

Interface Centric System Development

Chin-An Cheng, CSEP
Dave Mason, CSEP
SFBAC Meeting
July 8, 2008

Agenda

- Background
- Introduction
- System Development Process
- Change of Perspective
- Interface Anatomy
- Definitions
- Interface Centric System Development Approach
- Example – Motor Controller
- Summary

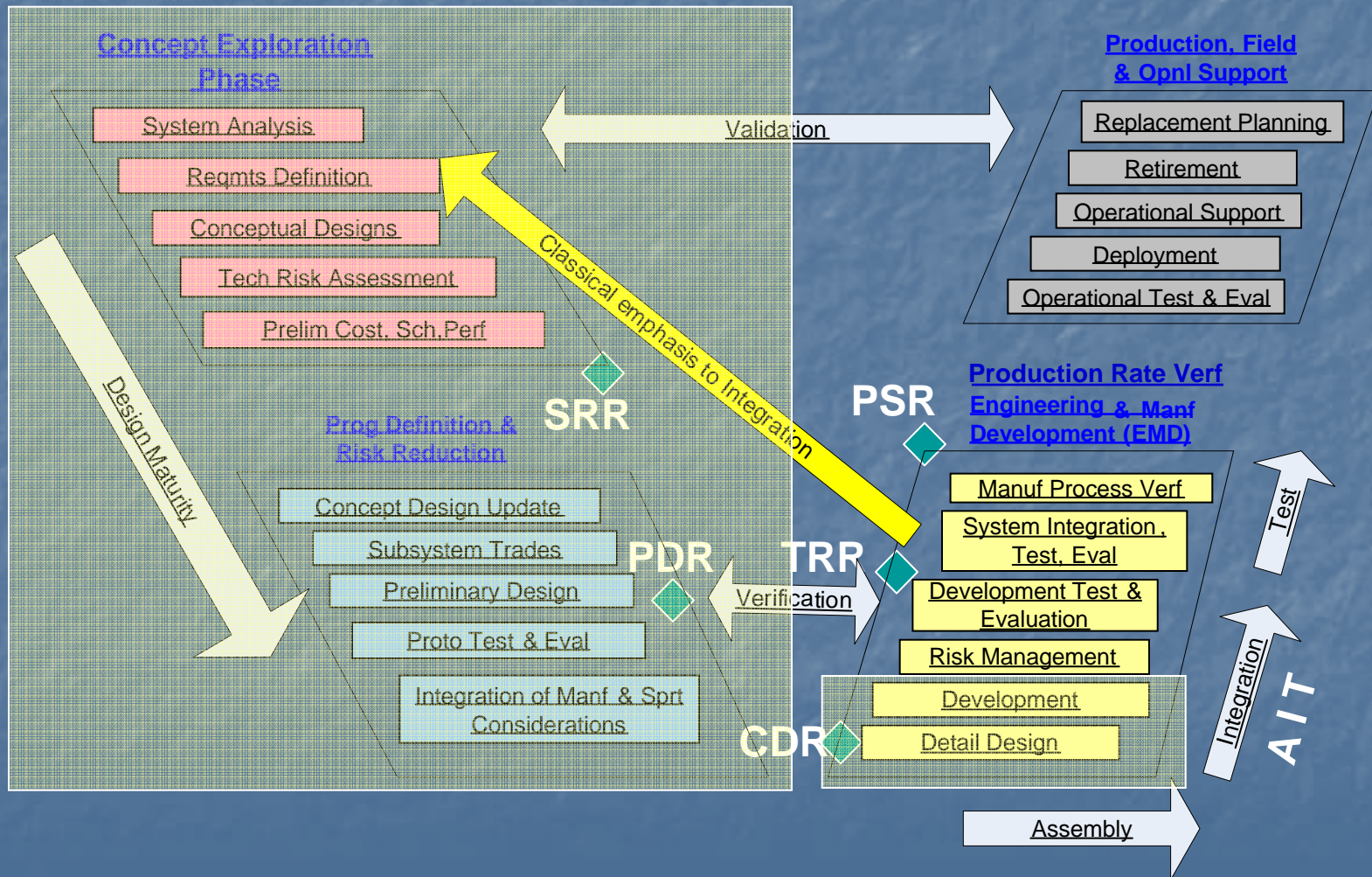
Background

- System development need to consider all aspects of the system
 - Tao of systems
 - We must look at the system as a whole (Macro view)
 - The three system descriptors
 - Parts
 - Provide individual function and building block
 - System architect
 - Establish relationship among the parts
 - System interface
 - The enabling factor of the added value
 - A balanced system is a good system (DM Presentation on SI)
 - Use multiple views
 - Architecture view
 - Traditional function view
 - Interface view
 - Operation
 - Interdependency

Introduction

- I/F Centric System Development (CSD) approach
 - An I/F CSD approach is a system development process focusing on the messages passed between system elements.
 - I/F CSD focuses in the messages and the resulting behaviors
 - I/F CSD can be performed in parallel with traditional SD
- I/F CSD is not a new approach
 - I/F CSD is an operation based design
 - I/F CSD is an Object Oriented design
 - We have development systems using interfaces at extreme high and low levels or IF CSD
 - Examples:
 - System Context Diagram
 - Lowest level components

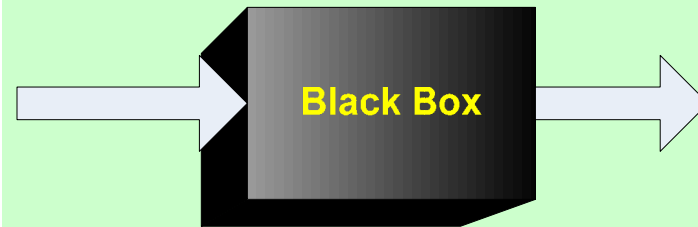
System Development Process



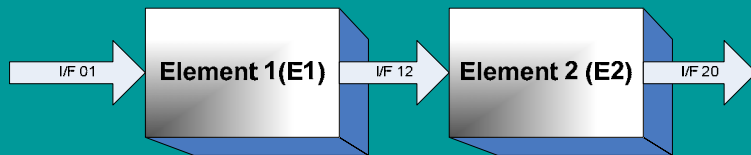
Change of Perspective



Vs



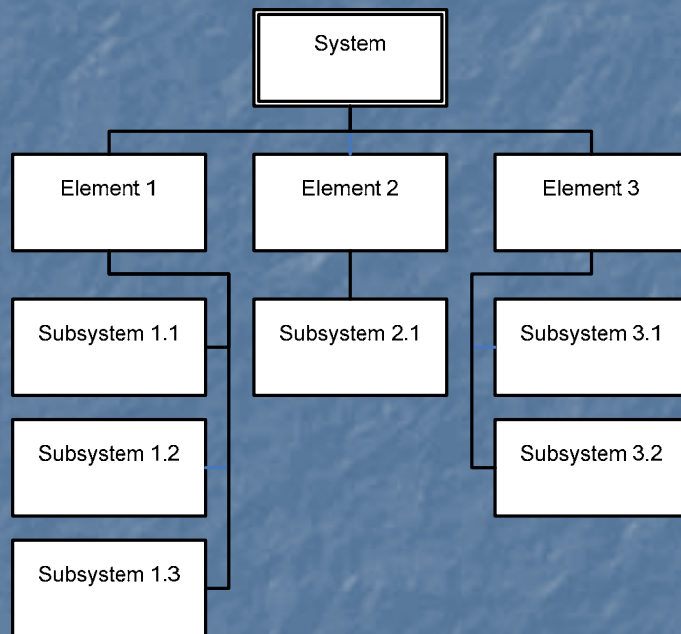
Vs



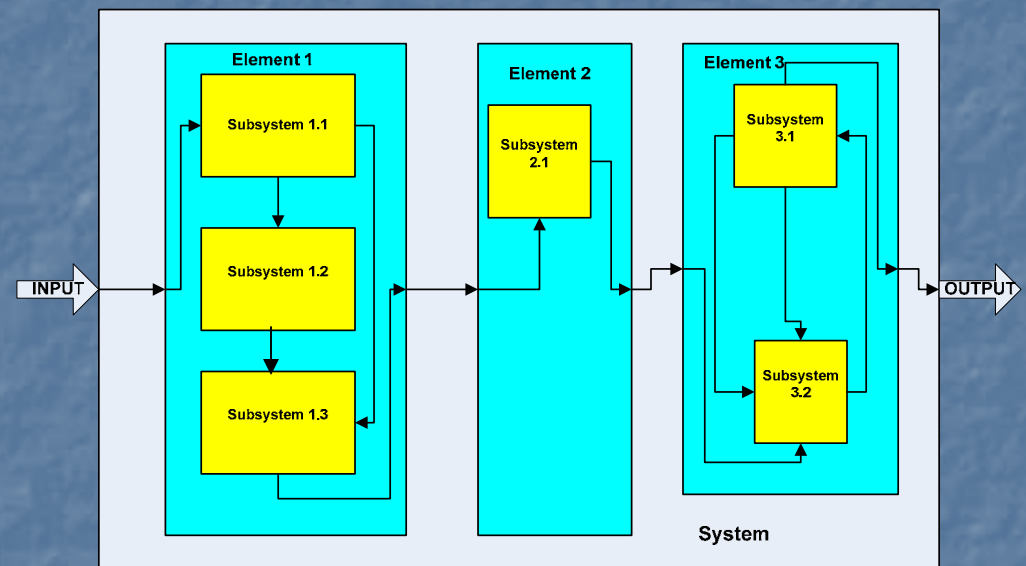
Traditional Perspective

I/F Centric Perspective

Architecture Views



Traditional View



I/F CSD View

Interface Anatomy

- Relationship (Parent-Child, Predecessor-successor, Master-slave)
 - Sequence (Operation or Procedure)
- Governing Rules (& Standards)
- Formats (form)
- Protocols (handshake) (fit)
- Messages (function, operation, parameters)

Definitions

- Objective of a system (redefined)

The objective of a system is to process the incoming message (inputs) and convert it into a desired outgoing message (outputs)

- This conversion process is called "behavior"
- I/O (interfaces) defines the "behavior" of the system

- Definition

- **Behavior** is the actions or reactions of an organism, usually in relation to the environment
- A **scenario** is a synthetic description of an event or series of actions and events
- A **thread** is a sequence of instructions which may execute in parallel with other threads

I/F CSD Approach

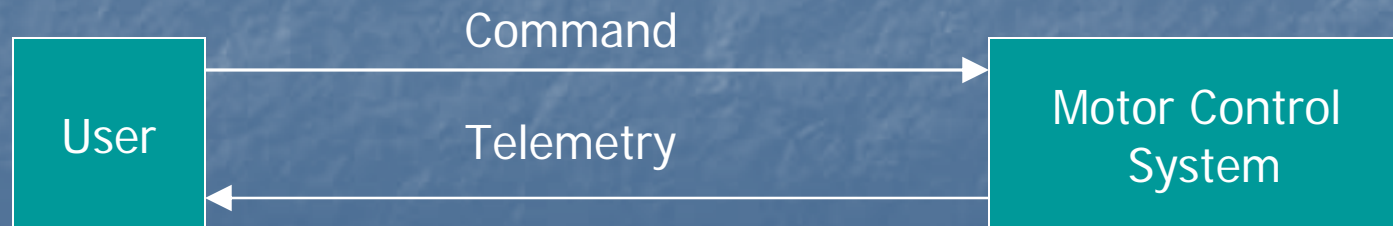
- Customer Requirements
 - Define Use interfaces
 - Scenarios
 - Threads
- Develop Con-ops
- Define high level architecture
- Define lower level interfaces (I/Os) to the lowest levels
- Develop operation flows
- Develop element behaviors (functions)
- Repeat the process at all element levels

Example-Motor Controller

- Objective
 - Control motor with external commands
- System architecture (three elements)
 - Control Software
 - Electrical System
 - Motor
- Interfaces
 - User to Control Software (Commands / Telemetry)
 - Control Software to Electrical System (Sub-Commands / Telemetry)
 - Electrical System to Motor (Physical Interfaces)
 - User sees motor move (by either the messages or actual motion)

Use Cases

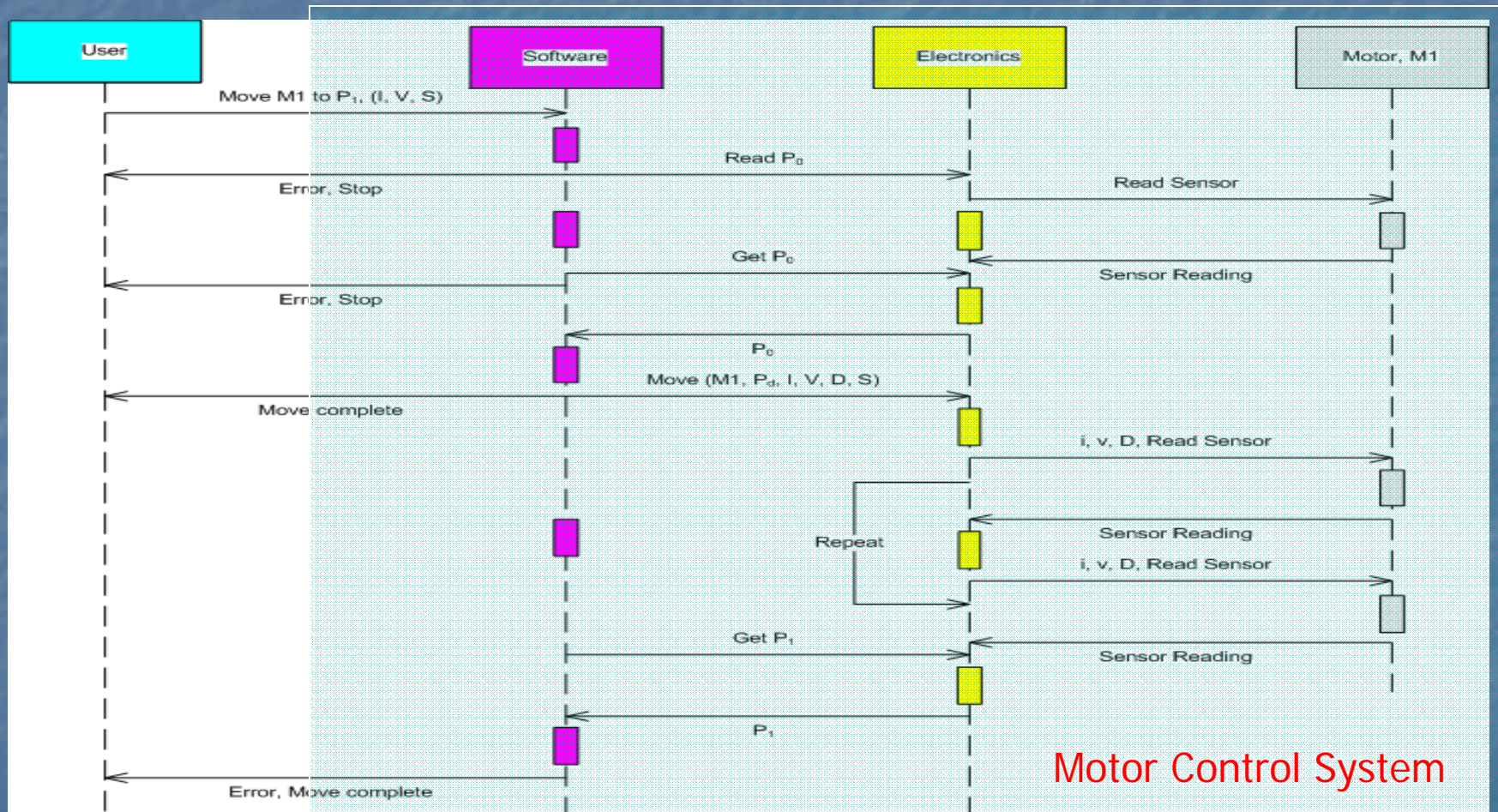
- User command system to move motor with a predefined position:
 - Speed, direction, voltage, current and position
 - Using Consultative Committee for Space Data Systems (CCSDS) packets
- Receive Telemetry with
 - H&S (Health and Safety)
 - Operation status: error and event message
 - Using CCSDS packets
- Con-ops
 - System to receive the command, software to process and issue sub-commands to electronics control motor



Identify Scenarios and Threads

- Scenario 1
 - Happy Path
- Scenario 2 - Command Problems
 - Format, Protocol
 - Message error – Missing, out of limit parameters
- Scenario 3 – Sub-command Problems
- Scenario 4 – Telemetry Error.....

Define & Allocate Messages



System Level N² Diagram

User	Command, control parameters		
Telemetry, Acknowledge, Error, etc.	Software	Sub- commands	
	Position Sensor	Control Electronic	Current, Voltage, etc.
Movement		Position Sensor	Motor

Software Level N² Diagram

User	Command							
	Inputs (Command Processor)		Parameters	Command Counter	Accept, Reject, Error		Function Code	
		Clock	Timer	Timer	Timer	Timer		
			Algorithm, Compare		Error, Complete	Parameters		
H&S				H&S	Status			
					Outputs (Telemetry Generation)			
Telemetry						Send (Sub-Commands)		Control parameters, Sensor Read, Sensor Get
			Preset Time, limits			Sub-Command Sequence	Database	
			Sensor Data					Control Electronics

Define Behaviors & Requirements

- Software
 - 1 Response command (check format, protocols)
 - Accept – acknowledge and command read
 - Reject – error message, stop
 - 2 Wait until $t = t_1$, then send request
 - 3 Compare value
 - Send error message, if no return value
 - Calculate and send move command if $P_1 <> P_0$
 - Stop and send success message if $P_1 = P_0$
 - 4
- Control Electronics

Summary 1/2

- System interfaces should be used to system development
- Benefits
 - Operation drives design
 - User interfaces are included in the design
 - System Interface and System Integration issues are addressed early in the development cycle
 - Less likely to over design

Summary 2/2

- System needs to be developed as a “whole”, by addressing all three descriptors of the system
- Interface Centric System Development
 - Should be used in conjunction with traditional SD process
 - Utilizes “Black Box” approach to the system development
 - Looks at all aspect of operation using use cases, scenarios and threads
 - Decompose interfaces though out all system levels and elements