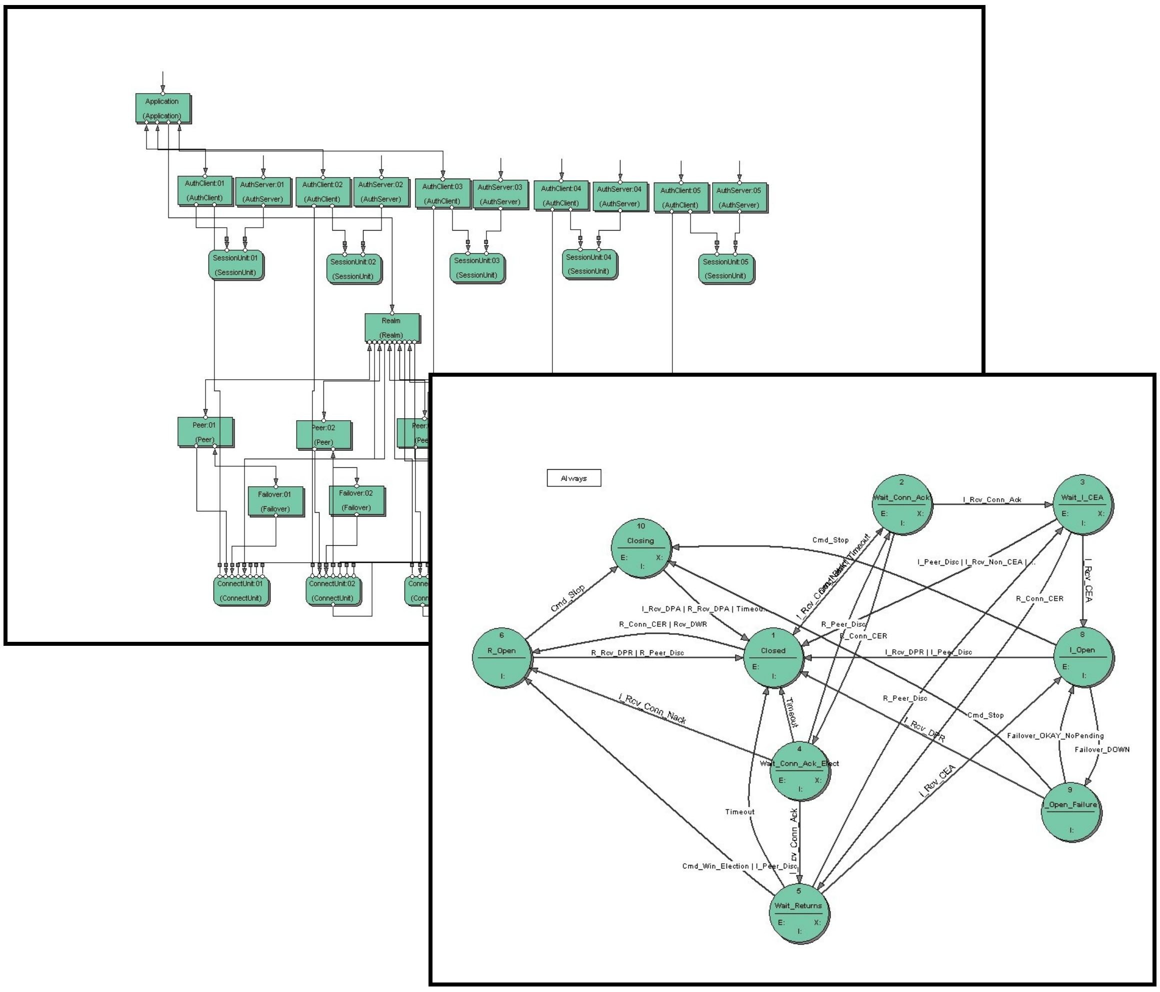
StateWORKS:

The Silver Bullet at last ?

"Not only are there no silver bullets now in view, the very nature of software makes it unlikely that there will be any — no inventions that will do for software productivity, reliability and simplicity what electronics, transistors and large-scale integration did for computer hardware." **Frederick P. Brooks, Jr.**

Frederick P. Brooks, Jr. wrote that in his classic article: "No Silver Bullet: Essence and Accidents of Software Engineering" published in the April, 1987 edition of IEEE "Computer". 20 years later, we can finally see that he was too pessimistic. But it took us a lot of hard work over that time to fully develop the StateWORKS concept! (http://www.stateworks.com)



StateWORKS

– bringing engineering design discipline to software development.

A real, hype-free, time-tested and mature product which you can use now, to gain huge benefits in developing your products. Design ultra-reliable and predictable software, reduce maintenance, simplify re-use of your software in new projects, etc...

StateWORKS: The Silver Bullet at last ?

Finally the answer to the big question – how can top-quality software for big projects (such as embedded or telecommunication applications) be produced reliably and on time? Such software systems are "reactive".

The software needs to work well when first installed, unlike desktop applications: it can not rely on thousands of users finding bugs. The answer is to just use two development steps:

- specify it
- run it.

This is what StateWORKS does. Using time-tested, proven and theoretically-sound finite state machine concepts, the implementer completely specifies behavior, in detail but at a high level, without assuming it will be coded. He creates a model, of unlimited complexity for the total system, but using small state machines; each with a well-defined task; each one easily designed and understood. The development system (I.D.E.) provides varied features for testing at the design stage. Then, the model is just loaded into the system, and is run. Perhaps a few hours of tuning will be needed, but nothing like the weeks of "testing" common in these areas. And of course we think that software can not wear out in use, so why should it need a huge "maintenance" effort?

Our basic rule is – never write complex code. Don't try to automate it, either, but avoid the problem altogether.

StateWORKS provides a method for handling complexity: very complex projects can be made simple to deal with, using our unique tools for handling "Systems of State Machines".

And the Gun?

Of course, the bullet needs a gun if it to be effective! So, a StateWORKS project will probably need some code, somewhere.

StateWORKS imposes a strict partition between data handling and control flow. But the control flow, through "actions" of the various finite state machines, is in charge of the data handling routines, which have to be written with normal tools such as C++ development systems. The same applies to input-output functions, for which we provide a very full Class Library.

The essential point is that very complex projects fail because it is impossible for the designers to cope with behaviour in all possible situations – not just the "sunny day scenario" represented by common "use cases" but in situations of multiple errors. Big systems can **never** be tested so completely as to ensure coverage of all possibilities.

The "skeleton" which StateWORKS provides for a project deals with the behaviour - where systems can often fail due to unexpected inputs - and also with the supervision of all the data-handling routines. Those can always be made reliable through careful development and testing, in the classical way. Behaviour can not!

StateWORKS Removes Risk! Hit that target first time and every time!

Four essential features of StateWORKS

1. StateWORKS is a method for creating high-quality software through models. Software behaviour is expressed as finite state machines: never very complicated individually, but with many of them working together to make a system. A strict partitioning strategy between control flow and data handling is employed, where the control flow supervises completely the data flow.

2. The StateWORKS design process is in fact a tool for exploring the requirements, so as to express them in such fine detail as to cause software to be generated without further human coding. Our multi-mode editors permit full complexity to be easily managed.

3. StateWORKS can deal effectively with very large and very complex systems, by using large numbers of state machines in a well-governed hierarchical structure: 50 or more state machines in a single processor is not unusual. So complexity is managed, and nearly all bugs eliminated in the design process: not left to a "testing and commissioning" phase or for "maintenance'.

4. StateWORKS has been around for several years, has been applied to a wide variety of projects, and is very reliable indeed.

StateWORKS uniquely brings, to software engineering design, practices as sound as those used to engineer hardware, bridges or airplanes , creating a true "Software Engineering" method for the first time.

How to Obtain StateWORKS

To learn about StateWORKS features in detail, there are two possible paths: buy our book or study our web site. A limited-power development system: StateWORKS Studio LE, may be downloaded from our web site free of charge, and can only be used for one month. Purchase of the book entitles the user to remove the time limit. This version can be used to study all the examples discussed in the book, or accessible on the web site, but designing with it is limited to single finite state machines. Since programming a single finite state machine is very easy, with hand coding or using tools from various suppliers, the LE version is mainly useful as a teaching tool, to show how to apply our concepts.

The huge power of the StateWORKS approach is only apparent for large-scale real-life projects, typically employing 10 to 100 finite state machines in a system. For these we can supply, either the Basic system which is limited to designs having up to 300 objects, or the Pro version which has no such limit. We can also arrange training courses and seminars.

- Book: "Modeling Software with Finite State Machines: a Practical Approach"- F. Wagner et al., Auerbach Publications, May 2006. ISBN 0-8493-8086-3
- Web Site: http://www.stateworks.com
- Inquiries: sw-info@stateworks.com

Six excellent reasons why not to use U.M.L.

1. StateWORKS is used to fully generate applications, and has to include detail which U.M.L. can not express. (This is also why we don't generate XMI files: we do generate XML files of designs and our format is published, and expresses aspects of detail which XMI can not handle.)

2. U.M.L has fundamental limitations which restrict it to top-level overviews of projects: it is O.K. for the systems analysts, but not for generating final software: despite claims made by some sponsors.
   In contrast, the StateWORKS design process is one of designing the final product, rather than just describing roughly how it should work. In fact the process brings all the problems to light, rather than leaving them to be resolved later when coding.

3. The finite state machine concept of U.M.L. is fundamentally flawed, and drives towards excessive and unmanageable complexity. StateWORKS encourages design with small modules, each simple enough to understand, and linked together hierarchically to make the entire system. The top levels of the hierarchy are clear enough that they can be used to discuss the project with management or customers.

4. UML does not offer solutions for communication between tasks in a multi-tasking system.

5. UML can be used for specification of sub-systems but cannot specify the detailed behaviour (meaning control flow) of entire software systems. Still less can it be used to automate the later stages of detailed software production, e.g. coding.

6. UML is overly-complex, particularly in its use of many different but confusing graphical symbols and its multiple options for expressing the same idea, while StateWORKS is solidly based on simple and elegant finite state machine concepts well-known for over 50 years, and has a shallow "learning curve".

SW *Software*
Nebenbach 8 Tel. +41 (81) 740 0717
CH-9470 Buchs Fax. +41 (81) 740 071

CYDON *Technology s.a.r.l.*
559, Montée de Font Vert Tel. +33 (494) 073 055
F- 83140 Six-Fours Fax +33 (488) 714 030

Dr. F. Wagner
Stauferstraße 99 Tel. +49 (791) 946 8853
D-74523 Schwäbisch Hall Fax +49 (791) 946 4558

Inquiries: sw-info@stateworks.com
and see http://www.stateworks.com

U.S.A. - Ask us about authorised resellers.