

Operating System for Structural Design

Alexander Yampolsky

December 2005

■ a **research** paper from upFront.reSearch
an imprint of upFront.eZine Publishing, Ltd.



■ upFront.reSearch

This research paper was prepared financially independent of any CAD vendor, although they have reviewed the content for accuracy. Public and private sources may have assisted in providing the information reported by this research paper. upFront.eZine Publishing, Ltd. provides this research paper as-is and does not guarantee its accuracy, and is not liable for any loss resulting from the use of its content. Over time, the information and summaries may change. upFront.eZine Publishing acknowledges that trademarks mentioned herein are property of their respective owners.

■ Production

Writer Alexander Yampolsky
Technical Reviewer Ralph Grabowski

■ Copyright and Fee

Copyright © 2005 by Alexander Yampolsky. All rights reserved worldwide.

The owner of the copyright, Alexander Yampolsky, does not give you permission to make electronic or print copies. You may not claim authorship or ownership of the text or figures herein.

Price List

- By email in Acrobat PDF format: **US\$100**. Allow for a 0.5 megabyte download.
- On CD in Acrobat PDF format: US\$125 (incl. shipping by FedEx). Allow 2-3 days to arrive.

Payment Options

For email delivery, use the PayPal account of editor@upfrontezine.com at www.paypal.com. PayPal accepts funds in US, Euro, Yen, Canadian, and many other currencies.

For CD delivery, make cheques or money orders payable to 'upFront.eZine Publishing, Ltd.' and mail to:

"Operating System for Structural Design"
upFront.eZine Publishing, Ltd.
34486 Donlyn Avenue
Abbotsford BC
V2S 4W7 Canada

We accept the following currencies:

- US funds drawn on a bank with its address in the USA.
- Canadian funds drawn on a bank with a Canadian address (includes GST).
- British funds drawn on a bank located in Great Britain.
- Euro funds drawn on a bank located in the EU.

www.upfrontezine.com/reSearch

■ ■ Table of Contents

Abstract	5
About the Author	5
About upFront.reSearch	5
Operating System for Structural Design	7
Project Data	8
Operating System	9
Advantages	10
Practice	10
Conclusion	10

■ ■ Abstract

Full-scale computer-aided design (CAD) systems are program shells similar to the operating systems (OS) of computers. This shell must be adapted for designing in specific fields.

Two necessary conditions are inherited from the OS: the ability to work with hierarchical structures, and access to independent application programs.

Building frames, foundations, floors, trusses, columns, beams, and other construction objects come out instead of folders and files. Appropriate information (object properties) may be linked to any object laying on any level of the hierarchical structure. Good practice is to store information with high level objects. Information about lower level objects may be recovered with the help of calculations.

When designing, objects from defined branches are loaded in memory for processing. Objects in memory usually have no hierarchical structure.

About the Author

Alexander A. Yampolsky is a design engineer with 20 years of professional practice, practicing in Tula, Russia. You can contact him through yampolski_a_a@rambler.ru.

About upFront.reSearch

This research paper was prepared by upFront.reSearch, an imprint of upFront.eZine Publishing, Ltd. upFront.reSearch has prepared research papers on behalf of Autodesk, Graphisoft, IMAGINiT Technologies, IMSI, IntelliCAD, and SolidWorks.

Its founder, Ralph Grabowski, has 20 years experience in the computer-aided design industry. He is the author of 80 books on CAD, the editor of two industry newsletters, publisher of upFront.eBooks, and manager of three WorldCAD Access weblogs.

www.upfrontezine.com/reSearch

■ ■ Operating System for Structural Design

When I noticed that a column at one level is the same compound structure as a building frame, I assembled a small design system based on tree structure. After doing some work with it, I saw that my system was similar to a computer operating system. In other words, I got an OS-like program shell adapted to structural design.

Two fundamental properties were taken from operating systems in general:

- Ability to work with hierarchical structures.
- Catering to independent application programs.

The scheme is similar to OS scheme, and can also be called a full-scale CAD system scheme, is depicted by Figure 1, as follows:

OS Operating System
Ap1, Ap2, Ap... Independent Application Programs.
PD Project Data

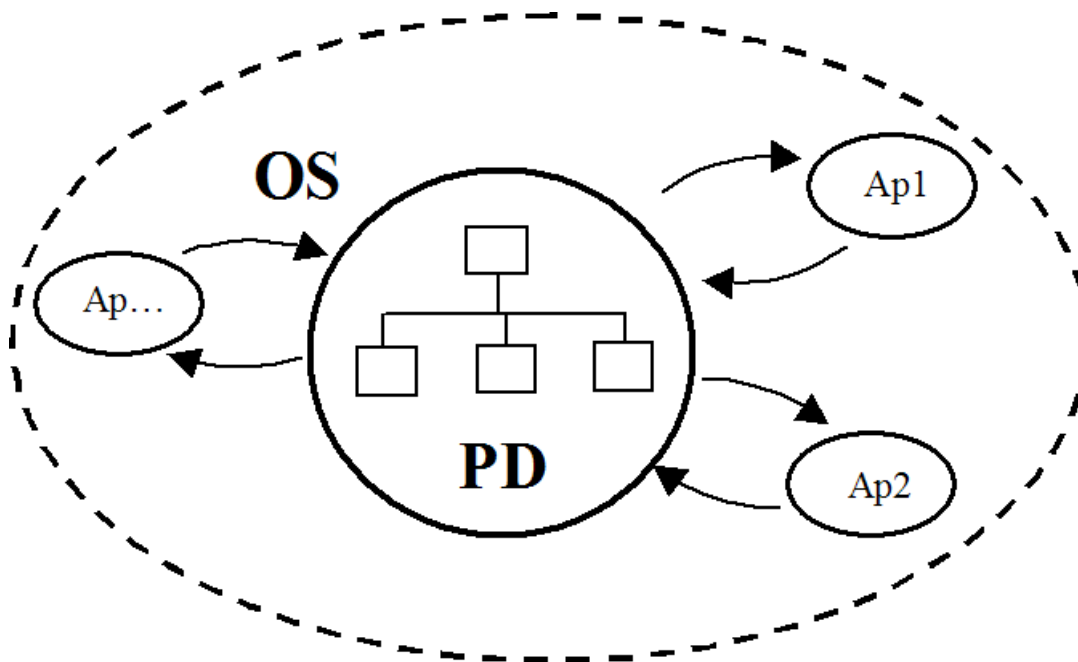


Figure 1. Scheme of an OS-like program shell.

If the OS and PD components of this scheme are filled with knowledge of the building field, we obtain an operating system specialized for structural design. I will describe some principles which might be useful. Most, but not all, of them have been examined in practice.

Project Data

Computer operating systems deal with folders and files. In specialized systems, however, their function is performed by construction objects, such as buildings as a whole, building frames, foundations, floors, trusses, columns, beams, and so on. The hierarchical structure of all these objects is evident. Building and construction projects mainly follow this structure.

The structure of a building project with concrete frame is depicted in the Figure 2, as follows:

F1, F2, ...	Foundations.
C1, C2, ...	Columns.
G1, G2, ...	Girders.
T1, T2, ...	Trusses.
Sc1, Sc2, ...	Spatial reinforcement cages.
Ft1, Ft2, ...	Embedded fittings.
Bar1, Bar2, ...	Reinforcement bars.
Te1, Te2, ...	Truss elements.
Fc1, Fc2, ...	Flat reinforcement cages.
Pt1	Plate.
An1, An2	Anchors.

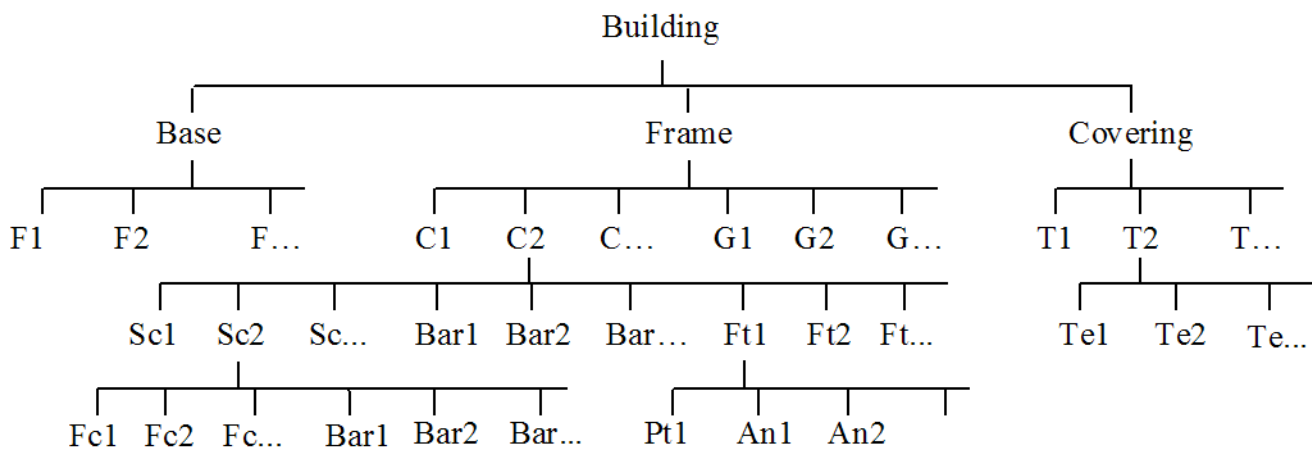


Figure 2. Structure of the building with concrete frame.

The generalized descriptors of objects are located at one or another level of the hierarchical structure, and is the “object type.”

The framed structure with hinged joints can be named the “truss” object. “Column,” “slab supported at the outline,” and “girder” are all object types. *Object types* define object properties and operations that can be done at the object.

The calculation of acoustic insulation, for example, can be performed for slabs, but it does not make sense for trusses. Many operations can be initiated without user intervention; i.e., the system becomes more intelligent when it is supplied with knowledge about typical object behavior. For example, “column” type objects can serve as a support for “girder” type objects, and so on.

Building objects have many links and references besides the hierarchical ones. Examples include references to nodes coordinates, property sets, and so on. Thus, PD components must be arranged as a database.

The core of the database is an indexed file of objects. The record of this file contains only the most necessary object information, such as:

- Object identifier (unique, invariable, hidden from the user).
- Object name (means of reference to the object from the user).
- Object type.
- Information about “ancestors.”

All the rest of the data can be called “object properties.” They are stored in multiple indexed files attached to objects and contain sets of logically connected data. Everything that is necessary to support interaction with application programs must be provided for in the database.

Operating System

The basis of operating system is a tree node traversal procedure. An arbitrary node (building object) is taken as the starting point. All node descendants are visited, then the descendants of the descendants, and so on. The descent to the subjacent level is performed only after visiting all objects of the current level. In the course of traversal, suitable objects are loaded in memory and processed. (Most of the time, this is a data preparation for the transfer to application program.)

A simple traversal can be a too cost-intensive procedure for big trees, so we can use traversal with “cuts.” The user selects one or more objects (one or more object tokens). In the course of traversal, the procedure comes across the selected objects, and then would act according to one of two options:

- Cuts all but selected branches;
- Cuts the selected branches and traverses all the others.

For example, the traversal order of the “building” structure according to the first option with two check-nodes specified – **Frame** and **FtI** – will be following:

```
Base
> Frame
> Covering...
  > C1 > C2...
  > G1 – G2...
    > Sc1, > Sc2...
    > Bar1, > Bar2...
    > FtI, > Ft2...
      > Pt1
      > An1, > An2...
```

Another useful way to traverse the tree structure is a traversal with “immersions.” This strategy is applicable in the case, for example, when the user requires static calculations of the T1, T2, T... trusses. When coming across the T1 node, the immersion takes place; i.e., an additional procedure of T1 descendants traversal is initiated. (In this case, the descendants are truss elements.) The source data are prepared in the T1 coordinate system and the truss calculation is carried out. Control is then transferred to the initial procedure and the actions are repeated when coming across the T2, T3 nodes, and so on.

The treatment of the full-scale CAD system as a sort of operating system implies that when designing,

the main load is borne by independent design programs. Therefore, the drivers providing interaction between the system and independent programs are a significant part of the system.

Advantages

The advantages of the approach under consideration result from the properties of hierarchical structures. First of all, this way of operation allows one to focus on designing objects separately (to make them the “current” one, to enter its coordinate system). The visible project area – the descendants of the current object – is available for processing. All the rest is “invisible” and does not distract from the design work.

Objects of the upper level are usually less mobile than their descendants. The latter are often subjected to changes: they are removed, added, shifted. Any of these actions entails a chain reaction of corrections affecting, as a rule, the lower-level objects. Thus, it is better to store the information at the top.

Another advantage inherent to hierarchical structures is the possibility to use property inheritance mechanisms. All properties attached to an object are inherited by its descendants. Missing object information can be restored visiting the parents. This is convenient from the point of view of saving disk space. Objects of the upper level are usually less “mobile” than their descendants are. The latter are often subjected to changes – they are removed, added, shifted. So, it is better to store the information “at the top”.

A small amount of information attached to the upper level objects provides a lot of information about the descendants, with the help of calculations. A local load applied to a “Floor” object completely defines local loads for all floor elements – slabs, beams, etc. Attaching the description of boreholes to a “Base” object fully specifies the ground conditions for all of the foundations included in “Base.”

The calculation procedures are initiated automatically every time when it is necessary to recover properties of lower level objects.

Practice

I use my system for actual designs since 1990. A theoretical problem for tree-based systems is the influence of the growth in the number of objects on system efficiency.

As a check, I once generated a tree structure consisting of two million objects. Then I worked on a small project had been opened to eighth level of the hierarchical structure. (In fact I tested “breadth first” traversal procedure based on search in a big indexed file.) I had no difficulties, and found that there was no dramatic productivity slowdown.

■ Conclusion

I think the approach discussed in this research paper may put a friendly, transparent, and sufficiently productive environment at the disposal of the designer. This provides a more homogeneous environment at each level of designing with a uniform code, with uniform methods of object operation.