

# Announcement

For the “Quality” report, just submit inspection reports for the SSRD and the SSAD. Since we asked you to do them in Excel, please submit them in XLS. Since we asked for a confusing set of things on very short notice, please accept our apologies and turn these in on Wed Nov 7.

For the LCA milestone, we will ask for another set of inspection reports covering the new versions of the SSRD and the SSAD based on the PBR approach. The SSAD inspections should focus not just on internal SSAD defects, but also on traceability of the SSAD elements to OCD and SSRD elements. This should provide you with much of what you need to complete your FRD Section 2.2's, most of which were pretty scanty in the draft LCO packages.

Also for the LCA package, we will ask for a completed COQUALMO form SPD-8, covering the results of both your LCO and LCA inspections. Thus, we're not asking for a separate LCO version of the SPD-8 now, but we'll need the LCO data then.

# MBASE LCO Wrap-Up

CS577A

2001

Congratulations...

You are engaged to your project!

**“Software Engineering”**: The disciplines which distinguish the coding of a computer program from the development of a software product.

	Requirements, Architecture	Design, Code	Implement, Maintain
Computer Science		CS Focus	
User Applications			
Economics			
People			

- **Accommodate new tools and techniques**
  - **Web browsers, GUI prototypers, WinWin, Spiral processes**
- **Integrate all these considerations**
  - **Via integrated models (MBASE)**

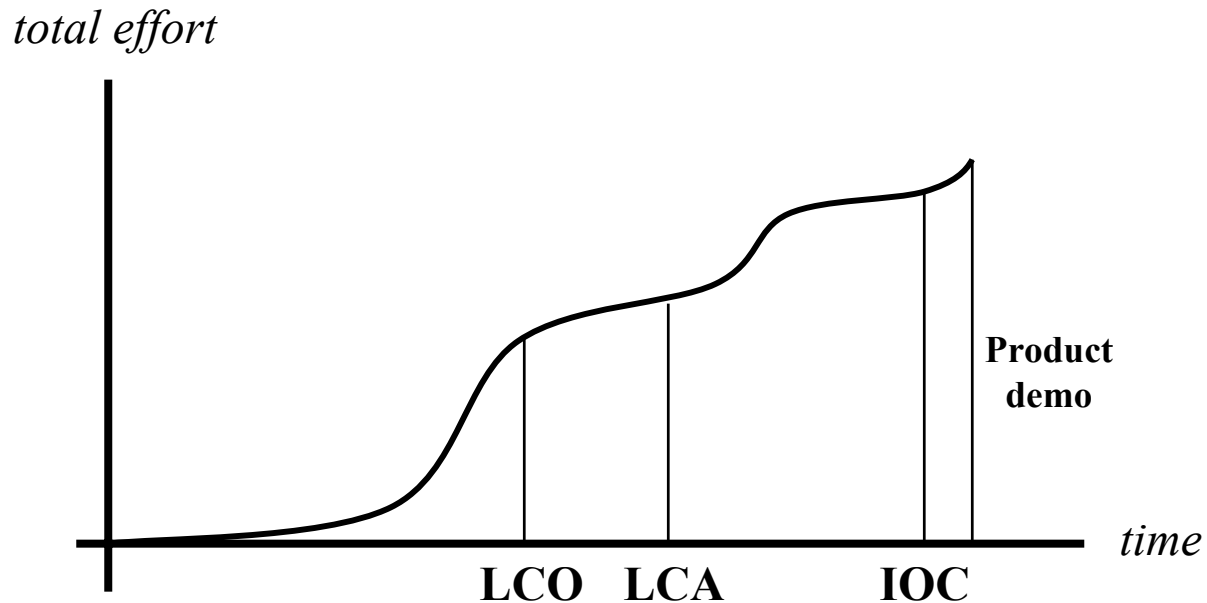
# Major Class Project Milestones

September 10	All teams formed
September 28	Prototype
September 28	WinWin Negotiation Results
October 22	LCO Drafts on Web Site
October 29-Nov 02	LCO Architecture Review Boards
November 05	LCO Package Due
November 28	LCA Drafts on Web Site
December 03-07	LCA Architecture Review Boards
December 10	LCA Package Due
December 12	Individual Critiques Due

# Class Project Artifacts

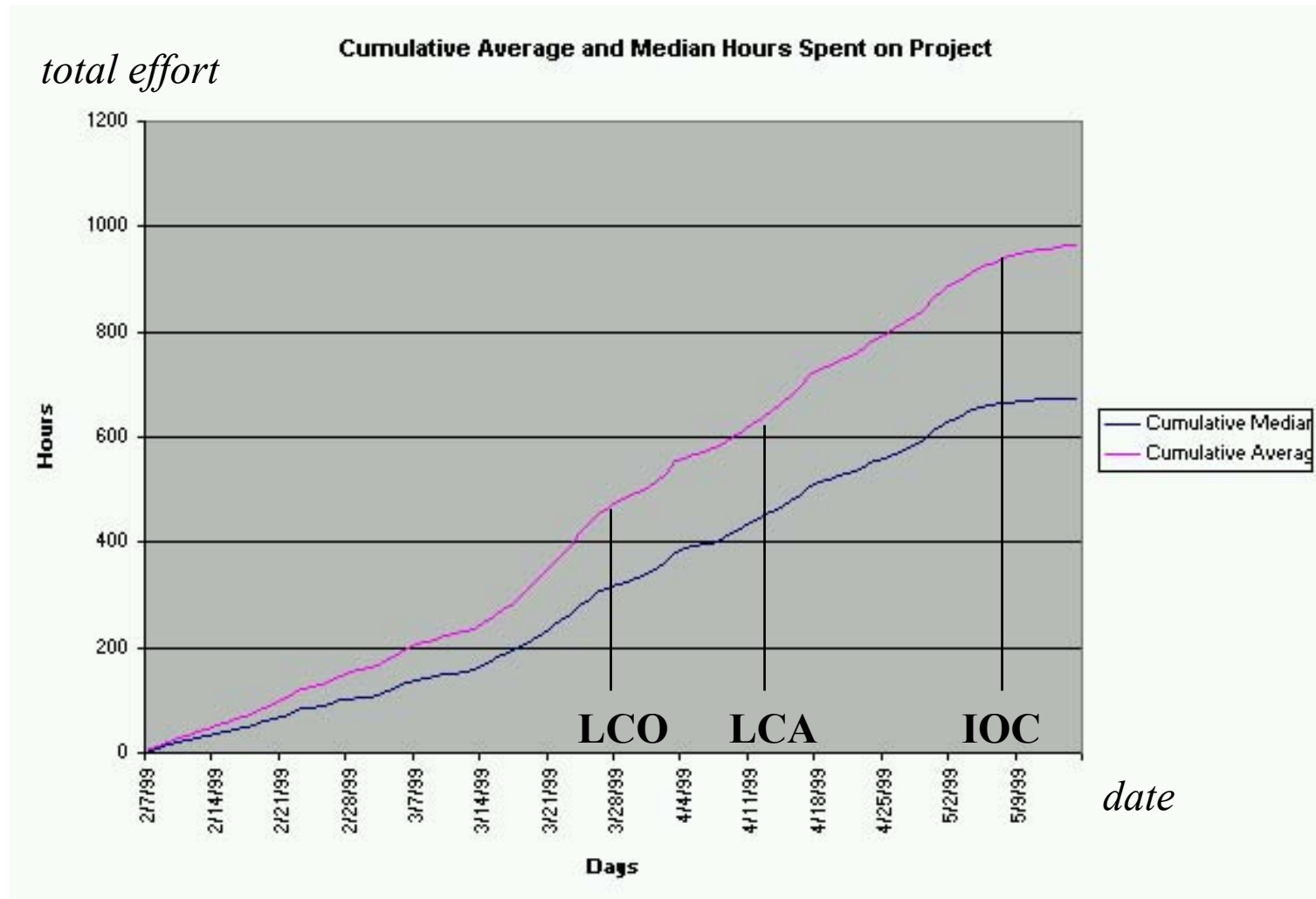
- 1. An Operational Concept Definition**
  - 2. A Prototype of Key System Features**
  - 3 A System Requirements Definition**
  - 4. A System and Software Architecture Definition**
  - 5. A Life Cycle Plan**
  - 6. A Feasibility Rationale, assuring the consistency and feasibility of items 1-5**  
*For 577b projects*
- 
- 7. A Detailed Construction Plan consistent with 1-6**
  - 8. The Test Results and Quality Assurance for 1-7**
  - 9. An Initial Operational Capability of the system derived from 1-8**

# Effort Graph 1-semester Columbia University Course



# Effort Graph CS3156 S99

<http://www.columbia.edu/~cl363/research/images2.html>



LCO – where are you at?

Success criteria:

**Identify at least one feasible  
architecture**

*(subject to risk considerations)*

# MBASE Milestone Elements

Milestone Element	Life Cycle Objectives (LCO)	Life Cycle Architecture (LCA)
<b>Definition of Operational Concept</b>	<ul style="list-style-type: none"> <li>• Top-level system objectives and scope               <ul style="list-style-type: none"> <li>- Shared vision of expected initiatives and outcomes [COTS strategy for reuse and legacy elements]</li> <li>- System boundary [major COTS component boundaries within system]</li> <li>- Environment parameters and assumptions [CBS success factors]</li> <li>- Evolution parameters [major COTS vendor product availability, upgrade cycles]</li> </ul> </li> <li>• Operational concept               <ul style="list-style-type: none"> <li>- Operations and maintenance scenarios and parameters [COTS maintenance]</li> <li>- Organizational life-cycle responsibilities (stakeholders) [major COTS vendors]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• Elaboration of system objectives and scope by increment</li> <li>• Elaboration of operational concept by increment [COTS operation and use summary, vendor support strategies, COTS adoption impact on organization]</li> </ul>
<b>System Prototype(s)</b>	<ul style="list-style-type: none"> <li>• Exercise key usage scenarios [demonstrate key COTS capabilities and validate interfaces, scope use and value of major COTS products]</li> <li>• Resolve critical risks [COTS selection and evaluation plan]</li> </ul>	<ul style="list-style-type: none"> <li>• Exercise range of usage scenarios [complete COTS evaluations, initial COTS configuration and training, initial COTS tailoring]</li> <li>• Resolve major outstanding risks [resolve COTS integration issues and level of use]</li> </ul>
<b>Definition of System Requirements</b>	<ul style="list-style-type: none"> <li>• Top-level functions, interfaces, quality attribute levels, including:               <ul style="list-style-type: none"> <li>- Growth vectors</li> <li>- Priorities</li> <li>- Legacy systems and environments</li> <li>- [Establish key COTS evaluation attributes and screening parameters]</li> </ul> </li> <li>• Stakeholders' concurrence on essentials [mandated COTS packages, suppliers and vendors, identify negotiable and non-negotiable COTS requirements, COTS evaluation win-conditions and constraints]</li> </ul>	<ul style="list-style-type: none"> <li>• Elaboration of functions, interfaces, quality attributes by increment               <ul style="list-style-type: none"> <li>- Identification of TBDs (to-be-determined items) [mapping of COTS capabilities to requirements]</li> </ul> </li> <li>• Stakeholders' concurrence on their priority concerns [concurrence on COTS imposed requirements and 90% re-worked requirements tradeoffs]</li> </ul>

# MBASE Milestone Elements (2)

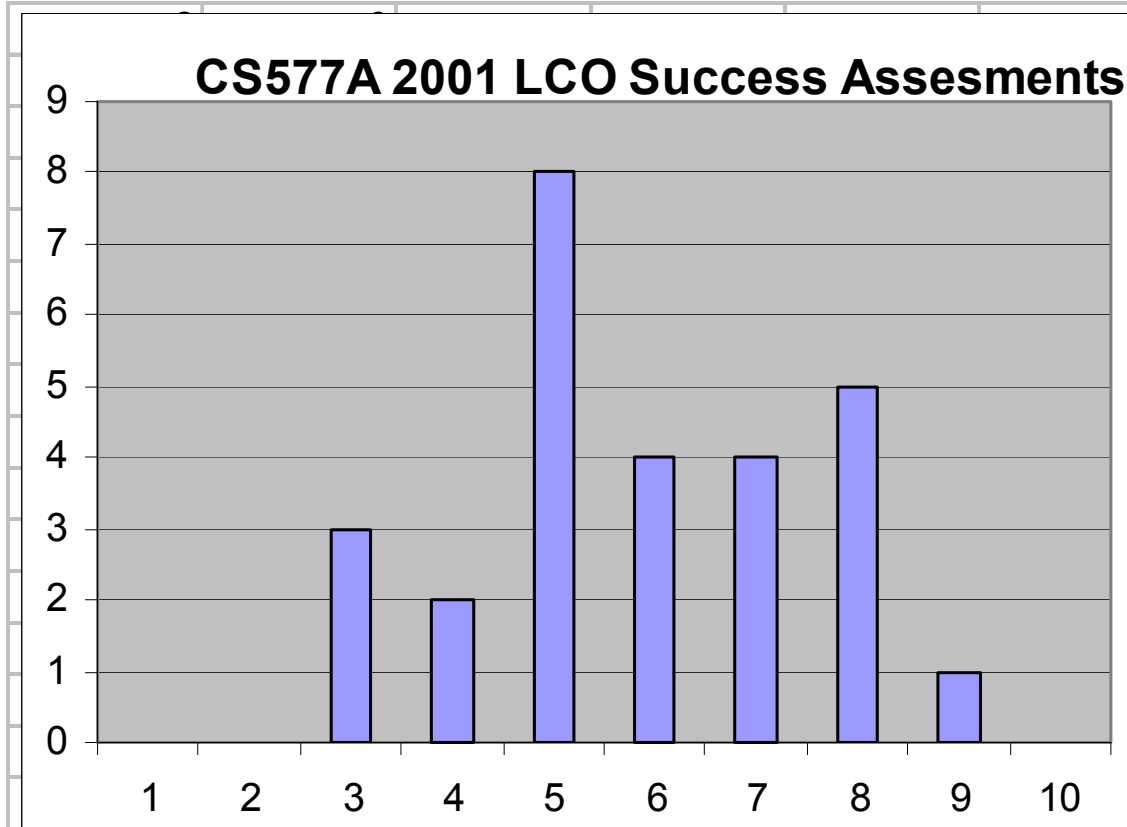
Milestone Element	Life Cycle Objectives (LCO)	Life Cycle Architecture (LCA)
<b>Definition of System and Software Architecture</b>	<ul style="list-style-type: none"> <li>• Top-level definition of at least one feasible architecture               <ul style="list-style-type: none"> <li>- Physical and logical elements and relationships, including top-level domain factoring and identification of possible design patterns</li> <li>- Choices of COTS and reusable software elements [mapping of critical system components to COTS products]</li> </ul> </li> <li>• Identification of infeasible architecture options</li> </ul>	<ul style="list-style-type: none"> <li>• Choice of architecture and elaboration by increment               <ul style="list-style-type: none"> <li>- Physical and logical components, connectors, configurations, constraints</li> <li>- COTS, reuse choices [CBS design (mapping of components and system factors to COTS), COTS package configurations]</li> <li>- Domain-architecture and architectural style choices [design for COTS glue code and wrappers]</li> </ul> </li> <li>• Architecture evolution parameters</li> </ul>
<b>Definition of Life-Cycle Plan</b>	<ul style="list-style-type: none"> <li>• Identification of life-cycle stakeholders               <ul style="list-style-type: none"> <li>- Users, customers, developers, maintainers, interpreters, general public, others [COTS suppliers and vendors]</li> </ul> </li> <li>• Identification of life-cycle process model               <ul style="list-style-type: none"> <li>- Top-level stages, increments [COTS deployment – prototyping, evaluation, schedule, risks impact of schedule and cost, stakeholders, license negotiations]</li> </ul> </li> <li>• Top-level W W W W W H H * by stage [COTS product lifecycles (releases, delivery dates, costs, training needs)]</li> </ul>	<ul style="list-style-type: none"> <li>• Elaboration of W W W W W H H * for Initial Operational Capability (IOC)               <ul style="list-style-type: none"> <li>- Partial elaboration, identification of key TBDs for later increments</li> <li>- [COTS upgrades, patches]</li> <li>- [License management (non-standard provisions)]</li> <li>- [COTS users, maintainer, and developer skill set attributes]</li> <li>- [COTS training plan]</li> <li>- [COTS risks realized; integration, cultural, economic]</li> </ul> </li> </ul>
<b>Feasibility Rationale</b>	<ul style="list-style-type: none"> <li>• Risk assessment and mitigation plans [COTS vendor issues – product availability, stability, costs]</li> <li>• Assurance of consistency among elements above               <ul style="list-style-type: none"> <li>- Via analysis, measurement, prototyping, simulation, etc.</li> <li>- Business case analysis for requirements, feasible architectures [buy versus build rationale, justification of CBS strategy, CBS tradeoffs]</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• Assurance of consistency among elements above [feasibility of COTS vendor relationships, COTS maintenance costs, organization readiness to implement COTS]</li> <li>• All major risks resolved or covered by risk management plan [completed license contracts, COTS product alternatives and marketplace issues, COTS organization culture issues, vendor commitments]</li> </ul>

# USC and Columbia Projects

Metric	USC 1996- 97	USC 1997- 98	USC 1998- 99	USC 1999- 00	Columbia U-grad. S99	Columbia Grad. S99	Columbia U-grad. F99	Columbia Grad. F99
<i>Fall Semester:</i>								
<b>LCA Package</b>								
<b>Teams</b>	15	16	19	22	20	13	10	7
<b>Students</b>	86	80	102	100	107	59	44	26
<b>Applications</b>	12	15	17	22	10	10	10	7
<b>Teams failing LCO</b>	4	4	1	1	10	6	5	1
review								
<b>Teams failing LCA</b>	0	0	0	0	0	1	1	0
review								
<b>Pages, LCO package</b>	160	103	114	-	124	116	107	95
<b>Pages, LCA package</b>	230	154	167	-	142	142	140	109
<b>Client</b>								
<b>Evaluation</b>	4.46	4.67	4.74	4.48	-	-	-	-
<b>(1-5, 5 best)</b>								
<b>Spring Semester: IOC</b>								
<b>Package</b>					Remained the same since projects were only one semester long			
<b>Teams</b>	6	5	6	8				
<b>Students</b>	28	23	28	35				
<b>Applications</b>	8	5	6	8				
<b>Teams failing IOC</b>	0	0	0	1	0	0	1	0
acceptance review								
<b>Applications satisfying clients</b>	5	5	6	7	20*	12*	10*	7*
<b>(* teams)</b>								
<b>Applications not overtaken by</b>	6	4	4	4	10	9	10	6
<b>events</b>								
<b>Applications continued</b>	3	3	4	4	2	3	1	2
<b>Applications used</b>	1	3	3	5	10	5	7	3
<b>Client evaluation</b>	-	4.15	4.3	4.75	4.44	4.21	3.9	4.38

# LCO ARB Assessments

	Mean	std
OCD	6.60	1.78
proto	6.89	1.80
SSRD	6.39	1.89
SSAD	5.33	1.59
LCP	4.31	2.05
FRD	5.31	2.19
LCO Success	5.72	1.63



Teams failing LCO  
success at ARB: 5

Teams with unknown  
ARB success: 2

# LCO ARB Best Practice Examples (with respect to project)

- Team 15 (DART)
  - Excellent OCD, Prototype, SSRD, and FRD
  - Non-COTS intensive
- Team 13 (PLASMA)
  - Good OCD, SSRD, FRD (particularly COTS screening mx)
  - COTS-based system
- Team 3 (MM Equipt. Sched.)
  - Solid prototype, SSAD, FRD
  - Uses COTS
- Team 1 (Dental Lib. New Books)
  - Excellent OCD, prototype, and SSRD

# OCD Feedback

- Results chains often missing initiatives outside of building proposed system
- Org. goals, results chains, benefits realized sometimes inconsistent
  - System capabilities sometimes presented as organizational goals
  - Results chains overfocused on software
  - Benefits realized and results chain ultimate outcomes should specifically address organizational goals
- Add software maintainer as key stakeholder
- Initial prototyping results should be part of the OCD 5, not a separate document

# Prototype Feedback

- Continue to explore usage issues with clients via prototype extension, end-user exercising
  - Focus on risk items, not easy-to-do things (password processing)
- For some projects, explore COTS level-of-service issues via prototyping, benchmarking, exercise

# SSRD Feedback

- Superfluous requirements
- Specification of requirements that are not implementable and testable
- MR, MRS, MARS integrations incorrect
  - Project refers to goals, L.O.S. refer to capabilities
- Simplifier and Complicator problems
- Validate all requirements with client
  - Levels-of-service, programming language
- Need thorough interface specs to critical interoperating systems
- Some level-of-service requirements overly vague
  - Ease of use, availability, “good” performance
- Work out evolution requirements
  - And use them to constrain architecture

# SSAD Feedback

- Content of UML views often confused
  - Enterprise, component, behavior, entity, class
  - More emphasis on use cases
- Some OCD-SSRD-SSAD traceability problems
- Need thorough interface components to critical interoperating systems
- SSAD Behavior model is different than Enterprise Class Model
- Component names do not reflect defining quality
- Incorrect and unclear notation for SSAD Design views
  - Especially physical view

# LCP Feedback

- Missing project specific details
  - Avoid general guideline stuff
- Be concise about deliverables (do not expand LCO, LCA package elements, reference guidelines if needed)
- Process choice arbitrary and not planned
- Be much more specific about 577b plans
  - Roles, skills, risks, tasks, milestones, increments, test and transition planning, preparation, and execution
  - CTS planning must be included now
- Risk mitigation and contingencies not planned
- More thorough risk management plans
- Provide rationale for COCOMO ratings
  - Use COCOTS also for COTS-intensive systems

# FRD Feedback

- Focus on 577b risks: skills, schedule, COTS
  - Big risks (timely or adequate COTS) should have fallbacks
- Process rationale should identify specific core capability
  - And propagate it to LCP milestone
- Quantify business-case cost and value via representative scenarios
- Operations and maintenance effort by people already on USC payroll isn't free
- FRD Business case
  - Cost estimates need to consider non-\$ costs such as effort, utils. etc.
  - Value estimates need to tie to OCD Org. Goals, Results Chain, Benefits Realized
  - ROI summaries need to consider initial costs, ongoing costs, and future costs as compared to initial value, ongoing value, and future value
  - Include COCOMO cost driver rationale when applicable

# General LCO Package Feedback

- Confusion on applying MBASE guidelines to COTS
  - COTS-based, COTS-driven, Uses-COTS
- Use of old version of MBASE guidelines and templates
- More consistency across documents
  - System definition across OCP, SSRD
  - Risks in LCP, FRD
  - Levels-of-service in OCD, SSRD, FRD
- Avoid duplicating text
  - Better to cross-reference; makes update easier
- Use client-oriented, not abstract terms
  - Option 1, 2, 3; data items a, b, c\
- Some projects need to start consideration of CTS plans now

# Easy WinWin Screening Matrix

	<i>Importance</i>	<i>Unix WinWin</i>	<i>WinWin Lite</i>	<i>GTE IWS</i>	<i>Gsys.com</i>
<b>W-I-O-A model</b>	***	***	***	***	***
<b>Ease of use</b>	***		**	**	***
<b>Low cost</b>	***	**	**		***
<b>Commercial Support</b>	***		*	*	***
<b>Windows platform</b>	***		***		***
<b>Unix WinWin API</b>	*	***	***	*	*
<b>Win C brainstorming</b>	***			*	***

# Initial Cognitive Demands Analysis for MBASE Inception Phase

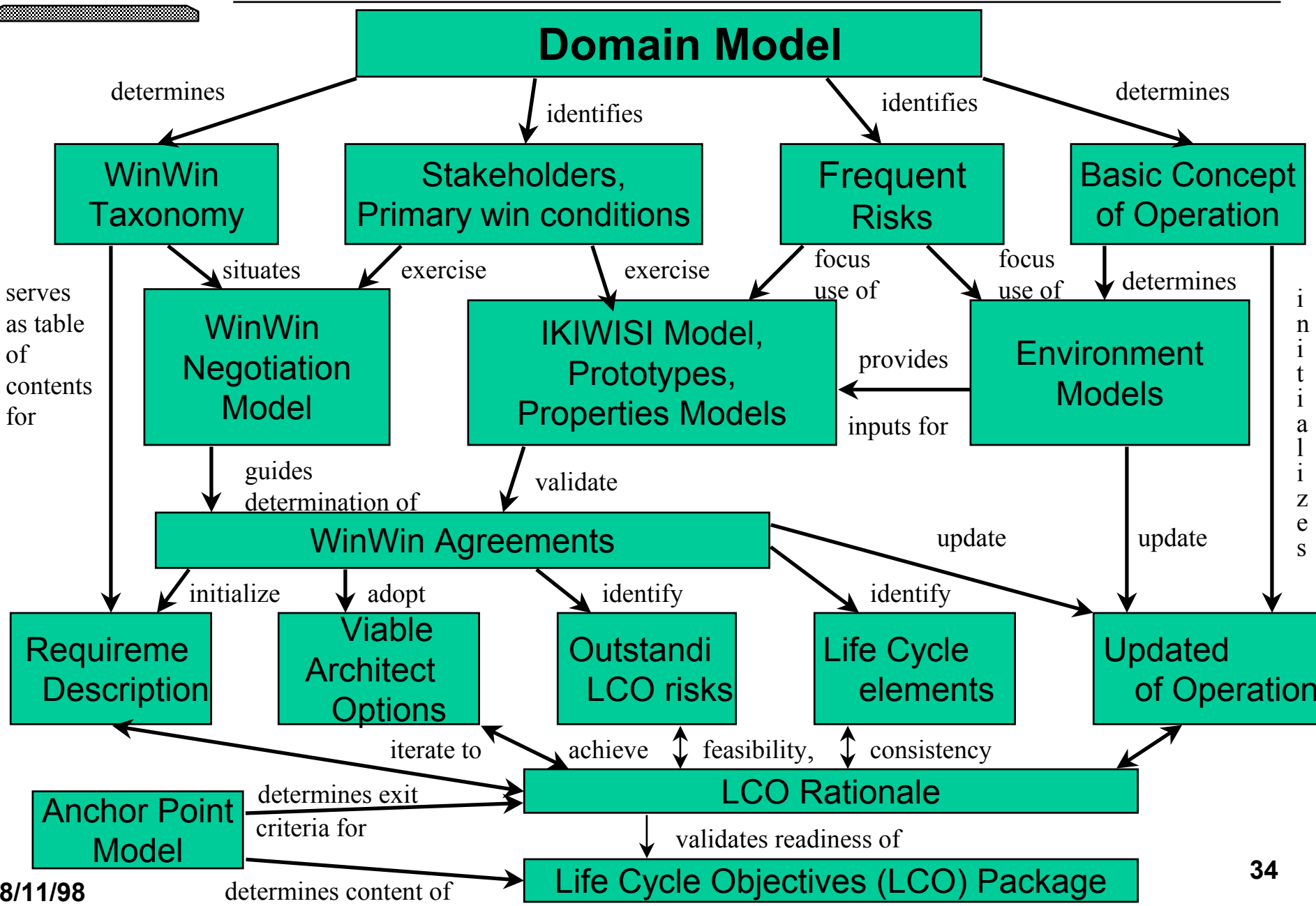
Technology Task	Artifact Guidelines	Domain Models	WinWin Tool	UML/Rose Tool	COCOMO Tool	Risk, SC Checklist	Other
Team Formation	Understand required artifact skills LRC	Understand required domain skills CH				Understanding assess team personnel risks LR	Collaborate among team; team formation negot. team rules LR
Select, tailor domain model	Understand OpCon guidelines LRC	Select, adapt domain model LRCH		Understand tool op. Develop top-level use cases LRCTH			Collaborate w/ client on domain model tailoring LR
Negotiate stakeholder win-win agreements	Understand OpCon, Rqts., Plan G/L's LRC	Apply domain taxonomy as checklist LRC	Understand tool oper., Devel., negot., WW artifacts LRCTH		Understand tool oper. Devel. negot. COCOMO est's. LRCTH	Manage stakeholder expectations LRH	Collab. among team members & client on winwin agreements LRCT
Prototype key product features						Formulate risks needing proto- Types LR	Understand, select proto. tools Develop proto's. Iterate with clients LRCT
Develop LCO package artifacts	Apply artifact guidelines LRC	Iterate adaptation of domain model LRCH	Use, iterate WinWin agreements LRC	Develop top- level OO artifacts LRCTH	Perform cost/sched. tradeoff analyses LRC	Develop, resolve list of top risks LRC	Collab. among team, client to balance LCO package LRC

V - videos, L - lectures, G/L - guideline, WW - winwin, S - simulations, R - readings, H - homework, SH - stakeholder, D - decision aids,

C - case studies, T - tutorial, S/C - simplifier/comp.



# MBASE Model Integration: LCO Stage



# The Road Ahead: LCA

Success criteria:

**Commit to one feasible architecture**

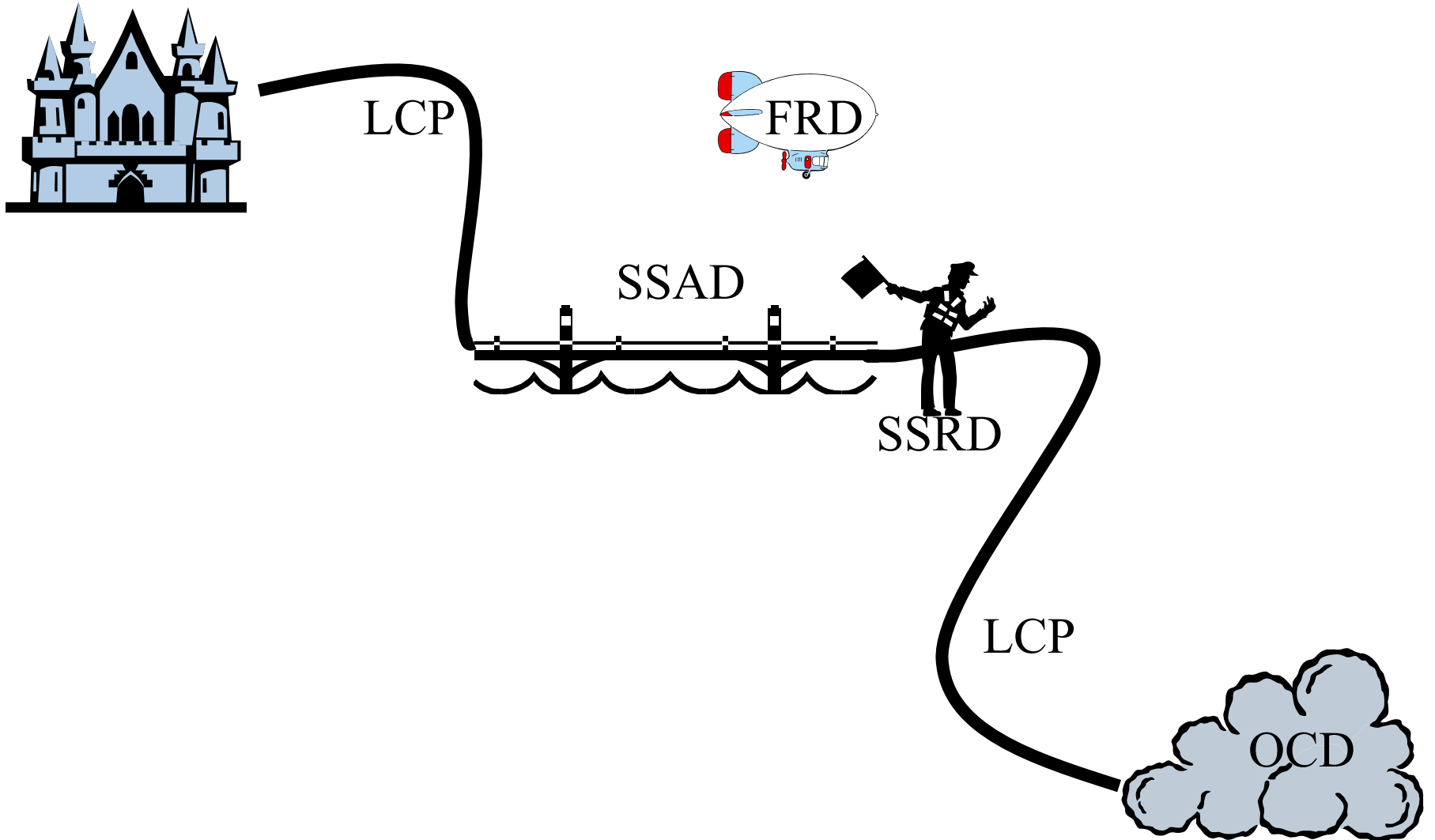
# Pre-wedding plans

- What do you need to do between LCO and LCA?
  - Design
    - Resolve all significant architectural issues
    - Advanced prototyping
  - Refine
    - Resolve model clashes
    - Add significant details
    - Remove non-significant details - no “fluff”
    - Schedule, roles, commitments, effort estimates, etc.
  - Justifine! (Justify)
    - Assure architecture is faithful to concept
    - Assure value of system vs. stakeholder investment
    - Reduce risk exposure

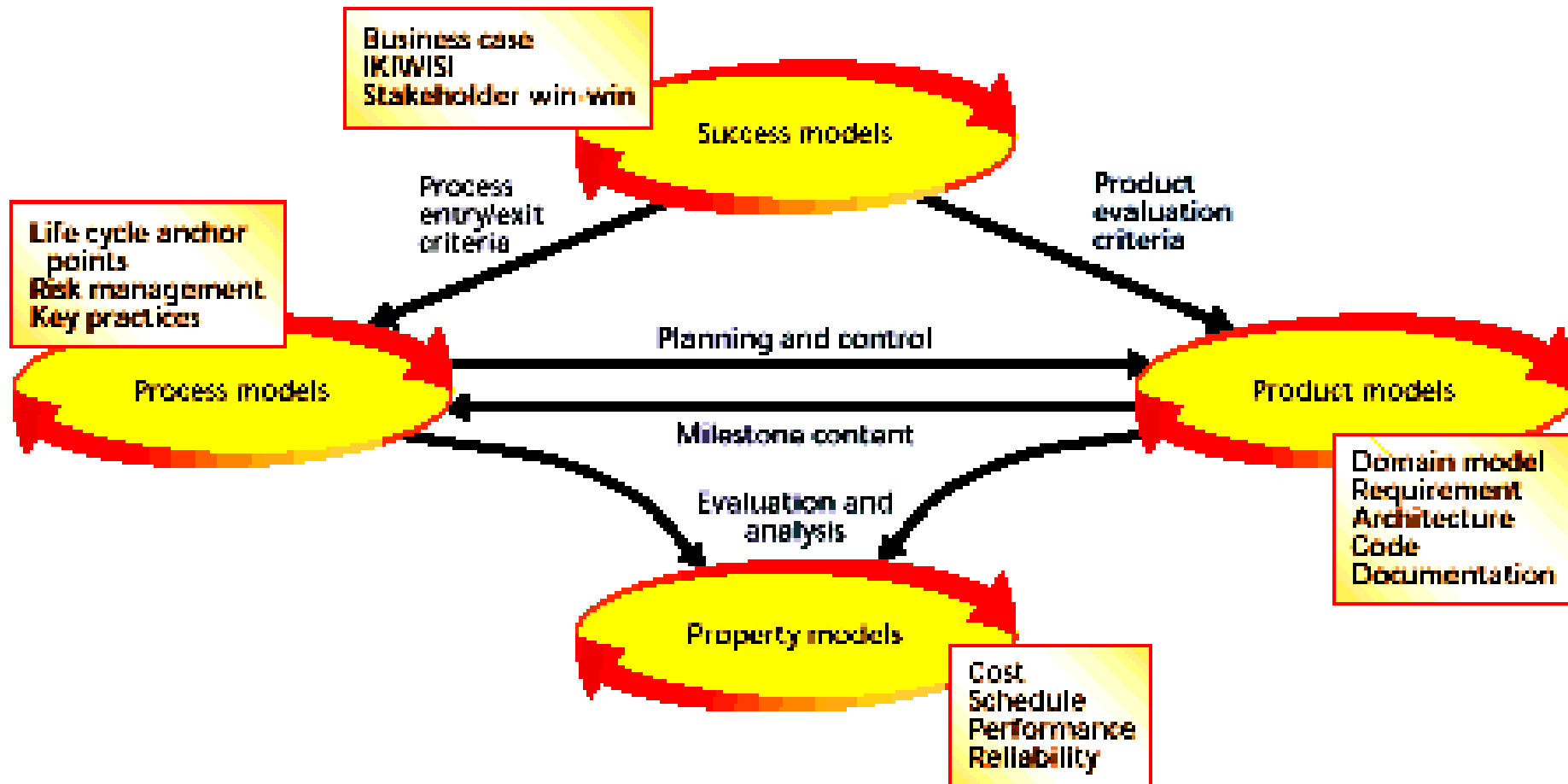
# What does this involve?

- Several iterations through MBASE models
  - Model integration and integrity paramount
- Refine WinWin negotiations
  - Close out old issues
  - Cover all win conditions
  - Identify and deal with new win conditions
- Writing code
  - Advanced prototyping to resolve risky architectural issues
  - Head start on implementing critical requirements (for assurance, schedule, etc.)

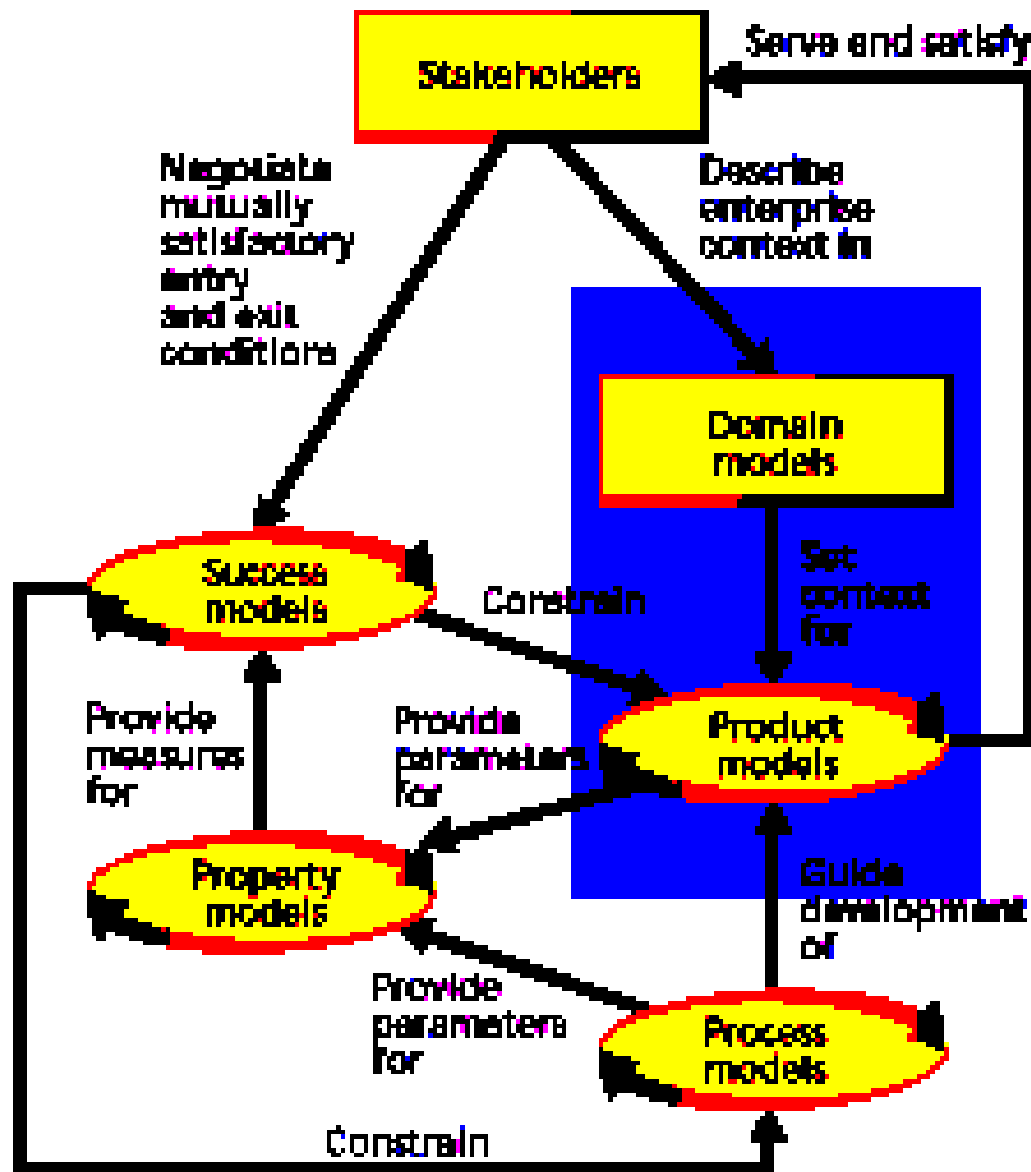
# Getting From A to B



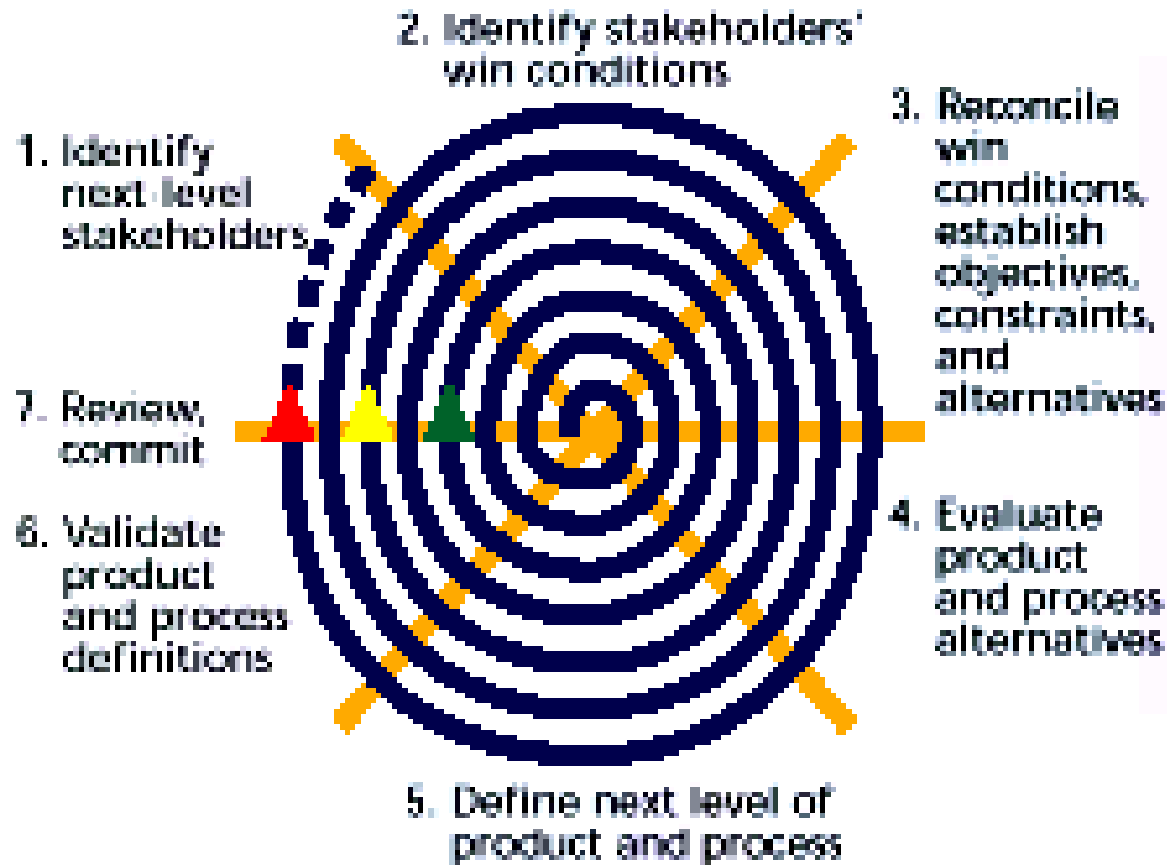
# MBASE Integration Framework



# MBASE Conceptual Framework

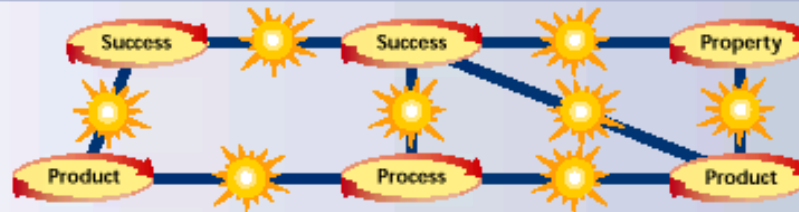


# MBASE WinWin Spiral



- ▲ - Life Cycle Objectives (LCO)
- ▲ - Life Cycle Architecture (LCA)
- ▲ - Initial Operational Capability (IOC)

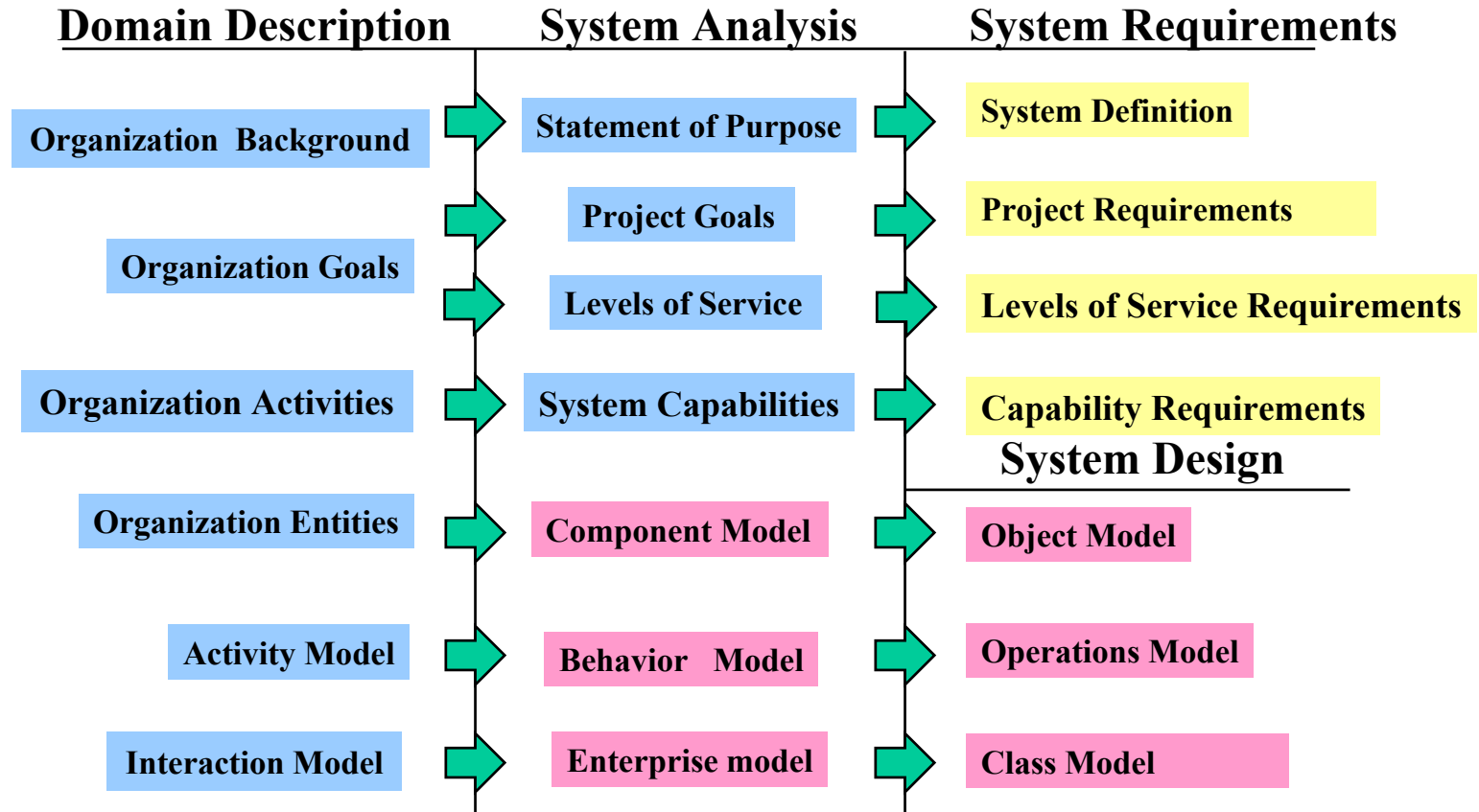
# MBASE Model Clashes



**Table 1. Success model clashes most often encountered by IT pros.**

Type	Example	Cause	Solution
Product-process	Developers try to freeze requirements and processes, and Users lose interest in the system.	Developers want stability and control. Users want features and the flexibility to change them.	Stakeholders negotiate which requirements must be stable and which can be more flexible.
Product-property	Entrepreneurs and Developers tend to overpromise features and schedules.	Users want many features very quickly. Entrepreneurs and Developers say what they need to say to get their business.	Entrepreneurs and Users prioritize features. Developers organize project so that features can be shed to meet schedule.
Product-product	Developers will tend to use their preferred platform, even if it creates compatibility problems down the road for Users.	Developers want to maximize profit and minimize risk. Users want compatibility with existing systems.	Entrepreneurs consider compatibility in selecting Developers. At the first anchor point, system feasibility is confirmed.
Process-process	Entrepreneurs and Developers won't let Sponsors or Users know what process is being used.	Sponsors and Users might try to overcontrol the project. Developer and Entrepreneur culture may suppress warnings.	Stakeholders agree to a thorough review at the anchor points. Progress reports make visible key progress metrics.
Property-property	Detailed agreements about system cost and performance are made at contract signing that can't be met.	Everyone assumes that system behavior is predictable, when in fact it is not. Developers reuse scalable components to meet Entrepreneurs' cost-schedule targets.	Use anchor points to prototype and verify performance and scalability of system components and configurations.

# Integration of MBASE System Definition Elements



 Operational Concept Description (OCD)

 System and Software Requirements Definition (SSRD)

 System and Software Architecture Description (SSAD)

# Modeling versus Documenting

- MBASE describes models, how to build them, and ways to communicate them.
- Documentation is a necessary consequence of modeling
  - why?
  - Also, the act of writing something down helps you solidify a model
- Communication is an essential part of the collaborative development approach
- Avoid documenting for documentation sake!
  - Everything you document should be the result of modeling
  - everything you document should have value and meaning to stakeholders (think about risk here)

The documentation are the models!!!

The Models are the documentation!!!

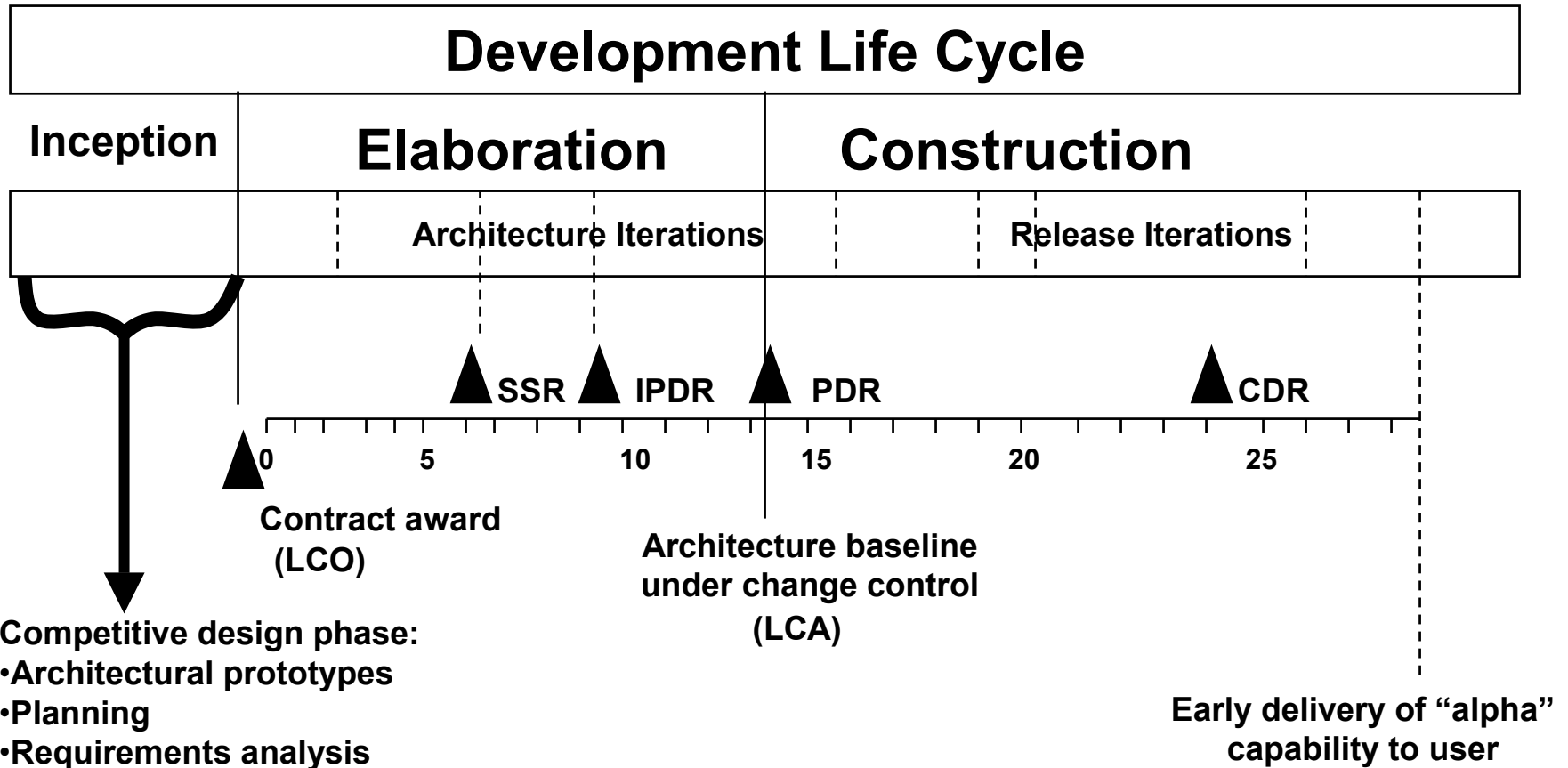
# Upcoming lectures

- Analysis and design
  - More in depth about components, objects, behaviors, operations
- Scheduling
  - Effort estimation
- Model clash resolution
- Project Management
- Construction plans

# Life Cycle Architecture (LCA)

- more formal, with everything tracing upward and downward
- no major unresolved issues or items, and closure mechanisms identified for any unresolved issues or items (e.g., “detailed data entry capabilities will be specified once the Library chooses a Forms Management package on February 15”)
- no more TBD's
- there should no longer be any "possible" or "potential" elements (e.g., Entities, Components, ...)
- no more superfluous, unreferenced items: each element (e.g., Entities, Components, ...) either should reference, or be referenced by another element. Items that are not referenced should be eliminated, or documented as irrelevant

# Common Subsystem Macroprocess



# The Road Ahead Ahead: IOC

Success criteria

**The initial operational capabilities of the system as constructed satisfy the architecture models**

\* This includes all the MBASE models, not just SSAD

\*\* Completeness is not as important as soundness

# **Requirements and Expectations: Domain Model Clashes**

- **Easy/hard things for software people**

**“If you can do queries with all those ands, ors, synonyms, data ranges, etc., it should be easy to do natural language queries.”**

**“If you can scan the document and digitize the text, it should be easy to digitize the figures too.”**

- **Easy/hard things for librarians**

**“It was nice that you could add this access feature, but it overly (centralizes, decentralizes) control of our intellectual property rights.”**

**“It was nice that you could extend the system to serve the medical people, but they haven’t agreed to live with our usage guidelines.”**