FREAC: A Technical Introduction to a Framework for Enhancing Research in Architectural Design and Communication

Reinhard Koenig¹, Torsten Thurow², Jörg Braunes³, Christian Tonn⁴, Dirk Donath⁵, Sven Schneider⁶
¹-⁵ Bauhaus-University Weimar, Germany, ⁶ Technical University Munich, Germany
¹-⁵ http://infar.architektur.uni-weimar.de/infar/index.html, ⁶ http://www.caad.ar.tum.de/
¹ reinhard.koenig@uni-weimar.de, ² torsten.thurow@uni-weimar.de, ³ joerg.braunes@uni-weimar.de, ⁴ christian.tonn@uni-weimar.de, ⁵ donath@uni-weimar.de, ⁶ schneider@caad.ar.tum.de

Abstract. This paper describes a framework for a collaborative, dynamically modifiable product model called FREAC built for the purposes of experimental software development. When developing FREAC, we attempted to realise the following properties that are typically lacking in currently available commercial systems: first, a high degree of flexibility so that it is highly adaptable to the needs of different disciplines; second, the ability to seamlessly connect different tools; third, real-time concurrent modelling by different remote partners; fourth, the ability to save a record of the entire modelling process; fifth, dynamic extensibility both for software developers as well as for the end users of the respective tools.

The term FREAC encompasses both the framework for developing and managing a product model (FREAC-development) as well as the tools developed to work with it (FREAC-tools).

Keywords: software development; experimental platform; product model; digital building model.

Introduction

Research in the field of Computer Aided Architectural Design (CAAD) consists not only in analysing, evaluating and classifying existing digital systems but more importantly in the development of new concepts and ideas for future software developments. Ideally these concepts are developed to a prototypical state for testing under real conditions. One possibility of realising such prototypes is to use commercially available systems as a basis and extend these with the necessary functionality. In many cases, however, commercial systems are – despite their extensive functionality – often too rigid and inflexible for the demands of new concepts such as interactive and generative design systems, open design methods or digital building surveying. Another means of realising prototypes lies in the development and programming of own tools from scratch. Aside from the not inconsiderable programming
task involved, a problem typical of proprietary solutions is a lack of compatibility with other programs and tools. Typically such tools are therefore developed to provide functionality sufficient to serve a specific purpose which in turn limits the scope of evaluation possible.

This situation shows clearly that for experimental software development in the field of CAAD research, there is a need for a supporting framework with which one can link together different software tools. This paper describes a framework for a collaborative, dynamically modifiable product model called FREAC that fulfils the above requirements and has already been largely implemented. The term FREAC encompasses both the framework for developing and managing a product model (FREAC development) as well as the tools developed to work with it (FREAC tools). FREAC was developed as an experimental research platform and numerous tools have already been built to work with it.

The concept of a dynamic product model

Current CAAD systems employ building information models to internally organise their data. These typically provide a fixed organisational structure with spaces and building element objects along with their respective geometric form. At present, these systems cannot be adapted in terms of their structure or geometric form to suit individual users or projects. However, to realise new software concepts and building typologies, this is absolutely necessary.

The core of the FREAC development consists, therefore, of a dynamically modifiable and extensible product model [1]. Individual aspects of the product model can be represented by a constellation of classes and objects which in turn consists of attributes and methods (according to object-oriented programming principles [2]). By definition, a dynamic model is a system with a structure that can be modified or extended. Such modifications include the addition or modification of classes, attributes and methods.

There are number of different approaches to realising a dynamic product model and it is necessary to take into account the degree of flexibility and performance requirements of the model:

- The overall flexibility of the entire system, i.e. the ability to change attributes and methods during run time,
- aspects where speed is an issue, for example for geometric presentation or for highly complex numeric operations,
- efficient use of memory, especially with regard to the geometric representation of larger constructions or highly detailed areas,
- the level of user expertise required to adapt the organisational structures.

These requirements help determine the relationship between flexibility on the one hand and performance – speed of operation and effective use of memory – on the other. To fulfil the greatest possible requirements, a heterogeneous system concept was proposed which allows programmers and users to adapt the system at different levels. The proposed approach differentiates between three different groups of persons [1]:

- Programmers
- Administrators
- Users

The programmers construct primarily submodels such as for representing geometries, describing the spatial and building element structure, structural calculations etc. The submodels are typically programmed in a high-level language such as C++ ensuring high performance and efficient memory usage. The dynamic adaptability of the submodels is comparatively limited and determined on compilation of the code. The constellation of submodels is known as the application module (Figure 1 Structure of the distributed product model as a system of shells).

Administrators can adapt the product model to the respective task at hand. Adaptations include the incorporation of new submodels which can then be subsequently modified and extended.
Interfaces provide a means of connecting the fixed programmed submodels with extensions to the attributes and methods made by users in the form of scripts. The application module together with administrative adaptations or extensions represent the basis scheme made available to the end users. At an application level, the users fill the model with the respective data. Changes to the product model are not possible at the level of the basis scheme.

The submodels developed up to now use the distributed product model as a building information model. The FREAC core can in principle also be used in other fields such as product design or urban design and can therefore be understood as a distributed product model.

The users of the model shown are typically not in the same location but distributed across individual offices or institutions. This necessitates the use of a distributed product model (cf. [3]) that makes it possible to access the model “online”. Ideally data should be retrieved from the central data repository, however, because this is not always possible, “offline” synchronisation techniques are necessary that facilitate the splitting and later merging of model data. FREAC uses techniques such as transactions and versioning using online-synchronisation to store the history of the development of the model, thereby making it possible to work in parallel on the same model. A FREAC model is modified on a local computer using a client application that serves a particular task. All clients synchronise their model data with a version stored on a central server. Using a client concept, it is possible to facilitate the seamless networking of different projects which can be worked on independently of one another based on a common model. The almost unlimited extensibility of the framework at both a user and developer level means that FREAC offers an approach for bringing together hitherto insular software solutions to form a continuous ‘continent’ of digital systems.

**The technical principles of FREAC**

**Model synchronisation**

In the basic principle behind FREAC, the synchronisation of modifications made to a data model are effected online in the form of short transactions – each completed action on a local model within a client is transmitted once via TCP/IP to the server model and from there to all other clients. This approach makes it possible to work quasi in parallel on a model by effecting sequential changes and obviates the need for merging mechanisms but limits the number of clients working in parallel and makes it necessary to keep the size of transactions small. This also means that there needs to be a permanent network...
connection with the server. The underlying principle of persistent data recording does, however, make it possible to fork local versions which could then be integrated offline using merging approaches.

The data of the FREAC-models is stored in parallel on the server and clients. This approach makes it possible to reduce load on the network, as only changes to the model need to be transmitted. In general, most of the object retrievals are reading actions so that, in contrast to remote retrieval methods, most operations need only access the local copy of the model.

A crucial aspect in the design of model management systems is the size of the model to be managed. The use of conventional databases makes it possible to work on very large models as the model size is not limited by available RAM. However the speed with which one can access content within the database is relatively slow in comparison to data models that are stored in their entirety in memory. The approach used by FREAC is a compromise of both concepts. This compromise means that there is a limit to the overall size of the model but that retrieval speed is much faster, compared with a database solution, as the content of objects can be held in memory. Furthermore, objects are only loaded into memory when their content is accessed for the first time. It is also possible to swap data out of active memory at any time when the respective object is no longer being accessed.

**Dynamic model structure**

In the context of the FREAC model structure, dynamic means that new objects can be added to an existing data structure at any time, that the properties of an object can be extended and that new methods of working on objects can be devised. The dynamic structure refers, therefore, both to the model structure as well as to the algorithms built off it, although we focus here primarily on extensibility. A dynamic product model of this kind, as mentioned earlier, is particularly important because in the case of complex artefacts, it is not possible to know all elements, their properties and interactions in advance. This applies not least to the representation of buildings and structures. In most cases each new planning or design process brings new special challenges.

FREAC is built using Microsoft's .NET software platform. FREAC modules with high performance or memory requirements are, however, implemented in unmanaged code. .NET offers a number of different features that are well-suited for realising such aforementioned dynamic systems. For example, the reflection mechanism provides information on available classes and data structures. Additionally, new assemblies (Dynamic Link Libraries, DLLs) can be added during runtime, e.g. without restarting the system. This possibility is used as a basis for creating new sub-models in FREAC.

FREAC extends the basic functionality of .NET by making it possible to extend objects and attributes during runtime and by facilitating their real-time monitoring. Here monitoring means that each object and attribute that is added in real time is able to inform other objects that are monitoring them about any changes. Monitoring is a fundamental mechanism for realising the dynamic extensibility of a model. A further mechanism is the addition of sub-models in real time in the form of new assemblies (DLLs).

The dynamic FREAC model structure will in future not just be for programmers but also for administrators and users. Here too .NET offers the advantage that different programming languages are available within this software platform. Because one cannot expect end users to have programming knowledge, visual programming or scripting methods are more suitable for the user level.

**Examples of use**

The following section describes some of the software prototypes (FREAC tools) that demonstrate the functionality of the FREAC model structure. The prototypes shown include test applications created to verify certain kinds of functionality as well as
advanced applications developed as part of research activities. Each of the applications is described in brief along with an explanation of its incorporation in FREAC development.

**FREAC Tools**

One of the FREAC tools is “Colored Architecture” (Tonn et al., 2006). This application aims to improve on the deficiencies of digital colour scheme and materials design and supports the entire design process from the initial planning phase right through to detailing. “Colored Architecture” adapts familiar approaches, means of representation and tools from existing planning practice such as colour scheme variants and studies, colour harmonies and colour contrasts, and supports designing with different material surface qualities. For the evaluation and assessment of colour and material concepts, a live-radiosity visualisation system was developed. The position of the sun, degree of cloud cover and the colours of the surfaces can be adjusted interactively and the near-real radiosity visualisation can be viewed immediately. The software also makes it possible to visualise and assess the interaction of different surfaces in real-time, for example colour reflections between different materials (Figure 2).

A further FREAC tool called “SketchClient” (Figure 3) was created for the rapid and simple modelling of 3D geometries. It provides simple polygonal modelling and editing functions. Each change to the model is saved as a version and represented in the form of a Versions-Graph (Figure 4). This makes it possible to rapidly switch between different design variants or to return to an earlier work stage. Each node of the versions graph contains a protocol of the user who made the change, when it took place and how many objects were affected. This versioning system is not limited to an individual client but is provided centrally by the FREAC model structure.

The third example is not a self-contained client

![Live-radiosity visualisation in “Colored Architecture”](image)
but an extended submodel that displays the spatial and building object structure of a building. This submodel is currently in development and for the most part follows a structure very similar to the IFC. Within the client, the building structure is represented in a tree view with the structure organised in terms of Project --> Site --> Building --> Storey --> Space or Building Element Category. The individual abstract elements of the spatial and building element model are linked with the respective 3D geometric items in the model.

**Submodel interaction**

Depending on the particular application, the individual FREAC tools use different submodels that fulfil specific tasks. “ColoredArchitecture”, for example, uses the “GeometryModel” to show the 3D geometry. The functionality for the colour and material choices as well as the live radiosity calculations are implemented in the “ColorModel”. Conversely, the “SketchClient” only makes use of the “GeometryModel”. The semantic representation of a building also employs the “GeometryModel” in conjunction with the “BuildingElementModel”. All the submodels and interactions between the clients are managed by the “SyncServer”, which transfers only the changes to a model between client and server (Figure 5). The clients only “see” the models, or the objects...
stored in the models, that are required for their specific purpose. Changes to the content of the model – in contrast to a repository – are communicated in real time to all the clients.

As explained above in the technical section of this article, the dynamic model structure of the FREAC development makes it possible to augment and adapt submodels at any time without needing to recompile any existing submodels. As a result, the set of FREAC tools can be extended by new software prototypes for new application areas. Newly developed clients can use both existing as well as new submodels.

**Outlook**

The aim of developing FREAC is to establish a flexible platform for different research projects in the field of Computer Aided Architectural Design. In a series of accompanying research projects, diverse clients have been and are being developed: computer-based building surveying, for different planning aspects such as colour planning, the potential maximisation of plot use, for generative architecture layouts and for coordination and communication in open collaborative design processes. The clients created in these projects are seamlessly connected with one another via the FREAC model concept. They demonstrate the potential of the approach described here for facilitating an exchange of expertise between different professional disciplines in the planning process – the connection of insular solutions to a model-‘continent’.

We intend to make FREAC freely available for research projects from September 2010.

**References**

