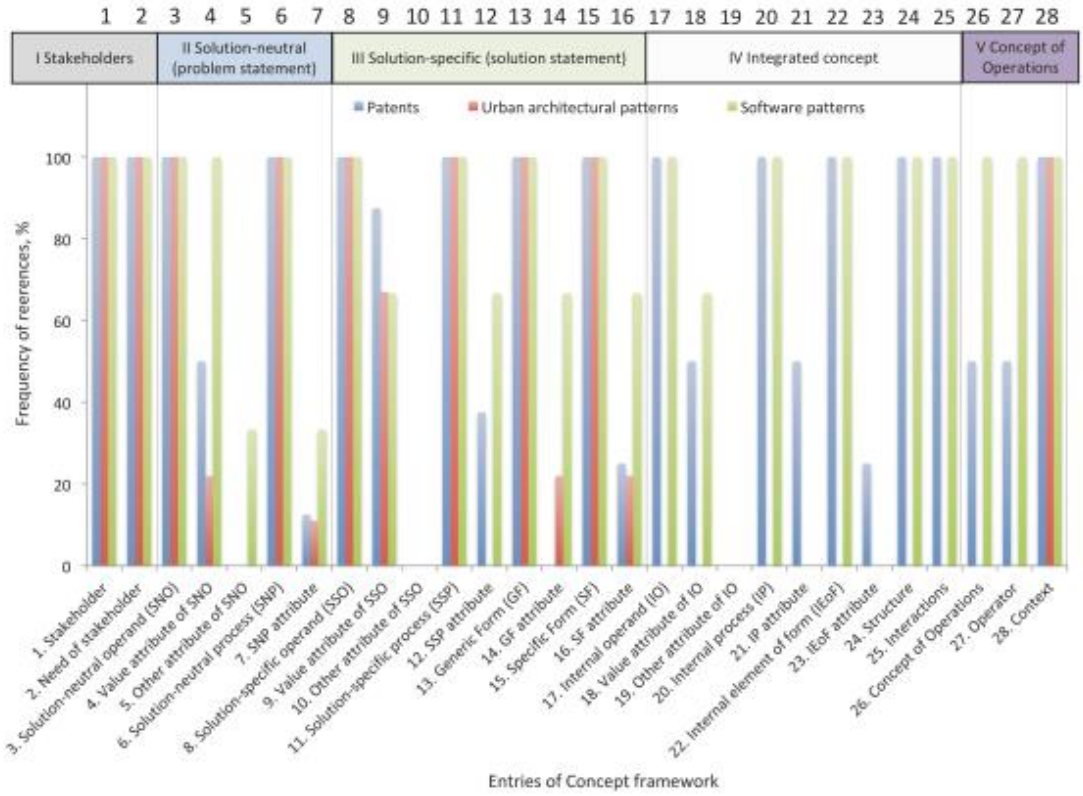


Figure 12. Occurrence of concept framework's entries in patents, urban architectural patterns, and software patterns

We have demonstrated that the conceptual content of a patent, urban architectural pattern, or software pattern can be mapped to proposed framework. *Virtually all of the entities of the framework are used* in some or another patent, urban pattern or software pattern. But not all entries are used universally, and there are profiles of the entries used by patents, by urban patterns and by software patters, as discussed in Sections 5.1 – 5.3. This result validates the design of our small-N experiment. Our choice of diverse examples to map to the framework provided this breadth of coverage.

Examining Figure 12 in more detail, we see that some entries are universal:

- The stakeholder and their needs (entries 1, 2) of the stakeholder proposition
- The solution-neutral operand and process (entries 3, 6) of the solution-neutral proposition
- The solution-specific operand, process, generic form and specific form (entries 8, 11, 13, 15) of the solution-specific proposition
- The context (entry 28) of the concept of operation proposition.

This is not surprising, as these are the core of conceptual design.

If we ignore the absence of the urban patterns, which convey much of the following information by figure, the other entries that are universal are the internal operand, internal process and internal elements of form, and the structure and interactions (entries 17, 20, 22, 24, 25) of the integrated concept proposition. This is surprising, as a conventional definition of concept may not contain this level of detail. But we have seen it is key in patents and software patterns.

Working from the low side, we identify two attributes that are never used: the other attributes of the solution-specific operand and internal operands (entries 10, 19). This is probably due to the fact that the value related attributes of the operands are more important, and the "others" go unnoticed.

In between important and rare are 12 entries that are sometimes used and sometimes not. Ten of these are various attributes – not critical but useful. Two of them are aspects of operations: concept of operations and operator (entries 26, 27). Due to the importance of operations in delivering value, one would think these would be more universal.

One sense of the potential completeness can be gained by comparison with representation of human thought in linguistics. Noam Chomsky discovered three intrinsic parts of human natural language[43]: a noun that is the instrument of the action, a verb describing the action, and a noun that is the object of the action. In order to describe some idea or concept, the human uses nouns and verbs to form a meaningful sentence. The framework proposed in our work is useful, as it provides a means to encode the system concept – expressed as object nouns (the operand), verbs (the processes), instrument nouns (the form), along with adverbs, and adjectives (the last two are represented by the "attributes" in the framework). For example, by taking the proposed system models of the specific patent/urban architectural pattern/software pattern the systems engineer could reconstruct the claims part of the patent, or to identify the concrete solution for the societal problem explained in an urban pattern, or the appropriate software pattern allowing to solve a specific problem in software implementation. The framework is a useful tool in the representation of knowledge contained in analyzed patents and patterns.

6. Summary and Conclusion

The objective of this work was to develop a system concept representation framework that reflects the concept's constituents, their definitions and interconnections. This was the observation of authors that the proposed concept framework comprises 28 entries, which span from stakeholder issues, solution-neutral function and solution-specific concept to integrated concept and concept of operations.

To achieve this objective, eight patents, nine urban architectural patterns, and three software patterns representing a broad spectrum of engineering and urban systems were mapped to concept framework. All were successfully mapped to the framework, in the sense that the key entries were identified. This confirmed that the content of patent, urban architectural pattern, or software pattern could be mapped to concept framework. This success is a necessary condition to showing utility of the proposed framework.

Within the results there were some interesting details. Virtually all 28 entries were used by the patents, urban patterns and software patterns, but each one did not use them all. There were some profiles to the usage.

For patents, an important specific finding is that the claims section of a patent, the key area of legal protection, contain information on form and function at one level of decomposition below the concept itself, in what we call the integrated concept.

The urban architecture patterns primarily addressed stakeholder needs, solution-neutral and specific domains and context, but left the integrated concept and concept of operations to figures, which were not analyzed.

The software patterns propose the way to satisfy the stakeholders needs focusing on the detailed explanation of stakeholders, solution-neutral/specific environments, and context; and only on the most essential entries of the integrated concept (operands, processes, and forms). But there was little information about attributes.

Looking across all of the mappings, we can identify a core of entries that appeared in all cases. These included stakeholders, solution-neutral and specific elements and context. More details on the integrated concept were identified than might be expected.

The proposed framework might have several forms of utility: encoding the core information about concepts such as contained in patents, urban architectural patterns, and software patterns; and generating concepts at early phases of the design process in a model-based environment, thus contributing to INCOSE Model-Based Conceptual Design initiative[3]. Such a framework represents the application of MBCD-based framework to practice. In addition to these two forms of utility, the framework might be useful as it can help to measure the distance between concepts, allowing the formal analysis, such as identification of similarity between alternative concepts. We leave such analysis for a direction of future work. Another utility of the framework is that the concept knowledge is reused in later stages of the design process – during the architecture development. Thus, the information from the conceptual design phase is spread throughout the other design stages.

There are several directions in which this work can be extended. The small-N study has allowed us to evolve the framework. Compared with the first draft, some entries were found unneeded and some added. Now we are in a position to further test the framework with a large-N study of patents, urban patterns, and software patterns. The framework could be extended to representing socio-technical and organizational systems. Another direction of future work is to apply the proposed framework to conceptual design phase of a specific system. In this capacity, a framework would serve as a guide for what information is required to be clarified from stakeholders to formulate the functional requirements and to develop the alternative concepts. This work would require an application of DSM-based methods[44-46] to systems analysis.

In this work the information from patents, urban patterns, and software patterns was mapped into the proposed framework by means of human reasoning supported by a clear definitions and criteria listed in Appendix A. However, such mapping could be done with textual analysis by means of machine learning. Thus, another direction of future work is the exploration of knowledge description in a machine-accessible way, which would employ the usage of OWL ontology[21] and a concept classification scheme representation in a SysML language.

References

1. Kroll E, Condoor S, Jansson D. *Innovative conceptual design: Theory and application of parameter analysis*. Cambridge: Cambridge University Press; 2001. http://doi.org/10.1017/cbo9780511612923.
2. Farid A, Suh N. *Axiomatic Design in Large Systems*. Switzerland: Springer; 2016. http://doi.org/10.1007/978-3-319-32388-6.
3. INCOSE. Model-Based Conceptual Design Working Group (MBCD WG) Charter. https://www.incose.org/docs/default-source/wgcharters/model-based-conceptual-design.pdf?sfvrsn=920eb2c6_6. Accessed August 13, 2019.
4. Heflin J. OWL Web Ontology Language Use Cases and Requirements. *W3C Recommendation*; 2004. http://www.w3.org/TR/2004/REC-webont-req-20040210/
5. Crawley E, Cameron B, Selva D. *System Architecture: Strategy and Product Development for Complex Systems*. Boston: Prentice Hall Press; 2015.
6. Ulrich K, Eppinger S. *Product Design and Development.* 5th ed. New York: McGraw-Hill Education; 2011.
7. Walden DD, Roedler GJ, Forsberg K, Hamelin RD, Shortell TM. *INCOSE Systems Engineering Handbook*: *A Guide for System Life Cycle Processes and Activities*. 4th Ed. Hoboken: Wiley; 2015.
8. Morris BA, Harvey D, Robinson KP, Cook SC. Issues in conceptual design and MBSE successes: insights from the model-based conceptual design surveys. *INCOSE International Symposium*. 2016;26(1):269- 282.
9. Pahl G, Beitz W, Feldhusen J, Grote, K. *Engineering Design: A Systematic Approach*. London: Springer; 2007. https://doi.org/10.1007/978-1-84628-319-2.
10. Andreasen MM, Hansen CT, Cash P. *Conceptual design: Interpretations, Mindset and Models*. London: Springer; 2015.
11. Hubka V, Eder W. *Theory of Technical Systems: A Total Concept Theory for Engineering Design*. New York: Springer-Verlag; 1988.
12. Andreasen MM. The theory of domains. *EDC-Workshop: Understanding Function and Function to Form Evolution*; 1992: 21-47.
13. Ferreirinha P, Grothe-Moller T, Hansen CT. TEKLA, a language for developing knowledge based design systems. *Proceedings of ICED*, 1990.
14. Jensen T. *Functional modeling in a Design Support System – Contribution to a Designer's Workbench*. PhD Dissertation, Technical University of Denmark, Copenhagen; 1999.
15. Mortensen NH. *Design Modeling in a Designer's Workbench. Contribution to a Design Language*. PhD Dissertation, Technical University of Denmark, Copenhagen; 1999.
16. Hatchuel A, Weil B. C-K design theory: an advanced formulation. *Research in Engineering Design*. 2009; 19(4): 181-192.
17. Siddharth L, Chakrabarti A, Ranganath R. Modeling and structuring design rationale to enable knowledge reuse. *Systems Engineering*. 2020;23(3):294-311.
18. Shafaat A, Kenley CR. Model-based design of project systems, modes, and states. *Systems Engineering*. 2020;23(2);165-176.
19. Selva D, Cameron B, Crawley E. Patterns in system architecture decisions. *Systems Engineering*. 2016;19(6): 477-497.
20. Raz AK, Kenley CR, DeLaurentis D.A. System architecting and design space characterization. *Systems Engineering*. 2018; 21(3): 227-242.
21. Antoniou G, van Harmelen F. Web ontology language: OWL. *In Handbook on Ontologies*. Berlin, Heidelberg: Springer, 2004:67-92.

22. McGuinness DL, van Harmelen F. OWL web ontology language overview. *W3C recommendation*. 2004;10(10).
23. Dori D. *Object-Process Methodology: A Holistic System Paradigm*. Berlin: Springer; 2002. http://doi.org/10.1007/978-3-642-56209-9.
24. Dori D, Reinhartz-Berger I, Sturm A. Developing complex systems with object-process methodology using OPCAT. *International Conference on Conceptual Modeling*. 2003:570-572 https://doi.org/10.1007/978-3-540-39648-2_46.
25. ISO 19450, Automation systems and integration - Object-Process Methodology.
26. Soderborg N, Crawley E, Dori D. System function and architecture: OPM-based definitions and operational templates. *Communications of the ACM*. 2003;46(10):67-72. https://doi.org/10.1145/944217.944241.
27. Yaroker Y, Perelman V, Dori D. An OPM conceptual model-based executable simulation environment: implementation and evaluation. *Systems Engineering*. 2013;16(4):381-390.
28. Suh ES, Chiriac N, Hölttä-Otto K. Seeing complex system through different lenses: impact of decomposition perspective on system architecture analysis. *Systems Engineering*. 2015;18(3):229-240.
29. USPTO, Glossary. 2007. Available at: https://www.uspto.gov/learning-and-resources/glossary.
30. WIPO. *Module 6: Patent Information*. 2016. Available at: http://www.wipo.int/export/sites/www/sme/en/documents/pdf/ip_panorama_6_learning_points.pdf.
31. Alexander C. *A Pattern Language: Towns, Buildings, Construction*. New York: Oxford University Press; 1977.
32. Alexander C. *Notes on the Synthesis of Form*. Cambridge: Harvard University Press; 1964.
33. Gamma E, Helm R, Johnson R, Vlissides J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading:, MA: Addison-Wesley Professional; 1994.
34. Trost J. Statistically nonrepresentative stratified sampling: a sampling technique for qualitative studies. *Qualitative Sociology*. 1986;9(1):54-57. http://doi.org/10.1007/bf00988249.
35. Al-Qaneei JMAJ. Vehicle mounted traffic light and system. US Patent 9318021. 2016.
36. Holzmac LLC. Traffic signal device for driver/pedestrian/cyclist advisory message screen at signalized intersections. US Patent 9153128. 2015.
37. Chebakov MS, Maslov IS. Kaspersky Lab ZAO. System and method for launching a browser in a safe mode. US Patent 9292701. 2016.
38. Golovanov SY, Monastyrsky AV. Kaspersky Lab ZAO. System and methods for detection of fraudulent online transactions. US Patent 9363286. 2016.
39. Palsson B, Charusanti P, The Regents of The University Of California. Method to generate novel bioactive molecules. US Patent 9080199. 2015.
40. Lewis D, Schmidt P, Hinds K, PR Pharmaceuticals. Methods and compositions for enhanced delivery of bioactive molecules. US Patent 6706289. 2004.
41. Murphy MJ, Zamora SP, United Technologies Corporation. Heat exchanger arrangement for turbine engine. US Patent 9212623. 2015.
42. Held TJ, Hoster S, Miller JD, Hume BF, Echogen Power Systems. Heat engine and heat to electricity systems and methods. US 8096128; 2012.
43. Chomsky N. Three models for the description of language. *IRE Transactions on Information Theory*. 1956;2(3):113-124.
44. Eppinger S, Browning T. *Design structure matrix methods and applications*. Cambridge, MA: MIT press; 2012.

45. Akhtyamov R, Vingerhoeds R, Golkar A. Measures and approach for modemization of existing systems. *IEEE International Systems Engineering Symposium (ISSE)*. 2018;1-8.
46. Menshenin Y, Crawley E. DSM-Based Methods to Represent Specialization Relationships in a Concept Framework. *International Dependency and Structure Modeling (DSM) Conference*. 2018;151-157.
47. Freeman RE. *Strategic Management: A Stakeholder Approach*. Boston: Pitman; 1984.
48. NASA Systems Engineering Handbook. 2016. Available at: https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20170001761.pdf. Accessed November 29, 2017.

Appendix A

Table A1. Criteria for inclusion of the data from patents, urban architectural patterns, and software patterns as the entries of system concept representation framework

№	Concept framework entry	Definition	Criteria
1	Stakeholders	Any group or individual who can affect or is affected by the achievement of the organization's objectives[47]	Whether the stakeholders are mentioned in the text. The stakeholders are usually represented by noun – "organization", "user", "community", etc.
2	Stakeholders' need	Needs "are defined in the answer to the question 'What problem are we trying to solve?'"[48]	Whether the stakeholders' needs are mentioned in the text. The needs are usually represented by verb + noun and are stated in an abstract way – "have fun", "create a healthy environment", etc.
3	Solution-neutral operand (SNO)	The operand is an object that is changed by the process.	Whether the operand is mentioned in the text. The operand is usually represented by noun. The fact that it is solution-neutral implies that the operand is mentioned in an abstract way. The example is "person", etc.
4	SNO value attribute	All those features which belong substantially to the operand. If the process changes the state of an object, we call such attribute a value-related one.	Whether the value-related attribute is mentioned in the text. It can be represented by noun ("location", "number"), or adjective ("safe"), and the choice of the attribute depends on the context.
5	SNO other attribute	All those features which belong substantially to the operand. Other attributes are those, which are important to be aware of, but are not changed by the process.	Whether the other attribute is mentioned in the text. It can be represented by noun ("speed"), or adjective ("safe"), the choice of the attribute depends on the context.
6	Solution-neutral process (SNP)	The process illustrates a dynamic nature of the function: it reflects the action.	Whether the process that acts on operand can be found in the text. The process is usually represented by verb. The fact that it is solution-neutral implies that the process is mentioned in an abstract way. The examples are "entertaining", "moving", etc.
7	SNP attribute	All those features which belong substantially to the process.	Whether the attribute is mentioned in the text. The attribute related to the process is usually represented by adverb – such as "safely", the choice of the attribute depends on the context.
8	Solution-specific operand (SSO)	The operand is an object that is changed by the process.	The same principles as for №3 are applied. The core difference is that solution-specific operand is specialization of solution-neutral operand. Thus, the example is "passenger".
9	SSO value attribute	All those features which belong substantially to the operand. If the process changes the state of an object, we call such attribute a value-related one.	Whether the value-related attribute is mentioned in the text. It can be represented by noun ("location", "number"), or adjective ("safe"), and the choice of the attribute depends on the context. It can be inherited from №4 or newly appeared.
10	SSO other attribute	All those features which belong substantially to the operand. Other attributes are those, which are important to	Whether the other attribute is mentioned in the text. It can be represented by noun ("speed"), or adjective ("safe"), the choice of the attribute depends on the context. It can be

		be aware of, but are not changed by the process.	inherited from №5 or newly appeared.
11	Solution-specific process (SSP)	The process illustrates a dynamic nature of the function: it reflects the action.	The same principles as for №6 are applied. The core difference is that solution-specific process is specialization of solution-neutral process. Thus, the examples are “flying”, “floating”, but it can be inherited from №6.
12	SSP attribute	All those features which belong substantially to the process.	Whether the attribute is mentioned in the text. The attribute related to the process is usually represented by adverb – such as “safely”, the choice of the attribute depends on the context. It can be inherited from №7 or newly appeared.
13	Generic Form	Form is the instrument that executes a function. The generic form is an abstracted version of an object that is usually associated closely with solution-specific process.	Whether the generic instrument that executes the function (№11 plus №8) is appearing in the text. The generic form is usually represented by noun. The example is “land vehicle”, or “flying vehicle”, etc.
14	Generic Form attribute	All those features which belong substantially to the generic form.	Whether the generic form attribute is mentioned in the text. It can be represented by noun (“cost”, “number”), or adjective (“safe”), and the choice of the attribute depends on the context.
15	Specific Form	Form is the instrument that executes a function. The specific form is a further specialization of the generic form.	Whether the specific instrument that executes the function (№11 plus №8) is appearing in the text. The specific instrument is usually represented by noun. The core difference comparing with №13 is that specific form is the specialization of generic form. The examples are “car”, “train”, or “aircraft”, “helicopter”.
16	Specific Form attribute	All those features which belong substantially to specific form.	Whether specific form attribute is mentioned in text. It can be represented by the noun (“cost”, “number”), or adjective (“safe”), and the choice of the attribute depends on the context.
17	Internal Operands (IO)	The operand is an object that is changed by the process. The internal operand implies that it is part of the decomposition.	Whether we can identify in the text the internal operand, on which the internal process (№20) is acting. Similarly to other operands, it is usually represented by noun.
18	IO value attribute	All those features which belong substantially to the operand. If the process changes the state of an object, we call such attribute a value-related one.	Whether the value-related attribute is mentioned in the text. It can be represented by noun (“location”, “number”), or adjective (“safe”), and the choice of the attribute depends on the context.
19	IO other attribute	All those features which belong substantially to the operand. Other attributes are those, which are important to be aware of, but are not changed by the process.	Whether the other attribute is mentioned in the text. It can be represented by noun (“speed”), or adjective (“safe”), the choice of the attribute depends on the context.
20	Internal Processes (IP)	The process illustrates a dynamic nature of the function: it reflects the action. The internal process implies that it is part of the decomposition.	Whether we can identify internal process, which is acting on internal operand (№17), and which is related to the corresponding internal element of form (№22). Similarly to other processes, it is usually represented by verb.

21	IP attribute	All those features which belong substantially to internal process.	Whether the attribute is mentioned in the text. The attribute related to the process is usually represented by adverb – such as “safely”, the choice of the attribute depends on the context.
22	Internal Elements of Form (IEoF)	Form is the instrument that executes a function. The IEoF implies that it is part of the decomposition.	Whether specific form (№15) is decomposed into internal element(s) of form and if so, whether we can find such decomposed element(s) in the text. IEoF is usually represented by noun.
23	IEoF attribute	All those features which belong substantially to IEoF.	Whether the attribute is mentioned in the text. The attribute can be represented by noun (“location”, “number”), or adjective (“safe”), and the choice of the attribute depends on the context.
24	Structure	Formal relationships among the elements – such relationships have a static nature such as connectivity and spatial arrangement	Whether we can identify the information about how the decomposed elements are physically connected. Usually it is explained in the text with support of such words as "connected", "attached", "within", etc.
25	Interactions	Functional relationships, which emphasize the dynamic interactive nature of the interchange of internal operands[5]	Whether we can identify the information about what is exchanged among the decomposed elements. Usually it is explained in text with support of such words as "provides power", "sends signal”, etc.
26.	Concept of Operations	The concept of operations (ConOps) “... describes the overall high-level concept of how the system will be used to meet stakeholder expectations, usually in a time sequenced manner”[48]	ConOps embraces the key operational phases of the system in terms of sequence of processes with corresponding operands, on which the processes are acting, and instruments, which are executing the functions (processes plus operands). In the text this information usually explained in the sequence of sentences, that are gradually uncovering the way how the system works.
27	Operator	Operator is the person, or group of people, or organization, or machine that will operate the concept.	Whether we can identify the individual, or organization, or machine that operates the system.
28	Context	Context contains elements that inform design and operations, but are not essential for function.	Whether we can identify the information about accompanying/supporting systems that are not directly related to our system, but which sheds light on how it is intended to operate.

Appendix B

Example of filling in the system concept representation framework for the US Patent 9318021 (Vehicle mounted traffic light and system)

Abstract

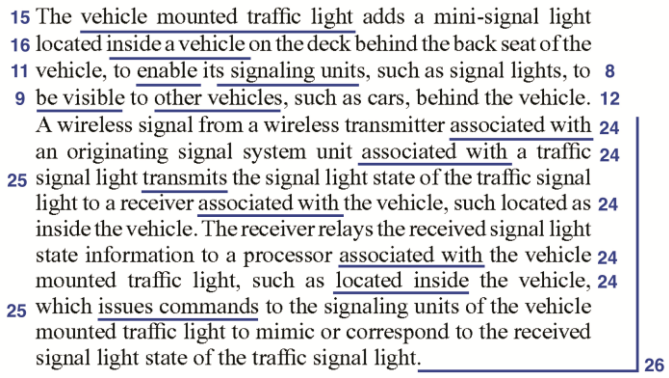


Figure B1. Entries of system concept representation framework in abstract section of a patent

Background of the Invention

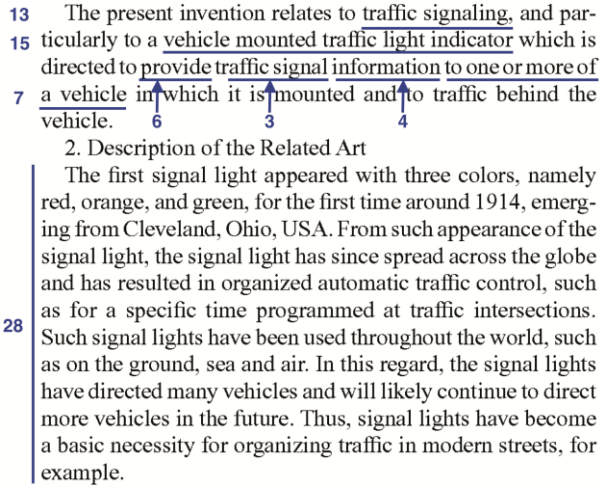


Figure B2. Entries of system concept representation framework in background of the invention section of a patent (1)

...

(Note that these are excerpts from the text of "Background of the Invention" section of the US Patent 9318021 “Vehicle mounted traffic light and system”)

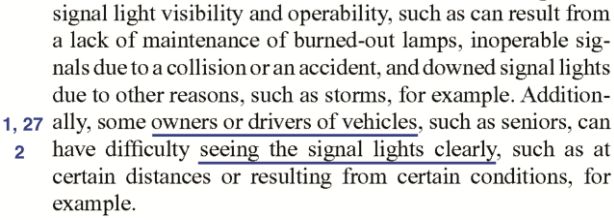


Figure B3. Entries of system concept representation framework in background of the invention section of a patent (2)

Claims

22 a signal unit associated with the receiver for generating 20
17 display signal of the determined state of the traffic con- 18
trol signal light;

Figure B4. Entries of system concept representation framework in claims section of a patent

Entries in a model-based environment

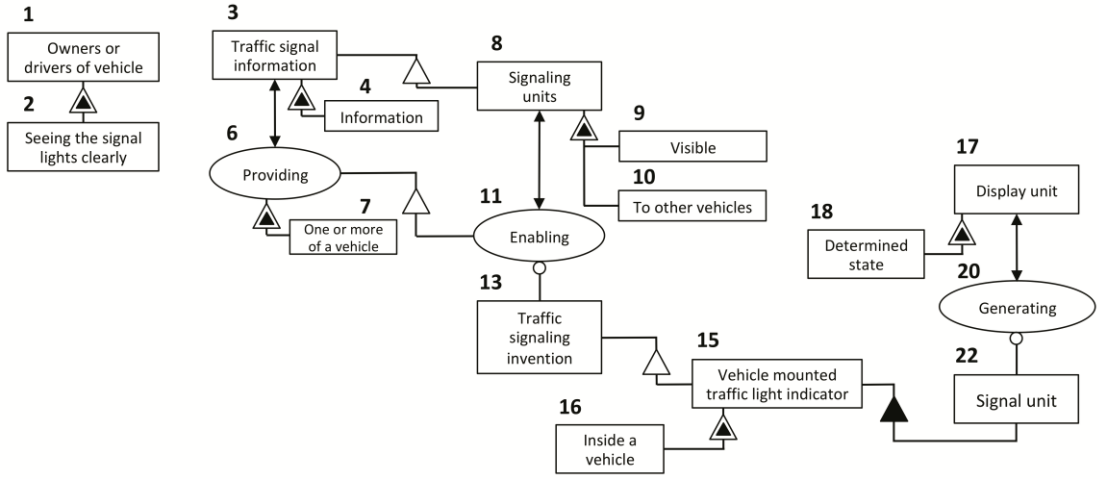


Figure B5. Model-based representation of stakeholders (proposition I of system concept representation framework); solution-neutral environment (proposition II); solution-specific environment (proposition III); and the part of the integrated concept (proposition IV). Note that for the integrated concept one of the internal elements of form (signal unit) is presented as an example

Appendix C

Example of filling in the system concept representation framework for the Urban Architectural Pattern №21 (Four-story limit[24])

Essence of the problem and Body of the problem

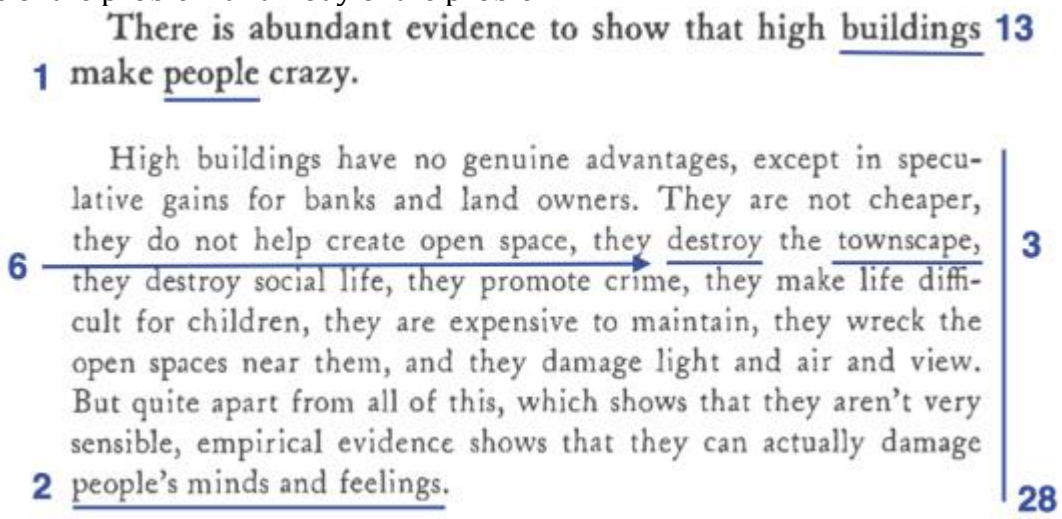


Figure C1. Entries of system concept representation framework in Essence of the problem and Body of the problem sections of an urban architectural pattern

Solution

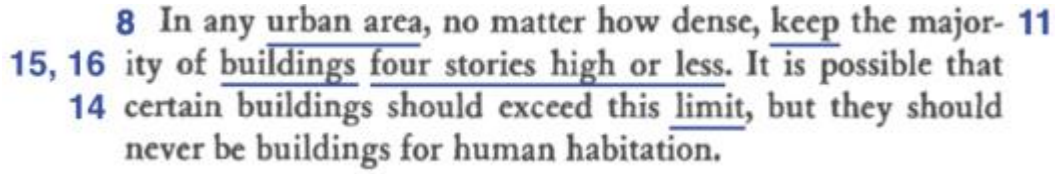


Figure C2. Entries of system concept representation framework in Solution section of an urban architectural pattern

Entries in a model-based environment

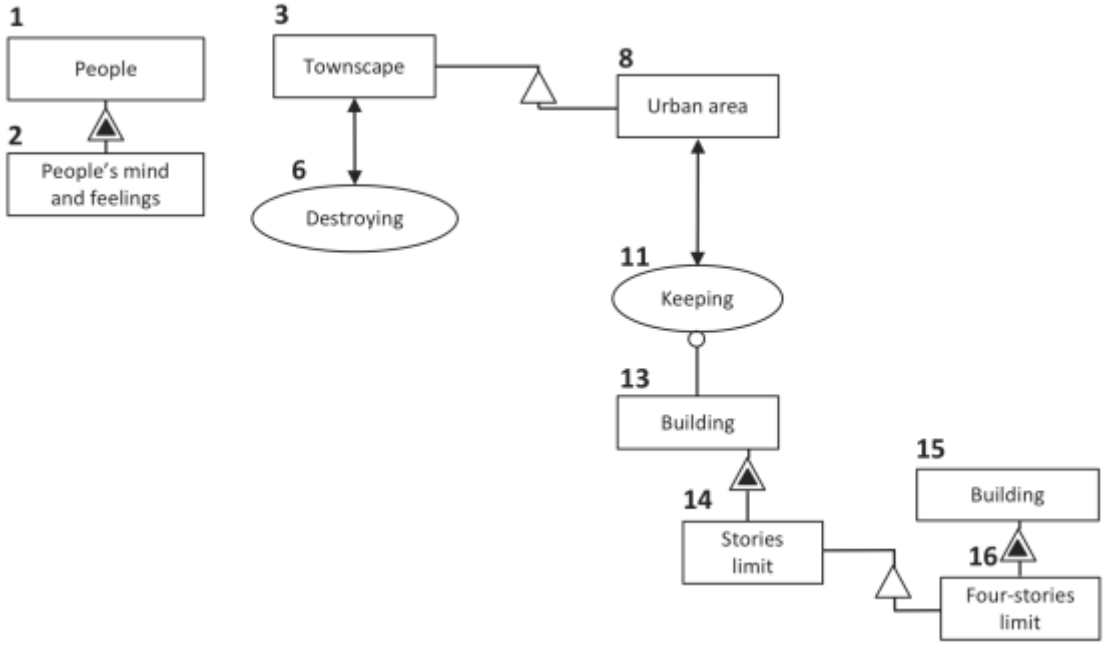


Figure C3. Model-based representation of stakeholders (proposition I of the system concept representation framework); solution-neutral environment (proposition II); solution-specific environment (proposition III)

Few important notes that should be taken into account while exploring urban architectural patterns:

- The proposed framework allows representing a problem-solution pair. For example, in Figure the problem is "destroying townscape". This problem is solved by "keeping urban area" through “four-stories limit” for the “building”. In this case the specialization occurs at the level of operands: “townscape” to “urban area”, and at the level of attributes of forms: “stories limit” to “four-stories limit”. As it was explained in sub-section 2.1 and 2.2 not all entries of solution-neutral environment are always specialized at the level of solution-specific environment. In this example we see that solution-specific process “keeping” appears not in a result of specialization from solution-neutral process "destroying".
- The entries do not necessarily change during the specialization process. For example, generic form (entry 13) and specific form (entry 15) are both "Building". The only thing that is really specialized is an attribute (from “stories limit”, entry 14, to “four-stories limit”, entry 16). We believe that there are some rules regarding when the specific form is specialized from generic form, and when is not. We leave this direction of work for future studies.

Appendix D

Example of filling in the system concept representation framework for the Software Pattern «Abstract Factory»[26]

Problem

6 Provide an interface for creating families of related or dependent objects without 3 specifying their concrete classes. 4

Figure D1. Entries of system concept representation framework in Problem section of software pattern

Solution

Consider a user interface toolkit that supports multiple look-and-feel standards, such as Motif and Presentation Manager. Different look-and-feels define different appearances and behaviors for user interface "widgets" like scroll bars, windows, and buttons. To be portable across look-and-feel standards, an application should not hard-code its widgets for a particular look and feel. Instantiating look-and-feel-specific classes of widgets throughout the application makes it hard to change the look and feel later. 28

13

We can solve this problem by defining an abstract WidgetFactory class that declares 11
8 an interface for creating each basic kind of widget. There's also an abstract class for each kind of widget, and concrete subclasses implement widgets for specific look-and-feel standards. WidgetFactory's interface has an operation that returns a new widget object for each abstract widget class. Clients call these 1, 27 operations to obtain widget instances, but clients aren't aware of the concrete classes they're using. Thus clients stay independent of the prevailing look and 2 feel.

Figure D2. Entries of system concept representation framework in Solution section of software pattern (1)

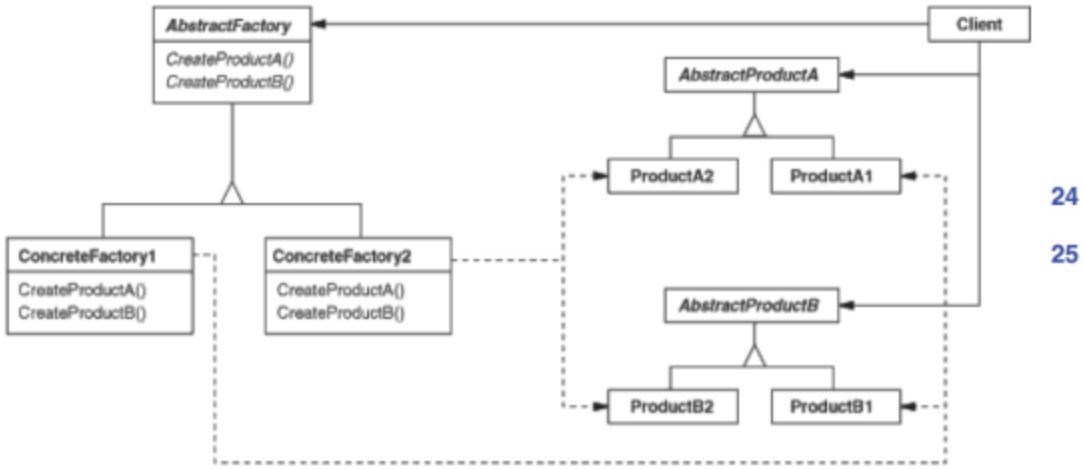


Figure D3. Entries of system concept representation framework in Solution section of software pattern (2)

Entries in a model-based environment

22
• **ConcreteFactory** (MotifWidgetFactory, PMWidgetFactory)
20 o implements the operations to create concrete product objects. 17

Figure D4. Entries of system concept representation framework in solution section of software pattern (3)

Entries in a model-based environment

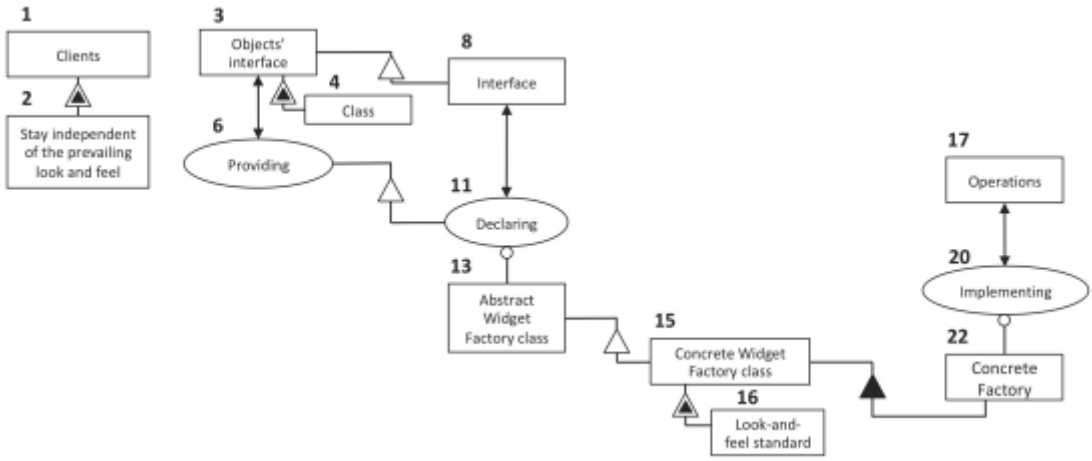


Figure D5. Model-based representation of stakeholders (proposition I of system concept representation framework); solution-neutral environment (proposition II); solution-specific environment (proposition III); and the part of the integrated concept (proposition IV). Note that for the integrated concept one of the internal elements of form (concrete factory) is presented as an example