Engineering, Operations & Technology
Boeing Research & Technology

Boeing
When Agile Software Development and Software Architecture Collide

Don O'Connell

Report Documentation Page	*Form Approved OMB No. 0704-0188*

1. REPORT DATE **MAY 2011**	2. REPORT TYPE	3. DATES COVERED **00-00-2011 to 00-00-2011**

4. TITLE AND SUBTITLE **When Agile Software Development and Software Architecture Collide**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Boeing,Engineering, Operations & Technology,Chicago,IL,60606**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Presented at the 23rd Systems and Software Technology Conference (SSTC), 16-19 May 2011, Salt Lake City, UT. Sponsored in part by the USAF. U.S. Government or Federal Rights License

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**26**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Bio

Don O’Connell is a Technical Fellow in SW architecture for the Boeing Company. He has been with Boeing for 27 years, and has worked on many dozens of projects, including domains of airplanes, satellites, weapon systems, helicopters, windmills and many more.

Don is on the team that develops agile processes for SW and system engineering use, and he leads the team that defines SW architecture processes as applied to agile SW development.

Contents

What is Agile SW Development?

Agile Manifesto – we have all seen these

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

Different types of agile

- Scrum, extreme programming, feature driven development, others

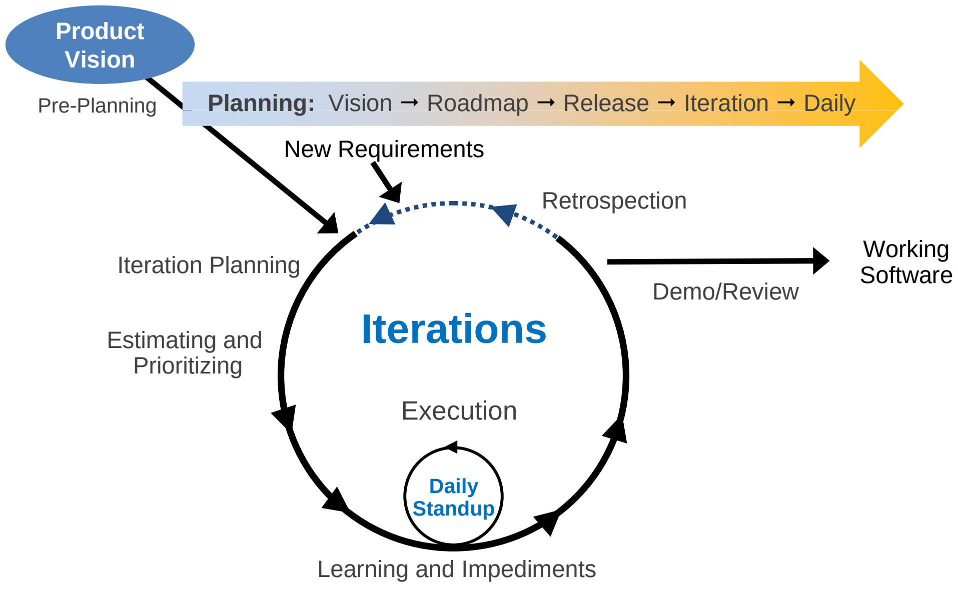
Agile Practices

- **Close customer collaboration**
- **Daily stand-up meetings**
- **Continuous integration**
- **Automated testing**
- **Planning and estimating**

- **Short iterations**
- **Test-driven development**
- **Prioritized requirements/reviews**
- **Product demonstrations**
- **Self-organized teams**

Agile SW Benefits

- Boeing builds upon Scrum agile practices
- Agile Benefits
  - **End result, higher quality, faster to market, adaptable development**
  - Higher quality and faster if the environment is changing
  - **Adaptive vs. Planned** – Chaotic systems are inherently unpredictable, often software development is seen as a chaotic process.
  - **Team involvement** , The team chooses how much work can be accomplished,
  - **Prioritizes work** based on highest value



Boeing Agile Scrum

Agile SW Development, My Take

- My take on this
  - Many practices of agile have been practiced for years, such as continuous integration, desktop integration, automated testing, collocation, etc.
  - Agile gathers these best practices, then adds to them
  - Team power, team work are pronounced in agile, power to the people
  - Completing high value, small chunks of software first, then moving on to the next ones

Power to the people

Agile SW Development, What is Done

- What is done
  - Done is when a chunk of SW is sold off
  - What about rework, refactoring?
    - Suppose to be rare
    - Just get it done again
  - All done is when “all” chunks are sold off
    - Or when the money is gone
    - Or when people all leave
  - How is this different than a “not agile” project
    - It isn’t, done when money is gone or people leave
    - It is different if the value of the work is greater

Software Architecture Defined

- SEI definition
  - **set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both [Clements, et al]**

- Phrased in another way
  - Architecture requirements
    - Quality attributes (performance, availability, safety, security, evolvability, adaptability, etc)
    - Business drivers, ways to make money, adhere to constraints
  - Architecture description
    - Overall decisions about how the system will be constructed how it will run, how it will be deployed
    - Tactics and strategies to accomplish the architecture requirements
    - Provides guidance for the design, code, integration of the system

Software Architecture Benefits

- Force intentional understanding of driving architectural requirements
- Help customer understand what they really need and can really get
- Create distinct architectural products that guide the design and development to achieve those QA
- Think before you build

Software Architecture, What is Done

- Often times, done means the architecture description is done
  - Code isn't done, nor is working software accepted by the customer
  - Customer may accept arch description to consider it done
- Sometimes it is a model that is done
  - But a model != architecture
- Rework and refactoring
  - Called architecture maintenance
  - The architecture should not change as code is designed, coded, tested
  - When arch change does happen, done means redeliver the description

How agile and architecture don’t coexist

- Classic take by the agile community
  - Architecture evolves or sprouts from iterations
  - Little bit of architecture happens each day by everybody
  - Everybody should contribute to the architecture
  - Scrum does not include distinct architecture phases, roles or products
- Their problems with architecture developers
  - Have to wait too long
  - Ivory tower
  - It will just be wrong anyways

How agile and architecture don’t coexist

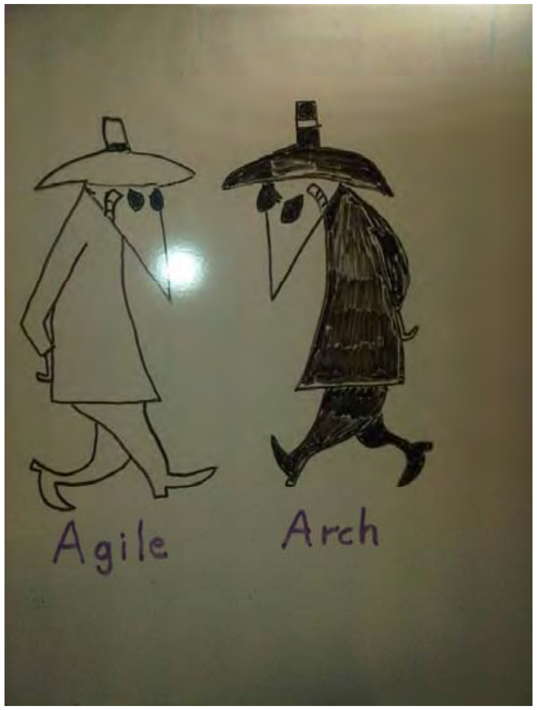
- Classic take by architects
  - Agile won’t scale
    - Large distributed teams can’t do daily standup
    - Rework will increase as time goes on
    - Refactoring will soon dominate all iterations
    - Building a house of cards
  - Agile won’t know or meet QA
    - Upfront effort required to know/understand QA
    - Meeting QA cannot sprout
  - Agile won’t do complex systems
    - Systems won’t integrate without upfront planning, interface work, etc
  - Agile works only if the architecture is already defined

Agile Foundation

Just refactor the Jack

Don’t coexist, my take

- My take on this lack of co-existence
- Agile community
  - Needs to recognize the need for architecture
  - Needs to recognize when architecture ends and design begins
  - Needs to accept that one size of development process does not fit all projects
- Architecture community
  - Needs to recognize their own ivory tower
  - Needs to get something done quickly so everybody is not waiting
  - One size of process does not fit all



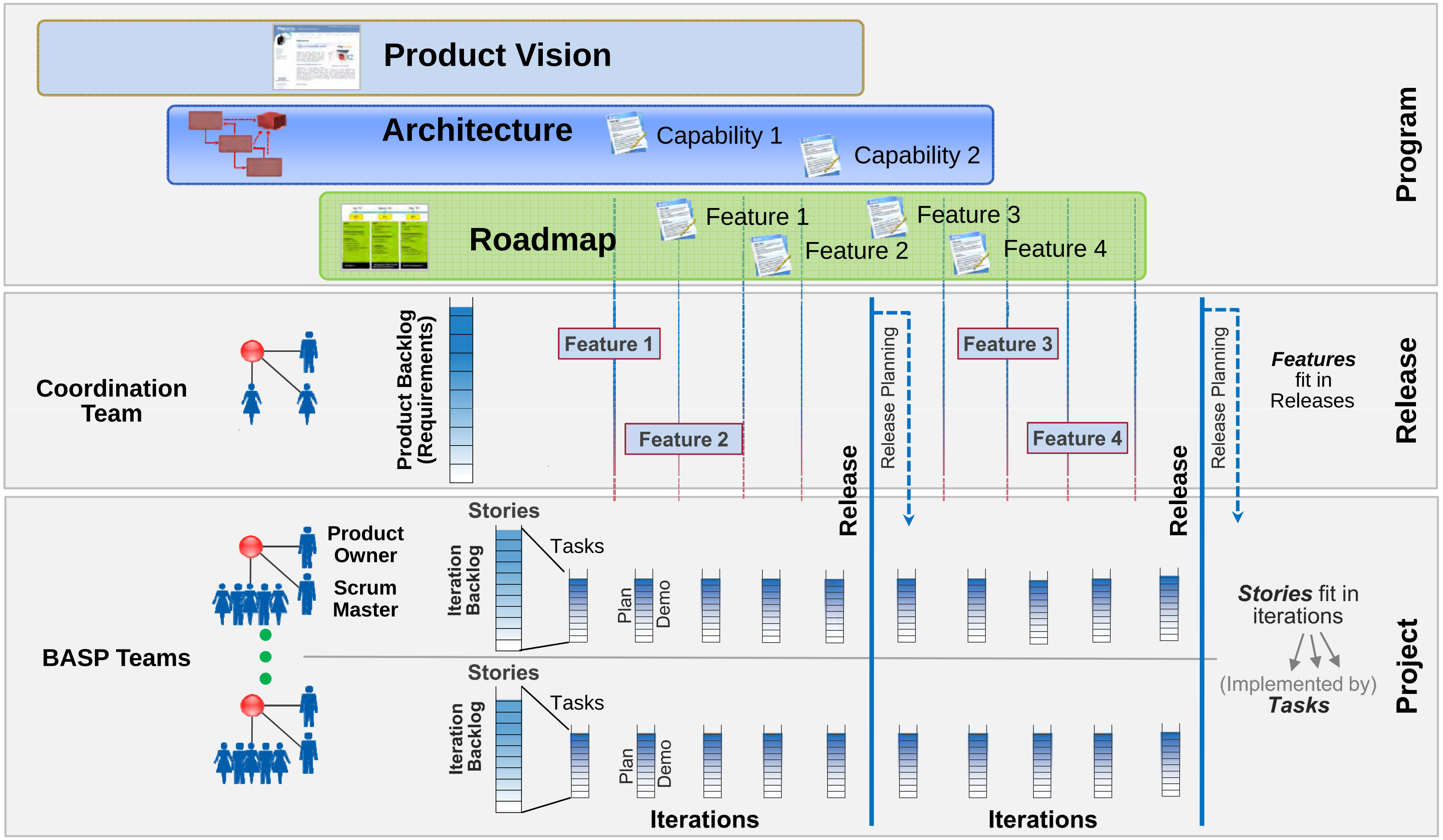
How agile and SW architecture do coexist

- Agile development of architecture
  - What is done, does architecture done count as done?
  - Develop in small chunks, most important chunks first
    - **We have been doing this for years**
      - Iterative development
      - Pick most important aspects first
      - Delay what you can
      - Could do these more, be more agile
  - Enable design, code, integration to start earlier
    - **Without large scale rework**
  - Agile development of arch alone is not worth much

How agile and SW architecture do coexist

- Agile development of system that needs architecture
  - Architect, design, code, integrate, test deliver in small chunks
    - **What chunks? What happens first?**
  - Agile SW designers
    - Work together with architects
  - Stop making those giant requirements specs and architectures that are late and significantly wrong
    - **Nobody intends for this, but, that is how it ends up, sometimes**
  - Now, what is done
    - Arch part done, some chunks are done
    - Eventually arch is done
    - Keep on finishing chunks

Boeing Approach for Agile including SW Architecture



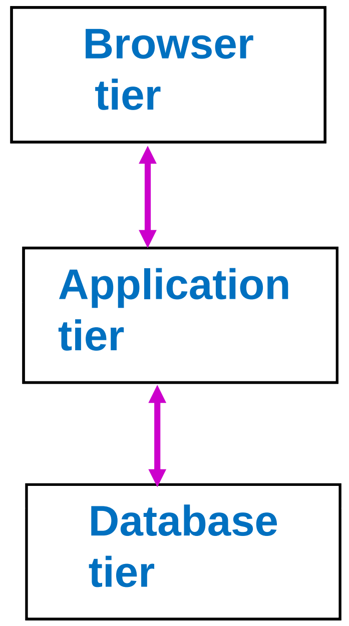
Based on an original diagram by Dean Leffingwell

Examples

- Examples
  - Well known architecture
  - Not well known architecture
  - Requirements well known in advance

Well Known Architecture

- 3 tier web based data delivery system
  - What is the architecture
    - Well, each tier is based on COTS elements, standards are defined, patterns are well known, QA are determined by all those choices, so it will be high availability, open, modular, . So, the arch is really defined by industry norms. There are dozens of books on how to build a system like this given these arch decisions
  - What is agile here.
    - Well, develop small chunks using these COTS, guidance and frameworks. Deliver those chunks to customer.
    - Software only is the deliverable
  - What if
    - **What if the team does not know the technology very well, or understand the arch very well?**
    - **COTS or standards change**



3 tier architecture

Well Known Architecture, Discussion

- What is the problem here
  - Arch decisions
    - Arch decisions may not be the right ones up front
      - Quickly selected COTS
      - Change COTS or change technology
      - Change display platform and framework, causing rework. Significance depends.
    - So, arch still has to be decided before significant SW development takes place
    - But, since it is classic COTS, guidance already exists. Many decisions do not require significant work
    - For those that do not know the technology, get the book
  - Does agile work
    - Sure, works great, in fact this is an ideal place for agile

Not Well Known Architecture

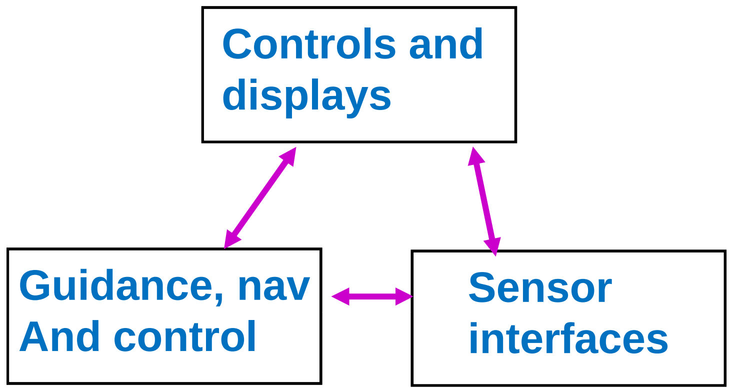
- Safety critical surveillance and weapons system
  - What is the architecture?
    - To be developed, QA are not all known, such as amount of availability, performance, reliability, safety, modifiability, configurability needed, etc. Not a lot of COTS, so patterns do not have books to back them up.
    - What is in and outside of the system, needs further definition.
  - What about agile?
    - Working software is part of the embedded system that includes hardware, missiles
    - What would one agile developer do on day 1?

Not Well Known Architecture, Discussion

- What is the problem here
  - Arch decisions
    - Architecture has to be developed, that will take time
    - What architecture to do first?
    - Software that is built using wrong COTS, wrong frameworks, wrong messaging, wrong etc will have to be redone
  - What about contracted subsystems, how to make them agile. One needs a contract, and scope, and SOW, and requirements, classically. These systems always have subcontracted parts.
  - Does agile work here , To wait or not to wait
    - Working software is part of the embedded system that includes hardware, missiles
      - Delivering small working sub parts of missiles is not practical or desirable
    - After many architecture decisions are made, agile will work, but
      - Customer to deliver to may be the system integration or test team, not the final customer

Well Known Requirements

- Performance intensive, deterministic Avionics system
  - Assumptions
    - **Seasoned veterans have built 10 of these in the past, know exactly what they want and how to do it. Requirements are well known**
    - Architecture is not fully defined, new tools, new technologies, techniques will be used
    - **Architecture needs to be developed**
  - What is agile here
    - **Some arch decisions have to be made before starting agile development**
    - **Again, an embedded system, so delivery of chunks to test team (not final customer)**



Well Known Requirements, Discussion

- What is the problem here
  - Arch decisions
    - Arch still has to be decided before significant SW development takes place
  - Does agile work
    - After some architecture decisions have been made, agile development can start
- What is the value of agile if the requirements do not churn?
  - Reduced differential value of agile over traditional

Another Take

- Creationists vs evolutionists
  - Creationists believe in an architect. They believe that a preconceived and intentional architecture decides the outcome.
    - After all, random lines of code don't just make the right things happen. Primordial goo doesn't just spring to life.
  - Evolutionists believe **it** all just happens, iterates and evolves to where we are today.
    - After all, the architect did not predict everything right. Things that did not evolve died.
  - Nobody knows for sure, both think they are right. Maybe it is a bit of both.
  - Same with agile and arch, both alone are partly right, perhaps together they are more right.

goo

Acronyms

- BASP – Boeing Agile Software Process
- COTS – commercial off the shelf
- Ivory Tower – where architects live
- SEI – software engineering institute
- SOW – statement of work
- SW – software – (not southwest)
- QA – quality attributes