**ABSTRACT**

We propose a novel computer-based design method for planning process-driven buildings which extends the traditional architectural schema to include processes. Each function in the schema can be tied to a process, giving us the ability to find (1.) functions that are not present in any process (2.) processes that lack some of their required functions. As benefit of our approach, we can keep functional program and process models of the building consistent and help bridge the communication gap between process planners and architects, simulating the entered processes as we go along.
1. 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 routines (i.e. processes), which are planned in parallel to the design and evolve with it over the course of the building project. In this context, the architect has to define a functional program that facilitates these work routines, in close cooperation with the process planners who define the process model of the organization in question. However, through the co-evolutionary nature of process model and functional program, two consistency problems arise that must be considered by both architect and process planner:

• Functions that are present in the functional program but are not used in any process are likely to be unnecessary or left over from previous iterations, lest the architect can give good reasons why there is indeed need for them with reference to some other argument than the process (restrooms, to give a trivial example, will not need to be rigorously justified).

• Steps in the planned work routine that are not linked to a function can be considered to be incompletely specified, as they remain abstract and without reference to “where” they will be carried out. There are two reasons why this might happen: First, the process planner might simply have forgotten to name the corresponding function. Second, the process planner might have named a function, but the functional program has changed, yielding a reference to a now non-existing function.

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 planning tool that links processes to a design, or vice versa. More precisely, we wish to contribute a novel kind of computer-based planning method called schematic systems that:

• Provides a combined diagram for processes and functions, as extension of the classical architectural schema (see ‘Extending the architectural schema’, beginning on page 202).

• Can find functions not present in any process and processes that lack functions (see ‘Executing Constraints’ on page 206).

• Can optionally be used to simulate the entered processes, helping both process planners and architects to get an overview of the planned operation of the building (see ‘Optional Extensions’ on page 207).

Our work complements other approaches that are focused on recording activities (see ‘Related Work’, page 200), but it is distinctively different: To the best of our knowledge, no previous work specifically considers using processes as constraints for the specification of functions. Furthermore, no
other approach explicitly allows for process modeling and simulation directly in the schema. In combination, these innovations lead to a very efficient and simple basis for communication between architects and process planners.

We have implemented our approach as software (see page 208) and are now evaluating it in actual planning work in the hospital domain (see page 210). Our insights indicate that the approach is well-suited for the early stages of conception of a process-driven building and for cases where a redesign in later stages is necessary. Furthermore, the addition of process simulation directly inside the schematic diagram has gained wide-ranged approval from the side of our project partners in the hospital domain.

2. Related work

Albeit being visually based on classical architectural schemas (described e.g. in [1]), our approach clearly relates to the ‘Activity Data Method’ developed in the 1960ies [2], in which activities are gathered as building user requirements. More recent work in recording activities as part of generic building modeling [3] directly in CAD systems has been given in [4,5]; however, process modeling only appears as future work item. Other work on office building user simulation [6] has a concept of processes; however, the main goal lies not in the simulation of planned processes but in replaying recorded behavior for the sake of understanding and predicting space utilisation in an office building environment.

Having hierarchical design methods in mind, our approach also relates to CASA [7], SEED [8] and KAAD [9]: The objectives are given in the form of processes, the design options are given as the arrangement of functions and subsystems planned by a body of collaborating professionals (architects, process planners). A difference which we clearly see here is that the named approaches clearly focus on the generation of alternative design solutions and their subsequent evaluation, which is not our goal. We try to achieve a rigorous design by justifying the required functions by processes.

As side-issue, we wish to note that the use of planning tools based on tree hierarchies for large structures is heavily debated [10]: 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 or building users think in neighborhoods [11], it is important to allocate functions according to their local perceptibility rather than efficiency. This effect also applies at smaller scales. For example, toilets 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 this level; however, the tools that would use this
data to support architectural programming are still missing. We may forward the interested reader to [12,13].

As visual representation of business processes, we have chosen a simple flowchart-like notation that could eventually be extended into the more generic Business Process Modeling Notation, which is described in detail in [14].

3. Elaboration

3.1. Re(de)fining the term ‘function’

The notion of function is ambiguous [15,16]: 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 (also see left image in Figure 1):

- **Capability**: the ability to perform a certain action in a space (e.g. capability can sit and can park bed in a hospital’s waiting area).
- **Action**: the implementation of a capability in space and time (e.g. action wait in the same waiting area that either uses can sit for patients that can walk or can park bed for reclined patients).

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.

The big difference between capability and function lies in the underlying semantics, as is shown with reference to the right part of Figure 1: A function represents a set of capabilities. The function of waiting stands for can sit and can park bed. However, a function of hygienic bed processing consisting of can park bed and can clean bed also has the very same capability park bed. Functions are therefore not disjointed, they might share some (or even all) of their capabilities. In contrast, capabilities are not further decomposable. They can be directly used in an action, as they refer to the intended work routine that underlies the function. This clarification will be vital when binding to functions by processes, or as we would now say: actions to capabilities.

3.2. Processes as seen by architects and process planners

Architects and process planners have radically different ways of expressing work routines of future building users:
Circulation paths (Figure 2, left) are used by architects to describe a work routine within the (preliminary) design.

Business processes (Figure 2, right) are used by process planners to describe processes in an abstract way, without needing a reference to the actual space where the action is to take place.

Either representation has its advantages and disadvantages: Circulation paths offer an intuitive depiction of trajectories; however, they do not state the actions out of which the process is composed. This is the domain of the business process, which names the sequence of actions that need to be carried out. However, unlike a depiction of paths on a design, business processes offer no control over the 'level of detail' or scale of each action, as process planners often mix highly detailed descriptions of process steps ("ask patient to wait", "call up patient") with very coarse ones ("perform operation") in the same process.

We argue that there is no need for two separate views on processes. In fact, the disadvantages of the one representation may be alleviated by the advantages of the other representation, as will be shown in the next subsection.

3.3. Extending the architectural schema into a schematic system

Creating a common view of processes in the architectural schema and using these to constrain function requires several extensions, which we are about to describe now in full detail. However, before starting off, we wish to briefly recapitulate what a schema is and for what it stands for, in order to be complete.

A schema (also see left in Figure 3) is a diagram that gives the rough placement of spaces and circulation of a possibly preliminary design. A space is a rectangle which stands for one or more functions that have been grouped (this is also often referred to as zoning). The rectangular form is in no way not meant to be taken literally, as the final form is to be determined separately in a phase called form finding. Circulation axes (depicted as lines with arrows) are furthermore used to give an idea of the accessibility of
As first extension of the schema, we now wish to transform functions to capabilities (refer to right part of Figure 3): Each function is substituted with its capabilities, either manually (by thinking about probable actions that are conductible in the context of the function) or automatically (with reference to a database that contains mappings from each function to the corresponding capabilities). Then, we introduce a process as ordered sequence of actions in a space, where each action is bound to at least one capability (see Figure 4). Actions are spatial – meaning that they have a concrete location. In contrast to this, capabilities are non-spatial – they are merely enumerated by their space and wait to be used by a process.

In the next extension step, we wish to take a deeper look at the hierarchies found in processes and transfer these into the schema. The constraint that each process must be contained in a space (i.e. actions may not cross space boundaries) is primarily targeted at achieving a common “level of detail” among actions in the process, as is the case with circulation paths (refer again to Figure 2, left): This type of process representation gives a common scale to all contained trajectories, as they are laid over a design, which is available in only one resolution. As additional thought, business processes have the ability to specify actions that are themselves composed of several sub-actions. When dealing with a process on a high level, all

![Figure 3. Left: An architectural schema. Right: Schema after transforming functions to capabilities.](image)

![Figure 4. A process with each action bound to at least one capability.](image)
sub-actions remain hidden, giving a hierarchy of processes within processes. We want to apply the same concepts to spaces, in the following manner (refer to Figure 5, left):

- Spaces can themselves contain sub-spaces. Each sub-space must be fully contained in the boundaries of its parent space, building up a spatial hierarchy in the form of a tree.
- The capabilities of a sub-space are inherited to the parent space (i.e. if I can operate in the operation theatre, I also have the capability to operate in the hospital). However, an action that references an inherited capability may only do so when located inside the volume of the capability's defining space (i.e. even though I can say that I can operate in the hospital, I will need to go to the operation theatre in order to perform this activity).
- Every process generates a capability by the same name in its space (i.e. the process perform operation in operation area generates a capability can perform operation). This process-generated capability is inherited to the parent space just like any other capability.

To build a process that spans multiple spaces, one goes to the parent space and creates a process that uses the inherited capabilities of the sub-spaces. As example of this, consider the process perform operation which was defined in the operation area (Figure 4). The task is now to extend this process into an emergency operation process, as seen from a global view. Going one step upwards in the hierarchy, we come to the hospital space, which has inherited all capabilities from the spaces operation area and emergency ward (Figure 5, left). Among these are the capabilities can examine and can...
perform operation, which are used in the right image of Figure 5 to actually build the new emergency operation process. Note that the usage of the inherited capabilities was restricted to the area of their defining spaces, given as dashed boundary.

Using the latter example, we have shown how a process can be composed of sub-processes. The addition of a spatial hierarchy furthermore leaves each sub-process in a defined scale, as all actions are on a common ‘level of detail’ that is defined by the containing space. As a side-note, the introduced hierarchy can also be used to effectively hide processes in subspaces, thus easing the visual load for the planner.

We have so far introduced processes in the form of actions (i.e. distinct points in the schema). What is still missing is the depiction of trajectories, which are obtainable automatically by using information about circulation axes and thresholds:

- Between each pair of consecutive actions, a directed line is drawn (left in Figure 6).
- If the line crosses boundaries of spaces, then it needs to be refined. We then have to consider all hit boundaries in a sequential manner, beginning with the one that lies nearest to the first action. If the boundary under consideration has thresholds, the line is split into segments so that it can pass over the nearest one (middle in Figure 6).
- After passing over the thresholds, we refine the line a second time. We consider each segment of the line, starting with the one that begins at the first action. If the segment under consideration crosses a circulation axis, the segment is again broken up into sub-segments, in order to be able to dock orthogonally at the circulation axis and to walk over it (right in Figure 6).

The proposed algorithm gives a unified view of business processes (actions) and circulation paths (line segments in between) in the changed architectural schema, which we wish to call schematic system, in order to distinguish it from the original diagram. The term system refers to the introduced hierarchy of spaces and processes, whereas the term schematic stands for the origins of our method.
3.4. Executing constraints and depicting warnings

Using the presented diversifications of functions and processes as well as the extensions of the architectural schema, the task of checking process model and functional model for consistency becomes trivial. If we are to find all functions without process and all processes that lack some of their required functions we need to:

- Check if all capabilities have an action that uses them: If we find a capability where this is not the case, this capability is unnecessary or must be justified by other arguments than its usage in a process. As each function is a set of capabilities, we can extend the argument further and say that a function which consists of a high number of unused capabilities indicates that it will remain underutilized. A function consisting of only unused capabilities is altogether unnecessary.

- Check if all actions refer to a capability: An action that does not have a reference to a capability remains abstract, which is not desirable – the question of “where” and “based on which function” an action will be executed must be answerable at all times. Either was the reference forgotten, or the architect has changed (or removed) the function, in which case the process is left empty-handed.

All unreferenced actions and capabilities found in the consistency check are to be brought to the attention of the planner as warning symbol (see left and middle image in Figure 7). In our preliminary implementation, warnings can be acknowledged and switched off, giving some reason as to why the function or capability does not have a reference. Furthermore, in processes, there is a type of action called condition that does not necessarily need to have a reference to a capability, since it represent a decision between two or more alternative paths in the execution and is genuinely abstract by its nature.

![Figure 7. Left: Warning sign for actions. Middle: Warning sign for capabilities. Right: Addition of icons.](image)

Besides using warning symbols, we can also use icons for actions (see Figure 7, right). This can help decrease visual load on the planner’s side and is in our view preferable to using text (especially in domains that have a standardized iconic language – think e.g. of airports). Furthermore, symbols that are normally used in flowcharting can also be utilized, which would make for a more “business process”-like look of actions.
3.5. Optional extensions of the proposed method

The nature of the circulative axes found in a schematic diagram is that they are always in the floor plane. It would, however, be nice to also have vertical circulation in the diagram (see Figure 8, left). However, since the 2D schema is unfit for depicting such a vertical circulation, we have extended schematic systems also to three dimensions. The right part of Figure 8 gives an example of such an extension, both for spaces as well as a process.

As further extension, we have looked at the introduction of process simulation directly in the diagram. The reasons for this are manifold:

- It is often the case that the usage of a process-driven building is non-equal over the run of the day. Therefore, there can be different ‘hot spots’ for processes at different times, which can only be visualized when resorting to animation.
- Process simulation may also give a hint at space requirements for areas that are directly dependent on the process (e.g. waiting areas in hospitals are directly dependent on expected queues before examination spaces).

Developing a process simulation involves defining a scheduler (see Figure 9, left) which “starts” the processes at given times, thereby putting an agent on the first action. We have looked at two possible start conditions: Time-triggered (e.g. ‘when the time has reached 1 minute’) and event-triggered (e.g. ‘when a process executes the action acute operation is needed’). Each generated agent then runs along the sequence of actions, performing additional task as he reaches an action (e.g. for junctions: decide on which action to take next). In our implementation, this behavior is momentarily given as programming code. However, there are other approaches that offer a more graphical and user-friendly interface for this task [17], which we would like to adopt in the future.

As another topic for an optional extension, we have looked at automatic categorization of circulation axes, using Bill Hillier’s Space Syntax method [18,19] (see Figure 9, right). In essence, the method can give a categorization...
of circulation from a reachability point of view. It takes a set of lines (the circulation) as input, computes a graph from it by taking the center point of the line as graph node, and connecting this to the center points of all the lines it touches (justified graph, also see middle in Figure 11). The process continues with a shortest path analysis on the graph, where the sum of all hops from one center point to every other center point in the graph is computed. The result is a number per center point, this is normalized and gives us a hint at the prominence of each circulation axis. The right part of Figure 9 shows circulations in a schematic system, scaled by their reachability (i.e. the thick circulations are ‘main circulations’ and so on).

4. Prototype, extensions and re-implementation

Our work has been initially implemented as a prototype 3D application (see left in Figure 10) which offers two distinct modes, a systems view and a process view:

- In systems view, schematic systems can be edited and attributed with capabilities. As systems are hierarchical, we have chosen to let the program focus on only one system (the active system) which we shown as a spatial container for its immediate sub-systems. The spatial hierarchy can be navigated either directly by double-clicking a sub-system (jump to subsystem) or double-clicking the active system (jump to parent), or by using a browser-like address-bar at the top of the application.

- In process view, the user can edit processes consisting of one or many actions. As each action is placed, a dialog (see right top in Figure 10) offers the opportunity to link the action to a capability.

A sanity check dialog (right bottom in Figure 10) furthermore tests the model for consistency, as it can find all capabilities that are not used by an action and vice versa. The user may react by removing the questionable actionable capability or action, or by affirming it as needed.
We have initially tested our tool using the case of a co-working scenario in a high-rise office building [20] as test bed. The task was to model a cross-corporate desk sharing environment with dynamic delivery and resupply processes on top of the three-dimensional schema of the building. As key users, we chose to involve three architects from inside our institute and one external process planner, as our emphasis was on gathering feedback rather than prematurely moving out into the field with the approach.

From the initial modeling runs, we observed that the architectonical schema with superimposed processes served both parties (architects/process planner) well as common communication media. Editing was conducted in a straightforward manner, even though the 3D interface was generally seen as cumbersome and non-intuitive. Changes on either side were accurately discovered and dealt with. A strict separation concerning who may edit what (i.e. architects edit the schema, process planners do the processes) was, however, questioned by both parties. Omitted functions and unreferenced actions were discovered in the course of the scenario, however, through the rather limited nature of the test project, no real revelations pointing to planning errors were observed. From a visual standpoint, the cluttering of the screen with actions posed severe problems. Therefore, we extended the program so as to include the possibility to selectively display processes either through switching process visibility on/off or by filtering, either of which was positively acknowledged. A further point of critique was the lacking integration of the application into other programs (CAD/process modeling package), which our users saw as essential for further re-use of the data in subsequent phases or projects.

After doing the prototype, several additional programs were written in order to explore possible solutions to shortcomings that our initial tests
had shown, or to supplement the method with more features that would truly enhance the planning of complex buildings:

- The implementation of a process simulation (see Figure 11, left) inside the schema was considered to be a natural extension by all our test users. Our architects said they would benefit from being able to see the temporal and causal usage of their functions while our process planner said that it would make sense for him to look into where a process was being utilized. As further benefit, the simulated usage of functions can reveal unobvious errors in the process model (example: functions that seem to be perfectly utilized by processes can be unnecessary if these processes are never triggered).
- Implementing the process simulation also required including automatic computation of trajectories between each pair of actions (shown in Figure 11, left), in order to send simulated persons along the circulation instead of walking straight from A to B, regardless of spatial boundaries.
- An implementation of space syntax (see Figure 11, middle) further provided us with a method of showing the prominence of each circulation axis. Although our test users were quite neutral towards this feature, we have nevertheless included it into the method as optional part, as it might turn out beneficial in the face of more complex buildings.
- Furthermore, we have also explored (but not included into the method) the use of pedestrian dynamics [21] (see Figure 11, right), so that (1.) agents will avoid bumping into each other and (2.) we can use the pedestrian flow data to visualize throughput for every trajectory in between two actions.

The sum of all findings from the initial tests and subsequent implementation of ‘explorative extensions’ has lead to the re-implementation of the prototype as 2D software (see Figure 12), which we now employ in a real hospital planning project in Upper Austria (LKH Vöcklabruck, 586 Beds). More specifically, it is our task to assist in the introduction of a novel triage system using our tool, in close cooperation...
with the hospital employees and administration. For the hospital, the most important points why a use of our approach seemed interesting were:

- The use of simulation as a means of objective comparison of different usage scenarios.
- Visualization of processes and schematic systems as means to communicate with building users, architects and process planners in an easily-understandable way.

Apart from having a design case in the hospital Vöcklabruck, we will continue evaluating and improving our approach, using our initial test candidates for feedback. After the actual design case is finished, we furthermore plan to roll out the software to a wider community (especially from the field of process planning).

5. 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 is based on the architectural schema and extends it to incorporate processes that can also be simulated. Bringing together processes and functions in a common diagram is the main benefit of our approach, as this reduces communication overhead between architects and process planners and eases presentation of the processes. Furthermore, our approach has the ability to keep functional model and process model consistent, which is of utmost importance in complex project situations we wish to address.

We have implemented our approach twice, first as 3D prototype and then as more refined 2D tool which is currently being employed in actual project work. As future extensions of the tool, we want to introduce a database for the mapping of functions to capabilities (as architects are
This database should be targeted at different areas of building design such as hospital planning, factory planning and airports. Second, we plan to integrate our tool into modeling packages (CAD / process modeling), in order to further support the workflow found in the planning of process-driven buildings.

Acknowledgements

Thanks to: Jan Michl (for diversifying functions), Georg Franck and Christa Illera (supervision), Sigrun Swoboda, Andreas Jonas, Kamyar Tavoussi (for their work) and Michael Bacher (for his hospital expertise).

References

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