Design phase transitions in object-oriented modeling of architecture

Issues in the development of IDEA+

BOEYKENS Stefan, GEEBELEN Benjamin, NEUCKERMANS Herman

Catholic University of Leuven, Department of Architecture, Belgium
http://asro.sbuild.com - http://www.asro.kuleuven.ac.be

The project IDEA+ aims to develop an "Integrated Design Environment for Architecture". Its goal is providing a tool for the designer-architect that can be of assistance in the early-design phases. It should provide the possibility to perform tests (like heat or cost calculations) and simple simulations in the different (early) design phases, without the need for a fully detailed design or remodeling in a different application. The test for daylighting is already in development (Geebelen, to be published).

The conceptual foundation for this design environment has been laid out in a scheme in which different design phases and scales are defined, together with appropriate tests at the different levels (Neuckermans, 1992). It is a translation of the "designerly" way of thinking of the architect (Cross, 1982). This conceptual model has been translated into a "Core Object Model" (Hendricx, 2000), which defines a structured object model to describe the necessary building model. These developments form the theoretical basis for the implementation of IDEA+ (both the data structure & prototype software), which is currently in progress.

The research project addresses some issues, which are at the forefront of the architect's interest while designing with CAAD. These are treated from the point of view of a practicing architect.

***Keywords**: CAAD; design; modeling; concept.*

IDEA+ – Why develop a new data structure?

The project IDEA+ didn't use any of the existing building models from recent research efforts (such as BAS°CAAD, SEED, KAAD, VEGA, BCCM and COMBINE2). Based on their evaluation we had some concerns (Hendricx, 2000):

- Most research projects don't elaborate on methodology.
- The data in these models often does not allow dynamic behavior of objects.
- Although most of these projects understand the importance of non-physical objects such as "Spaces" and "Activities", they all handle them differently. Therefore it was not possible to choose one of them as a clear "winner".

The "Core Object Model" was then developed as a "theoretical" framework that didn't rely on a rigid data structure, but is especially targeted at the

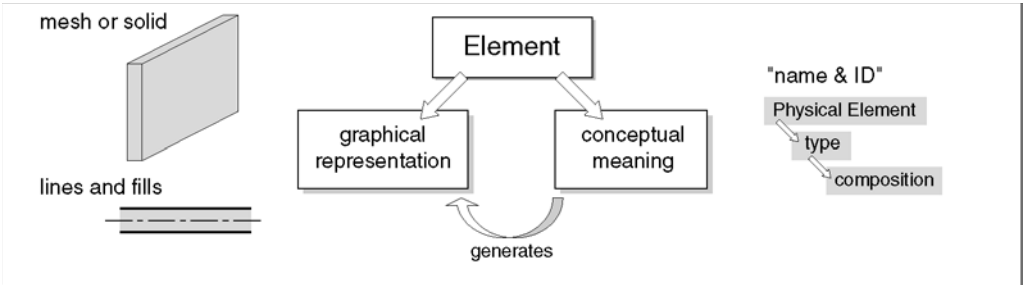

Figure 1. Separation between conceptual and graphical information.

design of a building model in the evolving workflow of an architect/designer.

Design phase transitions and scale transitions

One of the most important issues that will be investigated within IDEA+ is how the digital model is capable of following the evolution in the design model. In the course of a design activity, the architect can start, for instance, with a global mass study and elaborate this into a more detailed solution or, on the contrary, depart from some specific construction system and extend this into a larger layout. Instead of making two or more independent modeling studies, we would like to enable this transition in the data structure, without losing the possibility of a reverse transition, because the design process is not strictly straight forward.

The data structure is explained in (Hendricx, 2000). It makes a clear distinction between the conceptual and the graphical information for a particular element.

The graphical representation (using lines, surfaces or solids) will be automatically generated by the conceptual entities, which contain the real data. Each conceptual entity is an abstract object (such as physical element, space, masterplan block, grid etc…). Each has a name and ID and provides basic functionality for linking and making a hierarchy. By assigning a certain "type" to this conceptual entity, its real functionality is chosen (such as Wall or Floor for Physical Elements and UserSpace for Spaces). The type contains the instance-specific information (e.g. size, length, height, control points) together with the behavior of this particular kind of object. By choosing a "composition", information that is shared between different elements can be added (such as its internal set of material layers).

We distinguish between two kinds of transitions: design phase transitions and scale level transitions. These transitions have implications on the data structure to be developed.

Design phase transitions

Switching between design phases does not add or remove entities. It merely changes the type

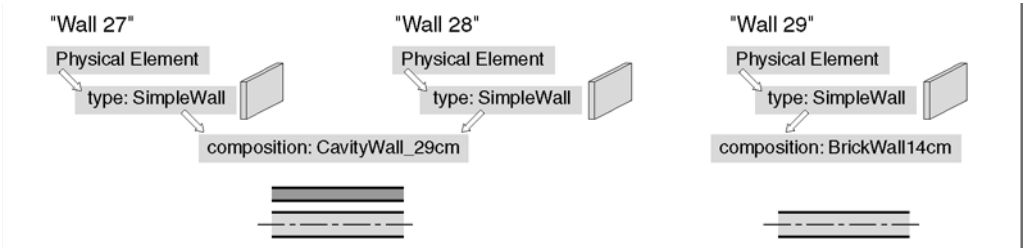

Figure 2. Different types can share the same composition.

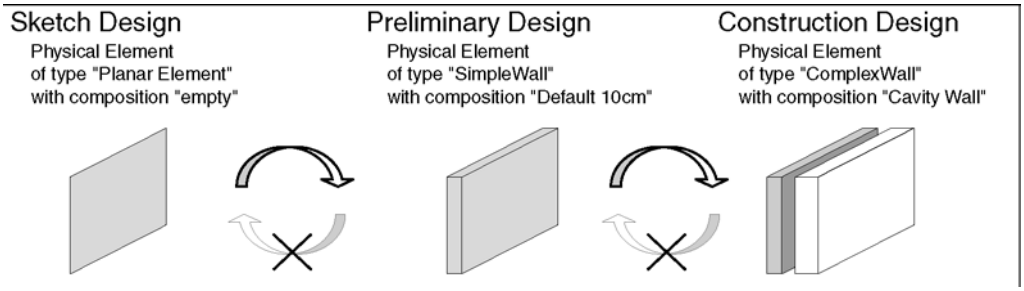

Figure 3. Transition from one design phase to the next for one particular level of scale

of an element and possibly chooses another composition. The basic type "Planar Element" is specified into a WallType (e.g. "SimpleWall", which only sets starting and ending point and height). Other possible types are FloorType, FoundationType, BeamType, and ColumnType. Theoretically, the data structure allows the modification of a Physical Element from one type to a completely different one, but the application will need to limit this to meaningful changes only (preferably by allowing the user to choose from a list of valid types).

A default composition is chosen, which sets the element's material and its overall thickness. Different elements can have the same composition, but the type is unique for one instance. The designer thus adds information to the Physical Element, without really changing it.

In the next step, the transition from the Preliminary Design into the Construction Design, the type is changed into a "Complex Wall", which can have more information (such as offsets for the different wall layers). The composition will also change.

These steps in the transition mimic the usual workflow from a designer's point of view.

The reverse transition, which is depicted in figure 3, is not obtainable by merely undoing (or reversing) these type and composition changes. Switching back from a WallType to a Planar Element Type would mean a loss of information, because our types contain instance-specific data. The designer will add information, but will preferably not remove this information. However, when the representation level is changed (to show another design phase), elements will seemingly switch back to a more simple type, because generating graphical entities for that particular representation will draw these elements differently: a complex wall that is shown in the sketch design phase can be represented as a surface without thickness, even when the wall element itself contains this information. As such, in the eyes of the designer, the reverse transition is achievable (without losing information).

This is only possible because we made a complete separation between any graphical representation and the actual building element's data. As such, the data structure doesn't depend on a graphical kernel or a particular CAD-tool. It can use solids, or surface models or even simple lines. The data-structure could even function using an existing CAD- or modeling application.

Scale level transitions

The second type of transition is between scale levels. We distinguish three scale levels: the masterplan, the block and the space (Neuckermans, 1992).

In a top-down approach, the designer might start on the "Masterplan level" and design complete buildings (e.g. urban planning, site design). By switching to the "Block level", the space entity (in this case a Masterplan block, possibly represented by a primitive volume) will be hidden (not deleted) and separate spaces are added into the design (such as different rooms or building stories). The sides of the volume will generate "Planar Elements"

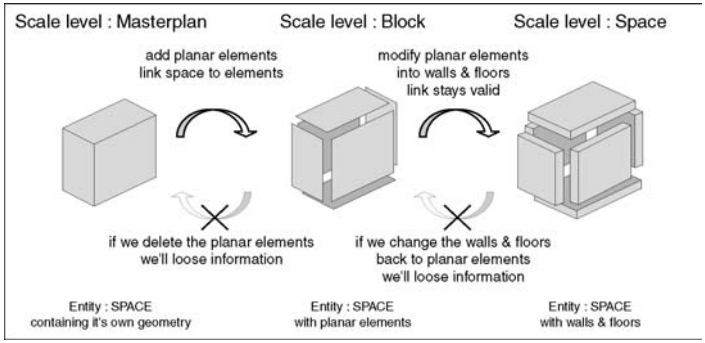

Figure 4. Transition between different levels of scale.

(flat surfaces, without further specification). The existing space will be linked to these planar elements, so changes in layout will affect both the planar elements and the space. When the designer switches to the "Space level", the planar elements are replaced by walls and floors and thus obtain more information.

The reverse transition, the bottom-up strategy, however, poses a problem: we would like to be able to go back and forth between these levels, without losing the valuable information from the lower levels. Therefore we cannot change the walls and floors back to simple planar elements. The solution lies in the representation. It can generate a more simple set of graphical entities (omitting thickness and/or openings), from the detailed elements with their specific compositions. When switching back to the Masterplan level, we can ignore floors and walls and thickness and only represent the contour of the building block: a union of the different spaces.

A design test (such as cost calculation) on the Masterplan level can thus generate a more detailed result, when all the information of the underlying levels has been set up, compared to doing this test in a top-down strategy, where no information about lower levels is known.

A wizard-like approach

The transition of one phase or scale into another and the possibility of a reverse transition cannot be a fully automatic process. The tools to be developed will provide a wizard-like approach, where the architect is presented choices, allowing the translation of his or her design decisions into the technical realization of a project.

One of the difficult choices the system cannot automatically generate is the positioning of elements, when a thickness is entered.

Conclusion

The project IDEA+ defines a new building model and data structure, targeted at early-design stages and allowing transitions between different design stages.

These transitions allow the model to follow the conceptual workflow of an architect/designer. Transitions can happen in opposite directions. The representation is responsible for hiding unneeded information (but also for preventing the loss of this valuable information).

References

Cross, N.: 1982, Designerly ways of knowing, Design Studies, 3(4), pp. 221–227.

Geebelen, B.: to be published, IDEA-I daylighting test for IDEA+, PhD thesis.

Hendricx, A.: 2000, A Core Object Model for Architectural Design, PhD thesis

Neuckermans, H.: 1992, A conceptual model for CAAD, Automation in construction, 1(1), pp. 1–6.