

Concurrent Increment Sequencing and Synchronization with Design Structure Matrices in Software- Intensive System Development



Dr. Peter Hantos
The Aerospace Corporation

NDIA Systems Engineering
Conference
October 23, 2008

Acknowledgements

- This work would not have been possible without the following:
 - *Feedback:*
 - Suellen Eslinger, Software Engineering Subdivision
 - Dr. Leslie J. Holloway, Software Acquisition and Process Department
 - Mary A. Rich, Software Engineering Subdivision
 - *Sponsor*
 - Michael Zambrana, USAF Space and Missile Systems Center, Directorate of Systems Engineering
 - *Funding sources*
 - Mission-Oriented Investigation and Experimentation (MOIE) Research Program (Software Acquisition Task)
 - *Inspiration*
 - Dr. Barry W. Boehm, University of Southern California



Presentation Objectives

- Introduce a research platform to address concurrent engineering concerns of software-intensive system development
- Propose new metrics to characterize increment coupling and cohesion in complex, aggregate life cycle models



Agenda

- Wisdom
- Introduction
- ULCM[®] (Unified Life Cycle ModelingSM)
- Challenges of Concurrent Engineering
- DSM (Design Structure Matrix)
- Mapping Anchor Points to DSM
- CICM (Concurrent Increment Coupling Metric)
- Relationship Between CICM and Schedule/Cost Risk
- Next Steps – Direction of Future Research
- Summary
- Acronyms
- References

[®] ULCM is registered in the U.S. Patent and Trademark Office by The Aerospace Corporation
SM Unified Life Cycle Modeling is a Service Mark of The Aerospace Corporation



Wisdom

“To understand a subject, one must tear it apart and reconstruct it in a form intellectually satisfying to oneself, and that (in the view of the differences between individual minds) is likely to be different from the original form. This new synthesis is of course not an individual effort; it is the result of much reading and of countless informal discussions, but for it one must in the end take individual responsibility. “

Quote from J.L. Synge, “Relativity: The Special Theory” (1956), p. vii



Introduction

- The National Security Space Defense Acquisition Challenge
 - *Chronic cost/schedule overruns in space acquisitions*
 - *Difficulty with validating the contractors' plans*
 - *Difficulty with implementing proper controls*
 - *Difficulty with successfully executing Evolutionary Acquisition and Spiral Development-related policies*
- One of the Most Significant Root-Causes Identified
 - *Concurrent Engineering is pursued without proper models and tools to manage concurrent process streams*
- Proposed solutions involve the use of ULCM[®] (Unified Life Cycle ModelingSM) and DSM (Design Structure Matrix)
 - *ULCM[®] is an Aerospace-developed research framework and methodology*
 - *DSM is a widely used, visual system representation tool*



ULCM[®] – The 64 Thousand Mile View

- ULCM[®] is an intuitive, pattern-based approach for specifying, constructing, visualizing and documenting the life cycle processes of software-intensive system development
- ULCM[®] is aspiring to become the “Occam’s Razor” of Life Cycle Modeling
 - *The medieval rule of parsimony: “Plurality shouldn’t be assumed without necessity”*
 - William of Ockham, 14th century philosopher
 - *The Life Cycle Modeling (LCM) rule of parsimony: All life cycle models are constructs or derivatives of a small number of basic life cycle modeling patterns*
- ULCM[®] is also a research platform
 - *It provides a foundation for a consistent and universal system development methodology*



The First Principles of Unified Life Cycle Modeling*

1. Covered process domains are acquisition and development of software-intensive systems
2. The fundamental building block of life cycle models is an increment
3. All life cycle models are constructs or derivatives of a small number of basic LCM patterns
4. LCM is synergistic with architecture, architectural concepts and architecture modeling
5. Proper representation of life cycle models requires multiple views
6. Concurrent processes are synchronized via anchor points

* **Source:** [Hantos 2007]



Principles #1, #2, and #3

- Principle #1: Covered process domains are acquisition and development of software-intensive systems
 - *ULCM[®] might be applicable in other domains as well, but such use was neither pursued nor verified*
- Principle #2: The fundamental building block of life cycle models is an increment
 - *Increment is a conceptual term, refers to the difference between two subsequent releases of the product*
 - Delivering any useful functionality requires the creation of at least one increment of a system
- Principle #3 : All Life Cycle Models are constructs or derivatives of a small number of basic LCM patterns
 - *Since the fundamental building block is an increment, the ULCM[®] definition of all LCM patterns must address their relationship to the creation and sequencing of increments*



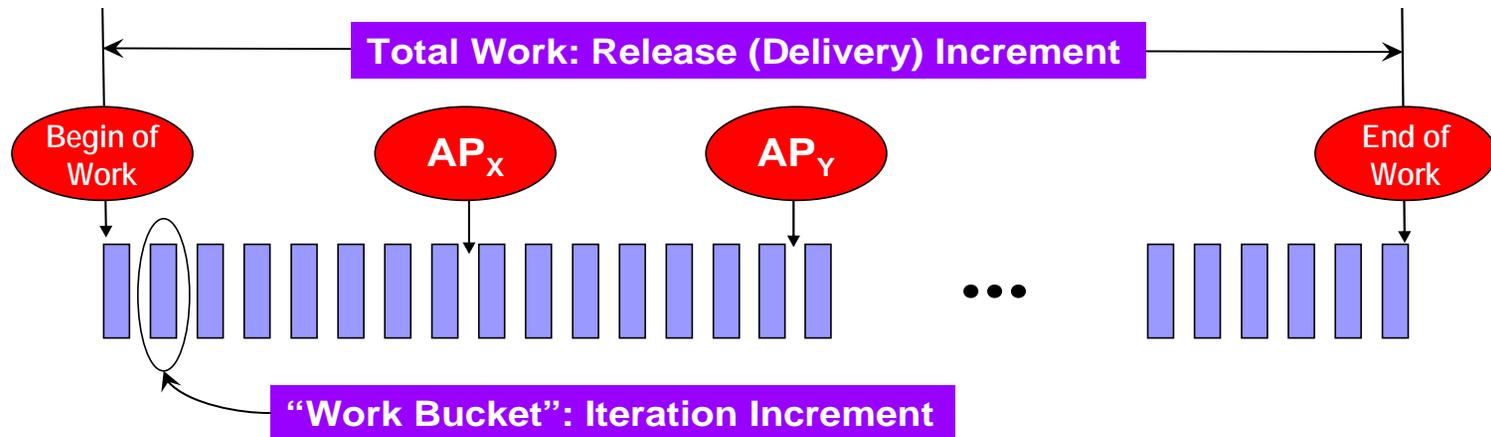
Principles #4 and #5

- Principle #4: Life cycle modeling is synergistic with architecture, architectural concepts, and architecture modeling
 - *Product Architects answer the “What” question*
 - *Process Architects/Project Managers answer the “How” question*
 - *However, both activities are concurrently iterated during the life cycle*
- Principle #5: Proper representation of life cycle models requires multiple views
 - *Based on related experience with architecture modeling, it is clear that having multiple views is always necessary when modeling complex entities*
 - *The question is how many is necessary and sufficient?*
 - Currently ULCM[®] assumes two views of any life cycle model
 - *However, only one of them, the Enactment View, will suffice to demonstrate concerns related to increment coupling and cohesion*



Principle #6

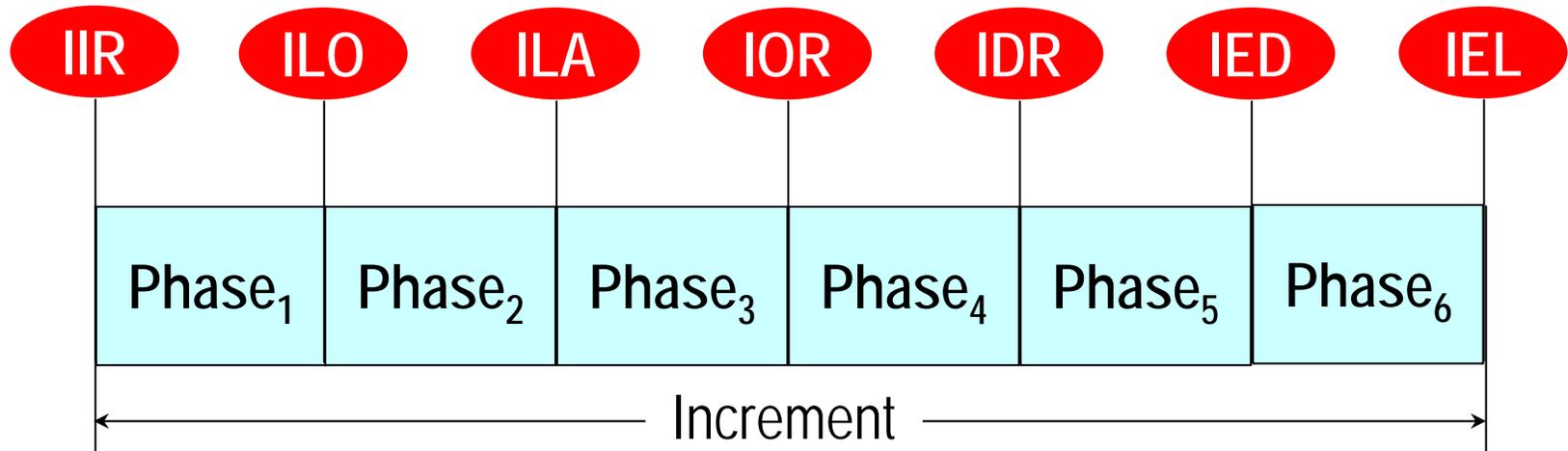
- Principle #6: Concurrent processes are synchronized via Anchor Points
 - *What are Anchor Points (APs)?*
 - Intermediate milestones with specific, focused objectives



- *The idea behind Anchor Points*
 - “Extreme” Planning and Monitoring & Control Approaches
 - *Ad-hoc, “code-and-fix”*: Planning horizon is the next iteration
 - *Waterfall*: Planning horizon is the end of the Increment
 - “Stop, Stabilize, and Regroup” Approach
 - *Iterative with APs*: Planning horizon is the next Anchor Point



ULCM[®] Enactment View of an Increment



Legend:

IIR – Increment Inception Readiness
ILO – Increment Life Cycle Objectives
ILA – Increment Life Cycle Architecture
IOR – Increment Operational Readiness
IDR – Increment Delivery Readiness
IED – Increment End-Of-Life Decision
IEL – Increment End-Of-Life

- In ULCM[®], life cycle phases of an increment are intentionally not named
 - *Specifying both phase content and anchor points is redundant*
 - *Phase content stays flexible; phase activities are not pre-determined*
 - *Focus is on achieving anchor point objectives*



Product-related AP Objectives During Development

- **IIR** – Increment Inception Readiness
 - *Its sole purpose is to mark the beginning of an increment*
- **ILO** – Increment Life Cycle Objectives
 - *Definition of operational concept, scope, and top-level requirements*
 - *Architectural and design options*
- **ILA** – Increment Life Cycle Architecture
 - *Refinement of operational concept, scope, and top-level requirements*
 - *Resolution of ILO option-explorations, commitment to a feasible architecture and technology solutions*
- **IOR** – Increment Operational Readiness
 - *Operation and quality is demonstrated in development environment*
- **IDR** – Increment Delivery Readiness
 - *The work product created in this phase is ready for*
 - Delivery to the end-user/customer, or
 - Higher-level integration and test



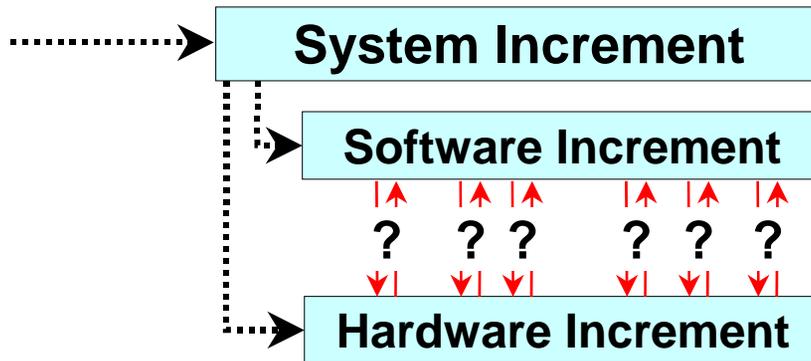
Challenges of Concurrent Engineering

- The usual HW/SW dialog
 - *Traditional SW Position: Give me the working hardware, and leave me alone!*
 - *Traditional HW Position: Here are the specs, see you at final integration. Now leave me alone!*
 - *What Really Takes Place: HW is frequently changing during design. SW people are frustrated and inefficient. SW always ends up being the bottleneck*
- Similar situation in case of concurrently developed software components
- Challenges, challenges ...
 - *The Project Manager's Challenge:*
 - Managing (estimating, planning, monitoring, and controlling) concurrent engineering processes
 - *The Process Architect's Challenge:*
 - Dealing with life cycle modeling complexity
 - *Concurrent engineering of hardware and software*
 - *Iterative/incremental processes*

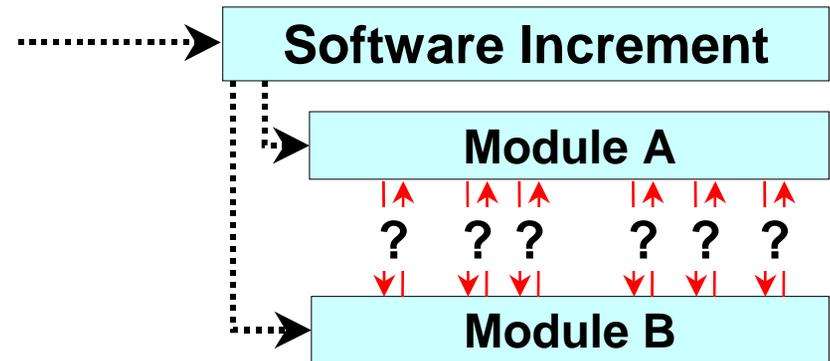


Anchoring Concurrent Engineering Processes in ULCM[®]

Hardware-Software Streams



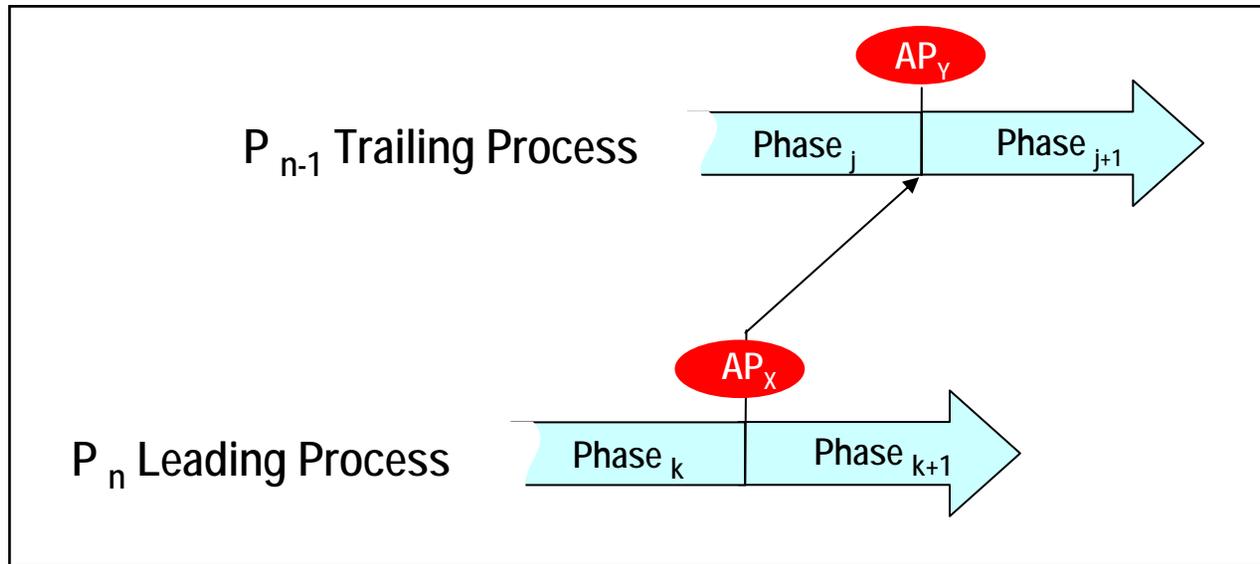
Software-Software Streams



- Specific Challenges Addressed
 - Design of interfaces and the tuning of Technical Performance Measures (TPMs) related to dependent, concurrently developed components
 - For concurrent engineering process streams, the determination of
 - Optimal number of interactions between concurrent streams, and
 - The optimal place of interactions in the life cycle (solved by using APs)



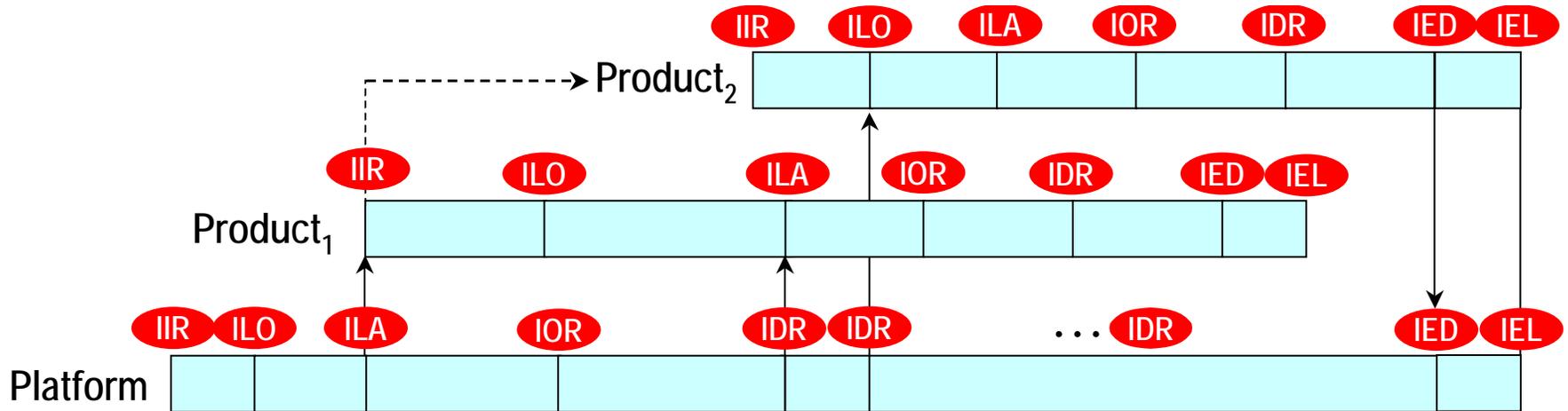
Synchronization Via Anchor Points



- How Anchor Points are used
 - *Concurrent process streams should not be arbitrarily shifted or overlapped*
 - *Connection is only planned at Anchor Points*
- Stakeholders of the process streams collaborate at Anchor Points
 - P_{n-1} stakeholders rely on P_n stakeholder deliveries at AP_x to satisfy AP_y objectives



Example Use of Anchor Points for a Product Line



- What is a product line?
 - A *product line* represents a product family, a set of related systems that are built from and leveraging off a common set of core assets*
- Product line challenges
 - *Technical considerations* – selecting/distributing product features
 - *Business constraints* – balancing cost and Time-to-Market
 - *Development strategy challenges* – determination of architectural structuring, development and production order
- LCM Challenge: Manipulating a complex, aggregate life cycle model

* **These core assets are also called the elements of the Product Line Platform**



DSM (Design Structure Matrix)

- The DSM method is widely used to design and optimize complex systems in various domains
 - *DSM describes the relationships between architectural elements of a system in a concise format*
 - *In each cell we might have simply a marker (like a circle) or, in more complex cases some kind of indicator characterizing the relationship between system design elements*
 - *A wide range of tools are available to manipulate DSM [Browning 2001]*
- Basic DSM Examples:

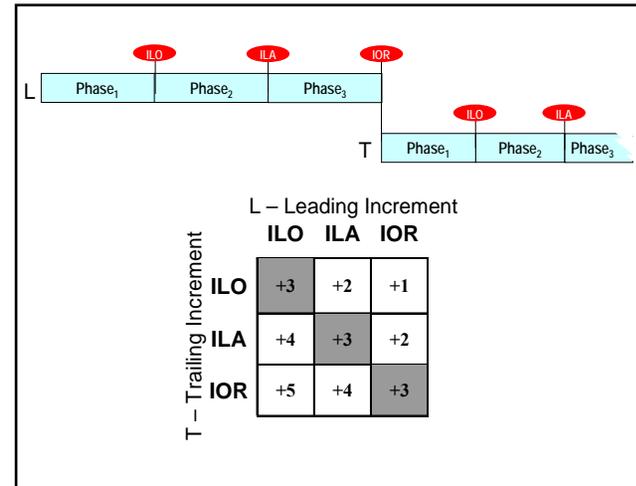
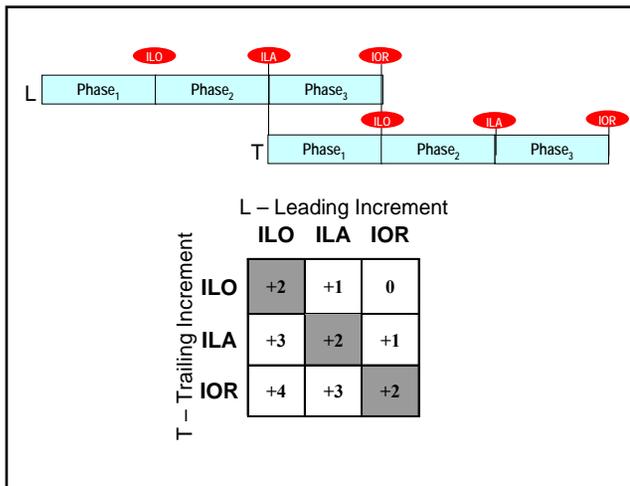
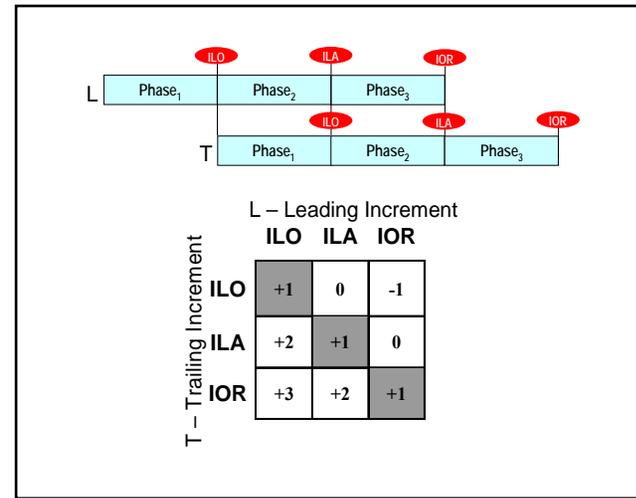
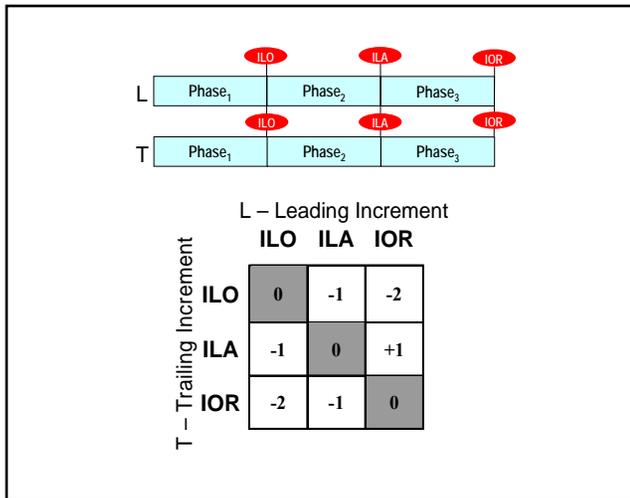
	D1	D2	D3	D4
D1				●
D2	●			
D3				
D4		●		

	D1	D2	D3	D4
D1				m_{14}
D2	m_{21}			
D3				
D4		m_{42}		

Legend: *D1 ... D4* – System Design Elements;
m_{ij} – Relationship between *D_i* and *D_j* Elements



Mapping Anchor Points to DSM



Concurrent Increment Coupling Metric (CICM)

- Coupling is a measure of strength of interconnection
 - *Uncoupled modules are independent*
- High or Low coupling is not “good” or “bad”
 - *Various pro’s and con’s are associated with different coupling levels*
 - *Author’s hypothesis is that for any concurrent engineering situation an optimal coupling exists*
- DSM-based CICM definition

$$\mathbf{CICM} = f(m_{11}, m_{12}, \dots, m_{ij}, \dots, m_{nn})$$

- For the shown DSM matrices a simple CICM definition

$$\mathbf{CICM} = 4 - m_{nn}$$

where m_{nn} is the value from the diagonal of the matrix



CICM Values for the DSM Examples

m_{nn}	CICM	
	Numeric Value	Ordinal Rating
0	4	Very High
1	3	High
2	2	Medium
3	1	Low
	0	No coupling (Independent)



The Relationship Between CICM and Schedule/Cost Risk

- Definitions
 - *Schedule risk in this context is risk to complete the project in the estimated timeframe due to unexpected rework*
 - *Cost risk in this context is risk to complete the project within estimated cost due to unexpected rework*
- A main source of these risks is architecture volatility stemming from concurrent engineering
 - *However, the relationship between concurrent increment process stream coupling and architecture volatility is not straightforward*
 - *For example, the classic “Iron Triangle” of Cost-Schedule-Performance does not apply anymore*
 - Depending on the chosen concurrency configuration of the increments, drastically different schedules are expected even though performance and cost are supposed to stay the same



Discussion Based on the Examples

- Very High Coupling (CICM=4)

- *Positive:*

- Increment phases overlap, all APs are aligned
 - The architecture of both increments is basically planned together, at the same time
 - *Being able to change both architectures provides flexibility that is considered positive*
 - This configuration promises the shortest schedule

- *Caveats:*

- Both architectures are volatile
 - No “hardening” provided for the leading increment
 - No learning from the development of the leading increment
 - There will not be any opportunity for early detection of defects in the leading increment
 - *This configuration results in the most costly rework*



Discussion Based on the Examples (Cont.)

- High Coupling (CICM=3)

- *Positive:*

- Architectural options for the leading increment are known when the design of the trailing increment starts
 - Actual architecture of the leading increment is known when the determination of the trailing increment architecture starts
 - The actual code of the leading increment is available when the implementation of the trailing increment starts

- *Caveat:*

- Increased cost of rework when correcting any problems with the leading increment that are discovered during the design of the trailing increment



Discussion Based on the Examples (Cont.)

- Medium Coupling (CICM=2)
 - *Positive:*
 - Actual architecture of the leading increment is known when the work on the trailing increment starts
 - The actual code of the leading increment is available when the architectural design of the trailing increment starts
 - *Caveats:*
 - Increased difficulty in correcting any problems with the leading increment that are discovered during the design of the trailing increment due to the fact that the leading increment's architecture has been determined
 - Final integration is further removed; correcting any problems with the leading increment that are discovered during final integration is becoming increasingly more expensive



Discussion Based on the Examples (Cont.)

- Low Coupling (CICM=1)
 - *Positive:*
 - The actual code of the leading increment is available when the planning of the trailing increment starts
 - Leading increment's code is considered sufficiently tested
 - *Caveats:*
 - High level of difficulty in correcting any problems with the leading increment that are discovered during the development of the trailing increment due to the fact that the leading increment has already been coded and tested
 - Final integration is further removed; Correcting any problems with the leading increment that are discovered during final integration is becoming very expensive



Next Steps – Direction of Future Research

- Extend CICM to cover more realistic increment positioning situations
 - *The shift involves more than one phase*
 - *Phase-lengths are not equivalent*
- Define LCPC (Life Cycle Plan Cohesion) Metric
 - *Cohesion is a measure of how tightly bound or related the concurrent increments are to one another*
 - *Coupling is one key factor, but not the only factor*
 - It seems to be plausible that tightly coupled increments create a life cycle plan with high cohesion
 - *However, the relationship needs to be researched and quantified.*

$$\text{LCPC} = f(\text{CICM})$$

- Develop quantitative evaluation guidance for LCPC
 - *Quantify metrics*
 - *Develop a methodology that allows the comprehensive evaluation of schedule, rework, and quality dimensions of different life cycle plans*



Summary

- A promising Aerospace research platform, ULCM[®] has been used to model concurrent engineering process streams of software-intensive system development
- DSM has been introduced to facilitate the easy manipulation of ULCM[®] models of concurrently engineered complex systems
- Two new metrics, CICM and LCPC has been proposed to characterize increment coupling and cohesion in complex life cycle models



Acronyms

AP	Anchor Point
CICM	Concurrent Increment Coupling Metric
DSM	Design Structure Matrix
IDR	Increment Delivery Readiness
IED	Increment End-of-Life Decision
IEL	Increment End-of-Life
IIR	Increment Inception Readiness
ILA	Increment Life Cycle Architecture
ILO	Increment Life Cycle Objectives
IOR	Increment Operational Readiness
IR&D	Independent Research & Development
LCM	Life Cycle Modeling
LCPC	Life Cycle Plan Cohesion
MOIE	Mission-Oriented Investigation and Experimentation
TPM	Technical Performance Measure
ULCM	Unified Life Cycle Modeling



References

- [**Browning 2001**] Browning, R. T., “Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions,” IEEE Transactions on Engineering Management, 48(3):292-306
- [**Hantos 2007**] Hantos, P., “Unified Life Cycle Modeling Tutorial, Version 1.0,” Aerospace Report No. TOR-2007(8550)-6966, August 31, 2007



Contact Information

Peter Hantos

The Aerospace Corporation

P.O. Box 92957-M1/112

Los Angeles, CA 90009-2957

Phone: (310) 336-1802

Email: peter.hantos@aero.org



All trademarks, service marks, and trade names are the property of their respective owners

