



2009

Mini-

Conference

Use Cases for Systems Engineering of a SoSs

- Biography - Malcolm Currie

UCLA: Control Systems Engineering Ph.D.

Engineering Consultant - Aerospace and Commercial:

Lear Seigler, Rockwell, Hughes, Boeing,
Rocketdyne, Schindler Elevator, ...

Missile/Aircraft guidance, navigation, and control systems; Military Systems requirements; Elevator and Security Systems control software, ...

Member of IEEE, GNSS, INCOSE ...

- Over 14 years experience in UML



2009

Mini-

Conference

What is a system?

- What makes a component of a SoS a system rather than a subsystem?
 - There are formal definitions of a system.
 - I will not quote them here.
 - Some examples in the context of a SoS will be used instead.
- What makes a component of a SoS a system rather than a subsystem?
 - The component system performs some function whether or not it is connected to the SoS.
 - An example is each automobile on a highway.
 - Another example is a modern day all-in-one printer.
- The primary focus of a Systems Integrator of a SoS must be on the interfaces
 - They are often whatever they are (the printer provides it).
 - Each system in the System has its own independent goals
 - It also has goals in its role in the SoS



2009

Mini-

Conference

How are SoS Use Cases different?

- The line is blurred between how much emphasis is to be placed on interfaces in a SoS compared to a system with subsystems.
 - **The less control a Systems Integrator has on the interfaces, the more his responsibilities shift** away from the working of the components and more **to how to use the interfaces of the components.**
 - **Use Cases are a valuable tool for defining and controlling system interfaces.**
 - From conception to final acceptance test.
 - Used by management to control the progress of incremental development.
 - Improve **meeting the primary needs of the customers.**
- Build **executable models** of the Use Cases
 - To, for example, **clearly define the interfaces between the system and its environment**
 - Do before allocating the functions to specific parts
 - Modeling the functionality first increase the likelihood of premature implementation.
- Use Cases are synergistic with the **principle of incremental development**
 - Use Cases provide a means to evaluate risks
 - technology risks
 - project cost and schedule risks.
- Use Cases also provide management a means to specify and monitor testing of the System.
 - Well constructed high level Use Cases map to final acceptance tests
 - Use Cases provide a means to **plan tests of the system early in development (very early)**



2009

Mini-

Conference

Highest level SoS Use Cases

- **What do the highest level of Use Cases look like for a SoS?**
 - A SoS is a system.
 - Highest level Use Cases looks like the highest level of Use Cases for any system.
- **What are the roles of the Actors?**
 - Not different than for any other system - yet.
 - Their may be constraints in the architecture of the SoS.
- **The component systems are not outside of a SoS**
- **The Use Case development will follow the usual description**
 - Preconditions, etc.,
 - Activity Diagrams and/or SD (Sequence Diagrams).
 - Can go directly to conceptual SDs.



2009

Mini-

Conference

Second level SoS Use Cases

- Conceptual Sequence Diagrams at the highest level can be used Use Cases at the next level
 - Do not decompose a Use Case to get next level Use Cases
 - Ignore as much as reasonable and possible that much of the functionality is performed by certain provided component systems.
 - Find conceptual objects with the responsibilities and interfaces to perform the operations that support the highest level SDs
- A Use Case diagram does not execute.
 - Each SD that supports a scenario of the Use Case executes.
 - No different than the process for developing a system with executable SDs.
- The differences: design constraints of the services provided by the supporting systems
 - The allocation of the conceptual objects to component systems may constrain available interfaces
 - Same types of constraints are present when developing a system using available component subsystems.
- The **essential difference** for usual SoS development:
 - The amount of control in specifying the interfaces with the components of the system.
 - It is not that the design of a SoS is different per se.
- **The provider of the components of the SoS may control the interface**
 - **along with the functionality of the components**
 - The SoS developer has to make compromises to the desired design to accommodate those constraints.
 - The same constraints could occur in the development of some systems with interfaces and functionality provided by available subsystems.



2009

Mini-

Conference

Development differences for SoS?

- **How is the development of a SoS any different that for a system of subsystems?**
 - The difference is control of the interfaces
 - If system integrator of a SoS has control of the interfaces and functionality of all the component systems, there is nothing new except the possibility that often the SoS will be larger and more complex than many other systems.
- **The Use Case approach to drive and control the development process is even more valuable and adaptable for large and complex systems development**
 - Especially when the interfaces and functionality of the components are not controlled by the system developer.
- **Well constructed high level Use Cases map to final acceptance tests of a system**
 - Use Cases provide a means to plan tests of the system early in development (very early).



2009

Mini-

Conference

Questions?

- Malcolm Currie
 - SEC_Services@Earthlink.net; 310 821-3081
- Special Thanks to Track Chairs
 - Track 3 **Evolving Systems Engineering To Address Soft Systems and Systems of Systems**
 - Josh Sparber (Joshua.Sparber@dcma.mil),
 - Jeff Lankford (Jeffrey.P.Lankford@aero.org)
- And to Conference Organizers
 - Technical Manager—Richard Emerson
 - r.emerson@computer.org, tel. 818.790.7017
 - Conference Manager—Shah Selbe
 - (Shah.Selbe@boeing.com)