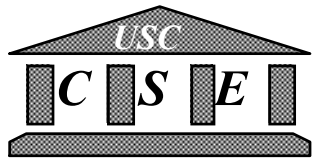


COTS Integration and Use of Prototypes in MBASE

CS577a

2001



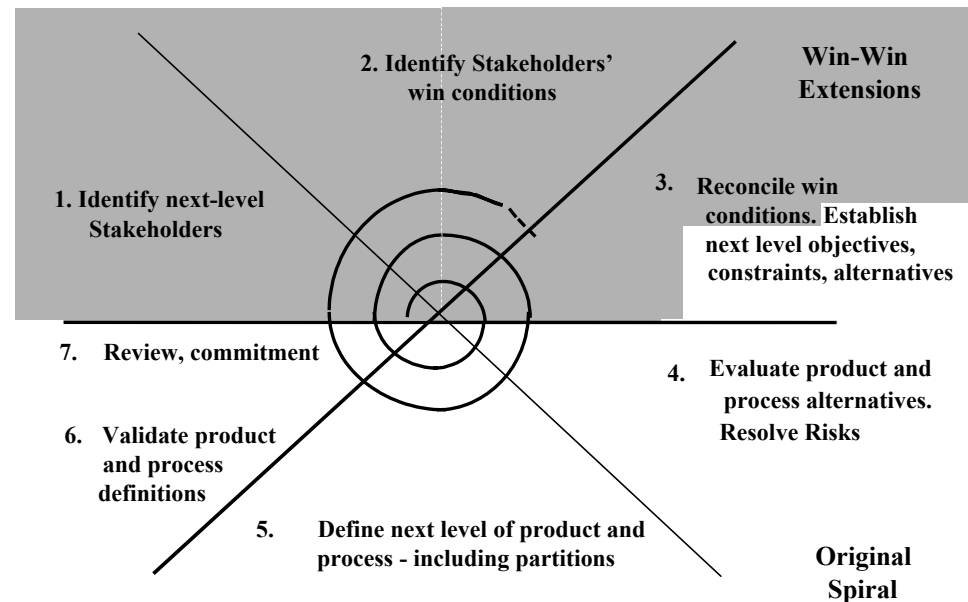
MBASE Approach to COTS Integration

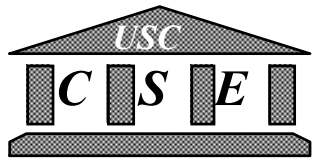
- **Preconditions**
- **Strategic Steps**
- **Tactical steps are situation-dependent**
 - **Primary situation categories**
 - **Easy WinWin example**
- **COTS integration MBASE Milestone Elements**
- **COTS Integration Spiral**
 - **General spiral**
 - **MBASE integration spiral**
- **MBASE Design and COTS**

MBASE COTS Integration: Process Preconditions

- **Success- critical Stakeholders**
- **Shared vision**
 - Details on next chart
- **Stakeholders have negotiated some primary**
 - Objectives
 - Constraints
 - Alternatives
 - Some COTS

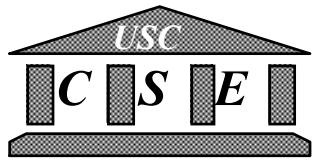
The WinWin Spiral Model





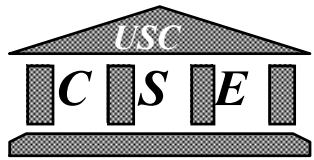
MBASE COTS Integration: Strategic Steps

- **Identify most critical objectives and constraints**
 - **Frequently: properties (cost, schedule, performance, ...);
enterprise architecture;
legacy interoperability or replacement;
core capabilities**
- **Search out more likely alternatives**
- **Screen alternatives with respect to critical constraints**
 - **Try using screening matrix**
 - **If no satisfactory alternatives, rework OC&A's**
- **Take most feasible alternative(s), work into LCO package**
 - **Several situation-dependent tactical processes for this**



Strategic Steps: Easy WinWin Example

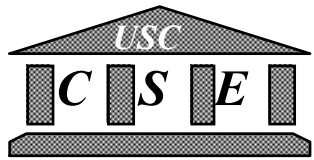
- **Most critical objectives & constraints**
 - WinWin negotiation model preserved (W-I-O-A)
 - Ease of use
 - Low cost
 - Commercial support
 - Windows platform
 - Less important: Unix WinWin API compliance
- **Search out new alternatives**
 - Paul Gruenbacher, GroupSystems.com
- **Screen alternatives: see next matrix**
- **Work into LCO package**
 - Later chart



EasyWinWin Activities

**thinkLet =
collaborative
reasoning tool +
facilitation script +
tool configuration**

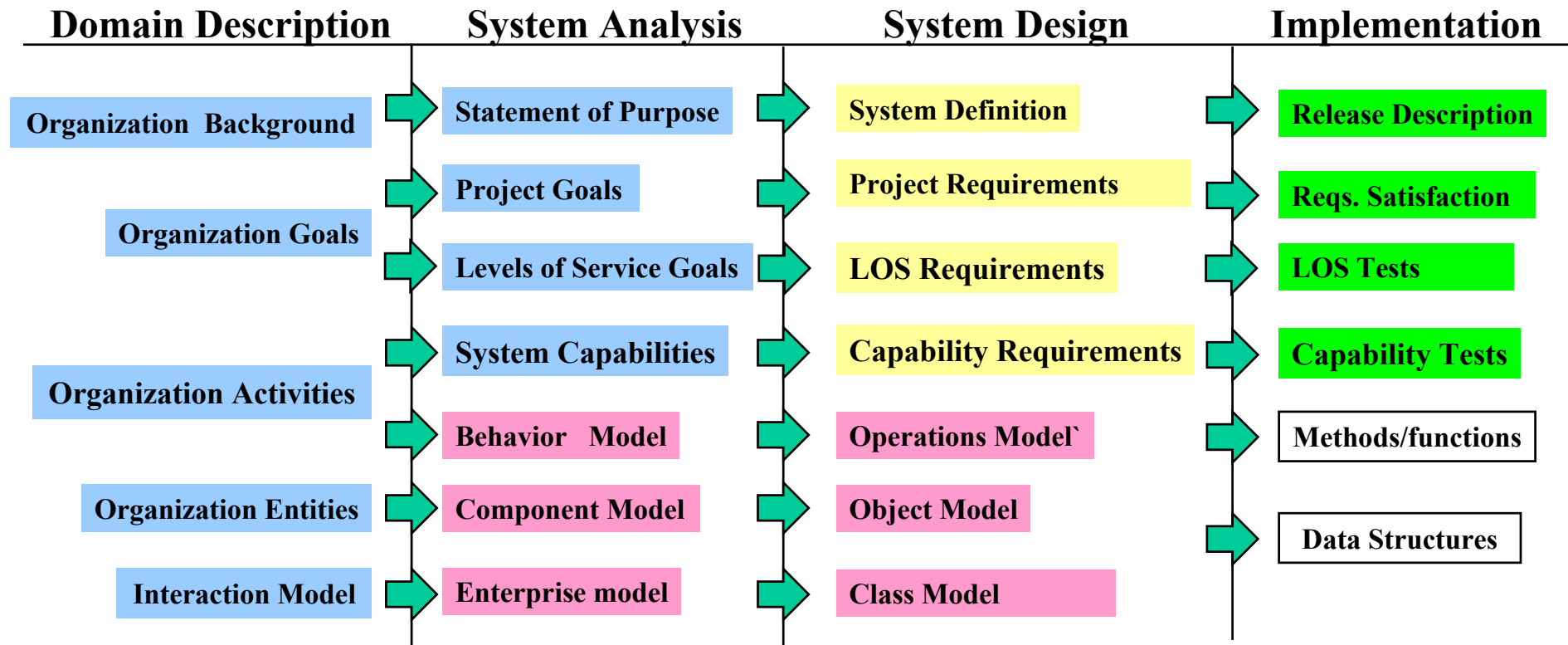
<i>Activity</i>	<i>"ThinkLet"</i>	<i>Tool</i>
Elaborate domain taxonomy	Could-be/ Should-be	GroupOutliner
Brainstorm stakeholder interests	Free Brainstorming	Electronic Brainstorming
Converge on win conditions	FastFocus	Categorizer
Capture domain language	TermCapture	Topic Commenter
Prioritize win conditions	MultiCriteria	Alternative Analysis
Elaborate Conflicts, Constraints, Issues	CrowBar, MultiPass	GroupOutliner
Elaborate Options	MultiPass	GroupOutliner
Negotiate Agreements	MultiPass	GroupOutliner



Easy WinWin Screening Matrix

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

Coverage/Traceability of MBASE Product Models*

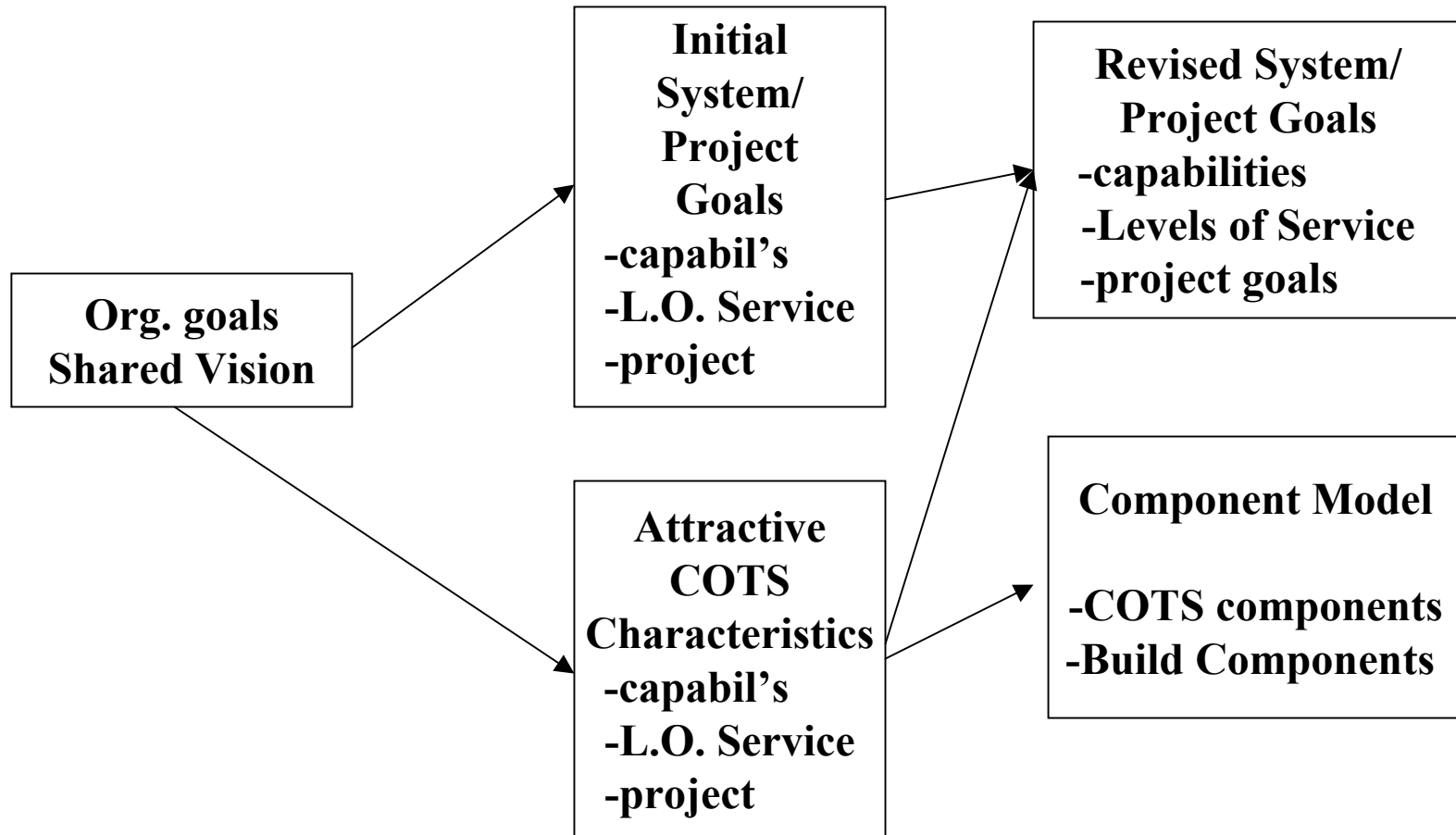


Operational Concept Description (OCD)
 System and Software Requirements Definition (SSRD)
 System and Software Architecture Description (SSAD)
 Construction, Transition, Support (CTS)
 External to MBASE

Operational Concept Description (OCD)
 System and Software Requirements Definition (SSRD)
 External to MBASE

System and Software Architecture Description (SSAD)
 ** Does not include all MBASE models*

Effect of COTS on Mapping Process

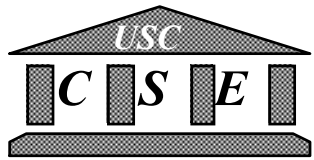


MBASE Milestone Elements

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

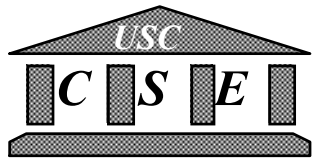
MBASE Milestone Elements (2)

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

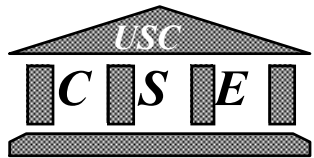


COTS Integration Spiral

- 1. Identify next level system elements, objectives, and constraints**
- 2. Factor elements into system partitions**
- 3. Identify patterns**
- 4. Map patterns to COTS**
- 5. Reconcile architectural mismatches, constraint violations, establish COTS alternatives**
- 6. Evaluate trade-off considerations (e.g. integration effort)**
- 7. Evaluate COTS alternatives with respect to objectives and trade-off considerations**
- 8. Define next level system elements, objectives, constraints**
- 9. Validate COTS integration design**
- 10. Review system elements represented and objectives met**



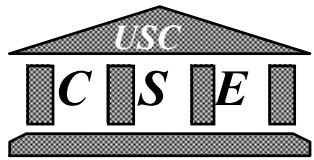
- 1. Identify next level system capabilities (OCD 4.3), goals and constraints (OCD 4.2), and L.O.S. (OCD 4.4)**
- 2. Factor system capabilities and requirements into system components and objects (SSAD 2.1, 3.1, 3.2)**
- 3. Identify architectural patterns and design patterns (from component model SSAD 2.1, design views SSAD 3.1, and object model SSAD 3.2)**
- 4. Map patterns to COTS**
- 5. Reconcile architectural mismatches, constraint violations, establish COTS alternatives (FRD 2.2, 5.2)**
- 6. Evaluate trade-off and risk considerations (OCD 5, FRD 2.1, 2.2, 4, 5)**
- 7. Evaluate COTS alternatives with respect to objectives and trade-off considerations (OCD 5.3, FRD 5)**
- 8. Define next level system elements, objectives, constraints**
- 9. Validate COTS integration design (prototypes, FRD 2.2, CTS Test Description and Results)**
- 10. Review system elements represented and objectives met**



Component Design Views and COTS

- **Since Components are often a simple *object + mechanism*, many COTS products have been developed to handle common situations (patterns) reducing complex, tedious, repetitive, or unnecessary implementation details**
- **The Component model (SSAD 2.1) helps you identify and analyze *architectural patterns* for your system independent of technology implementation details e.g. information self service, distributed services**
- **The design views (SSAD 3.1) help you identify *design patterns* e.g. publish and subscribe, client-server**
- **COTS often exist to implement, partially implement, or assist in implementing design patterns!**

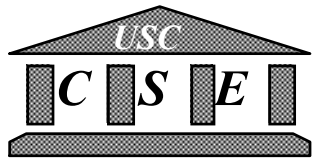
Warning: You must carefully and explicitly account for trade-offs for identifying and integrating COTS into your system



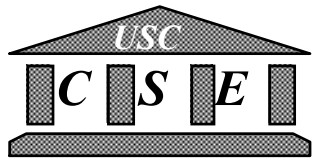
COTS in Design Views

Design Views are a natural place to match COTS to design patterns:

- **Topology (SSAD 3.1.1)**
 - Connectors and layers in are often associated with COTS
- **Design Components (SSAD 3.1.2)**
 - Typically COTS will exist for common complex patterns in Component model. Watch for applications and object libraries
- **Frameworks (SSAD 3.1.3)**
 - Frameworks are constructed to deal with common design needs and thus often are COTS
- **Deployments (SSAD 3.1.4)**
 - Legacy systems, common hardware and operating systems usually have COTS
- **Logical Blocks (SSAD 3.1.5)**
 - Many common block patterns, often these imply COTS



- **Prototypes directly explore design issues**
 - Design views help connect system analysis to design and thus to prototypes
 - Care should be taken to ensure prototypes do not drive the system analysis e.g. do not choose components or capabilities based on prototypes, use for risk reduction
- **Topology (SSAD 3.1.1)**
 - Prototypes explore component connectivity
- **Design Components (SSAD 3.1.2)**
 - Prototypes explore utility and feasibility of COTS for design use
- **Frameworks (SSAD 3.1.3)**
 - Prototypes often make extensive use of frameworks that imply design (watch for risk factors here)
- **Deployments (SSAD 3.1.4)**
 - Prototypes and final system often have similar deployment elements
- **Logical Blocks (SSAD 3.1.5)**
 - Prototypes must relate to logical system elements, helps with design



COTS and Prototypes

- **Prototypes are also a natural place to map patterns to COTS**
 - **Often use COTS in prototypes that imply use in design**
 - **COTS often have competition that may be more suitable for the final product e.g. MS Access → Oracle, MySQL, MS SQL**
 - **Common to use prototyping to establish COTS trade-off factors (integration effort, L.O.S. qualities, etc.)**
 - **Good planning can help make this proactive, advanced prototyping may still be required to resolve details**
 - **Design Views identify what COTS need prototyping**