Function & Action

Verifying a functional program in a game-oriented environment

Gabriel Wurzer[1], Antonio Fioravanti[2], Gianluigi Loffreda[3], Armando Trento[4]
[1]Vienna University of Technology, [2,3,4]Università degli Studi di Roma "La Sapienza"
[1](http://www.iemar.tuwien.ac.at), [2](http://www.dau.uniroma1.it)
[1]wurzer@iemar.tuwien.ac.at, [2]antonio.fioravanti@uniroma1.it, [3]gianluigi.loffreda@uniroma1.it, [4]armando.trento@uniroma1.it

Abstract. *The finding of a functional program for any kind of building involves a great amount of knowledge about the behavior of future building users. This knowledge can be gathered by looking at relevant building literature (Adler, 1999; Neufert and Neufert, 2000) or by investigating the actual processes taking place in similar environments, the latter being demonstrated e.g. by (Schütte-Lihotzky, 2004) or new functionalist approaches of the MVRDV group (Costanzo, 2006)). Both techniques have the disadvantage that the architect might assume a behavior which is seldom experienced in real life (either through lack of information or by failing to meet the building user's expectations). What is needed is a verification step in which the design is tested on real users. We have devised a game-like environment (Figure 1a) in which it is possible to capture the behavior of future building users in order to verify the relevance of the design even at a very early stage. As result of applying our approach, we can find previously overlooked usage situations, which may be used to further adapt the design to the user's needs.*
Keywords. *Requirements Checking, Participative Design.*

Introduction

Lawson states that *"...the best test of most design is to wait and see how well it works in practice"* (Lawson, 1997). We feel that this 'reality check' has to be done well before the building goes into operation, in order to meet the future building users expectations. It is an often quoted dilemma that people cannot state their expectations about the daily life routine in a new building without reference to some sort of solution. As architect, taking assumptions in the form of an initial functional program is good, what is missing is a validation step in which the preliminary design is verified on real users.

We have devised a game-like environment in which users can enter their would-be behavior in a planned building, based on their normal work and life routine. Our key goal is

- to let building users *provide feedback on the presented design* (e.g. what they think is still missing, see page 391)
- to be able to *tell which planned functions are actually used* (see page 393) and at what time or because of what cause

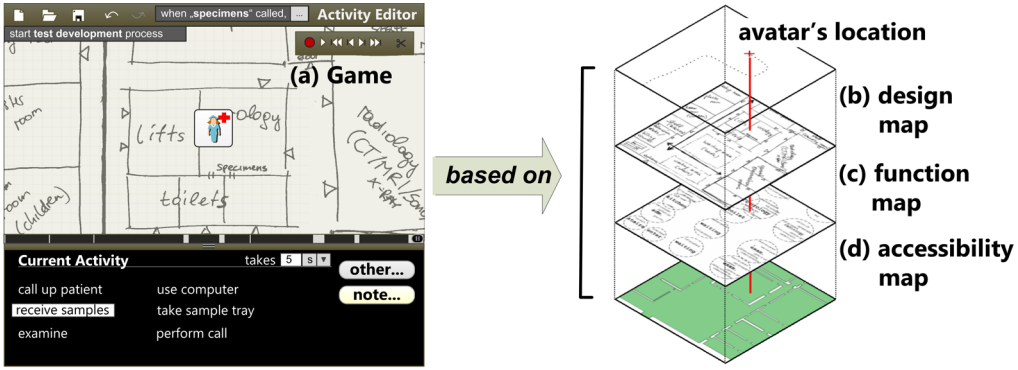

Figure 1
(a) Game Environment based on (b) preliminary design, (c) function map (d) accessibility map to record behavior of user.

As game setting, we take the preliminary design (Figure 1b) which the architect has annotated with function map (refer to Figure 1c): For us, functions are spatial and have an influence boundary in which they can be utilized. We adopt the notion of function as proposed in (Wurzer, 2009), in which functions are further detailed into basic elements of work routine (e.g. kitchen into *cook, prepare, store* etc.). As final input into our program, we use an accessibility map (Figure 1d) that specifies what pieces of the design can be walked on.

Given a goal verbally (e.g. "go through your working day"), a future building user may now walk his virtual representation through the design, being able to perform *actions* as he goes along. The actions the user can perform in any location correspond to *all functions in close proximity* (i.e. functions that have the player's position contained in their boundary). Should none of these functions fit the desired activity of the user, he can enter one himself ("other"-button in Figure 1a). Furthermore, the player might leave a comment for the architect at his current location ("note"-button), similar to what e.g. (Moloney, 2002) proposed in his "String Collaborative Virtual Environment".

Another important piece of information about the building user's behavior is the time or event that triggers it (e.g. "at 11am", "from 9am to 6pm", "when a call comes in"). This is entered every time the user wishes to create a new behavior. As a matter of fact, we can infer temporal and causal usage of functions, which would otherwise be hard to draw from just using the functional program.

Contribution, benefits and applicability of our approach

Out of all behaviors that have been entered by future building users, we can infer the *usage of functions* both temporally and causally (see page 393). The architect can view the behaviors on top of his design and can check whether his initial assumptions concerning usage were correct. Furthermore, he may notice which functions remain rarely used and which functions seem to be missing. The latter is done by taking all actions for which the user has entered a function himself.

It shall be noted that the process of testing a design should begin in a planning stage which still brings the possibility to adapt some of the planned functions (or else it would make no sense to verify). As input, we take the design (CAD drawing or image, concrete or sketch) that is manually attributed with functions by the architect. The work on the side of the building users - going through their daily work routines, specifying when each part of the work routine starts etc., can be done in parallel to the further planning work. After that, we require an analysis

phase (see page 393) from the architect's side to view the gathered data, go through it and check the initial assumptions made in the design. This process can be iterated – or adapted to gradually bring more details into the work routine (e.g. users first give their high-level tasks, then go to special cases in subsequent phases).

Evolution and Related Work

The design of the game environment proposed herein is a natural extension of recent work that lets planners enter planned processes in an architectural schema (Wurzer, 2009). However, those processes are only correct if the planner can correctly anticipate the usage of the building by different building user groups. Because this can be a daunting task, we thought of an approach that could be used by the building users themselves to enter their activities, either alone (i.e. in an unattended game session) or in a workshop environment.

Apart from our previous work, the possibility to enter activities and behavior in a CAD environment has previously also been proposed by (Ekholm, 2001). However, the novelty in this case is the questioning of the building user as means of participative design, as this has (to the best of our knowledge) not been presented before. We are aware of approaches that use game technologies to measure user satisfaction on design variations (Orzechowski, Timmermanns and de Vries, 2003; Orzechowski, Timmermanns and de Vries, 2000), however, these seem to be of more use in the late stages of the design process when the final form is being generated. Others have tried to formalize the future building user's requirements as rules and to execute these in a game environment (Trento, Loffreda and Kınayoğlu, 2009). However, as the authors conclude, this formalization involves a lot of design knowledge, which the users may not possess. A mediator (program or human) should aid in translating the user's preferences into formal rules. We definitively support these considerations, as an analysis step is always needed after data (being it rules or behavior) has been gathered, which we also do in this approach.

Apart from architectural planning and verification, there are other fields also pointing into the direction of developing a more graphical questionnaire for paths taken and actions conducted: NIST has conducted a survey with occupants of a 32-story high-rise office building who had to evacuate due to a fire incident (Kuligowski and Hoskins, 2009). One of the main interesting points asked was the set of specific actions taken by the respondents. However, as the survey was form-based, there was no hint on "where" actions were performed. If a location-based interviewing tool had been available, survey data would have been more context-specific. Another important aspect that has lead to the invention of the tool is the sampling of usage for simulation. However, we will not cover this aspect in this paper, as we rather rely on using statistical (i.e. spreadsheet) analysis of the entered behaviors. Also interesting is the relation of this work to storytelling approaches such as (Kelleher, 2006), the true difference being that we want to enable a user to tell a story about himself, not somebody else.

Description of the game environment

Entering behavior (building user)

As seen in Figure 1a, the game environment is a tool that basically offers an avatar centered on a design. This avatar is then moved over the design with the cursor keys, taking the accessibility map (spots on which walking is permitted) into accound. At every location of the design, the program furthermore knows of utilizable functions using the underlying function map. Accordingly, it will update the user interface in which the user can choose what function to call, or put differently: what action to perform. The separation between function and action is crucial to us (also refer to Wurzer, 2009): While the function is defined by the architect, defining "what could be done in every space", the action is an actual

instantiation of a function (i.e. a function that was put into effect). A user may enter a different action than the ones that are suggested, therefore indicating that the functional program was somehow incomplete. Furthermore, one may give feedback for the avatar's current standing position as a note on the map , which appears as a label.

As the user moves the avatar over the layout map, all steps which he does are recorded. In the user interface, the corresponding functionality is included as a metaphor in the form of *a video recorder*: Pressing the "new" button starts a new (empty) process. It is noteworthy that we have the clear concept of allowing only one process to be edited at a time, of which we display the name in the top-left corner. Toggling the "record" button records the movements of the user. The timeline at the middle of the screen gives the current position within the process that is described. Once the user has toggled the "record" button, he may not move his avatar further without having record on (i.e. we do not allow for jumps in the process, it has to be continuous). At any time, the user may drag the time slider at the bottom of the screen to a certain instance in the recorded time, thereby updating the position of the avatar. The user can insert behavior from the current time onward using again "record". Furthermore, as in film, we have the ability to cut at a specified position, using the "scissors" button. The cut will range from the current position to the end, having again the ability to add new behavior at the end.

The play button will play back the process beginning at the current time. Should the current time indicator surpass the recorded duration, the time indicator is reset to the begin of the process and the playback continued. Furthermore, we have "jump to start" and "jump to end of behavior" buttons to ease temporal navigation. A process may be saved by using the "load" and "save" buttons at the top of the tool.

If the user selects a function from the list of functions at the very bottom of the screen, the process inserts a new action into the process. The action is similar to a key frame in animation – it is a fixed point in the behavior that executes a function. As per default, each action has is one second long. However, the user has the opportunity to specify how long it takes by specifying a value in the "takes" editing field. One can jump to the next and previous action by pressing the "next" and "previous" button. Furthermore, as the time slider is dragged on an action, the used function becomes selected in the functions view at the bottom of the screen (and can be changed to another function if the user wishes).

Besides actions, *notes* can be left at the avatar's current position using the "note" button at the lower right part of the screen. They are a valuable tool for communication between architect and building user, since they can give a connotative feature to the process (from a metaphoric standpoint very similar to *subtitles*). Once created, notes can be selected and edited using the mouse.

To end with, each process has a starting condition. This is either (1.) a time when the process is scheduled to start, or (2.) an event which triggers it. Events can either be the execution of a function or the performing of an action by a named process.

- If a time is given as process start criteria, an according dialog appears. The user can determine whether to enter a range (e.g. 1am-2am) or a single time stamp.
- If a function is taken as starting condition, the user has to select from a dropdown list which function should trigger the process. If there still is no such function, the user can enter a new one.
- If an action is taken as starting condition, the user is presented with a list of all processes and contained actions and selects one from it. Two processes that have the actions of the same name are handled as different (internally, the process name is prepended when issuing the event).

Behavior analysis (architect)

The architect can perform analysis in the application in multiple ways:

- By loading the activities the users have entered into the application depicting the trajectory and function usage on the design (static image, no simulation or animation of the routes) and comparing this to the initial assumption over the building user's behavior.
- He can see functions with no usage in any process displayed prominently on the map. These functions are likely to be unnecessary – i.e. they are candidates for removal.
- Vice versa, functions which have been entered by the user himself (because there was no function that matched the action in the mind of the user) can also be shown prominently, as they give a hint at forgotten usages.
- The notes that were entered by the building users can also be shown on the design.

Furthermore, there are cases when a usage itself is to be questioned – the interesting questions being *when and why* this behavior occurs. Using the entered starting conditions for every behavior as basis, we can generate a report (standard spreadsheet) containing a specific view on the data:

- Outputting times at which processes start can be a hint at a possible incompleteness in the description of the usage of the building (*when*).
- We can also output for each process the corresponding start function or action (*why*). This can be beneficial when there process is complex and its cause is not clearly known (being part of the daily work routine does not necessarily mean that it is reasonable).
- Exporting quantitative measures can furthermore give a measure of importance for each function. The key figures we use are *how many processes a function starts* and *how often it is used in a process*.

Limitations

We have certain limitations in our approach that we know can annoy advanced users, but have novices enter processes in a more intuitive way. First, when using our cutting tool during behavior editing, we discard everything in the timeline to the very end of the recording. Second, we have no notion of junctions in behavior (i.e. "alternative behavior" that executes from a certain time on). This could, in principle, be beneficial from the planner's point of view (i.e. seeing all alternative behavior at once), however, we have left it away since alternative behavior can be decomposed into several behaviors. Third, actions can only be attributed durations by stopping the recording and getting back to the last action and then setting the duration to the correct value (the time is inserted into the timeline). Last, we cannot distinguish between a note left because of thoughts concerning the process and a note left because of the preliminary design. Therefore, clicking on a note has to bring the corresponding process up, in order to provide further context.

Performing User Tests

We have developed our ideas by using mock-up testing on five un-trained candidates with no background in architecture or CAD. In detail, we took a cardboard frame with an attached player figure in the middle and laid it over a large design of a high-rise office building. In order simulate the games world moving, we then subsequently moved the design. Furthermore, flash-cards corresponding to the functions at each locality in the map were laid before the candidates, with the possibility to choose one so as to perform an action. Recording, playing back, cutting and inserting behavior in close correspondence to a video recorder was taken up after a temporal list of actions failed to convince. Another feature originating from input by our users was the possibility to leave notes or feedback at locations on the map.

For the analysis part, we have performed mock-up tests with two architects being the subjects, taking only the graphical part (missing functions / unnecessary functions / notes) into account. We could, however, not observe accumulations of missing functions but rather isolated cases. A variety of such

missing functions were proposed, not all legitimate, and some through misinterpretations of existing functions. Clearly, there has to be an analysis phase for guaranteeing validity of the entered data.

As next step after the mockups, we are currently in the phase of programming the actual application (a web application in Adobe flex) which will be ready until the eCAADe takes place in September. Until then, also, data collection in the context of actual project work (in the hospital domain) will be gathered in order to have a real case to put forward.

Concluding

We contribute a novel approach that uses a game environment to let prospective building users walk an *avatar* through the preliminary design, performing their daily work routine as they go along. Each action within the work routine is the invocation of a function from the pre-defined functional program of the building. Should the user want to perform an action for which he cannot invoke a function, he creates one himself (*missing function*). Functions which were not used in any work routine, on the other hand, are probably not needed (*unnecessary function*). As benefit, both of these situations can be brought to the attention of the architect even in a very early stage. Furthermore, our approach may also be used for the *temporal* and *causal* origins of usage scenarios.

References

Adler, D. and Littlefield, D.: 2007, Metric Handbook – Planning and Design Data (3rd Edition), Architectural Press, Oxford.

Neufert, E. and Neufert, P.: 2000, Architect's Data (3rd Edition), Blackwell Publishing, Malden.

Schütte-Lihotzky, M.: 2004, Warum ich Architektin wurde, Residenz-Verlag, Salzburg.

Costanzo, M.: 2006, MVRDV: Works and Projects 1991-2006, Skira.

Lawson, B.: 1997, How Designers Think (3rd Edition), Butterworth Architecture, London.

Wurzer, G.: 2009, Systems – Constraining Functions Through Processes (and Vice Versa), in G. Cagdas and B. Colakoglu (eds), Proceedings of the 27th eCAADe,
Istanbul, pp. 659-664.

Moloney, J.: 2002, String CVE: Collaborative Virtual Environment software developed from a game engine, in K. Koszewski and S. Wrona (eds), Proceedings of the 20th eCAADe, Warsaw, pp. 522-525.

Ekholm, A.: 2001, Activity objects in CAD-programs for building design, in B. de Vries, J. van Leeuwen and H. Achten (eds), Proceedings of the Ninth International Conference on Computer Aided Architectural Design Futures, Kluwer Academic Publishing, Dordrecht, pp. 61-74.

Orzechowski, M., Timmermans, H. and de Vries, B.: 2003, Virtual Reality CAD system for non-designers, in Proceedings of the 7th Iberoamerican Congress of Digital Graphics, Rosario, pp. 133-137.

Orzechowski, M., Timmermans, H. and de Vries, B.: 2000, Measuring user satisfaction for design variations through virtual reality, in H. Timmermans and B. de Vries (eds) Design & Decision Support Systems in Architecture - Proceedings of the 5th International Conference, Nijkerk, pp. 278-288.

Trento, A., Loffreda, G. and Kınayoğlu, G.: 2009, Participative Technologies: an Internet-based Environment to Access a Plural Design Experience: Knowledge Modeling to Support User's Requirements Formalization, in G. Cagdas and B. Colakoglu (eds), Proceedings of the 27th eCAADe, Istanbul, pp. 515-522.

Kuligowski, E. and Hoskins, B.: 2010, Analysis of Occupant Behavior during a High-rise Office Building Fire, in Proceedings of PED 2010 (to appear), Gaithersburg.

Kelleher, C., 2006, Motivating Programming: Using storytelling to make computer programming attractive to middle school girls, PhD Dissertation, Carnegie Mellon University.