Agile Software Development in Defense Acquisition – A Mission Assurance Perspective

Dr. Peter Hantos
The Aerospace Corporation

Computers and Software Division/Software Engineering Subdivision
Tuesday, April 9, 2013
Acknowledgements

• This work would not have been possible without the support of the following people of The Aerospace Corporation
  – Asya Campbell
  – Suellen Eslinger
  – B. Zane Faught
  – Leslie J. Holloway
  – William S. Macaulay

• Special thanks
  – Steven Kropp, Florida Department of Economic Opportunity, Labor Market Statistics Center

• Funding Source
  • The Aerospace Corporation’s Aerospace Technical Investment Program (ATIP,) Software Acquisition Long Term Capability Development (LTCD) Project
Background

• Emergence of new buzzwords in software development
  – Competitive pressures of the 1990s forced software companies to reexamine their development processes and adopt radical approaches. As a result, the industry has been flooded with buzzwords like “internet time,” “extreme,” and “agile,” just to mention a few

• Management buzzwords have been flooding over the past 30 years…
  – There has been a “bandwagon effect” of popular management movements such as total quality management (TQM), management by objectives, reinventing government, reengineering, the balanced scorecard, lean, and Six Sigma®. However,
    • companies that claimed excellence on the basis of these practices later turned out to be mediocre or outright failure [Paparone 2009]
  – Consequently, a relatively recent, interesting recommendation to the Pentagon brass: “Stay away from management bestsellers…” [Erwin 2009]
    • Despite of Ms. Erwin’s advice, Pentagon brass does not seem to be able to stay away from management bestsellers after all 😛

* Six Sigma has been registered in the U.S. Patent and Trademark Office by Motorola
Agility in Defense

• The warfighter perspective
  – There is a confusion about the need for systems enabling war-fighter agility vs. the need for agile acquisition of weapon systems
  • No argument about the value of war-fighter agility. However,
    – War-fighter agility can be primarily supported via weapons design and flexible architecture
    – Faster access to new weapons is not always the right solution
    – The trade-off between faster access and features is promoted, but the underlying, hidden quality concessions are always controversial and the associated decisions are very difficult

• The acquisition perspective
  – There are essential concerns that need to be clarified and answered
  • To what extent would agile software development contribute to the achievement of agile acquisition of weapon systems?
  • How is fast procurement different from agile acquisition?
  • Under what circumstances is agile software development acceptable or even desirable for weapon systems acquisition?

For operational responsiveness we need “agile products” and not “agile processes”
Determines required capabilities ("Requirements")

Provides funding

Controls implementation, flow of funding

Legend:
- **DOD**: Department of Defense
- **JCIDS**: Joint Capabilities Integration & Development System
- **JROC**: Joint Requirements Oversight Council
- **OSD**: Office of the Secretary of Defense
- **PPBE**: Planning, Programming, Budgeting & Execution
- **R**: Performance & “Time to Need” Requirements
- **S**: Allocated Funding
Agile software development affects only a small fragment of the acquisition system.
Key Stakeholders in the Big “A” Acquisition Process

Note how removed development is from the actual user and customer
The Risk Pendulum – Who is Going to Bear the Risks?

<table>
<thead>
<tr>
<th>Basic Funding Patterns*</th>
<th>Cost-based</th>
<th>Time-based</th>
<th>Fixed Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Promise</td>
<td>Best effort</td>
<td>Best effort</td>
<td>Shall deliver</td>
</tr>
<tr>
<td>Cash flow</td>
<td>As incurred</td>
<td>As incurred</td>
<td>On delivery of item</td>
</tr>
<tr>
<td>Customer control</td>
<td>Maximal</td>
<td>Maximal</td>
<td>Minimal</td>
</tr>
<tr>
<td>Risk to contractor or developer</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Risk to customer or management</td>
<td>High</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>

* Note that these patterns have their formal, contracting equivalents and variations in the Federal Acquisition Regulation (FAR)

The interesting paradox is that despite higher customer control - which, perceived to drive down risk - cost-based and time-based patterns are still risky…
Agile Time-box Life Cycle Concerns

“Traditional”
Iterative-Incremental Development (IID)
Content (Requirements) Driven

Factors to be compared

<table>
<thead>
<tr>
<th></th>
<th>IID</th>
<th>Time-box</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration/Increment duration</td>
<td>varying</td>
<td>set</td>
</tr>
<tr>
<td>Iteration content in the context of an increment</td>
<td>Planned up-front</td>
<td>Not planned up-front</td>
</tr>
<tr>
<td>Difficulty of iteration planning</td>
<td>moderate</td>
<td>easy</td>
</tr>
<tr>
<td>Difficulty of increment planning</td>
<td>difficult</td>
<td>difficult</td>
</tr>
<tr>
<td>Micro-estimation fidelity</td>
<td>moderate</td>
<td>higher than IID</td>
</tr>
<tr>
<td>Macro-estimation fidelity</td>
<td>high</td>
<td>low</td>
</tr>
<tr>
<td>Naturally fitting contracting pattern</td>
<td>cost-based</td>
<td>time-based</td>
</tr>
</tbody>
</table>

“New”
Agile Time-box*
Calendar (“Clock”) Driven

* Note that it needs to be compared to IID and not the Waterfall

Red flag marks the customers’ primary concerns
Main Time-box Risk: Violating the Iron Triangle Principle

- What is the Iron Triangle Principle?
  - *One can only fix two of the cost, requirements, and schedule triad; any attempt to pre-determine all three results in an non-executable plan*

- (Unfortunately,) typical scenario: the number of time-boxes, like sprints in Scrum, and ultimately the launch-date are pre-determined
  - *At that point cost (i.e., manpower loading,) is also pre-determined*

- Risks:
  - *No guarantees that all desired requirements can be fully implemented; In fact, the adaptive process will successively defer and drop requirements*
  - *Since the process is already over-constrained, delivering predictable quality is also a challenge*
Examining Agile Software Development Values

• Agile software development values from the Agile Manifesto*
  – Individuals and interactions over processes and tools
  – Working software over comprehensive documentation
  – Customer collaboration over contract negotiation
  – Responding to change over following a plan

• During the analysis the following, typical figures were considered
  – Space vehicle (embedded, large, including bus software and payload(s)):
    • ~512 Thousand Delivered Source Instructions (KDSI)
  – Ground systems:
    • Space Shuttle software ~2,000 KDSI
    • Satellite control systems software ~4,700 KDSI
  – The development of 512 KDSI would require roughly a 6,420 person-month effort, spreading over 41 months, involving ~157 full-time equivalent software personnel

* [Agile 2001]
Individuals and Interactions Over Processes and Tools

- Agile proponents believe that one should only declare and rely on **practices** instead of **processes** to increase the agility of development
  - A **practice** usually refers to an individual activity while a **process** is an aggregate structure of multiple activities
  - However, relying only on fluidly defined practices certainly ensures a greater level of flexibility, but this flexibility comes with unavoidable ambiguities and may create tension amongst the stakeholders

- **Lean** - The term “lean production” was coined in the 80’s [Krafcik 1988]
  - The underlying ideas represent the so-called lean thinking about processes

- Current (mis)use of the term
  - Lean is a popular buzz-word for general cost cutting efforts
  - Lean may be used in conjunction with Six Sigma®, another, also manufacturing-rooted, process improvement method (“Lean Six Sigma”)

- Mission assurance exposure
  - Difficult to sort out what is really important due to stakeholder conflicts
  - Lean Six Sigma rule of thumb is that usually only 5% of total process cycle time adds value to outputs; **mission assurance is valued low by developers**
### Major Areas in a Typical Software Development Standard*

<table>
<thead>
<tr>
<th>System and Software Architecture</th>
<th>Transition to Operations and Maintenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human Systems Integration</td>
<td>Software Configuration Management</td>
</tr>
<tr>
<td>Interoperability and Standardization</td>
<td>SW Peer Review/Product Evaluation</td>
</tr>
<tr>
<td>Reliability, Safety, Information Assurance</td>
<td>SW Quality Assurance</td>
</tr>
<tr>
<td>Project Planning and Oversight</td>
<td>Corrective Action</td>
</tr>
<tr>
<td>SW Development Environment</td>
<td>Joint Technical and Management Reviews</td>
</tr>
<tr>
<td>System Requirements Analysis</td>
<td>Risk Management</td>
</tr>
<tr>
<td>SW Requirements Analysis</td>
<td>SW Management Indicators (Metrics)</td>
</tr>
<tr>
<td>SW Design</td>
<td>Security and Privacy</td>
</tr>
<tr>
<td>SW Implementation and Unit Testing</td>
<td>Subcontractor Management</td>
</tr>
<tr>
<td>Unit Integration and Testing</td>
<td>Interface with Software IV&amp;V Agents</td>
</tr>
<tr>
<td>SW Qualification Testing</td>
<td></td>
</tr>
</tbody>
</table>

The “lean” question: Which ones do not add value? Which ones to get rid off?

* Source: [Adams 2005]
What Does My Dentist Know About Mission Assurance?

Sign in my dentist’s office:
“Brush only those teeth you wish to keep…”
Individuals and Interactions Over Processes and Tools-2

• Tools
  – The typical 3-4 year long development and a minimum 5-10 year long operation and sustainment for a space vehicle require strong tools support
    • Development must be based on an architecture-first approach
      – Architecture modeling artifacts need to be documented with rigorous notation and handled with appropriate (preferably visual) modeling tools
      – The dynamics of concurrent workflows by different teams working on shared artifacts necessitates a rigorously controlled change management environment
    • Tools are also necessary to keep all the engineering information in different formats synchronized and to support bi-directional traceability
      – System requirements, software specifications, design models, source code, executable code, scripts, test cases, test data, etc.
  • True change freedom cannot be realistically achieved without the support of an appropriate, integrated environment [Royce 1998]

Even if the labor force would be stable, tacit knowledge sharing is not sufficient
Work Force Volatility

- The work force in the information sector is very volatile* even during recessions when the overall, net employment change is lower than average.

<table>
<thead>
<tr>
<th>Periods of Recession**</th>
<th>Information Sector</th>
<th>Federal Sector</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hires</td>
<td>Separations</td>
</tr>
<tr>
<td>2001-2002</td>
<td>36.5%</td>
<td>43.3%</td>
</tr>
<tr>
<td>2008-2010</td>
<td>23.7%</td>
<td>27.8%</td>
</tr>
</tbody>
</table>

- How to interpret the data
  - Unfortunately, the Bureau of Labor and Statistics (BLS) is not collecting the exact data we would be interested, i.e., programming-related turnover in the defense industry.
  - However, one can see that the turnover rate is quite high even in the federal sector, which is considered less volatile than the private sectors.
  - Additionally, the BLS database does not track internal, company turnover.

* Source: Bureau of Labor and Statistics database; ** [Bruyere 2011]
Working Software Over Comprehensive Documentation

- This value statement is not about the merits of documentation but about interim progress assessment
  - The idea is not new; modern processes are already using the demonstration-based approach to assess intermediate artifacts [Royce 1998]
- The concern regarding the agile approach is the impact on the customer
  - Principle #8 of the Agile Manifesto represents a strong imposition on the customer: “Sponsors, developers, and users maintain a constant pace”. Unfortunately, maintaining such pace is not feasible on large projects
  - Issues:
    - Embedding users/customers with the necessary expertise into every team
    - Customers need to approve technical decisions in relatively short cycles
    - Coordination of a network of government customer representatives

In short, this agile value does not scale up in a large project
Customer Collaboration Over Contract Negotiation

• As it was shown, actual users and customers are far removed from the development organization
  – JROC, DOD, and Congress are high-inertia organizations with complex, bureaucratic processes for interaction
  – These are stakeholders with different political weights; building true collaborative relationships is difficult if not impossible

• With the current, rigid “upstream” relationship the flexibility of the surrogate customer is very limited
  – Agile development will not improve the agility of the acquisition process; in fact, insisting on developer agility may exacerbate the existing tensions

• It is an unfortunate fact of life that when things do not go well, collaborative resolution becomes less and less feasible
  – The stakeholders have their own, different risk perspectives and motivations and their differences cannot be easily reconciled via voluntary actions

• You would not remodel your kitchen without a detailed contract, so why would you deemphasize the importance of contracts for billion-dollar weapon system acquisitions?
  – Well, actually we did it in the 1990s, it was called “Acquisition Reform”
Responding to Change Over Following a Plan

• The essential motivation is the recognition that solution details to complex problems cannot be successfully determined up-front
  – This is not a new idea; that’s why modern, but pre-agile software development methods are adaptive and use iterative/incremental processes. How requirements risks are handled in modern methods:
    • On micro-level: The emphasis during the planning of iterations is on facilitating a successively refined understanding of requirements
    • On macro-level: New or changing requirements are expected to be handled via evolutionary acquisition and development strategies
• Agile principle #2 (“Welcoming changing requirements”) is directly flowing from the discussed agile value statement
  – Unfortunately, this is a disingenuous statement, to say the least
    • In reality, everybody likes to work on stable grounds with clear, unchanging expectations; Don’t you?
• However, if anyone still has doubts, listen to Yogi Berra:

  “If you don't know where you are going, you will wind up somewhere else”
**eXtreme Programming (XP) Practices**

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning game, on-site customer</td>
<td>Burdensome customer participation</td>
</tr>
<tr>
<td>Small releases</td>
<td>Implied customer responsibility for validation</td>
</tr>
<tr>
<td>Metaphor</td>
<td>None</td>
</tr>
<tr>
<td>Simple design</td>
<td>Rigid use of the YAGNI principle</td>
</tr>
<tr>
<td>Continuous integration &amp; testing</td>
<td>Implied customer responsibility for success</td>
</tr>
<tr>
<td>Refactoring</td>
<td>Practice is not a replacement for proper architecting</td>
</tr>
<tr>
<td>Pair Programming</td>
<td>Practice is not a replacement for formal inspections</td>
</tr>
<tr>
<td>Collective code ownership</td>
<td>Practice does not scale up</td>
</tr>
<tr>
<td>40-hour work week</td>
<td>None</td>
</tr>
<tr>
<td>Coding standards</td>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Practice Changes in 2004**</th>
<th># of Practices</th>
</tr>
</thead>
<tbody>
<tr>
<td>No change</td>
<td>4</td>
</tr>
<tr>
<td>Eliminated or deemphasized</td>
<td>4</td>
</tr>
<tr>
<td>Renamed or changed</td>
<td>2</td>
</tr>
<tr>
<td>New practice introduced</td>
<td>11</td>
</tr>
</tbody>
</table>

XP is a good example of how difficult it is to pin-down specific agile practices

* [Beck 2000], **[Beck 2004]
The Main Exposure to Mission Success: Software Failures

• Deterministic vs. random failures*
  – Deterministic (“Bohrbugs”)
    • Repeatable
    • Traceable to root cause(s) under control of developer or user
      – Deterministic failures can be prevented through the use of a disciplined development process
  – Random (“Heisenbugs”)
    • Not repeatable; many such failures can be fixed by reset
    • Caused by transient states of the software (timing, buffer overflows, queues, memory leaks, etc.)
    • Indistinguishable from single event upsets (SEUs,) power fluctuations or hardware timing errors

• Recoverable vs. non-recoverable software failures (space example)
  – Recoverable software failures are events that occur in spacecraft processors that cause a loss or performance degradation of the bus or payload, which can be restored via either onboard or ground corrective actions

Even application of a disciplined development process itself is not a guarantee for preventing random failures or mitigating recoverable failures

* Courtesy of Myron Hecht [Guarro 2008]
Agile Software Development from a Commercial Perspective

• Using agile software development is a business strategy, based on a particular value proposition

• Choosing a development method should be based on the home grounds of the organization and project, characterized by the following factors*
  – Project size (Expressed by the number of development personnel involved)
  – Criticality (Loss due to impact of defects)
  – Level of software understanding in development personnel
  – Dynamism (%Requirements-change/month)
  – Culture (Thriving on chaos vs. preferring order)

• When these factors are considered, agile software development certainly seems to be a promising approach for small, low criticality projects with rapidly changing requirements, where the organization’s culture embraces high degrees of freedom, and the developers are highly experienced

Unfortunately, the applicability of these methods outside of the above described home ground has not yet been proven

* [Boehm 2004]
Top 12 Commercial Concerns in 2010 About Adopting Agile Software Development*

- Loss of management control
- Lack of upfront planning
- Management opposed to change
- Lack of documentation
- Lack of predictability
- Lack of engineering discipline
- Development team opposed to change
- [Lack of] engineering talent
- Inability to scale
- Regulatory compliance
- Reduced software quality
- Other

It is not on the list, but one of the main concerns should be lack of consistent metrics and reliable data to verify if any of the original objectives that triggered agile adoption in a company have been met.

* [VersionOne 2010]
Defense Acquisition - How do We Know that it is Broken?

- Defense Acquisition Performance Assessment (DAPA) summary in 2006
  - “As early as 1971 it has been identified that [defense] acquisition processes have significant shortcomings leading to loss of confidence by congress and the defense community”
  - “Many improvements to the DOD’s acquisition system have been made as a result of past reviews … However, the ability to deliver operational performance of major systems within predicted cost and schedule has not improved over the last 20 years”
Selected* DAPA Recommendations in 2006

- Replace the Joint Capability Integration Development System (JCIDS) with a new, two-year recurring planning process based on the 15-year extended plans submitted by combatant commands
- Stabilize the Planning, Programming, Budgeting, and Execution (PPBE) process
- Introduce a new requirements process with 2-year duration
- Establish a distinct, stable Program Funding Account
- Increase program predictability
- Program all accounts to a high, 80/20 confidence level
- Establish very early a realistic capability delivery rate
- Establish very early all test plans
  - Complete Test & Evaluation Management Plan (TEMP) and Initial Operational Testing & Evaluation Plan (IOT&EP) prior Milestone B

Clearly, the DAPA panel valued stability and predictability as opposed to agility

* There were more recommendations but those did not have potential agile implications
Acquisition Problems Identified in 2011 by the Government Accountability Office (GAO)*

- **Alternatives not considered**
  - Clearly, no relationship to agile development

- **Funding unstable**
  - Agile development might be helpful in dealing with unstable funding, but only at the price of delaying or dropping requirements

- **Inadequate contracting strategy**
  - The report is referring to the failure of Total System Performance Responsibility (TSPR) and lack of evolutionary strategies in certain acquisitions; neither has agile software development implications

- **Inadequate contractor oversight**
  - This concern is also related TSPR; While some agile principles would embed more government personnel in the development process, due to lack of contracting rigor this involvement would be costly and ineffective
  - Also, increasing the acquisition work force has been suggested, but in the current climate of drastic budget cuts it is not feasible
    - $148–$178B DOD cuts are planned between 2012 and 2016**

* [Chaplain 2011], **[Weisgerber 2011]
More, GAO-identified Acquisition Problems

- **Optimistic cost and schedule estimates**
  - The operative word seems to be “optimistic”, which has nothing to do with the details of development methodologies. Additionally, due to the difficulties with macro-estimation in agile development, one can expect further dissatisfaction with the accuracy of cost and schedule estimates.

- **Requirements unstable**
  - Due to its adaptive nature agile development is supposed to help with handling unstable requirements. However, regardless of the implemented agile project management strategy, volatile requirements will yield inaccurate cost and schedule estimates, ultimately resulting in customer dissatisfaction.

- **Software needs poorly understood**
  - This is also a requirements and early architecting issue. Again, selected agile development practices do facilitate the gradual, more effective discovery of software-level requirements, but still, software estimates, particularly the early ones, will be grossly inaccurate.

- **Technology immature**
  - This has not been a software issue on the reviewed acquisitions.

Agility and agile software development still do not seem to be the answers.
Section 804

• A recent directive by Congress to shape the future of defense acquisitions is

• The directive targets the streamlining of Defense Business System (DBS) Acquisitions

Weapon systems and National Security Systems acquisitions are still must be carried out according to DOD 5000.02
Does Section 804 Direct the Incorporation of Agile Methodologies in DOD Software Acquisition?

• Why are we even asking the question?
  – *Quotes from the November 14-15, 2011 National Defense Industry Association (NDIA) Agile Scrum Workshop’s invitation*
    • “The law [Section 804] directs the incorporation of Agile methodologies in DOD software acquisition … Agile cannot fail. Unequivocally, Agile cannot fail.”

• However what Section 804 actually requires is an acquisition process for DBS acquisitions only, with the following characteristics
  – *Early and continual involvement of the user*
  – *Multiple, rapidly executed increments or releases of capability*
  – *Early, successive prototyping to support an evolutionary approach*
  – *Modular open systems approach (MOSA)*

• Section 804 requires a new acquisition process but congress cannot (and should not) legislate a software development process

The wording is indeed inspired by agile ideas, but the connection to specific agile software development practices is very weak or nonexistent
There is No Silver Bullet – Particularly in Software Engineering! 😊

Everybody quotes Fred Brooks* but still, we keep trying to make one…

* [Brooks 1987]
More on Silver Bullets - Spiral Development vs. Agile Methods

• Spiral development is a precedence for a fiasco when the government tried to mandate a development process
  – *Spiral development, first introduced in 1988, is one of the most complex and most powerful adaptive process models*
  – *It looked so promising that in 2003 it was declared as the DOD’s preferred development strategy in the DOD 5000.2 instructions*
  – *However, people struggled with the implementation of the process and the lack of success culminated in the elimination of the spiral model from the DOD 5000.02 instructions in 2008*

• Agile development is essentially an “extreme” adaptive process model
  – *Apparently an adaptive process model is being pushed blindly again, without any attempts to understand why spiral development failed and what it takes to manage major acquisitions with adaptive approaches*

“It is déjà vu all over again”
~~~ Yogi Berra
What May the Future Bring?

• There is numerous evidence that the government wants to do away with cost-plus contracts and wants to pursue fixed price contracts*
  – Presidential memorandum
  – Office of Management and Budget (OMB) memorandum
  – Revised procurement regulations and additional barriers to cost-plus contracts

However, remember the Iron Triangle? Pursuing agile development and fixed price contracts at the same time is not feasible

* Source: [Chierichella 2012]
Conclusions

• What Acquisition Program Office personnel needs to do
  – *Continuously educate itself on the emerging development methods*
  – *In the contracting phase must insist on the use of robust development standards*

• The government should not settle for vague references to agile programming; must insist on a detailed Software Development Plan (SDP) that fully characterizes all planned life cycles, their internal relationships, and the planned implementation details of all life cycle processes and associated activities
  – *During contract monitoring must implement effective mission assurance*

• Mission assurance is essentially an ingrained instrumentation of the development process; a necessity that must not be allowed to be viewed by the development organization as a “nice to have” negotiable feature

"The temptation to 'cut corners,' even in the name of being efficient or 'expedient,' is ever-present, especially in a global business that is economically unforgiving...
That is why 'getting it right' must be a 24/7 commitment."
~ Dr. Wanda Austin, President and CEO, The Aerospace Corporation
## Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>APO</td>
<td>Acquisition Program Office</td>
</tr>
<tr>
<td>ATIP</td>
<td>Aerospace Technology Program</td>
</tr>
<tr>
<td>BDUF</td>
<td>Big Design Up Front</td>
</tr>
<tr>
<td>BLS</td>
<td>Bureau of Labor &amp; Statistics</td>
</tr>
<tr>
<td>CEO</td>
<td>Chief Executive Officer</td>
</tr>
<tr>
<td>DAPA</td>
<td>Defense Acquisition Performance Assessment</td>
</tr>
<tr>
<td>DBS</td>
<td>Defense Business Systems</td>
</tr>
<tr>
<td>DoD</td>
<td>Department of Defense</td>
</tr>
<tr>
<td>GAO</td>
<td>General Accountability Office</td>
</tr>
<tr>
<td>IID</td>
<td>Iterative-Incremental Development</td>
</tr>
<tr>
<td>IOT&amp;EP</td>
<td>Initial Operational Testing &amp; Evaluation Plan</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>IV&amp;V</td>
<td>Independent Verification &amp; Validation</td>
</tr>
<tr>
<td>JCIDS</td>
<td>Joint Capabilities Integration &amp; Development System</td>
</tr>
<tr>
<td>JROC</td>
<td>Joint Requirements Oversight Council</td>
</tr>
<tr>
<td>KDSI</td>
<td>Thousand Delivered Source Instructions</td>
</tr>
<tr>
<td>LTCD</td>
<td>Long Term Capability Development</td>
</tr>
<tr>
<td>MOSA</td>
<td>Modular Open System Architecture</td>
</tr>
<tr>
<td>NDIA</td>
<td>National Defense Industry Association</td>
</tr>
<tr>
<td>OMB</td>
<td>Office of Management &amp; Budgeting</td>
</tr>
<tr>
<td>OSD</td>
<td>Office of the Secretary of Defense</td>
</tr>
<tr>
<td>PPBE</td>
<td>Planning, Programming, Budgeting &amp; Execution</td>
</tr>
<tr>
<td>SDP</td>
<td>Software Development Plan</td>
</tr>
<tr>
<td>SEU</td>
<td>Single Event Upset</td>
</tr>
<tr>
<td>SW</td>
<td>Software</td>
</tr>
<tr>
<td>TEMP</td>
<td>Test &amp; Evaluation Management Plan</td>
</tr>
<tr>
<td>TQM</td>
<td>Total Quality Management</td>
</tr>
<tr>
<td>TSPR</td>
<td>Total System Performance Responsibility</td>
</tr>
<tr>
<td>XP</td>
<td>eXtreme Programming</td>
</tr>
<tr>
<td>YAGNI</td>
<td>You Aren’t Gonna Need It</td>
</tr>
</tbody>
</table>
## References

<table>
<thead>
<tr>
<th>Year</th>
<th>Citation</th>
</tr>
</thead>
</table>
Use of Trademarks, Service Marks and Trade Names

Use of any trademarks in this material is not intended in any way to infringe on the rights of the trademark holder. All trademarks, service marks, and trade names are the property of their respective owners.

The clip art on slide 21 is courtesy of Animation Library
The clip art on slide 8 is courtesy of Florida’s Educational Clearing House
The clip art on slide 14 is courtesy of PicGifs
The clip art on slide 30 is courtesy of Wallpapers4Desktop
The clip art of a witch on slide 32 is courtesy of All-free-download
All other clip arts on slide 32 are owned by The Aerospace Corporation