Software Intensive Space Systems Engineering
Fall 2010 Course Syllabus

Dr. Doug Buettner, Dr. Kirstie Bellman

Course Description

Software Intensive Space Systems Engineering is a survey and methods course that provides experience with engineering a software intensive system in addition to the theoretical background for engineers tasked with designing and building our next generation of space systems, of which software is an integral and increasingly important part; the course includes sufficient detail to allow class participants to pursue detailed research into areas of interest.

Topics Covered By This Course

1) *Introduction to Software Intensive Space Systems* (space system software history and software/firmware functionality in space systems, historical software development approaches – structured, object oriented and other paradigms, radiation and thermal environment effects on software/firmware and ground impacts, examples of space system software issues, class project and information)
2) *Software Lifecycles, Theory and Planning* (software lifecycle models, stakeholders, tools, project planning and reviews, lifecycle documentation, configuration management, software quality assurance, build and test planning, lifecycle staffing, creating basis of estimates, methods for estimating software size, COCOMO, schedule assessment methods such as Monte Carlo, critical path, resource loading, and schedule coordination on large projects)
3) *Software Metrics, Measurement and Management* (software metrics measurement, requirements volatility, requirements traceability, action item burn down, action item, inspection findings, defect report trends/clustering, EVM, KPM and KPIs, software rework, software risks management, methods for identifying risks, risk compounding, mitigation techniques, dynamic nature of risks, building successful teams, team productivity, software metrics use)
4) *Software Architectures, and Design* (computer architecture impacts, trade studies – space vs. ground tradeoffs, mission autonomy considerations, fault management, redundancy management, safety and security considerations, 4+1 and other architectural representations, UML, AADL, SysML, use cases, state charts, design patterns, distributed system communication, interfaces – HW to SW, bus to payload, and space to ground, data transfer rates, data busses, rapid prototyping, MBD-Simulink, MFESA engineering approach, databases, design reviews)
5) *Requirements Engineering and Review Methods* (methods for requirements elicitation, analysis and documentation, text – graphical method overview, management of requirements and tools, traceability, inspection method history, types of peer reviews, training, roles, action items)
6) *Software Implementation* (programming language selection, types of operating systems, OS selection, code/design centric, concurrent processing, threads, graphical user interfaces, human factors, real-time systems, SIMD/MIMD, code reuse)
7) *Software Verification and Validation Overview* (the V-model, defect management lifecycle, cost of late lifecycle defect discovery, ground test systems, database testing, static analytical techniques, dynamic testing techniques)
8) *Software Verification Details* (method details – unit, software component integration, and software/hardware integration, qualification methods, formal qualification, system testing, how to write a good bug report, test API, stubs, test harness)

9) *Team Processes, Dynamics and Roles* (RUP, managers, SQA, testers, developers, leads, architects, customer, team software process, CMMI, test process maturity, entrance and exit criteria, checklists, cleanroom, distributed teams, continual process improvement)
10) *Advanced Analytical Methods* (reliability growth models, system process dynamics, process metrics, ODC, RCA, formal methods, FMEA/FMECA, FTA, HazOps, Monte Carlo, Markov models, Petri Nets, other probabilistic risk assessment methods, model simulation analysis, rate monotonic analysis, logic rules)
11) *Advanced Topics and Review* (standards, large team dynamics organizational maturity and structure, scaling model-centric vs. code-centric projects, integration of model-centric with code-centric development, large team communication overhead, collaboration environments, pressures - schedule and funding, issues with long development cycles, product dependency maturity and obsolescence, training, technology refresh, refactoring, maintenance, where do we go from here?)

Expectations

Schedule: 9:30 to 10:50 AM on Monday and Wednesday

Location: OHE 120

Pre-requisites: students should be proficient in computer programming or willing to learn while taking the course, have access to a computer with a compiler, MS Word or an equivalent text editor, MS Excel or Project, a UML modeling tool.

Attendance: students are expected to attend all classes either in person or remotely via DEN. Exams must be taken at a monitored site, the location of which will be provided prior to the exam. Students are encouraged to actively participate in the class with questions and discussion.

Grades: grades will be determined by a student's performance in three areas: mid-term exam 30%, final exam 30%, and a collaborative class project 40%. Active class participation contributes to grade improvement. Course passing grades are A+:100-97; A: 96-93; A-: 92-90; B+ 89-87; B: 86-83; B-: 82-80; C+: 79-77; C: 76-73; C-: 72-70. Total cumulative grades ≥ 70 passes the course.

Class Project: a collaborative competitive project will be provided at the end of the 1st lecture. Individuals will be assigned to work in virtual collaborative work groups and will be required to provide progress reports each week. Team sizes are set depending on class enrollment. The first phase of the project requires submitting to the instructor a plan that describes how the team intends to meet the goals of the project within the class's schedule culminating in a documented project plan. The second phase requires the creation of a preliminary design, and presenting the design at an in-class design review (via in person or the class 1-800 number and Blackboard). The third phase requires collaboratively taking the preliminary design into the detailed design phase, and presenting the design at a detailed design review. The final phase requires implementation of the design, and submission of an individually written test and experience report.

Statement for Students with Disabilities

Any student requesting academic accommodations based on a disability is required to register with Disability Services and Programs (DSP) each semester. A letter of verification for approved accommodations can be obtained from DSP. Please be sure the letter is delivered to me (or to TA) as early in the semester as possible. DSP is located in STU 301 and is open 8:30 a.m.–5:00 p.m., Monday through Friday. The phone number for DSP is (213) 740-0776.

Statement on Academic Integrity

USC seeks to maintain an optimal learning environment. General principles of academic honesty include the concept of respect for the intellectual property of others, the expectation that individual work will be submitted unless otherwise allowed by an instructor, and the obligations both to protect one's own academic work from misuse by others as well as to avoid using another's work as one's own. All students are expected to understand and abide by these principles. Scampus, the Student Guidebook, contains the Student Conduct Code in Section 11.00, while the recommended sanctions are located in Appendix A: http://www.usc.edu/dept/publications/SCAMPUS/gov/. Students will be referred to the Office of Student Judicial Affairs and Community Standards for further review, should there be any suspicion of academic dishonesty. The Review process can be found at: http://www.usc.edu/student-affairs/SJACS/

Course Calendar – Fall 2010

Date	**Lecture/Activity**	**Due at end of class**
Aug 23rd	*Intro to Software Intensive Space Systems and Class Info*	
Aug 25th	*Continue - Introduction to Software Intensive Space Systems*	Team Background
Aug 30th	*Software Lifecycles and Theory*	
Sept 1st	*Software Planning and Estimation Methods*	Team Progress 1
Sept 6th	**No Class – Labor Day**	
Sept 8th	*Detailed Basis of Estimate (BOE) Example*	Project Plan
Sept 13th	*Software Metrics and Measurement*	
Sept 15th	*Standards and Documentation*	Team Progress 2
Sept 20th	*Inspection and Review Methods*	
Sept 22nd	*Systems and Requirements Engineering*	Team Progress 3
Sept 27th	*Space and Ground System Architectures*	
Sept 29th	**Preliminary Design Review – Design Presentations**	Preliminary Design
Oct 4th	*Software Designs and Modeling*	
Oct 6th	*Risk Assessment and Management Methods*	Team Progress 4
Oct 11th	*Configuration Management and Review of Topics for Midterm*	
Oct 13th	**Midterm – Open notes/book**	**Nothing due**
Oct 18th	*Team Processes, Dynamics and Roles*	
Oct 20th	*Implementation, Languages and Tools*	Team Progress 5
Oct 25th	*Verification and Validation Overview*	
Oct 27th	*Unit Testing*	Team Progress 6
Nov 1st	*Integration Testing*	
Nov 3rd	*Qualification Testing*	Team Progress 7
Nov 8th	*Defect and Fault Reporting and Classification Methods*	
Nov 10th	**Critical Design Review – Design Presentations**	Critical Design
Nov15th	*Reliability Growth and Models*	
Nov 17th	*Other Advanced Analytical Methods*	Team Progress 8
Nov 22nd	*Advanced Topics in Software Intensive Systems Development*	
Nov 24th	*Advanced Topics in Software Intensive Systems Development*	Team Progress 9
Nov 29th	*Interactive Class Discussion on Futuristic Concepts*	
Dec 1st	*Review of Topics for Final*	Project Reports
Dec TBD	**Final – Closed book, one 8.5x11cheat sheet is allowed**	

Required Text:

Estimating Software-Intensive Systems: Projects, Products and Processes (Richard D. Stutske)

Supplemental Reading Material *(a small sampling of literature)*

Ada:

Programming in Ada 95, 2nd Edition (John Barnes)

Programming in Ada 2005 (John Barnes)

Concurrent and Real-Time Programming in Ada (Alan Burns and Andy Wellings)

C/C++:

C Programming Language, 2nd Edition (Brian W. Kernighan and Dennis M. Ritchie)

Programming in C, 3rd Edition (Stephen Kochan)

The Joy of C, 3rd Edition (Lawrence H. Miller and Alexander E. Quilici)

The C++ Programming Language: Special Edition (Bjarne Stroustrup)

Programming: Principles and Practice Using C++ (Bjarne Stroustrup)

C++ Coding Standards: 101 Rules, Guidelines, and Best Practices (Herb Sutter and Andrei Alexandrescu)

Markup Languages (HTML, XHTML, XML, CSS):

Beginning XML, 4th Edition (David Hunter, Jeff Rafter, Joe Fawcett, and Eric van der Vlist)

Professional XML (Bill Evjen, Kent Sharkey, Thiru Thangarathinam, and Michael Kay)

HTML, XHTML, and CSS, 6th Edition (Elizabeth Castro)

The Ultimate HTML Reference (Ian Lloyd)

Java:

Introduction to Java Programming, 7th Edition (Y. Daniel Liang)

Effective Java, 2nd Edition (Joshua Bloch)

Perl:

Learning Perl, 5th Edition (Randal Schwartz, Tom Phoenix, and brian d foy)

Structured Query Language (SQL):

SQL Queries for Mere Mortals®: A Hands-On Guide to Data Manipulation in SQL, 2nd Edition (John L. Viescas and Michael J. Hernandez)

Learning SQL (Alan Beaulieu)

Real-Time Embedded Software:

Meeting Deadlines in Hard Real-Time Systems: The Rate Monotonic Approach (Loic P. Briand and Daniel M. Roy)

Real-Time Systems: Design Principles for Distributed Embedded Applications (Hermann Kopetz)

Specification and Design Methodology for Real-Time Embedded Systems (Randall S. Janka)

Software and System Architectures:

Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives (Nick Rozanski, and EOin Woods)

Software Architecting in Practice, 2nd Edition (Len Bass, Paul Clements, and Rick Kazman)

Software Architecture: Foundations, Theory, and Practice (R. N. Taylor, N. Medvidovic, and E. M. Dashofy)

Software Architecture: Perspectives on an Emerging Discipline (Mary Shaw, David Garlan)

The Art of Systems Architecting, 2nd Edition (Mark W. Maier, and Eberhardt Rechtin)

The Method Framework for Engineering System Architectures (Donald G. Firesmith, Peter Capell, Dietrich Falkenthal, Charles B. Hammons, DeWitt T. Latimer IV, Tom Merendino)

Software and System Design:

Design Patterns: Elements of Reusable Object-Oriented Software (Erich Gamma, Richard Helm, Ralph Johnson, John M. Vlissides)

Managing Software Requirements: A Use Case Approach, 2nd Edition (Dean Leffingwell and Don Widrig)

Model-Driven Software Development: Integrating Quality Assurance (Jorg Rech, and Christian Bunse)

Object-Oriented Analysis and Design with Applications, 3rd Edition (Grady Booch, Robert A. Maksimchuk, Michael W. Engel, and Bobbi J. Young)

Object-Oriented Analysis and Design with the Unified Process (John W. Satzinger, Robert B. Jackson, and Stephen D. Burd)

Requirements Engineering: From System Goals to UML Models to Software Specifications (Axel van Lamsweerde)

Software & Systems Requirements Engineering: In Practice (Brian Berenbach, Daniel Paulish, Juergen Kazmeier, and Arnold Rudorfer)

System Analysis, Design, and Development: Concepts, Principles, and Practices (Charles S. Wasson)

Systems Analysis and Design with UML (Alan Dennis, Barbara Haley Wixom, and David Tegarden)

Software Requirements, 2nd Edition (Karl E. Wiegers)

The Engineering Design of Systems: Models and Methods (Dennis M. Buede)

Software and Systems Modeling Languages:

A Practical Guide to SysML (Revised Printing): The Systems Modeling Language (Sanford Friedenthal, Alan Moore, and Rick Steiner)

Introduction to Simulink with Engineering Applications, 2nd Edition (Steven Karris)

Systems Analysis and Design with UML Version 2.0: An Object-Oriented Approach (Alan Dennis, Barbara Haley Wixom, and David Tegarden)

Systems Engineering with SysML/UML: Modeling, Analysis, Design (Tim Weilkiens)

SysML for Systems Engineering (J. Holt and S. Perry)

The Unified Modeling Language User Guide, 2nd Edition (Grady Booch, James Rumbaugh, and Ivar Jacobson)

The Unified Modeling Language Reference Manual, 2nd Edition (James Rumbaugh, Ivar Jacobson, and Grady Booch)

Use Cases: Requirements in Context, 2nd Edition (Daryl Kulak and Eamonn Guiney)

Software and Systems Engineering:

Aerospace Software Engineering (Christine Anderson, and Merlin Dorfman)

Information Systems Engineering: From Data Analysis to Process Networks (Paul Johannesson and Eva Soderstrom)

Modern Systems Analysis and Design, 4th Edition (Jeffrey A. Hoffer, Joey F. George, and Joseph S. Valacich)

Software Engineering: A Practitioner's Approach, 7th Edition (Roger Pressman)

Software Engineering, 8th Edition (Ian Sommerville)

Software Engineering Best Practices: Lessons from Successful Projects in the Top Companies (Capers Jones)

Software Engineering: Theory and Practice, 4th Edition (Shari Lawrence Pfleeger and Joanne M. Atlee)

Systems of Systems Engineering: Principles and Applications (Mo Jamshidi)

Systems Engineering and Analysis, 4th Edition (Benjamin S. Blanchard and Wolter J. Fabrycky)

Systems Engineering Principles and Practice (Alexander Kossiakoff and William N. Sweet)

Software Cost and Economics:

Estimating Software Costs: Bringing Realism to Estimating (Capers Jones)

Software Engineering Economics (Barry W. Boehm)

Software Cost Estimation with Cocomo II (Barry W. Boehm, et al.)

Software Databases:

Concepts of Database Management, 6th Edition (Philip J. Pratt and Joseph J. Adamski)

Data Analysis for Database Design, Third Edition (David Howe)

Database Modeling and Design: Logical Design, 4th Edition (Toby J. Teorey, Sam S. Lightstone, Tom Nadeau, H.V. Jagadish)

Database Systems: Design, Implementation, and Management (Peter Rob, Carlos Coronel)

Modern Database Management, 9th Edition (Jeffrey A. Hoffer, Mary Prescott, and Heikki Topi)

Software Inspections:

Peer Reviews in Software: A Practical Guide (Karl E. Wiegers)

Modern Software Review: Techniques and Technologies (Yuk Kuen Wong)

Software Inspection (Tom Gilb and D. Graham)

Improvement of Defect Detection with Software Inspection Variants: A Large-Scale Empirical Study on Reading Techniques and Experience (Dietmar Winkler)

Software Management:

Applied Software Project Management (Andrew Stellman, and Jennifer Greene)

Applied Software Risk Management: A Guide for Software Project Managers (C. Ravindranath Pandian)

Death March, 2nd Edition (Edward Yourdon)

Managing Risk: Methods for Software Systems Development (Elaine M. Hall)

Managing and Leading Software Projects (Richard E. Fairley)

Managing the Testing Process: Practical Tools and Techniques for Managing Hardware and Software Testing (Rex Black)

Project Management: A Systems Approach to Planning, Scheduling, and Controlling (Harold Kerzner)

*Project Management with MS Projec*t (Clifford Gray and Erik Larson)

Real World Software Configuration Management (Sean Kenefick)

Software Configuration Management Implementation Roadmap (Mario E. Moreira)

Software Configuration Management Handbook, 2nd Edition (Alexis Leon)

Software Configuration Management Patterns: Effective Teamwork, Practical Integration (Stephen P. Berczuk, Brad Appleton, and Kyle Brown)

Software Project Management: A Unified Framework (Walker Royce)

The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition, 2nd Edition (Frederick P. Brooks)

Software Measurement and Metrics:

Handbook of Software Reliability Engineering (Michael R. Lyu)

Metrics and Models in Software Quality Engineering, 2nd Edition (Stephen H. Kan)

Object-Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems (Michele Lanza, Radu Marinescu, and S. Ducasse)

Software Measurement: Establish - *Extract* - *Evaluate* - *Execute* (Christof Ebert and Reiner Dumke)

Software Metrics: A Rigorous and Practical Approach, Revised (Norman E. Fenton and Shari Lawrence Pfleeger)

Software Reliability Engineering: More Reliable Software Faster and Cheaper, 2nd Edition (John D. Musa)

Software Process:

Agile Software Development, Principles, Patterns, and Practices (Robert C. Martin)

Agile Software Development: The Cooperative Game, 2nd Edition (Alistair Cockburn)

Cleanroom Software Engineering (Jesse H. Poore)

CMMI®: Guidelines for Process Integration and Product Improvement, 2nd Edition (Mary Beth Chrissis, Mike Konrad, and Sandy Shrum)

CMMI: Improving Software and Systems Development Processes Using Capability Maturity Model Integration (CMMI-DEV) (Ralf Kneuper)

Introduction to the Team Software Process℠ (Watts S. Humphrey)

Practical Software Process Improvement (Robert Fantina)

Software Process Dynamics (Raymond J. Madachy)

The Unified Software Development Process (Ivar Jacobson, Grady Booch, and James Rumbaugh)

Software Refactoring:

Refactoring: Improving the Design of Existing Code (Martin Fowler, Kent Beck, John Brant, and William Opdyke)

Refactoring in Large Software Projects: Performing Complex Restructurings Successfully (Martin Lippert and Stephen Roock)

Refactoring to Patterns (Joshua Kerievsky)

Software Verification and Testing:

Exploiting Software: How to Break Code (Greg Hoglund and Gary McGraw)

Implementing Automated Software Testing: How to Save Time and Lower Costs While Raising Quality (Elfriede Dustin, Thom Garrett, and Bernie Gauf)

Software Test Automation (Mark Fewster and Dorothy Graham)

Software Testing and Analysis: Process, Principles and Techniques (Mauro Pezze and Michal Young)

Software Testing: Testing Across the Entire Software Development Life Cycle (Gerald D. Everett and Raymond McLeod Jr.)

Software Testing Techniques, 2nd Edition (Boris Beizer)

Software Verification and Validation: An Engineering and Scientific Approach (Marcus S. Fisher)

Small Sampling of Internet Resources

USC Center for Software and Systems Engineering http://csse.usc.edu/csse/

The Software Engineering Institute http://www.sei.cmu.edu/

The Mathworks™ Technical Literature on Real-Time Workshop 7.3 *http://www.mathworks.com/products/rtw/technicalliterature.html*

The Object Modeling Group http://www.omg.org/

InterNational Committee for Information Technology Standards http://www.incits.org/