simpleXecutive Methodology

White Paper

By



*“Our simpleXecutive Methodology is Design Automation for Developing Complex Systems.”*

Contact:

info@comcomsys.com

Problems in the Development and Upgrade of Complex Software

Traditional complex system development de-emphasizes the value and importance of front end systems engineering because not enough time or analysis is allocated to initial design. People commit to implement the first plausible solution and rush to implementation. This results in inefficiencies and expensive integrations because excessive time is spent on the back end fixing problems that resulted from ineffective front end decisions.

Often unsatisfactory performance observed in integration is the result of inefficiencies hidden in run-times, sequencing, resource management, communication deadlocks, processing overloads, etc. These invisible inefficiencies drag down the overall performance of the system.

Traditional system integration is fraught with error when merging individual components that were not designed in concert with each other. Static interfaces are interpreted differently by each implementer and sometimes using different reference frames.

When system integrity is added piecemeal and not cohesively engineered as a holistic system solution, all the pieces function correctly when evaluated individually. But in the operational environment, dynamic interactions are subtle leading to reliability and quality and consistency problems. Dynamic interfaces can be complex, asynchronous, non-repeating, etc. making problems very difficult to identify and resolve.

Systems tend to become more fragile as complexity increases. Unexpected, apparently random, failures or faults are very difficult to duplicate. Diagnosing such errors is very expensive. Without an understanding of the nature of the problem, a successful resolution is, at best, a hit and miss proposition.

System complexity grows when problems are solved during integration. System complexity as measured by a large number of off-nominal or special cases expands to eventually exceed the size of the basic functionality. Ugly integration surprises necessitate changes, schedule and budget pressure preclude fundamental redesign to correct the erroneous mistake. Instead, a special case adjunct is patched into the system to alleviate the particular failure. This spiral makes future maintenance increasingly more expensive.

Cost overruns and delivery delays result from ineffective budget and schedules as unplanned system errors are discovered in the lab and rushed to be reconciled, constrained by deadlines and budgets.

Modern Weapon Systems encounter really nasty problems involving temporal concurrent inconsistencies and hazards especially in asynchronous systems with multiple processors tightly coupled in a node and with multiple interconnected networks. These conditions quickly overwhelm traditional analysis techniques. The simpleXecutive Methodology was designed to specifically address these problems.

simpleXecutive Methodology

Our simpleXecutive Methodology applies system engineering to the front-end of system development to specifically address the fundamental source of these problems. Through our simpleXecutive Methodology, we can substantially increase system performance by more effectively managing system resources - without modifying existing infrastructures. We synthesize the integration process from the beginning to resolve design errors early by modeling the system integration from the beginning. Our methodology can be applied at all phases of the development cycle.

* We consider the entire system as a whole throughout the entire development process.
* We continuously validate the interfaces between individual components and determine the effectiveness of their interactions in operational contexts.
* We simplify complex systems using our REAL Blueprints which look at the system from five unique perspectives, separating the domains between the blueprints.
* We use our Real-time Executable Architecture Language (REAL) to construct a formal REAL model from the design REAL Blueprints.
* REAL models are imported by our patented SIMPLEX analysis and simulation tools.
* SIMPLEX tools explicitly perform the following:
	+ They evaluate all temporal and spatial interfaces and interactions between each individual component.
	+ They automatically apply advanced hazard analysis to identify execution sequences which compromise system integrity.
	+ They establish full spectrum integrity protection and synthesizes the run-time execution overhead and blocking required for safe execution in any defined context.
	+ They construct a full fidelity temporal and spatial simulation of the system.
	+ The dynamic analysis of simulation logs automatically exposes the effective system behavior.

What We Do and What We Don’t

Popular development strategies stress early implementation and demonstration of selected functions. This becomes the backbone of the solution as additional functionality is incrementally incorporated. Each expanding functionality demonstration constrains future decisions by limiting the remaining solution space. Truth becomes apparent when the particular components are integrated and forced to interact with each other in the integration lab.

simpleXecutive Methodology……..

We evolve a system design at the system level by iteratively adding system level details.

*We don’t look at individual parts individually or at individual components.*

*We don’t apply a piecemeal approach to designing a system*

*We don’t demonstrate selected functions operating with actual code*

We test overall system designs from the beginning by simulating virtual integration using component characteristics.

*We don’t design components and then force them to integrate.*

*We test the system design to evaluate system performance*

*We determine the hardware configuration based on the application to be hosted*

We always assess impacts of individual changes at the system level.

*We don’t wait for integration errors to expose the consequences of changes.*

*We purposely stress a system design to failure to understand sensitivities and limitations*

*We systematically change the design to evaluate options and conduct system performance trades*

We apply the scientific method to methodically experimentally evaluate alternative designs prior to establishing the recommending a preferred design.

*We don’t commit to a design, or start coding until we have an overall accepted system design.*

*We don’t commit to the first plausible design, instead we test multiple alternate design options.*

*We provide implementers with a valid basis for estimating schedules and budgets prior to coding*

We minimize integration surprises and reduce overall integration expenditures by always modeling the complete system design.

*We don’t experience surprises in the lab because components won’t “fit” together.*

*We model component integration from the very beginning and push loading beyond operational limits*

*We strive to find and fix poor design decisions early in development, before serious implementation begins.*

We optimize the system design for more effective performance before committing to implementation

*We don’t enhance performance by adjusting the integrated operational system in a refactoring cycle*

*We engineer the design to achieve effective system performance before implementing components.*

*We understand the behavior at the system level before recommending a design be carried forward.*

How the User Experiences simpleXecutive as being Fundamentally Different

simpleXecutive is an advanced development methodology that targets the front-end design development emphasizing large scale, complex, real-time systems. Our simpleXecutive Methodology systematically assesses the Temporal, Spatial and Thermal dimensions of system design. System engineers utilize our simpleXecutive development methodology to improve system *integration*, validate system *integrity* and enhance system *performance* by focusing on the front end system design.

Under our methodology the user evaluates alternate system solutions to select the most effective system design. With our methodology, the user proposes fundamental changes and evaluates options based on total system performance. Our methodology encourages total system design evolution from crude notional concepts to detailed top level designs addressing all required functionality. Successive refinements add detail and reduce risk at every level by synthesizing the system consequences of individual design decision.

The detailed system level design REAL Blueprints provide value across multiple levels. They proved a standardized vehicle for communication between developers and customers. They graphically expose the structure and operation of a proposed system from the earliest stages. They provide a valid basis for estimating budgets and schedules. In Later phases they can display progress, risk, schedule, budget, and status information for management and project control.

Our methodology works in the system domain and does not require that the individual code and hardware components actually exist to synthesize the effective performance based on component characteristics. simpleXecutive development methodology has been applied at different points in the development lifecycle of various sized systems, conceptual to mature.

There are numerous system adjustment opportunities available to improve performance of a mature system. These range from tuning without changing Application or Architecture components to changing these components. There are a plethora of possible adjustments including combinations of the following:

|  |  |
| --- | --- |
| **Adjustments to Existing System Parameters** | **Changes to System Components** |
| Adjust the initial phasing or relative priority of specific Processes | Subdivide or revise blocking Task procedures or substitute H/W (FPGA , etc.) |
| Elevate the preemption priority ceiling of particular Tasks | Add or upgrade processing components (programmable computers or ASICS) |
| Remap Processes to different processing resources | Modify Algorithms  |
| Double buffer datasets determined to be critical blocking contributors | Add software components (Tasks) |
| Add alternate instances of non-reentrant code to Tasks determined to be major blocking contributors |  |
| Restructure the data flow or revise selected Task sequence order to significantly change the temporal relationship between major blocking contributors |  |

Because performance model changes are straightforward and since SIMPLEX automatically constructs the performance simulations, the designer is encouraged to experiment with various options and combinations of options to find a design with improved performance.

The Extensive Impact of our Value

simpleXecutive teaches users not to be satisfied with the first plausible system design solution. The methodology encourages early system level trades beginning at the Blueprint level. Changes to the REAL Blueprints and formal model are easily incorporated. The SIMPLEX tools quickly produce full fidelity simulations to synthesize the system level consequences of the alternate designs. Systematically evaluating a range of alternatives constitutes a system level trade suggesting more effective design solutions.

Performance tuning the system level design has consistently produced stunning results versus simply optimizing individual components, and for relatively trivial investment.

What we Offer

Commercial Computer Systems offers Systems Engineering consultation and System Design support including:

**Requirements** – CCS documents requirements in diagrams: Mission Functional Flows (what the system does), Physical and Functional External Interfaces, Trigger Diagrams, and Measures Of Effectiveness (MOEs define how well)

**System Architecture** – CCS documents a system designs in Functional Schematics (static component accesses), Process Threads (dynamics and engagement logic), Object Flows (loading and behavior), Resource Configuration (hardware interconnections), System Allocations (how applications are managed on the resources)

**Formal System Model** – CCS documents a system design in the Real-time Executable Architecture Language (REAL) composed of English statements and standard mathematical expressions using common dimensional units.

**System Analysis** – CCS patented tools automatically document a system’s structure and checks for completeness and consistency from a formal model then directly synthesizes the real-time performance by virtually integrating the dynamic component interaction in time and space on processors, memories, and buses.

**Performance Assessment** – CCS uses our exclusive analysis tools to uncover and repair errors, expand details, and to experiment with alternate designs to evolve a superior system design prior to actual implementation.

**Greg Scallon, greg@comcomsys.com, 206.914.7607**

Founder and chief engineer of Commercial Computer Systems, brings over four decades of software research experience specializing in engineering effective real-time distributed system designs based on performance modeling to improve system effectiveness.

* Greg has applied his extensive engineering skills to create the simpleXecutive Methodology which includes REAL Blueprints, REAL Language and SIMPLEX simulation design automation tool, which models a system’s performance based on its defined structure and characteristics. These tools synthesize system level interactions between hardware, software and communication subsystem modules. The newly patented tools have successfully verified system design performance against complex distributed system effectiveness goals early in system development, i.e. prior to initiating coding.
* As Chief Computing System Architect for multiple successfully deployed large scale real-time projects, Greg has acquired a broad experience base developing complex systems. This experience includes commercial avionics, manned Apollo Moon Mission, medical diagnostic and therapeutic systems, DoD weapon systems, and Star Wars experiments. Optimized system level performance on complex real-time distributed systems is the hallmark of his designs.

Greg retired from Boeing as an Associate Technical Fellow after 30 years of successful aerospace experience specializing in complex real-time systems design, performance modeling and evaluation, and basic research for a wide variety of large scale embedded real-time distributed systems.

His subsequent CCS career includes successfully solving many complex real-time medical and avionics system design development challenges.

**Education:**

* Seattle University, BSEE, ’65, minors in Math and Nuclear Physics
* Post grad studies at MIT and FSU in digital signal processing and filtering
* Extensive formal systems engineering training from NASA and USAF

**Honors:**

* Appointed Technical Fellow of the Washington Technology Center
* Founding Technical Contributor to the Software Productivity Consortium
* Patented inventor (of both hardware and software) and published author
* Awarded 9th annual Simulation Symposium best paper by IEEE & ACM
* Invited speaker: NATO Software Engineering & NRL S/W Research conferences,
* Earned numerous Boeing awards for technical achievement, leadership, etc.

**Suzanne Welton,** **suzanne@comcomsys.com****, 425.922.7107**

Director of Business Development for Commercial Computer Systems, has over fifteen years’ experience bringing pre-released software and hardware to market for the first time. Her career spans over twenty years focusing on marketing and consumer experience. She has a proven success record of managing high risk, complicated, large budget and high profile projects. She has a proven track record of leading multi-discipline teams to achieve award winning cohesive executions that are on time and on budget.

After spending almost 15 years at Nintendo and working with executives and teams in both the US and Japan, she’s ready to take on her next challenge and bring this start up to the system development world.

**Education:**

* Bachelors of Communications/Advertising, University of Washington
* Executive Masters in Business Administration, University of Washington