Systems

Constraining Functions Through Processes (and Vice Versa)

Gabriel Wurzer
Vienna University of Technology, Digital Architecture and Planning
http://www.iemar.tuwien.ac.at
wurzer@iemar.tuwien.ac.at

Abstract: *We propose a novel computer-based design method for planning process-driven buildings. The method binds functions to processes, in order to avoid (1.) functions that are not present in any process and (2.) processes that lack some of their required functions.*

Keywords: *Functional planning; process design; design rigor; diagram.*

Introduction

The planning of complex buildings such as hospitals, airports and industrial facilities is *process-driven*: Above all, these buildings need to support their users in their daily work routine. The role of the architect as part of a much larger planning team is to define a functional program in which no functions are missing, while at the same time ensuring that every planned function is absolutely necessary. To accomplish this, architects must work in close cooperation with the planners who define the process model of the organization.

Keeping functional program and process model consistent throughout the project is the key problem we want to address in our work, as there is to date no integrated tool that links processes to a floor plan, or vice versa. More precisely, we wish to contribute a novel kind of computer-based planning method called *systems* (see Figure 1) that

- provides a combined diagram for processes and functions (see Figure 1, elaboration in pp.2-3)
- can find functions not present in any process and processes that lack functions (see elaboration step 4, p.3)
- is hierarchical, inheriting and provides non-contradictory relations between its components (see elaboration step 1, p.2)

Our work complements other approaches that are focused on architectural programming (see Related Work, p.5), but it is distinctively different: to the best of our knowledge no previous work specifically considers using processes as constraints in the specification of functions. Furthermore, no other approach explicitly allows for process modeling directly in the floor plan. In combination, these innovations lead to a very efficient and simple basis for communication between architects and process planners.

To be fair, this claim remains to be tested. We have implemented a preliminary version of our approach as 3D software, and are currently in the course of evaluating it in a real project. First results look promising, yet we still have too little hard data for a full-fledged analysis. Nevertheless, we present some remarks by users in course of presenting the software (see Implementation and preliminary results, p. 4).

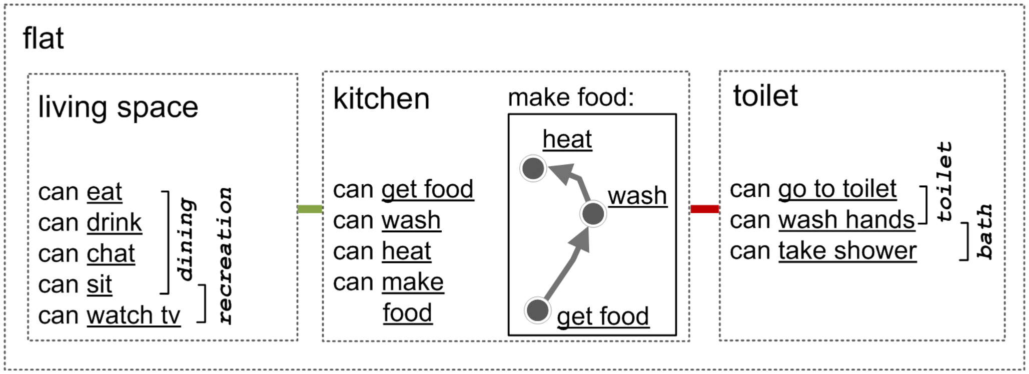

Figure 1
Example of a system ("flat") containing three sub-systems ("living space", "kitchen" and "toilet"). A system is a space (dotted boundary) that contains capabilities (e.g. can eat) and processes (e.g. make food). Furthermore, systems have relations (e.g. line between living space and kitchen). Functions can be attributed to capabilities (e.g. function toilet consisting of can go to toilet, can wash hands)

Elaboration

Step 1: Representing space as a hierarchy

Our approach is based on *spaces*, which are: geometrically bounded areas that are given a name (e.g. the space "flat" in Figure 1). Each space may have subspaces which are fully contained in its boundaries, thus building up a hierarchy of spaces within spaces. As additional constraint, we require that subspaces do not overlap, or put differently: we require that the hierarchy forms a tree (see Related work, p.5).

As in classical functional planning (White 1986), we state the adjacency *relation* between two spaces as being either 'attractive' (relation between living space and kitchen space in Figure 1), 'repellent' (relation between kitchen space and toilet space) or 'neutral' (no line between living space and toilet space). Since our system is hierarchical, we must distinguish between two different kinds of relations: One between siblings, and one between system and subsystem. This distinction is quite important: While sibling relations can be of any kind, relations between systems and subsystems must either be 'related' or 'neutral', but not 'repellent'. The reason for this is that we do not allow contradictions (e.g. a toilet in a hygienic area, when the hygienic area 'repels' the toilet).

Step 2: Re(de)fining functions and introducing processes

The notion of function is ambiguous (Michl, 1995): Architects use the term more in the sense of purpose (or intent), while an outside observer would derive function from an action that was actually performed. It is hard, if not impossible, to build up a rigorous basis for the statement of functions in the presence of such dualities. We therefore have to introduce two new termini in order to make difference between these two notions clear:

- *capability*: the ability to perform a certain action in a space (e.g. capability can eat of space kitchen)
- *action*: the implementation of a capability in a space and time (e.g. action eat in space kitchen)

Extending on this point, we argue that architects are in fact planning the capabilities for their spaces, while process designers plan a sequence of actions (also called *process*), which uses these capabilities. An example of the latter is given in the 'make food' process in Figure 1. This process consists of three sequential actions, get food, wash and make food.

Processes must be fully contained in a space (i.e. they may not cross space boundaries). This constraint is alleviated by the fact that processes themselves are capabilities: The 'make food' process generates a capability 'can make food', which (as will be shown in the proceeding description on inheritance) can be used by a process one level up the hierarchy.

Coming back to functions, we see that the terminus only describes an aggregation of capabilities under a new name. Consider the example of the toilet space in Fig. 1. This space has three capabilities: can go to toilet, can wash hands and can take shower. A place where one can go to toilet as well as wash hands could be considered to have the function of toilet (see square bracket in Figure 1). On the other hand, a space where one can take shower and can wash hands can also be seen as shower.

This latter example has shown that functions are not disjoint; they might share some (or even all) of their capabilities. Furthermore, functions might contain capabilities which are not used in any process, and thus be the key problem when asking the architect that he should not put unused capabilities into the design, while at the same time demanding that there should be no capabilities missing. We therefore opt to leave the naming of functions to the phase when the functional program finally has to be handed in, which is when all capabilities and actions have settled. Furthermore, the naming of functions can be left to an algorithm, applying rules that match function names against the set of capabilities.

Step 3: Introducing inheritance and connecting processes

As benefit of having a hierarchical method, we introduce *inheritance* between parent system and subsystem:

- capabilities are inherited upwards from the subsystem to the parent system (i.e. if I can watch TV in my living room, I also have the capability to watch tv in my flat space). This property is important, as it gives us the possibility to set the granularity at which we are considering the design, or put differently: to choose a level of detail. If we are focusing on flat space level, then we can hide all subspaces of the flat space. As capabilities are inherited upwards, they are still available at flat level
- actions are not inherited, they stay bound to the space in which they were defined (i.e. I can watch TV in my flat space, but in order to perform this action, I have to go to the living space)

Through inheritance, processes may cross boundaries between spaces. As mentioned, every process generates a capability (i.e. the process 'make food' generates the capability 'can make food'). This capability is inherited upwards, and can be used by a process at a higher level. In this way, a process may connect two processes in subsystems.

Step 4: Defining systems and executing constraints

A *system* is the sum of the preceding points, i.e. it is a hierarchical, inheriting space that accommodates both capabilities and processes. A search for functions that have no process (or, as we would now put it, capabilities that lack actions) is now straightforward, and involves checking all capabilities against all actions. In case we find any such candidates, we may either ask the architect to remove it from the functional program, or to enter a reason why this function needs to be put in effect. Vice versa, we may either disallow processes that perform actions for which no capability exists, or we can allow them, but give notice to the architect that he needs to define a corresponding capability.

Implementation and preliminary results

Our work has been implemented as prototype 3D application, in order to prove its usefulness in real project situations. We have initially tested our approach using a high-rise office building and we are in the course of also testing it in the hospital domain. Our results to date are based on a too small basis in order to give an elaborate discussion. We therefore give some first impressions gathered orally, rather than in a more formal way. As side-note, given the amount of feedback brought forward by our initial users, it seems beneficial to first adapt the application to the comments given, rather than conducting a broader survey at this early stage.

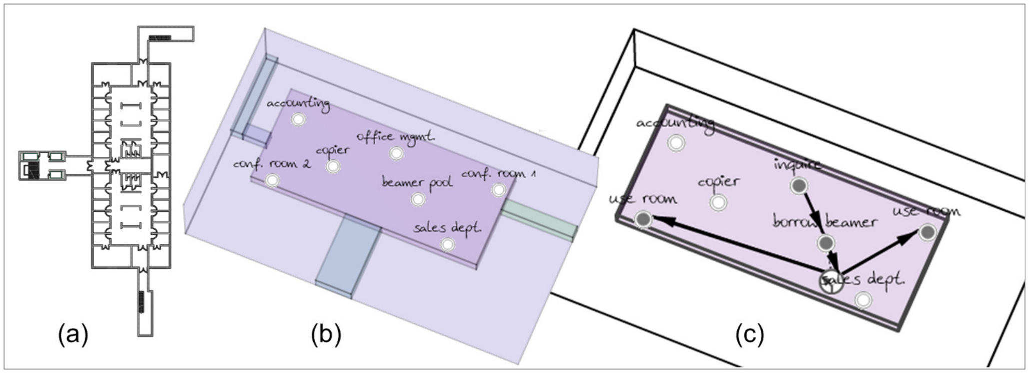

Figure 2
Prototype implementation. (a) original building layout (b) nested systems with capabilities (c) processes

The architect's perspective
We started off by giving architects an existing building design (Fig. 2a) that was converted into systems, asking them define capabilities and processes (Fig. 2b and 2c). Our initial observation was that our test candidates tried to use the application for the generation of form ("where are the shapes I can use?"). We had to explain that, even though the approach is spatial, it does not necessarily correspond to the final form which the building should take. It is rather a method for quickly laying out a hierarchy of spaces and subspaces, capabilities and processes, and form finding would happen in a further step, by integration with the CAD package. This point was also considered a "must-have" by our test candidates.

As further thought, our test users argued that processes would never be defined directly by architects, who clearly think more in terms of functions. Instead, they would expect to find the processes (as defined by the process planners) in a to-do list or catalogue which needs to be fulfilled.

The process planner's perspective
Quite interestingly, the latter point would also be true for process planners. Re-using a list of already predefined processes would form the basis for adaptation targeted at the building project at hand. An import and export functionality of processes to a library would be beneficial for re-use in subsequent projects, as well as the possibility to export function to a more specialized business process modeling software.

Another feedback was that filtering of processes and capabilities should be possible, as it would help in keeping an overview when there are many capabilities and functions on the screen. Furthermore, one should be able to add user-defined properties (or tags) to capabilities and actions, which could also be filtered against.

Another aspect still missing from the application was that of a process documentation generator: Given a set of spaces, it should be possible to generate a process map describing the used processes from the point of view of the building users. This information could then be used in the commissioning of a building and in training for the building users.

Related work

Our approach clearly relates to collaborative and hierarchical design methods such as CASA (Jones, 1992): Our objectives are given in the form of processes, the design options are given as the arrangement of functions and subsystems carried through by a body of collaborating professionals (architects, process planners).

On the other hand, our approach could also be seen as one of many graphical tools which aid in architectural programming; however, we are not aware of any such tools in which the influence of the building user's processes is considered explicitly as requirement of the design. For example, Ekholm (2001) describes an extension of a CAD system to support entering user activities, building on previous work by Eastman and Siabiris (1995); however, process modeling only appears as future work item. Other integrated approaches such as SEED (Flemming, Coyne and Snyder, 1994) and KAAD (Carrara, Kalay, and Novembri, 1994) are similar to us in the sense that they try to decompose functions into a hierarchy ("functional units" or "space units"), but they focus more on the generation of alternative design solutions and their subsequent evaluation than on justifying the required functions by processes. Therefore, we can safely claim that our work is novel.

The use of planning tools based on tree hierarchies for large structures is heavily debated some authors (Minett, 1975). A systems planning approach would imply that all needed resources should be available in the system as a whole. However, since we know that inhabitants of a city think in neighborhoods (Lynch, 1960), it is important to allocate functions according to their local perceptibility rather than efficiency. This effect also applies at smaller scales. For example, restrooms can never be planned efficiently; rather, one has to ensure immediate availability throughout the building.

Another interesting area well beyond the scope of this paper is the representation of buildings and activities on the data level, taking into account the ongoing efforts for standardization in the Building Information Modeling (BIM) domain. In brief, we can say that there are ways of representing activities on the data level, however, the tools that would use this data to support architectural programming are still missing. We may forward the interested reader to (Ilal, 2007; Eastman, Teicholz, Sacks and Liston, 2008).

The visual inspiration for our approach builds on the excellent work on the design space of TreeMaps by (Schulz, Luboschik and Schumann, 2007). More specifically, we use TreeMaps to depict the parent-child relationships of systems through containment; we use cubes as graphics primitives, which we layout manually using architectural tools. As visual representation of processes, we have chosen a simple flowchart-like notation approach that could eventually be extended into the more generic Business Process Modeling Notation (Grosskopf, Decker and Weske, 2009). For a comprehensive review of different notations and patterns we suggest reading (www.bpmn.org/Documents/Notations%20and%20Workflow%20Patterns.pdf: May 2009).

Conclusions and future work

We have shown a novel method for planning process-driven buildings which binds functions to processes in order to avoid (1.) functions that are not present in any process and (2.) processes that lack required functions. Our approach has been implemented as 3D software, and we are currently in the course of evaluating it in a real project. Preliminary results indicate we are so far in good shape, even though we are aware that we still have a lot of work ahead when it comes to adapting the prototype to architectural every-day practice. As possible extensions to the presented work, we currently consider

- introducing a capability catalogue listing capabilities specifically targeted at different areas of building design (e.g. hospital planning, factory planning, etc)
- introducing explicit circulation paths which also constrain process links (i.e. the path from action 'get food' to action 'wash' is bound to a certain path)
- adding process simulation by letting different agents (e.g. patients, nurses, doctors) visually execute processes (e.g. as in Wei, 2006).
- making the defined systems with their processes available in the form of a web-accessible process/space database and further integration with business process servers

Acknowledgements

We wish to thank Jan Michl for his diversification of the notion of function, Christa Illera for inspiration, Georg Franck for supervision, Sigrun Swoboda and Kamyar Tavoussi for modeling and data, Andreas Jonas for his work, Michael Bacher for his expertise on hospitals and the team at away3d.com for their support.

References

Carrara, G., Kalay, Y. E. and Novembri, G.: 1994, Knowledge-Based Computational Support for Architectural Design, ACADIA Conference Proceedings, ISBN 1-880250-03-9, pp. 5-12.

Eastman, C. M. and Siabiris, A.: 1995, A generic building product model incorporating building type information, Automation in Construction, 4(4), pp. 283-304.

Eastman, C., Teicholz, P., Sacks, R. and Liston, K.: 2008, BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors, John Wiley & Sons, Inc., New Jersey.

Ekholm, A. and Fridqvist S.: 1997, Concepts of Space in Computer Based Product Modelling and Design, eCAADe Conference Proceedings.

Ekholm, A.: 2001, Activity objects in CAD-programs for building design. A prototype program implementation, Proceedings of the Ninth International Conference on Computer Aided Architectural Design Futures, ISBN 0-7923-7023-6, pp. 61-74.

Flemming, U., Coyne, R. and Snyder, J.: 1994, Case-Based Design in the SEED System, Computing in Civil Engineering, 1, pp. 446-453.

Grosskopf, A., Decker, G., and Weske, M.: 2009, The Process: Business Process Modeling using BPMN, ISBN 978-0929652269, Meghan Kiffer Press, Tampa.

Ilal, M.E.: 2007, The Quest for Integrated Design System: A Brief Survey of Past and Current Efforts, METU Journal of Faculty of Architecture Faculty of Architecture Middle East Technical University, 24(2), pp. 149-158.

Jones, J.C.: 1992, Design Methods, ISBN 0-471-28496-3, John Wiley & Sons Inc., New York.

Lee, G., Eastman, C.M. and Sacks, R.: 2007, Eliciting Information for Product Modeling using Process Modeling. Data & Knowledge Engineering 62(2), pp. 292-307.

Lynch, K.: 1960, The Image of the City, M.I.T Press, Cambridge.

Michl, J.: 1995, Form Follows WHAT ? The modernist notion of function as a carte blanche, 1:50 – Magazine of the Faculty of Architecture & Town Planning (Technion Univ.), 10, pp. 31–20.

Minett, J.: 1975, If the City is not a Tree, nor is it a System, Planning Outlook New Series, 16, pp. 4 -18.

Schulz, H.-J., Luboschik M. and Schumann, H.: 2007, Exploration of the 3D Treemap Design Space (Poster), Poster Compendium of IEEE Information Visualization.

Wei, Y.: 2006, Integrating web 2D and 3D technologies for architectural visualization: applications of SVG and X3D/VRML in environmental behavior simulation, Proceedings of the eleventh international conference on 3D web technology, pp. 37-45.

White, E. T.: 1986, Space Adjacency Analysis, Architectural Media LTD, Tucson.