CMMI and the Balance of Discipline and Agility

Barry Boehm, USC

CMMI Technology Conference ‘02
November 13, 2002

(boehm@; http://) sunset.usc.edu
Outline

• Clausewitz and De Marco: Armor vs. Mobility
  – Software CMM and Agile Methods

• Characteristics of Future Systems
  – Range of sizes and criticalities
  – All need to balance discipline and agility

• Using Risk to Balance Discipline and Agility
  – No one-size-fits-all solution

• Representative Future Example: Future Combat Systems
  – Complex system of systems (CSOS)

• Conclusions
Clausewitz and De Marco: Armor and Mobility

- Clausewitz: Armor and mobility alternate dominance

  Greeks

  Romans

  Franks

  Castles

  Maginot Line

  Panzers

  Field Artillery

  Mongols

  Vandals, Huns
Clausewitz and **De Marco**: Armor and Mobility

- **Clausewitz**: Armor and mobility alternate dominance
- **De Marco**: Same is true for software methods

- Whither CMMI?
The Agile Manifesto - I

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

• **Individuals and interactions** over processes and tools
• **Working software** over comprehensive documentation
• **Customer collaboration** over contract negotiation
• **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.
Various Agile Methods Available

• Adaptive Software Development (ASD)
• Agile Modeling
• Crystal methods
• Dynamic System Development Methodology (DSDM)
  * eXtreme Programming (XP)
• Feature Driven Development
• Lean Development
• Scrum
XP: The 12 Practices

- The Planning Game
- Small Releases
- Metaphor
- Simple Design
- Testing
- Refactoring
- Pair Programming
- Collective Ownership
- Continuous Integration
- 40-hour Week
- On-site Customer
- Coding Standards

-Used generatively, not imperatively
The Planning Spectrum

Hackers

XP

Adaptive SW Devel.

Milestone Risk-Driven Models

Milestone Plan-Driven Models

Inch- Pebble Ironbound Contract

Agile Methods

Software CMM

CMMI

11/13/2002 ©USC-CSE
## Agile and Plan-Driven Home Grounds

<table>
<thead>
<tr>
<th>Agile Home Ground</th>
<th>Plan-Driven Home Ground</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Agile, knowledgeable, collocated,</td>
<td>• Plan-oriented developers; mix of skills</td>
</tr>
<tr>
<td>collaborative developers</td>
<td>• Mix of customer capability levels</td>
</tr>
<tr>
<td>• Above plus representative, empowered</td>
<td>• Reliance on explicit documented knowledge</td>
</tr>
<tr>
<td>customers</td>
<td>• Requirements knowable early; largely stable</td>
</tr>
<tr>
<td>• Reliance on tacit interpersonal</td>
<td>• Architected for current and foreseeable requirements</td>
</tr>
<tr>
<td>knowledge</td>
<td>• Refactoring expensive</td>
</tr>
<tr>
<td>• Largely emergent requirements, rapid</td>
<td>• Smaller teams, products</td>
</tr>
<tr>
<td>change</td>
<td>• Premium on rapid value</td>
</tr>
<tr>
<td>• Architected for current requirements</td>
<td>• Premium on high-assurance</td>
</tr>
<tr>
<td>• Refactoring inexpensive</td>
<td></td>
</tr>
<tr>
<td>• Smaller teams, products</td>
<td></td>
</tr>
<tr>
<td>• Premium on rapid value</td>
<td></td>
</tr>
</tbody>
</table>
Outline

• Clausewitz and De Marco: Armor vs. Mobility
  – Software CMM and Agile Methods

→ • Characteristics of Future Systems
  – Range of sizes and criticalities
  – All need to balance discipline and agility

• Using Risk to Balance Discipline and Agility
  – No one-size-fits-all solution

• Representative Future Example: Future Combat Systems
  – Complex system of systems (CSOS)

• Conclusions
# Information Technology Trends

<table>
<thead>
<tr>
<th>Traditional Development</th>
<th>Current/Future Trends</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standalone systems</td>
<td>Everything connected-maybe</td>
</tr>
<tr>
<td>Stable requirements</td>
<td>Rapid requirements change</td>
</tr>
<tr>
<td>Rqts. determine capabilities</td>
<td>COTS capabilities determine rqts.</td>
</tr>
<tr>
<td>Control over evolution</td>
<td>No control over COTS evolution</td>
</tr>
<tr>
<td>Enough time to keep stable</td>
<td>Ever-decreasing cycle times</td>
</tr>
<tr>
<td>Small to big systems</td>
<td>Plus very big systems of systems</td>
</tr>
<tr>
<td>Repeatability-oriented process, maturity models</td>
<td>Adaptive process models</td>
</tr>
</tbody>
</table>
The “Separation of Concerns” Legacy

“The notion of ‘user’ cannot be precisely defined, and therefore has no place in CS or SE.”
– Edsger Dijkstra, ICSE 4, 1979

“Analysis and allocation of the system requirements is not the responsibility of the SE group but is a prerequisite for their work.”
– Mark Paulk at al., SEI Software CMM v.1.1, 1993
I wonder when they'll give us our requirements?
The CMMI Software Paradigm

• System and software engineering are integrated
  – Software has a seat at the center table
• Requirements, architecture, and process are developed concurrently
  – Along with prototypes and key capabilities
• Developments done by integrated teams
  – Collaborative vs. adversarial process
  – Based on shared vision, negotiated stakeholder
How Much Planning Is Enough?
- A risk analysis approach

• Risk Exposure $RE = \text{Prob}\ (\text{Loss}) \times \text{Size (Loss)}$
  
  – “Loss” – financial; reputation; future prospects, …

• For multiple sources of loss:

  \[
  RE = \sum_{\text{sources}} \left[ \text{Prob}\ (\text{Loss}) \times \text{Size (Loss)} \right]_{\text{source}}
  \]
Example RE Profile: Planning Detail
- Loss due to inadequate plans

\[ RE = P(L) \times S(L) \]

- high \( P(L) \): inadequate plans
- high \( S(L) \): major problems
  (oversights, delays, rework)
- low \( P(L) \): thorough plans
- low \( S(L) \): minor problems

Time and Effort Invested in plans
Example RE Profile: Planning Detail

- Loss due to inadequate plans
- Loss due to market share erosion

\[ RE = P(L) \times S(L) \]

- high P(L): inadequate plans
- high S(L): major problems (oversights, delays, rework)
- low P(L): few plan delays
- low S(L): early value capture
- high P(L): plan breakage, delay
- high S(L): value capture delays
- low P(L): thorough plans
- low S(L): minor problems

Time and Effort Invested in Plans
Example RE Profile: Time to Ship
- Sum of Risk Exposures

\[ RE = P(L) \times S(L) \]

- low P(L): thorough plans
- low S(L): minor problems

- high P(L): plan breakage, delay
- high S(L): value capture delays

Sweet Spot

- low P(L): few plan delays
- low S(L): early value capture

- high P(L): inadequate plans
- high S(L): major problems
  (oversights, delays, rework)

- low P(L): thorough plans
- low S(L): minor problems
Comparative RE Profile: Plan-Driven Home Ground

$RE = P(L) \cdot S(L)$

Higher $S(L)$: large system rework

Mainstream Sweet Spot

Plan-Driven Sweet Spot

Time and Effort Invested in Plans
Comparative RE Profile: Agile Home Ground

\[ RE = P(L) \times S(L) \]

Mainstream Sweet Spot

Agile Sweet Spot

Lower S(L): easy rework

Time and Effort Invested in Plans
Outline

• Clausewitz and De Marco: Armor vs. Mobility
  – Software CMM and Agile Methods

• Characteristics of Future Systems
  – Range of sizes and criticalities
  – All need to balance discipline and agility

• Using Risk to Balance Discipline and Agility
  – No one-size-fits-all solution

• Representative Future Example: Future Combat Systems
  – Complex system of systems (CSOS)

• Conclusions
Future Combat Systems: A Network-Centric Example

From This...

Small Unit UAV
Other Layered Sensors
Network Centric Force
Distributed Fire Mechanisms
Robotic Direct Fire
Robotic Sensor
Robotic NLOS Fire
Manned C2/Infantry Squad

To This...

Network Centric Distributed Platforms

Exploit Battlefield Non-Linearities using Technology to Reduce the Size of Platforms and the Force

11/13/2002 ©USC-CSE
Total Collaborative Effort to Support FCS

- **FY00**
  - Design Competition
  - Concept Development / Modeling and Simulation
  - Government-Run Experiments

- **FY01**
  - Preliminary Design
  - PDR

- **FY02**
  - Detailed Design
  - Build

- **FY03**
  - Design Competition

- **FY04**
  - CDR

- **FY05**
  - T & E

- **FY06**
  - Shakeout

**CHPS**
- **Robotics**
  - AUTONOMY
  - VISION

**SUO**
- **Unmanned Ground Vehicle**
  - MOBILITY
  - DESIGN

**AFSS**
- **Maneuver BLOS Networked Fires Weapon**
  - LOITER ATTACK
  - PRECISION ATTACK

**A160 ($5.0M)**
- **Organic All-Weather Targeting Vehicle**
  - 3D PLATFORM
  - NEAR ALL WEATHER

**All-Weather Surveillance and Targeting Sensor**
- ALL WEATHER
- PRECISION SENSING

**SOG review**
- IOR 1
- IOR 2
<table>
<thead>
<tr>
<th>CSOS Characteristics</th>
<th>and</th>
<th>Software Benefits (relative to hardware)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Many component systems and contractors with wide variety of users and usage scenarios—including legacy systems</td>
<td>• Ease of accommodating many combinations of options</td>
<td></td>
</tr>
<tr>
<td>• Need to rapidly accommodate frequent changes in missions, environment, technology, and interoperating systems</td>
<td>• Rapidly adaptable</td>
<td>• Ease of tailoring various system and CSOS versions</td>
</tr>
<tr>
<td>• Need for early capabilities</td>
<td>• Flexibility to accommodate concurrent and incremental development</td>
<td>• Rapidly upgradeable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Near-free COTS technology upgrades</td>
</tr>
</tbody>
</table>
CSOS Software Benefits, Risks, and Strategies

- Accommodating many combinations of options
  - Development speed; integration; cross-system KPP’s
- Accommodating many combinations of systems and contractors
  - Subcontractor specifications, incompatibilities, change management
- Rapid tailoring and upgrade of many combinations of options
  - Version control and synchronous upgrade propagation
- Flexibility, rapid adaptability, incremental development
  - Subcontractor chain increment synchronization; requirements and architecture volatility
- Near-free COTS technology upgrades
  - COTS upgrade synchronization; obsolescence; subcontractor COTS management

- Compound risks
How Soon to Define Subcontractor Interfaces?

Risk exposure $RE = \text{Prob}(\text{Loss}) \times \text{Size}(\text{Loss})$

-Loss due to rework delays

$RE = P(L) \times S(L)$

Many interface defects: high $P(L)$
Critical IF defects: high $S(L)$

Few IF defects: low $P(L)$
Minor IF defects: low $S(L)$

Time spent defining & validating architecture
How Soon to Define Subcontractor Interfaces?

- Loss due to rework delays
- Loss due to late subcontract startups

\[ RE = P(L) \times S(L) \]

- Few delays: low \(P(L)\), short delays: low \(S(L)\)
- Many delays: high \(P(L)\), long delays: high \(S(L)\)
- Many interface defects: high \(P(L)\), critical IF defects: high \(S(L)\)
- Few interface defects: low \(P(L)\), minor IF defects: low \(S(L)\)
How Soon to Define Subcontractor Interfaces?

- Sum of Risk Exposures

\[ RE = P(L) \times S(L) \]

- Many interface defects: high P(L)
- Critical IF defects: high S(L)
- Few delays: low P(L)
- Short delays: low S(L)
- Few IF defects: low P(L)
- Minor IF defects: low S(L)
- Many delays: high P(L)
- Long delays: high S(L)

Time spent defining & validating architecture
How Soon to Define Subcontractor Interfaces?

- Very Many Subcontractors

$$RE = P(L) \times S(L)$$

Higher $P(L)$, $S(L)$: many more IF’s

Mainstream Sweet Spot

High-Q Sweet Spot

Time spent defining & validating architecture
How Much Architecting is Enough: A COCOMO II Analysis


Percent of Project Schedule Devoted to Initial Architecture and Risk Resolution

Added Schedule Devoted to Rework (COCOMO II RESL factor)

Total % Added Schedule

Percent of Time Added

Percent of Time Added for Architecture and Risk Resolution
Conclusions

• Future systems need to balance discipline and agility
  – Need both high dependability and high adaptability

• Can interpret CMMI in two ways
  – Rigorous, heavyweight, one-size fits-all standard
  – Risk-driven combination of discipline and agility

• Need ability to accommodate agile methods
  – Liberal as well as literal compliance interpretations
  – Risk-driven content of processes and artifacts
  – Outreach to commercial sector