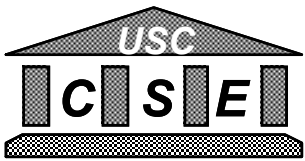


SAIV/CAIV/SCQAIV

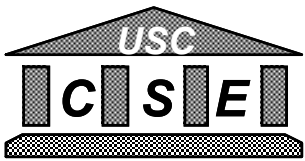
CS577A

2002



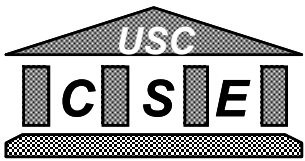
Is There A Problem?

- **The clients impose cost, schedule, or quality constraints**
 - **Want the system on a ridiculously short schedule, low cost, high quality**



Is There A Problem?

- **The clients impose cost, schedule, or quality constraints**
 - **Want the system on a ridiculously short schedule, low cost, high quality**
 - **And they aren't even willing to commit on just exactly what they want!**



Is There A Problem?

- **The clients impose cost, schedule, or quality constraints**
 - **Want the system on a ridiculously short schedule, low cost, high quality**
 - **And they aren't even willing to commit on just exactly what they want!**
- **Usually, it leads to a Death March**

Is There A Problem?

- **The clients impose cost, schedule, or quality constraints**
 - **Want the system on a ridiculously short schedule, low cost, high quality**
 - **And they aren't even willing to commit on just exactly what they want!**
- **Usually, it leads to a Death March,**
 - **but, an opportunity to succeed with strategic methods...**

What is The Problem?

- **Accepting schedule [cost, quality] goals before understanding how they relate to the requirements and constraints, often results in risky behavior:**
 - **Retrofitting required capabilities in an ad-hoc way (constraints?)**
 - **Reducing scope or drop features arbitrarily (requirements?)**
 - **Painful “heroic” efforts that succeed by luck than by design (fail later?)**

The Advice...

Plan from the outset to meet schedule, budget, quality, etc. goals

 *Strategic Methods*

- Must know what these goals are
- Must explicitly consider non-technical factors of *risk, value, and cost*
- These define success (or failure) and dictate the path development must take, but are often not given adequate consideration

Tactical Vs. Strategic

- Tactical
 - Small-scale actions serving to contain or respond to risks made or carried out with only a limited or immediate end in view
 - Examples: Risk identification and assessment, Top-10 risk monitoring, Risk contingents
- Strategic
 - As an integrated whole or to a planned effect (e.g. Expected Return on Investment)
 - Examples: Risk driven “how much is enough?” questions, Risk/value based feasibility assessments (GQM), Risk based development processes (e.g. SAIV)
 - For our purposes a risk profile over an independent variable
 - time, effort, cost, testing amount, defects, etc.

Risk-Driven Specifications and Activities

- **Basic driving principle for strategic activities and specifications (i.e. modeling, model content, degree of detail, etc.)**

If it's risky to do something, Don't

e.g. specify firm GUI requirements early

If it's risky not to do something, Do

e.g. document shared protocols

- **Seems obvious, but often not explicitly planned or managed!**

The SAIV Process Model

“If schedule is your independent variable, then just modulate your functionality to meet schedule.”

- 1. Shared vision and expectations management**
- 2. Feature prioritization**
- 3. Schedule range estimation**
- 4. Core capabilities determination**
- 5. Architecture Flexibility Determination**
- 6. Incremental development**
- 7. Change and progress monitoring and control**

1. Shared Vision and Expectations Management

- **Obtain stakeholder agreement**
 - Meeting a fixed schedule is the most critical objective
 - The other objectives (e.g. IOC features) can be variable
- **Simplifier/Complicator Analysis**
 - Helps manage expectations and avoid risky design elements from the outset

Example: Fulltext Title Database

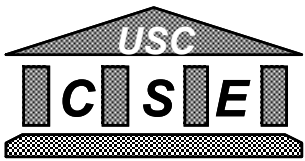
- **Fixed 24-week schedule**

- 33 student person-weeks for 5 member team
(students are not full time employees!)

Full lifecycle: operational concept, shared vision, architecture, prototyping, design, testing, “cold turkey” transition, documentation, business case, ...

- **System to provide users ability to search multiple library periodical vendor databases**

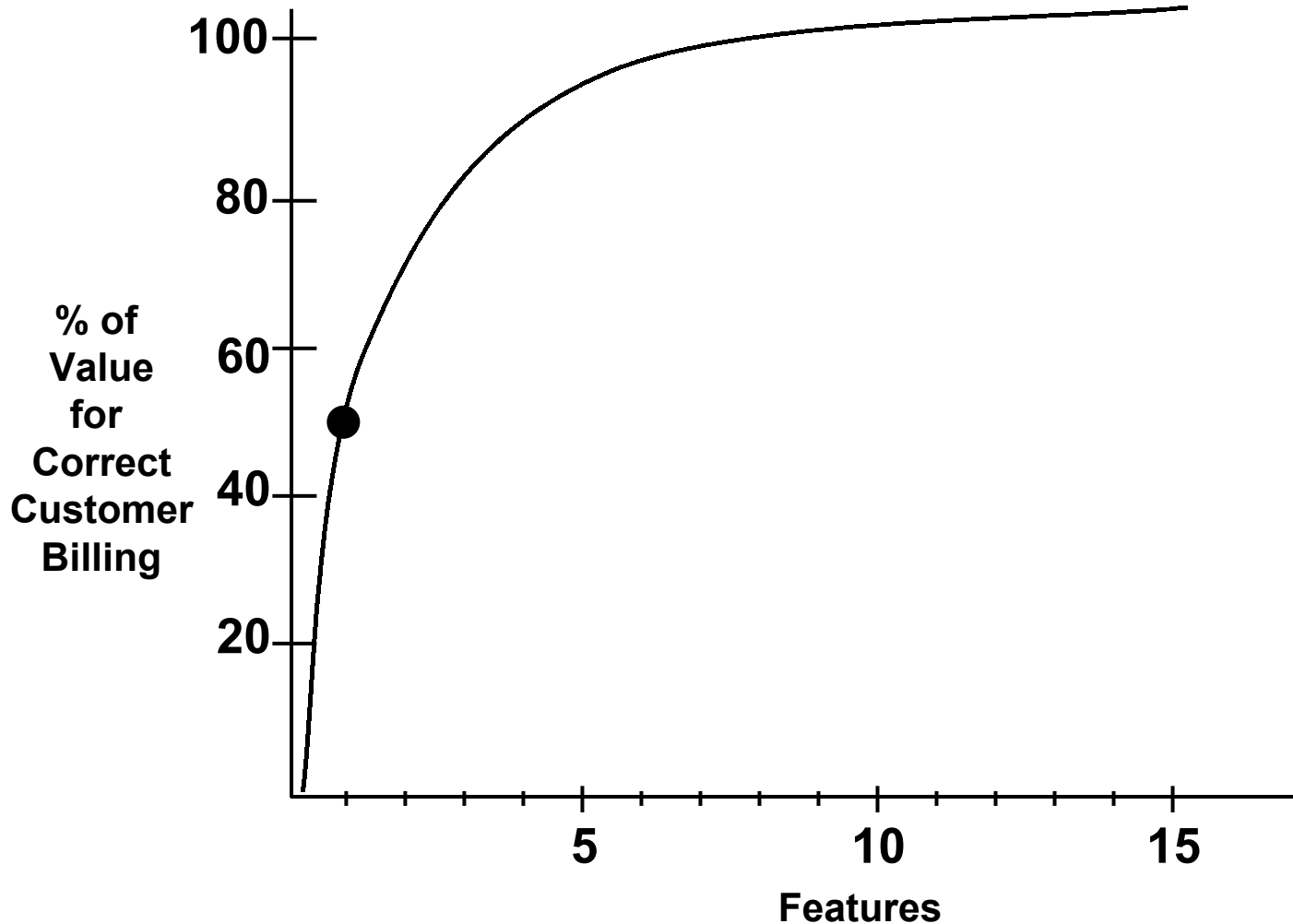
- 12 prioritized features via WinWin negotiation
- Requirements volatility?



2. Feature Prioritization

- **Feature Prioritization**
 - Stakeholders use USC/GroupSystem EasyWinWin requirement negotiation tool to converge on project requirements
 - Initial system prototyping in parallel with capability and requirements generation
 - Focus on value of features

20% of Features Provide 80% of Value: Focus on These (Bullock, 2000)



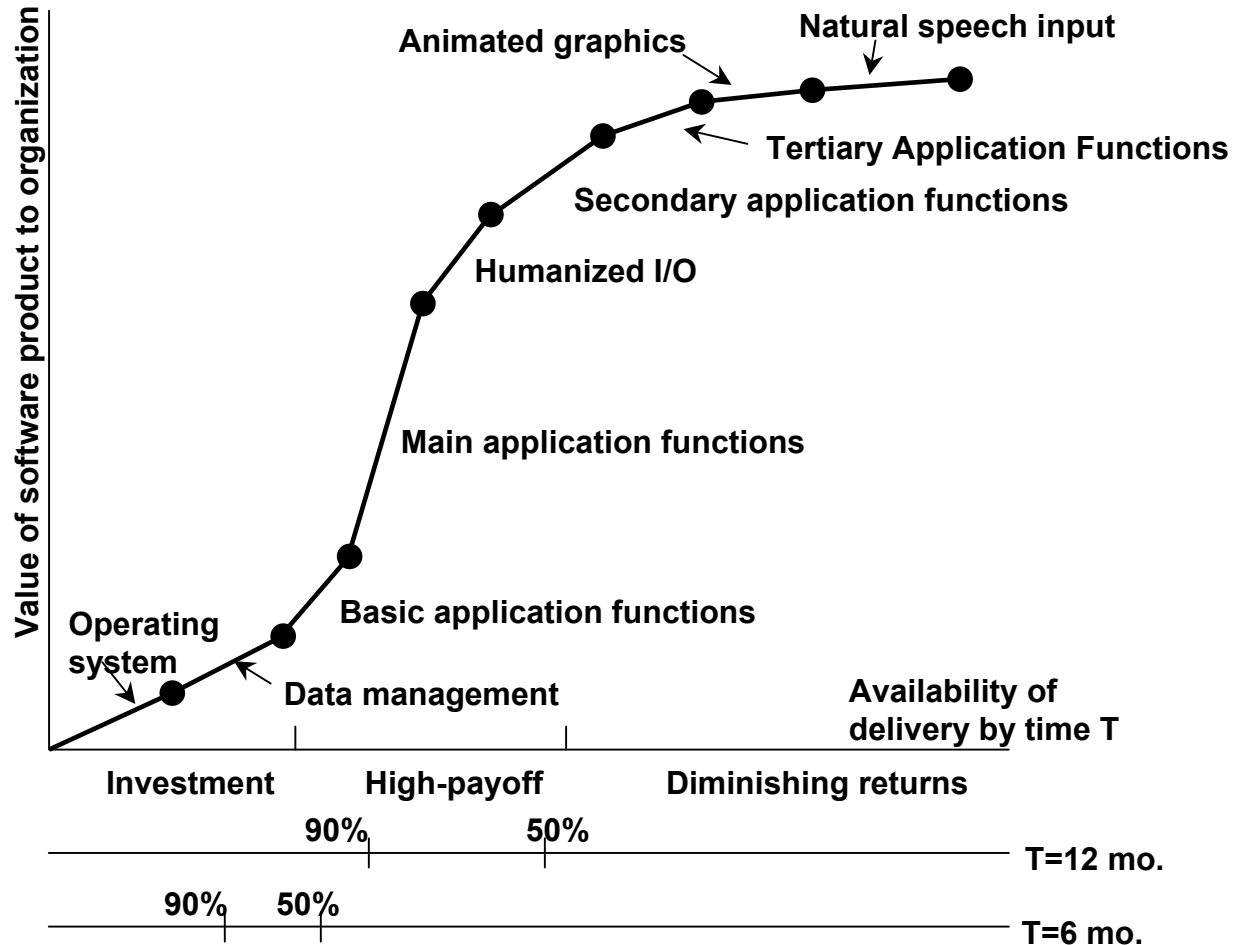
Prioritized Product Features

Priority	Features	
Very high	F1	Search and locate full-text journal titles by title keywords
Very high	F2	Search and locate full-text journal titles by title keywords and date, title keywords and volume or number, title keywords with any combination of the other three attributes
Very high	F3	Provide hyperlinks to vendors' databases in the searching results
Very high	F4	Update Fulltext Title Database using current vendors' title lists
High	F5	Automatically FTP downloaded title lists from administrator's local machine to remote server
High	F6	System administrator authentication
High	F7	Administrator password maintenance
Medium	F8	Add new vendor's title list profile
Medium	F9	Delete existing vendor's title list profile
Medium	F10	Modify existing vendor's title list profile
Medium	F11	View existing vendor's title list profile
Low	F12	Allow more searching options starting with searching by ISSN, etc.

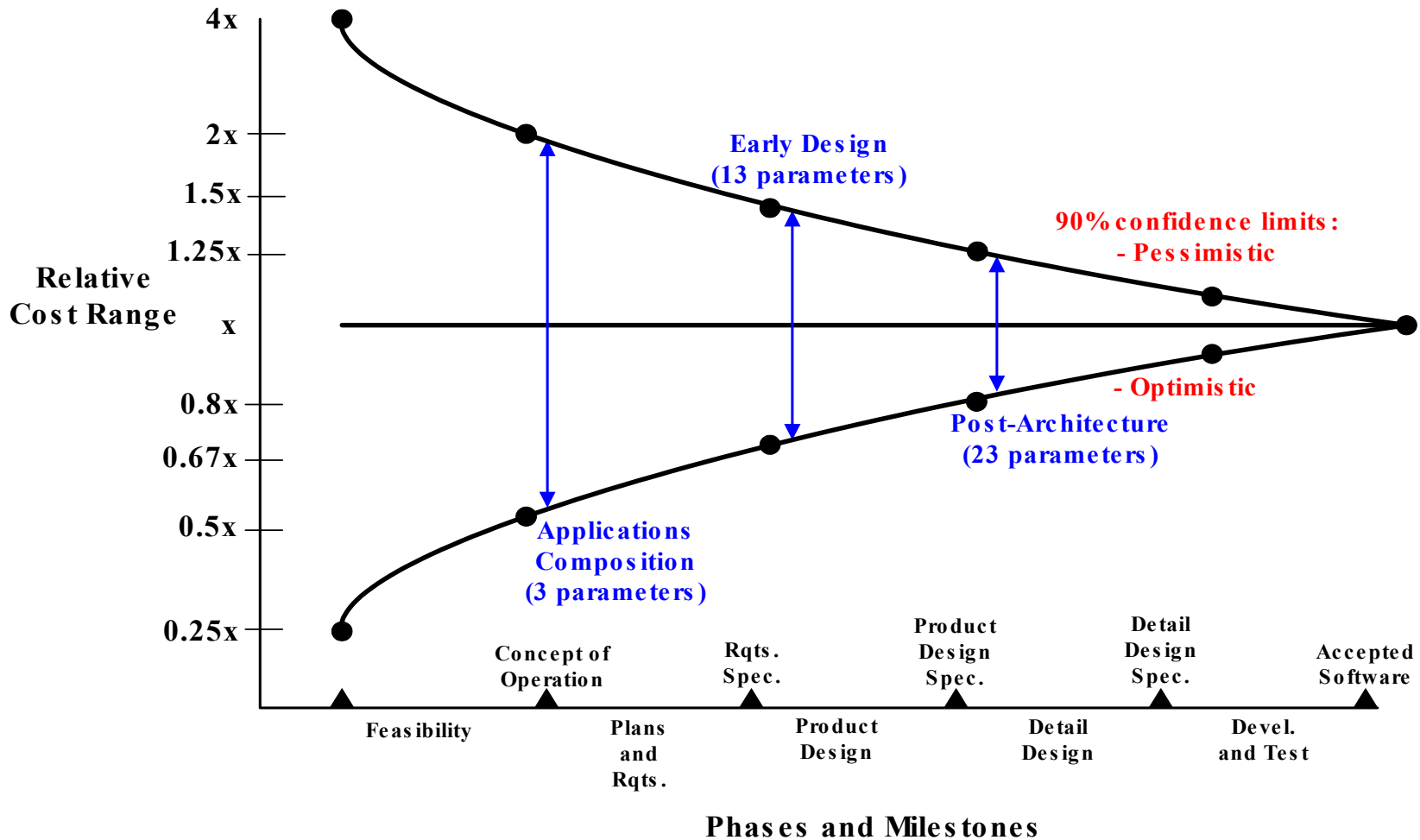
3. Schedule Range Estimation

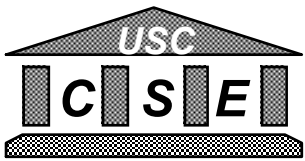
- **Schedule Range Estimation**
 - Not how long to deliver capabilities, but how likely capabilities can be delivered within given times
 - Expert judgment
 - Parametric cost modeling (e.g. COCOMO)
 - Relate schedule estimates to value
 - Consider project production function

Software Product Production Function



COCOMO II Cost/Effort Estimate Ranges

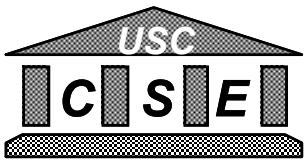




COCOMO II Analysis

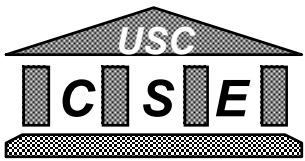
- F1-F12
 - 40 student person-weeks
 - Very risky to attempt
- F1-F7 (80% of value)
 - 17 student person-weeks
 - Leaves time to build architecture flexibility to ease change or addition of features later

Pessimistic estimate with 90% confidence



4. Core Capability Determination

- **Core capability not just top-priority features:**
 - **Coherent and workable end-to-end operational capability**
 - **Provides significant value**
 - **Facilitate evolution to full operational capability**
 - **Goal is to architect for ease of adding, dropping marginal features without significant additional risk**



Core Capability Determination

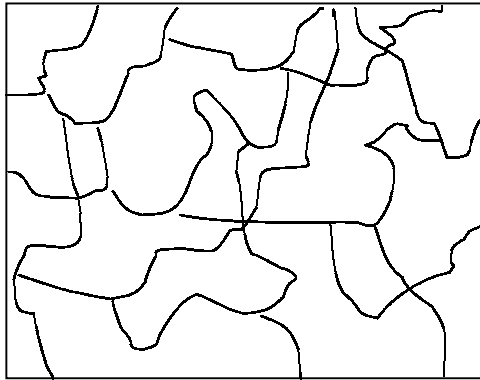
- **Goal: Get into “High Payoff” range with low risk**
- **Negotiated three core capabilities for the Fulltext Title Database system**
 - 1) Provide a full-text journal title search capability (features F1,F2,F3, optionally F4)
 - 2) Update the Fulltext Title Database (features F4, F5, F6)
 - 3) Administrator password maintenance (feature F7)
- **Highly desired, but not could be dropped if necessary**
 - 4) Accommodate vendors’ title lists in various formats

5. Architecture Flexibility Determination

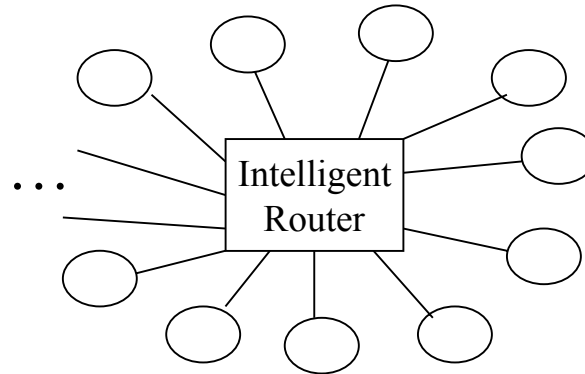
- **Simply modularizing the core capabilities is not enough**
- **The source of anticipated changes must be taken into account**
 - New features and interaction with each others
 - Their interaction with existing core capabilities
- **Strategically design an architecture with an appropriate level of flexibility without introducing an excessive risk of schedule overrun**
 - Use an approximate combinatorial expected effort overhead model

Architecture Choices

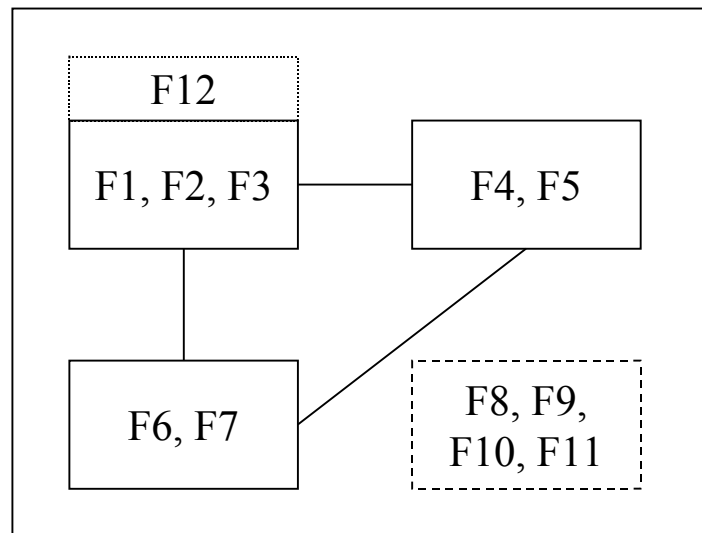
Rigid



Hyper-flexible

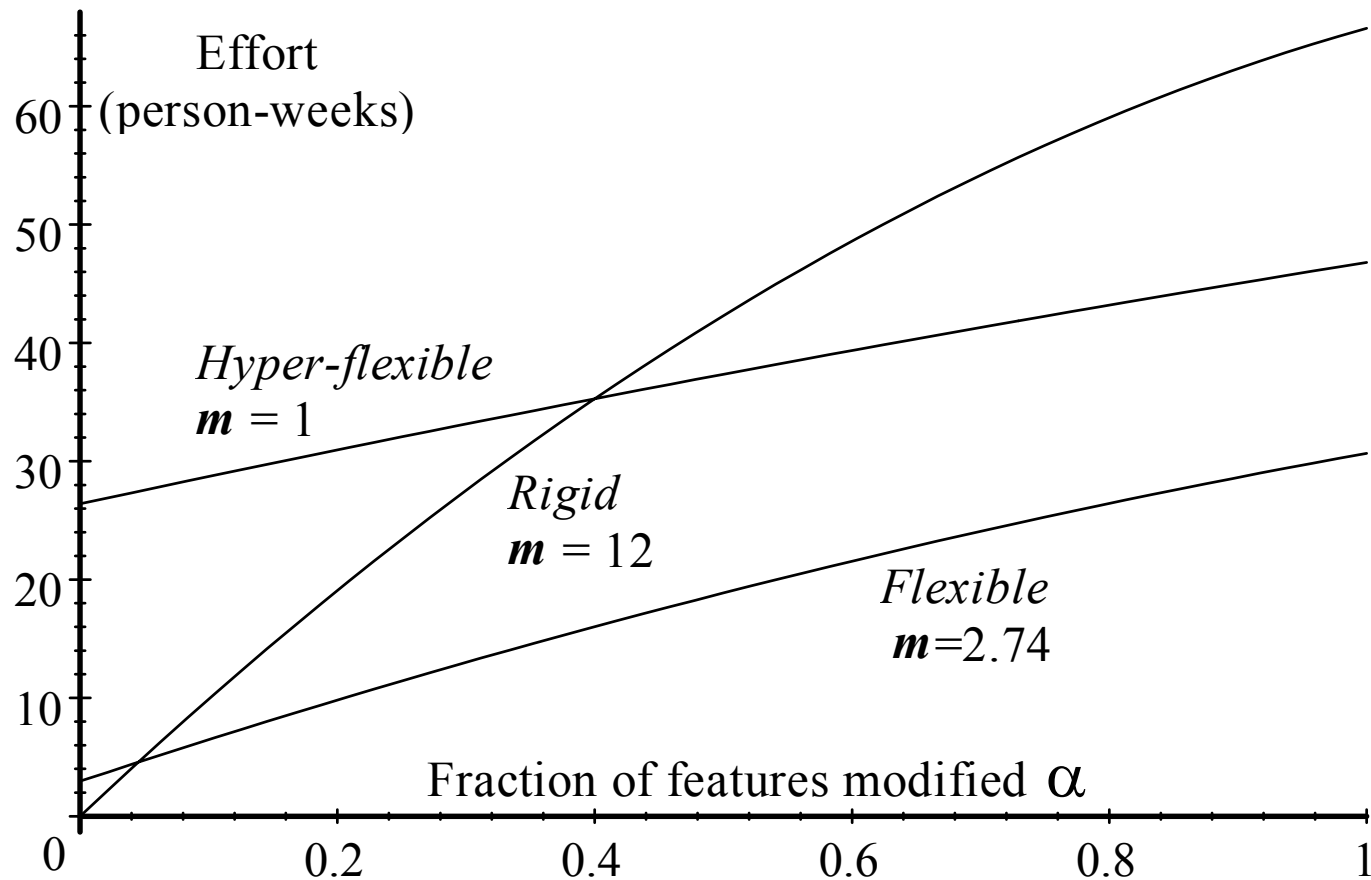


Flexible



- How much flexibility?
 - Avoid risking core implementation
 - Increase value by reducing schedule risks for changes and additions

Architecture Flexibility Determination



Architecture Flexibility Determination (cont.)

$$E = c_1 \alpha N + c_2 \binom{N/m}{2} + c_3 \binom{\alpha m}{2} N/m + c_4 \alpha (1 - \alpha) m N$$

E – total change effort

N – total number of features ($N=12$)

m – modularity factor ($m = 2.74$)

α – a fraction of anticipated changes ($\alpha = 5/12$)

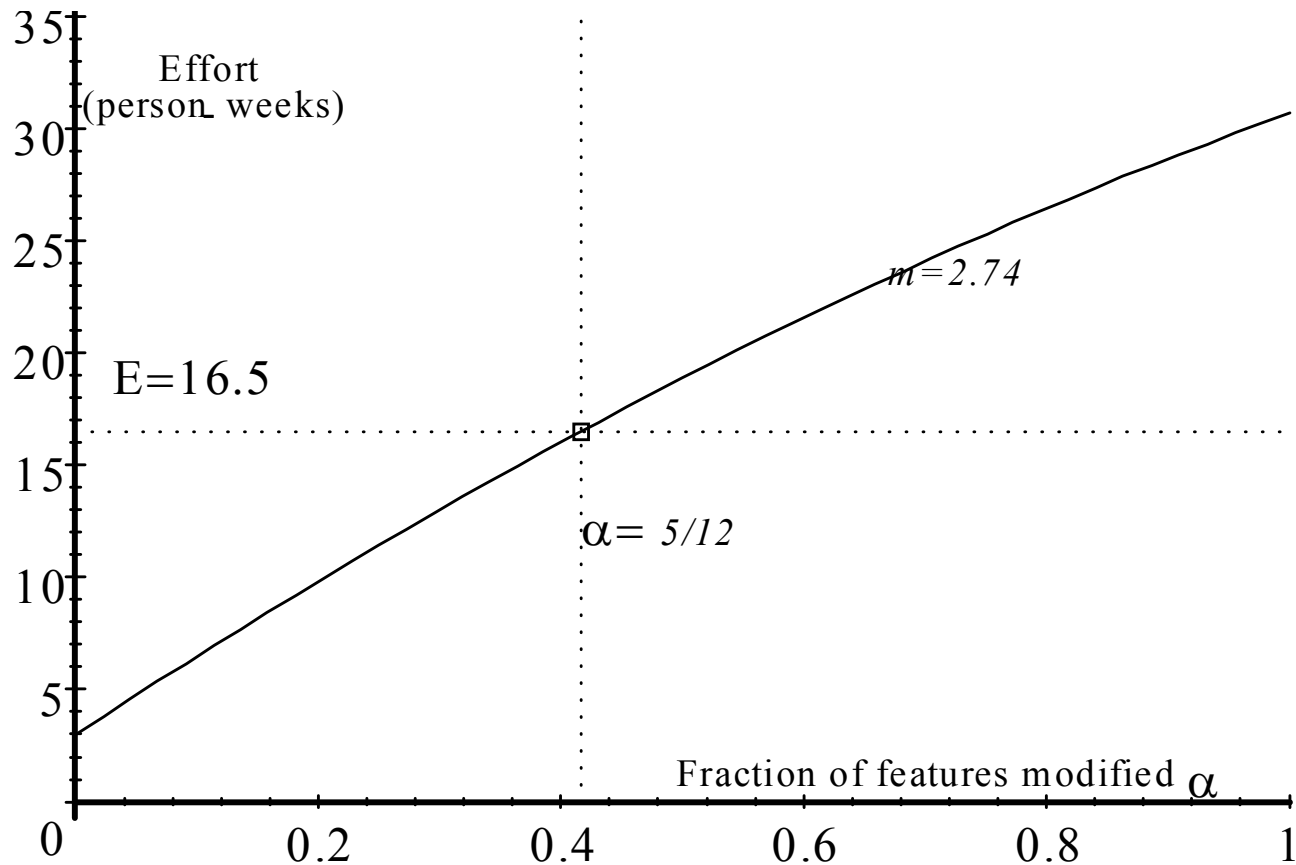
c_1 – average effort to develop a feature

c_2 – average development effort of per inter-module interaction

c_3 – average development effort of per intra-module feature interaction

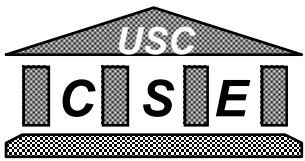
c_4 – average effort to accommodate a new feature with each existing feature in a module

Architecture Flexibility Determination



6. Incremental development

- **LCO package establishes the core operational concept and shared vision**
 - Core capabilities, critical risks, project value
- **LCA package establishes basic (flexible) architecture and includes an incremental development plan**
 - Initial Operational Capability Iteration I
(70~80% of the schedule most likely)
 - Milestone: **Core Capability Demonstration**
 - Initial Operational Capability Iteration II (add the next-highest priority features into IOC)

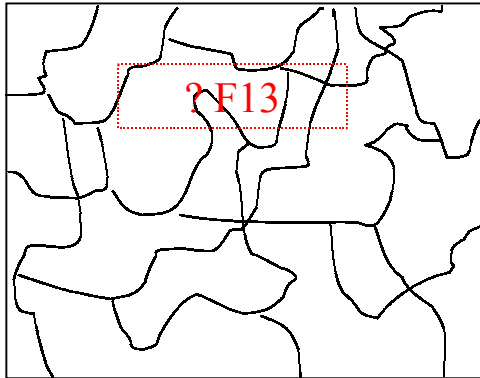


Incremental Development, and Coping with Rapid Change within Fulltext

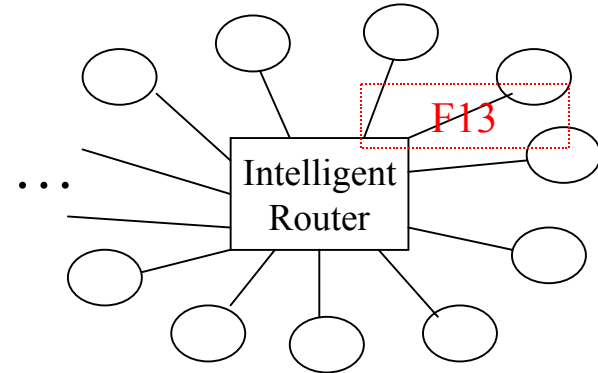
- **Initial Operational Capability Iteration I: Build Core Capabilities (70~80% of the schedule)**
 - Provide a full-text journal title search capability
 - Update the Fulltext Title Database
 - Administrator password maintenance
- **Initial Operational Capability Iteration II**
 - Accommodate vendors' title lists in various formats
 - Negotiated feature demotion
 - **Add** F13: Partial keyword searching
 - **Drop** F12: Allow more searching options starting with searching by ISSN, etc.

Revised Architecture

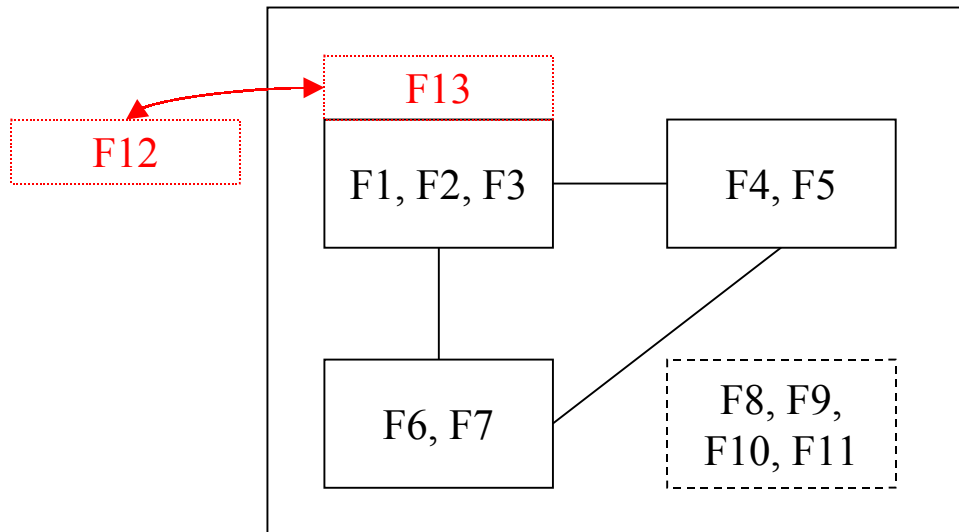
Rigid

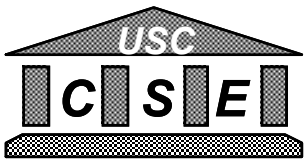


Hyper-flexible



Flexible





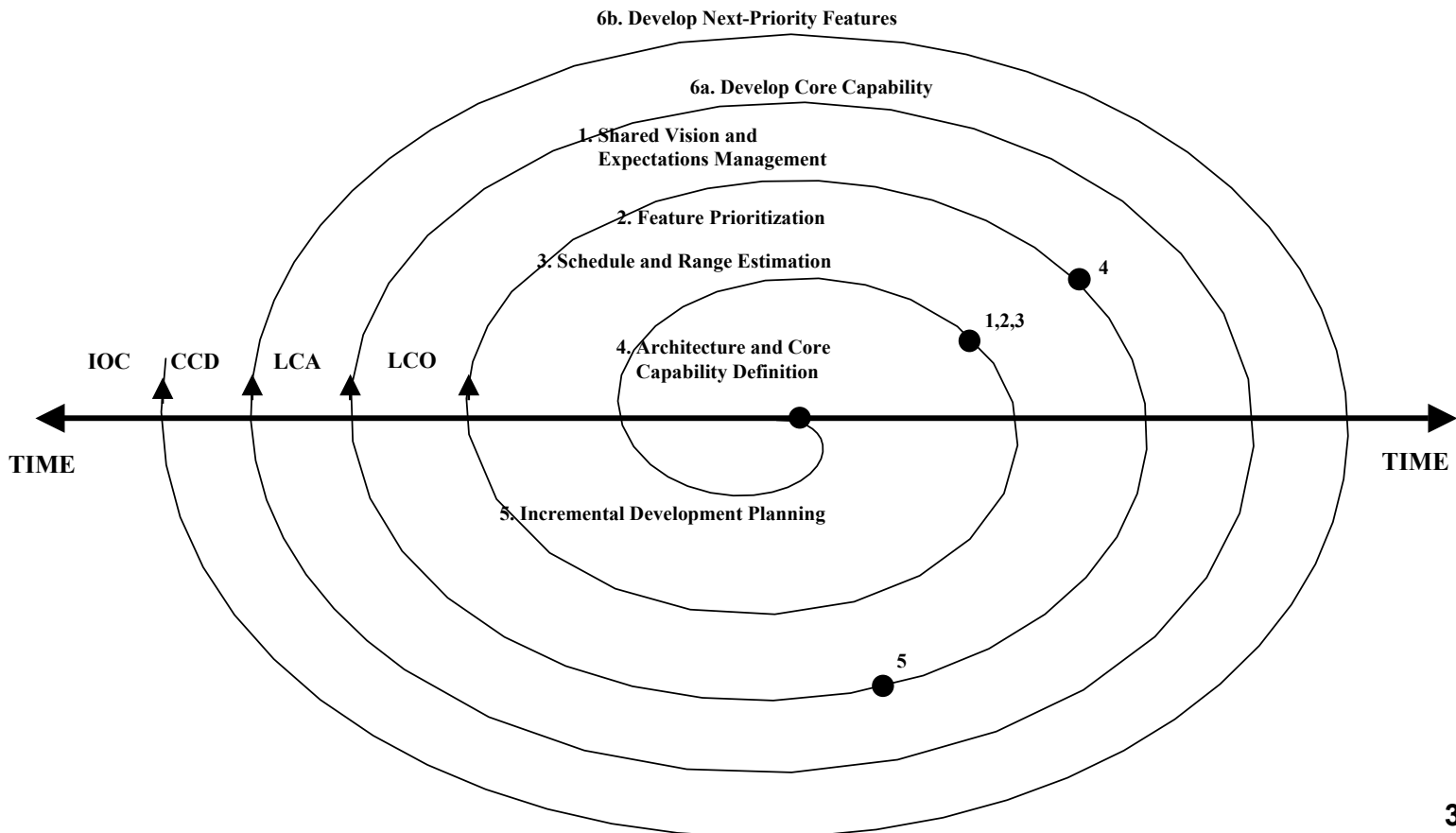
7. Change and progress monitoring and control

- **Three major sources of changes**
 - Schedule slips
 - Requirement changes
 - Project changes
- **Essential to monitor progress and match to plan**
- **Accommodate possible changes within the existing plan**
- **Encapsulate the foreseeable changes within modules**
- **Evolutionary requirements with architecture support**

Risk-driven MBASE-Spiral approach and SAIV

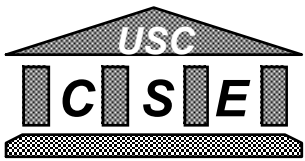
- **Uses the risk of schedule or cost overrun to invert the usual software-intensive-system acquisition process**
 - **Schedule becomes the independent variable**
 - **Lowest-priority features become the dependent variable**
 - **This requires several sub-processes in SAIV (7 steps)**

Mapping of SAIV Spiral Process Elements onto Win-Win Spiral Model



Conclusions: SAIV Critical Success Factors

- **Working with stakeholders in advance to achieve a shared product vision and realistic expectations;**
- **Getting clients to develop and maintain prioritized requirements;**
- **Scoping the core capability to fit within the high-payoff segment of the application's production function for the given schedule;**
- **Architecting the system for ease of adding and dropping features in an appropriate level of flexibility;**
- **Disciplined progress monitoring and corrective action to counter schedule threats**

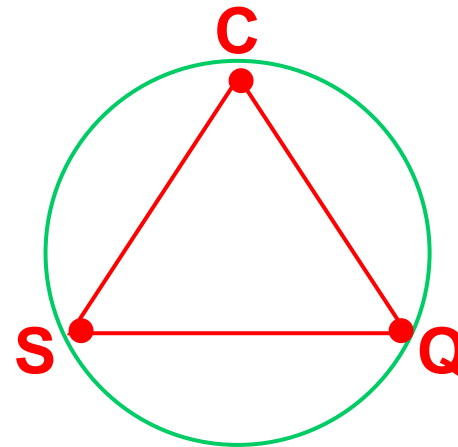
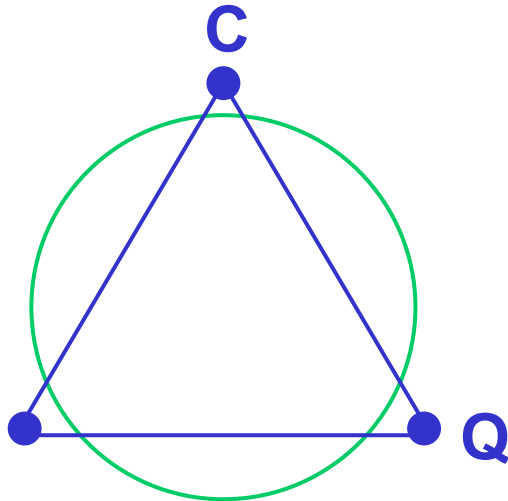


CAIV and SCQAIV

- **SAIV process model also works for Cost as Independent Variable**
- **SCQAIV model is a straightforward extension of CAIV and SAIV**
 - **And “Cost, Schedule, Quality: Pick Any Two?”**

Cost, Schedule, Quality: Pick any Two?

- Consider C, S, Q as Independent Variable
 - Feature Set as Dependent Variable



“Cost, Schedule, Quality: Pick All Three”