# Systems Engineering Lite

James Kolozs

syncroness™
innovative product development

July 12th, 2012

# Intended Audience

- Technical project managers
- New systems engineers
- Small/medium sized product development companies
- Large product development companies seeking a simplified SE process and tools for smaller projects

# Background

- Syncroness founded 1998; located in Colorado, USA
- Syncroness assists companies with outsourced:
  - Research
  - New product development
  - Sustaining engineering
- Multiple functional groups (~70 engineers)
  - Mechanical, Electrical, Firmware, Software, Project Management, Systems Engineering
- Mature product development and project management processes
  - ISO 13485 certified; 21CFR820 compliant
  - Initiated formal systems engineering 2 years ago

# The Challenge

- How much systems engineering?
  - We use detailed SE process/tools and model based SE (CORE) for larger projects
  - Initially a struggle to determine the appropriate level of SE for our small/medium sized projects:
    - 4-26 weeks duration
    - 8-80 man-weeks of effort

# The Challenge

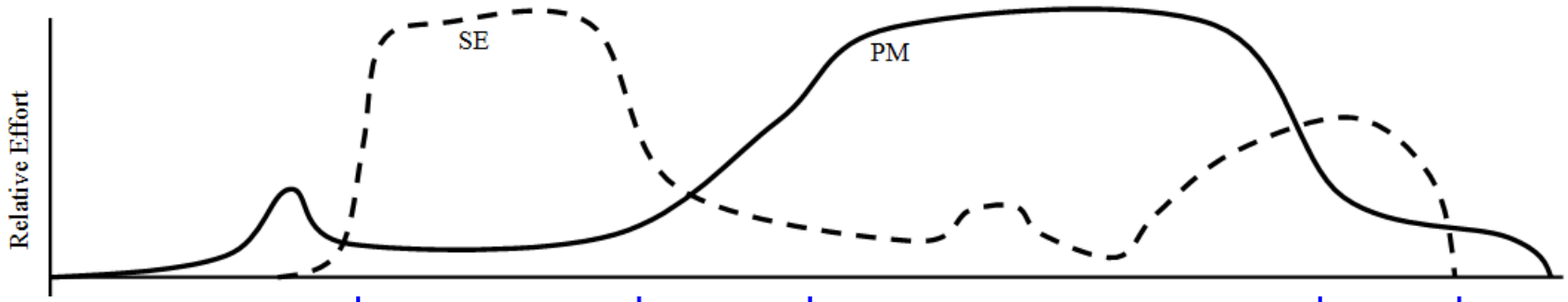
- Characteristics of smaller projects:
  - Fewer requirements (<100) and simpler deliverables
  - Fewer engineering disciplines involved (perhaps only one)
  - Less staff (2-8 on a project) and simpler organizational structure (at most 3 tiers)
  - Less complex tradeoffs (The architecture is dictated or the basic architecture is obvious)
  - Fewer and simpler interfaces

# The Need to Simplify SE

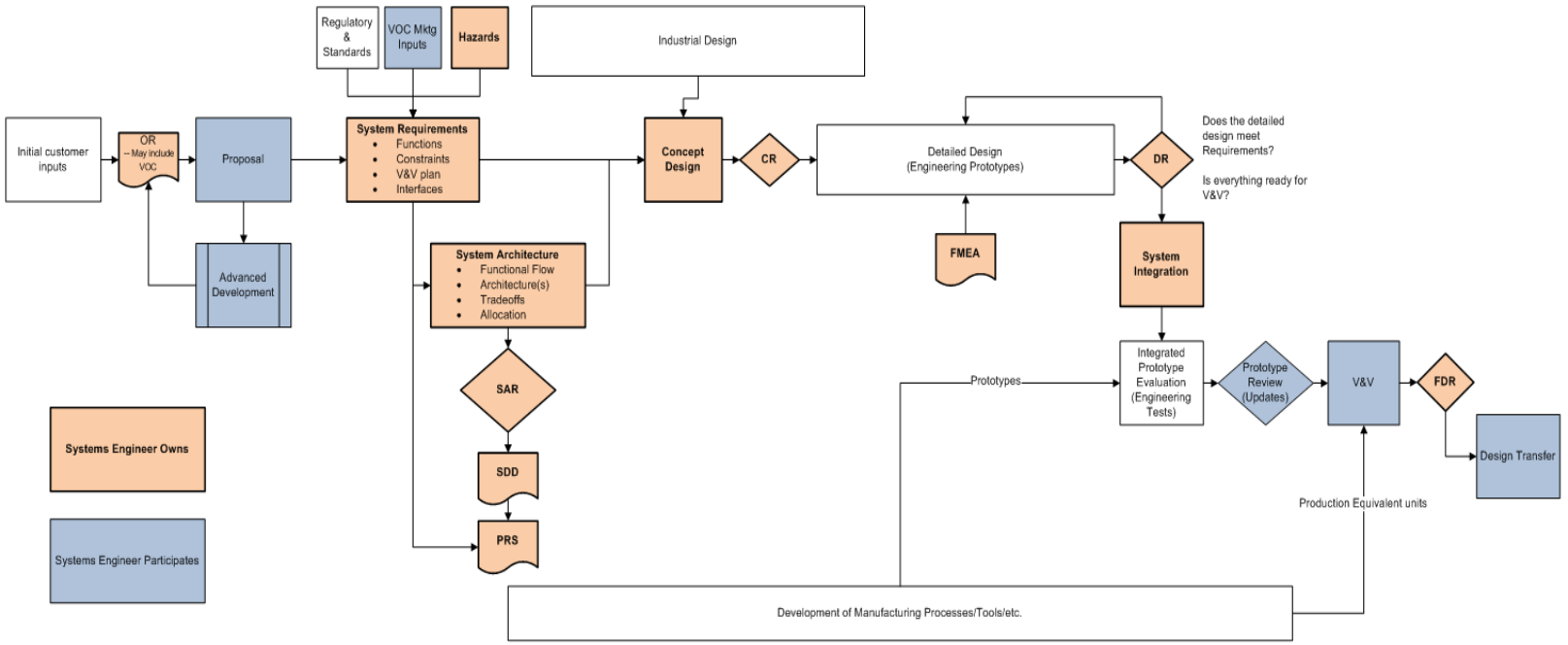
- Few resources available for SE for small projects/businesses
  - Integration of multiple technologies and fields means that even small projects can face complexity issues
- You are expected to tailor SE, as it is scalable, but how?

- *…Just found out about the Very Small and Micro Entities Working Group!*

# PM/SE Effort On Smaller Projects

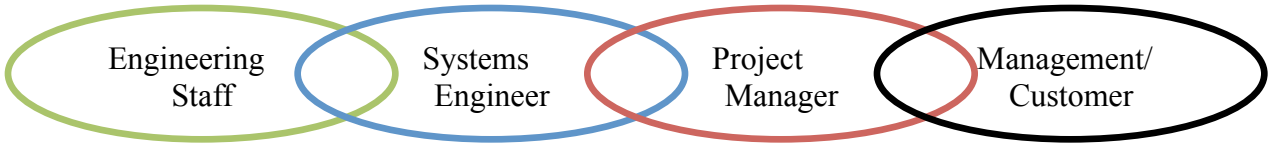



SE Activities



# PM/SE Paradigm

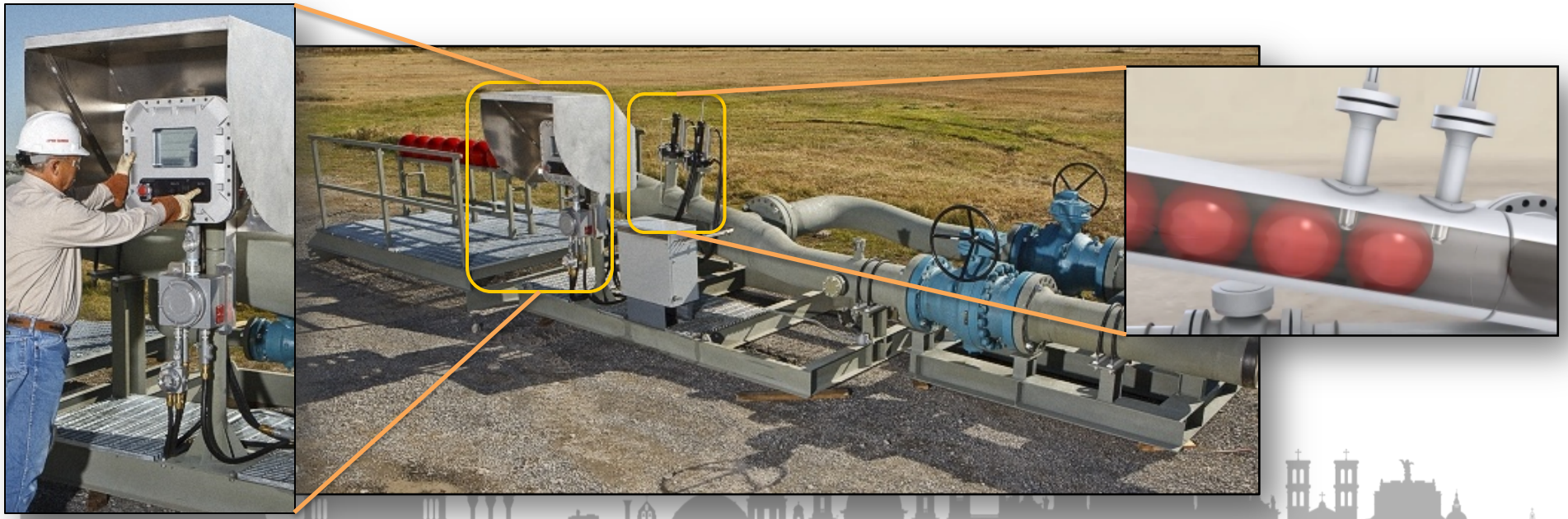
- Overlapping domains on a large project




- SE-Lite project

# Example Project

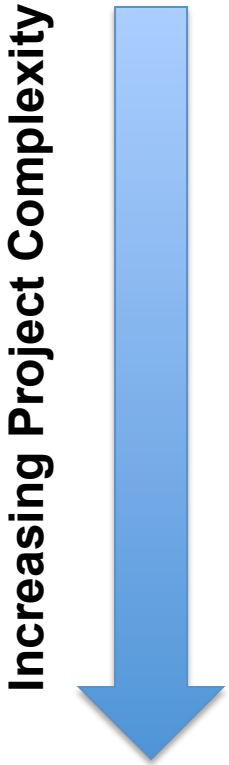
- T.D. Williamson, Inc. SmartTrap® Automated Combo Pigging System
  - Automatic, timed launching of spherical “pigs” to clear gas pipelines of water buildup
- Design hydraulic dual pin subassembly (many configurations)
- Design control subsystem with GUI interface
- 6 month duration from initiation to 3 production systems



# Scalable Solution – SE Lite

- Goals:
  - Make basic systems engineering tools accessible to a technical project manager on smaller projects
    - PM/SE
    - But don't overburden them – just enough SE to add value
  - Improve system definition
  - Improve project quality
  - Improve communications within the team and with the stakeholders

# SE-Lite Tools




| Project Type | Tool |
|---|---|
| All | Product Requirements Specification |
| Most | Physical Block Diagram |
| | Enhanced Functional Flow Block Diagram (Behavior) |
| Medium Sized | Physical Architecture |
| | Interface Control Document |
| | Trade Studies |
| Heavy User Interaction or GUI | Use Cases |
| | User Interface Document |

- *Does not require specialized SE software tools!*

# What's Missing?

- A lot! But that is the point.

- *We are adding SE capabilities to the PM, not making the PM a SE*

# Gathering Requirements

- The stakeholder(s) might give you “originating requirements” at the start of the project:
  - Document
  - Email
  - Napkin sketch
  - Conference call
  - Meeting notes
  - Regulatory specifications
- Often need to use a questionnaire to “pull” requirements from the stakeholder
  - Start with system level requirements, then work down to subsequent level(s)
- Derive and manage requirements during the early phases

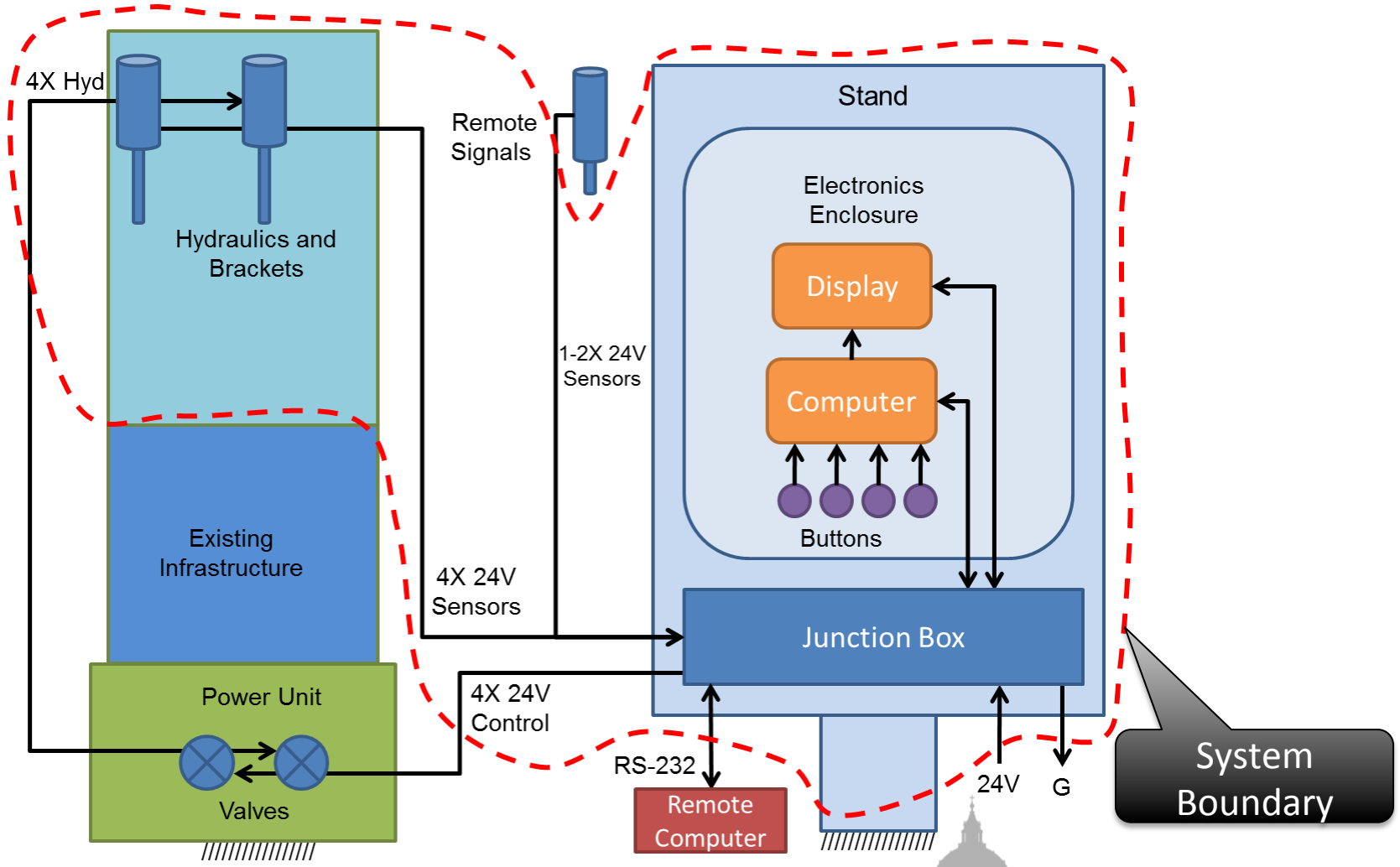
# Product Requirements Specification (PRS)

- All projects, regardless of size, require a PRS
- Requirement attributes (recommended)
  - ID (number)
  - Requirement (name)
  - Specification (detailed, quantified description)
  - Status
  - Source (traced)
  - Verification (traced)
  - Verification Owner
  - Verification Results

# Physical Block Diagram

- Combination view of the system
  - Overall layout and relationships of important subassemblies and components; not to scale
  - Rough interfaces and signals
- Helps define system boundary – what is “in” and what is “out”
- Can use it to allocate functions
  - Which component executes a particular function?

# Physical Block Diagram Example



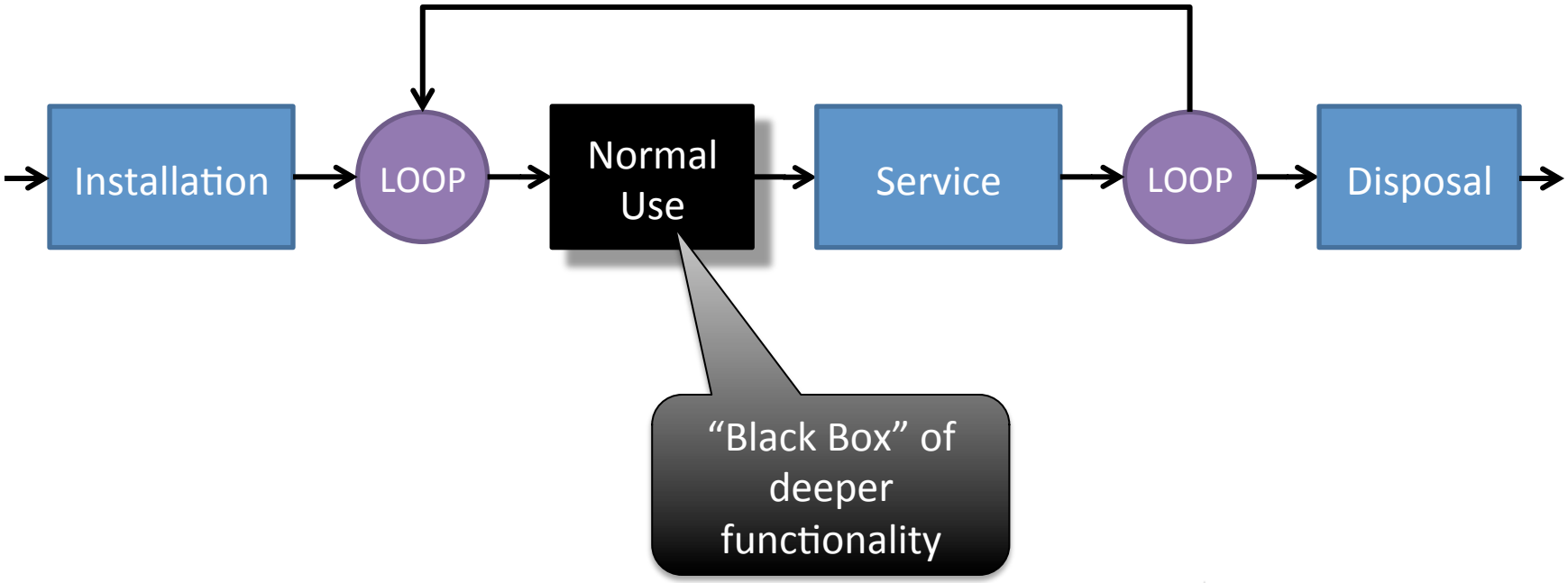
# Enhanced Functional Flow Block Diagram (EFFBD)

- Meet with multiple stakeholders to determine the behavior (logic) of the system
  - Pull from Use Cases (if available)
  - Users
  - Management/Executives
  - Marketing
  - Engineering
  - Installation/Repair/Maintenance
- Do not define the implementation
  - What does it do, not how it does it
- Work in layers (functional black boxes)
  - Start at the system level
  - Break down sub-system functions into their own EFFBDs

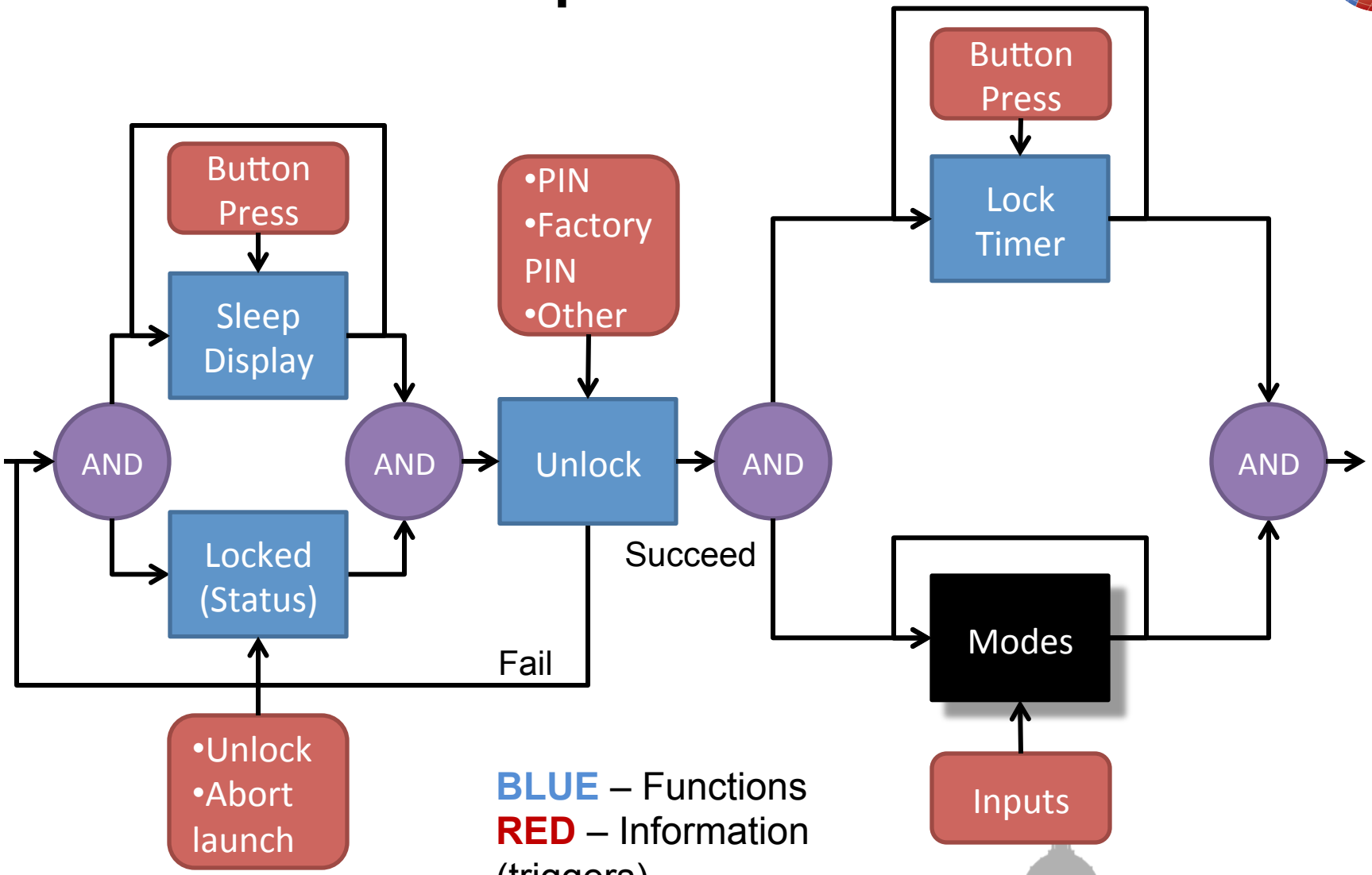
# Enhanced Functional Flow Block Diagram (EFFBD)

- Use detailed constructs to understand the functional relationships
  - "Fancy" block diagram
  - Sequence, And (Parallel), Or (Select), Decision, Iterate, Loop (w/exit)
- Use information blocks to understand data flow between functions
  - Inputs/Outputs/Triggers

# EFFBD Example – **Top Level**

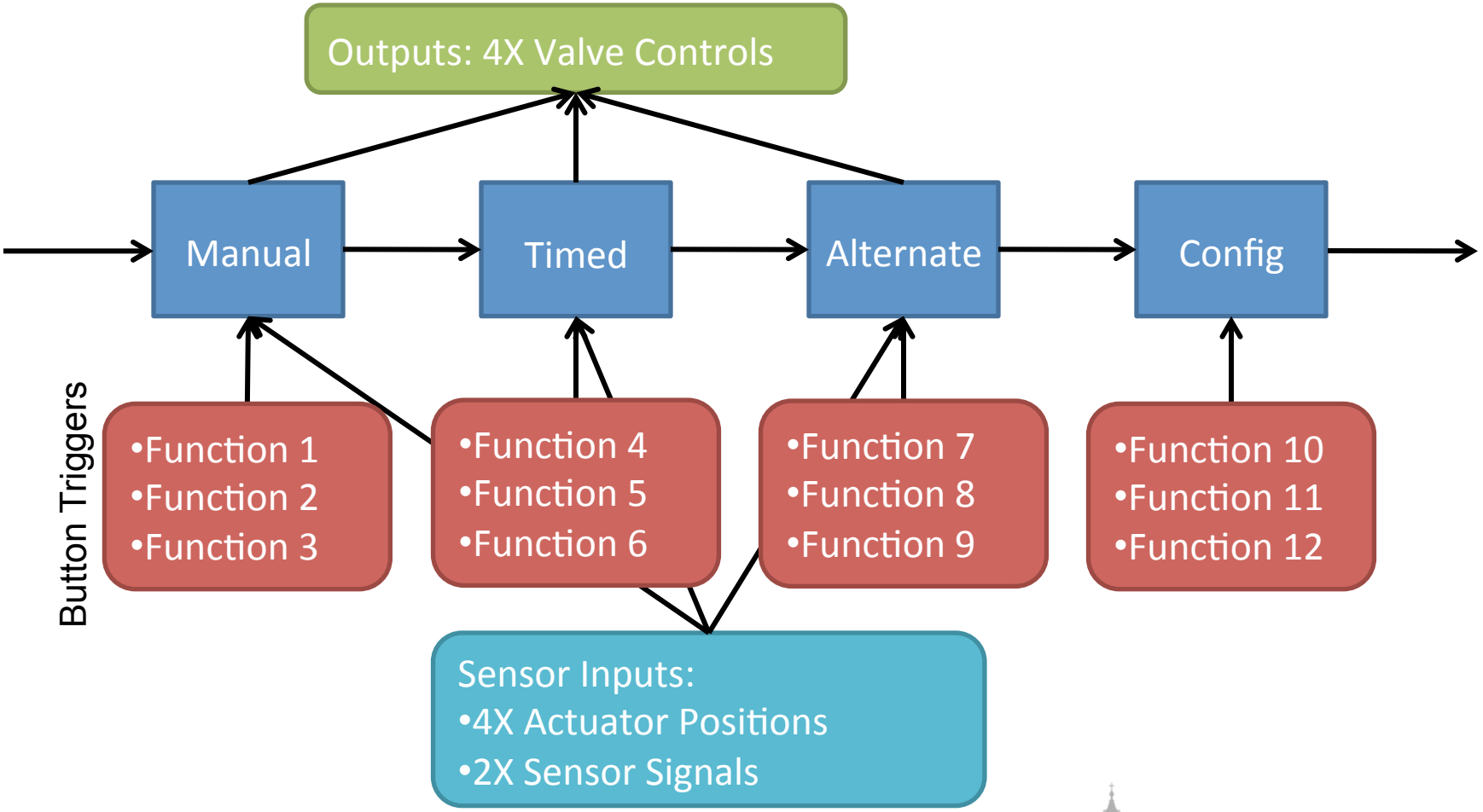


# EFFBD Example – **Normal Use**




**BLUE** – Functions
**RED** – Information (triggers)

# EFFBD Example – **Modes**

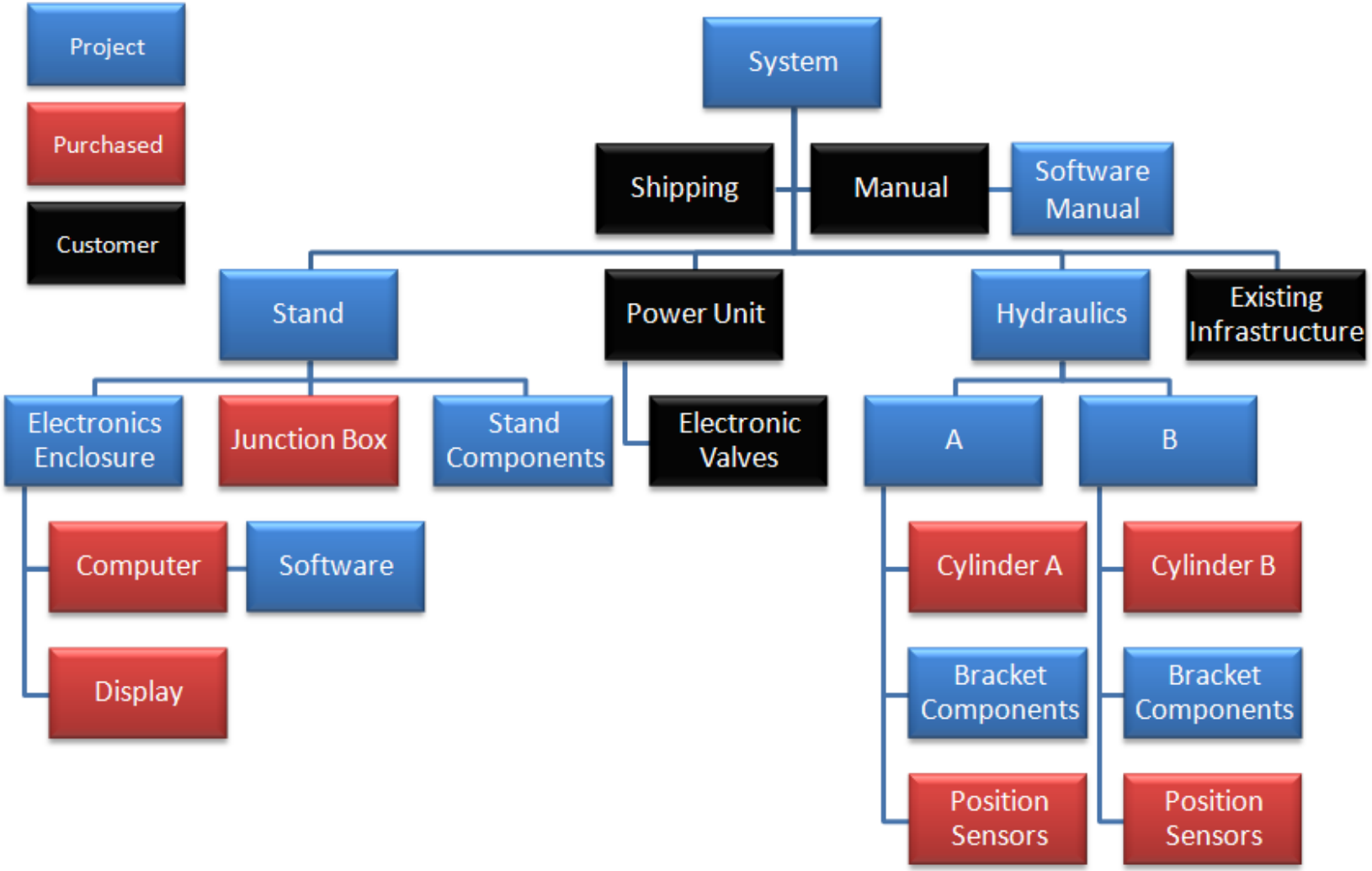


# Physical Architecture

- Hierarchical view of the system, assemblies, subassemblies, and major components
  - Basically, a high-level BOM
- Can use it to clearly assign responsibilities, costs, weights, etc.
- Functions map to components in the physical architecture
  - Known as “allocation”
  - Should be obvious in a smaller system

# Physical Architecture - Example

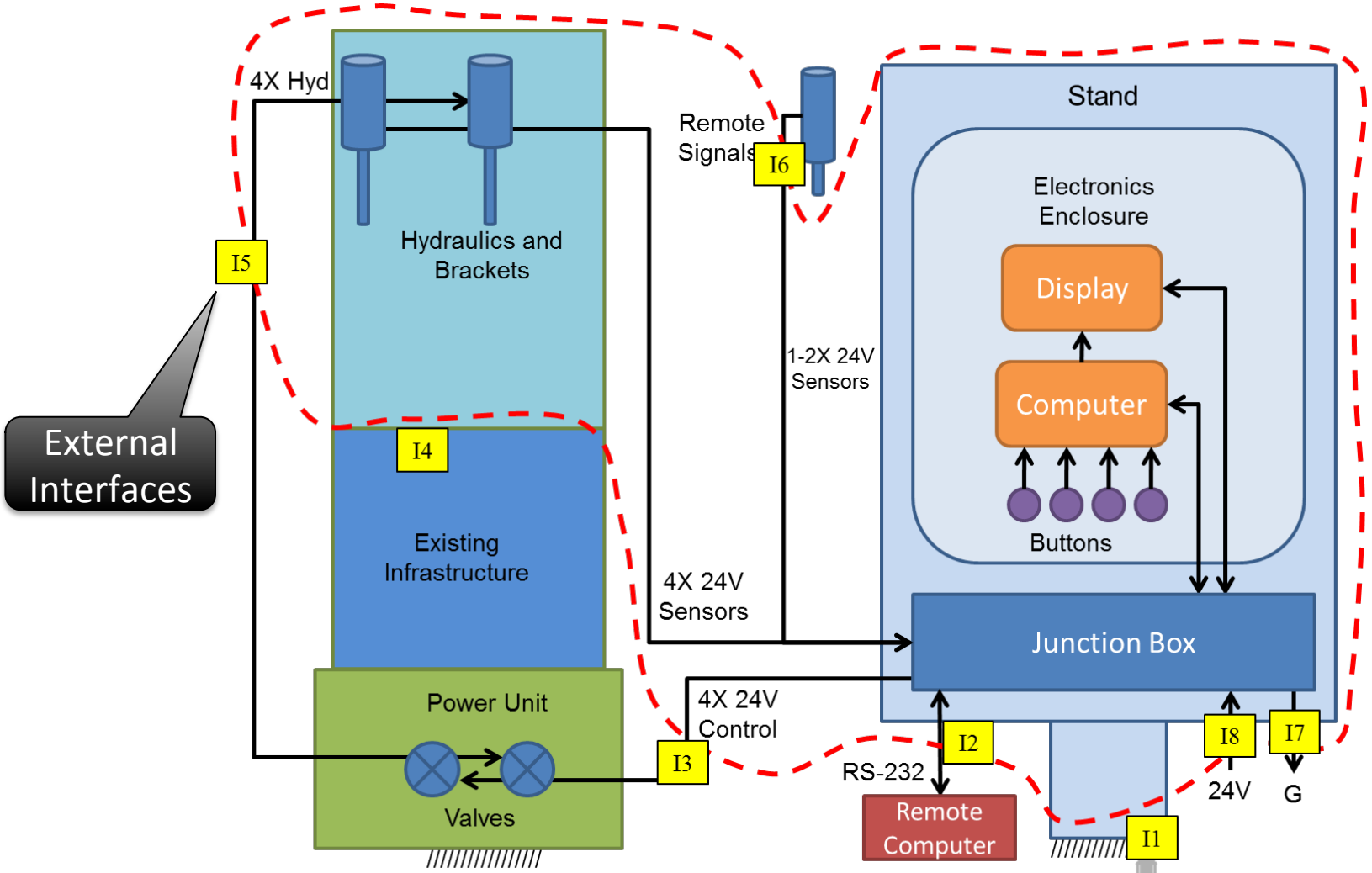
Responsibility



# Interface Control Document

- Trace the system boundary in the Physical Block Diagram to identify external interfaces
- Also useful to document critical internal interfaces
- Each interface is defined by:
  - Name
  - Number
  - Description
  - Definition of Electrical Interface
  - Definition of Mechanical Interface
  - Definition of Software Interface
  - Interface Verification and Owner

# Interface Control Document



# Trade Studies

- Needs to occur when architecture could take different forms
  - Forms have different pros/cons
  - Best solution is not obvious
- Typical Process:
  1. Brainstorm alternatives
  2. Use Go/No-Go criteria to weed down ideas
  3. Conduct rapid feasibility study, determine rough answers to decision criteria
  4. Use decision matrix to choose best candidate architectures
  5. Deep investigation on candidate solutions and/or down-select to begin development

# Trade Study – Decision Matrix

- Decision matrix is used to quantify opinions and establish hierarchy of solutions
  - Determine evaluation criteria, weighting factors
  - Evaluate each concept against each criteria
- Be careful with biases
  - Not a substitute for engineering experience
  - Stakeholder performs separate analysis and then reconcile

Criteria

Concepts

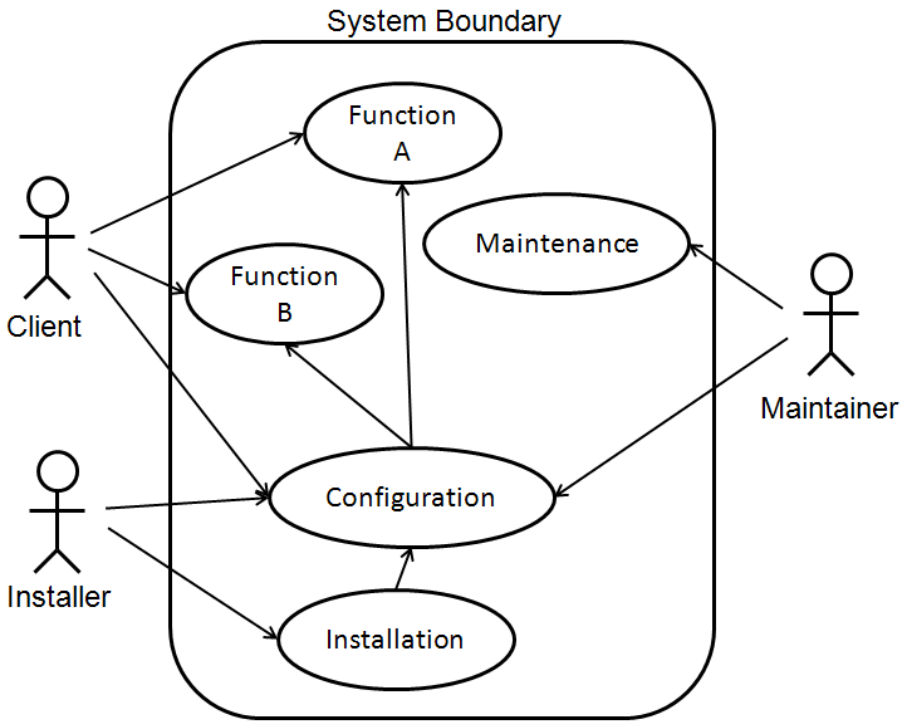
| # | Weighting Factors→→→ | 10 High Performance | 8 Small Size | 7 Low Weight | 5 Low COGS | 3 Short Development Time | Un-Weighted Totals | 33 Weighted Totals | Notes |
|---|---|---|---|---|---|---|---|---|---|
| | Sorting and Filtering→ | | | | | | | | |
| 1 | Concept 1 | 9 | 8 | 6 | 7 | 4 | 6.8 | 7.4 | |
| 2 | Concept 2 | 9 | 5 | 5 | 6 | 4 | 5.8 | 6.3 | |
| 3 | Concept 3 | 5 | 5 | 6 | 6 | 5 | 5.4 | 5.4 | |
| 4 | Concept 4 | 5 | 7 | 7 | 7 | 9 | 7.0 | 6.6 | |
| 5 | Concept 5 | 4 | 6 | 6 | 5 | 10 | 6.2 | 5.6 | |

# Use Cases

- If there are significant or complex user interactions with the system, use cases should be determined
- Use cases outline all of the “actor” interactions with different parts of the system
  - Note that some actors may not be people
- Make sure that the use cases are clearly communicated and reviewed by ***all*** stakeholders:
  - Users
  - Managers/Execs
  - Marketing
  - Engineering
  - Production
  - Installation/Repair/Maintenance
- *Not all stakeholder opinions carry the same weight*

# Use Cases Diagram

- Generate a use case diagram
- Separately describe and document each “actor’s” story
- Serves as input to the EFFBD



# User Interface Document

- A User Interface Document is helpful to generate requirements when there is heavy user interaction and feedback

- Graphic User Interface
  - Industrial design guidelines
  - Screens; permanent vs. temporary information
  - Graphics, animations, status bars
  - These can be PowerPoint mockups, to be replaced by screenshots as GUI is developed
- Ergonomics
  - Buttons, handles, mouse, touch interface, etc.
- Labeling
- Configuration
- Languages/Special character sets
- Messaging/Warnings/Errors
- Logging

# Building Consensus

- Each document helps align stakeholders and the design team
  - Smaller projects are often ill-defined, so this process helps to “fill in the blanks”
- Minimizes impact to development cost/schedule by forcing consensus early on, when design changes are easy to implement
  - **Very helpful to have stakeholders sign off on the key requirements and architecture documents**
- Expect changes
  - SE-Lite will provide a framework for everyone to understand the impact of later changes

# Summary

- Systems engineering needs to be scaled down for smaller projects!
  - "You shall not overwhelm small projects with heavy systems engineering processes, tools, and documentation."
- Systems Engineering Lite:
  - Brings SE discipline to technical PMs in measured doses, using common tools
  - Increase communication and understanding between stakeholders and design team
  - Has resulted in reduced costs, happy customers, and more successful projects

# Thank You

syncroness™
innovative product development

# Happy Customers = Repeat Business

- *"Syncroness used several risk analysis and design requirements vetting tools, in addition to their normal, weekly design review conference calls (with accompanying visual presentation materials) to create very efficient and effective communication and collaboration with us on this somewhat complex "mechatronic" product development project. The results were relatively easily implemented into production and customers have been very satisfied with the product."*
  - **Eric N. Freeman, PE, T.D. Williamson, Inc.**
    ***Manager, Engineering & Technology Development***

# Example Project

- Phase 1
  - 4 week duration; ~8 man-weeks effort
  - Trade study on pin styles
  - Trade study on electrical control and interface
- Phase 2
  - 20 week duration; ~33 man-weeks effort
  - SE-Lite
  - Design/document system
  - Procure/assemble/verify three systems
  - Onsite integration
- Phase 3
  - Production support as needed

# PRS – Status

- Requirements on small, fast moving projects are often not well defined
- Use “Status” to push for clearer requirements with each stakeholder meeting
  - **Firm** – Known, quantified, customer approved
  - **Preferred** – Known, quantified; not certain or not approved
  - **Estimate** – Qualitative, or quantitative values not known
  - **Unknown** – Placeholder requiring more research

# PRS - Verification

- Each requirement needs to identify how it will be tested, and by whom
  - Removes ambiguity, strengthens confidence in the design
  - Assigns responsibility for verification
  - Understand the costs of testing
- Common verifications:
  - Inspection
  - Analysis
  - Test
  - Demonstration
- If verifications are simple enough, report directly in PRS