Emerging Technologies for the Enterprise
Philadelphia, Pennsylvania April 8-9, 2010

Airplanes to Application Development

JonKern@comcast.net / Architect & Agile Coach

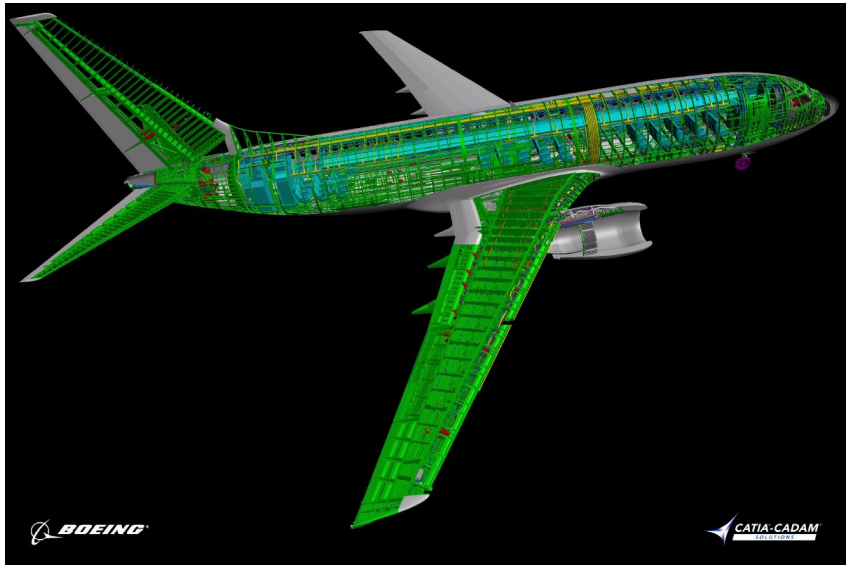
JonKernPA

Overview

- A look at B777 aircraft development project
- Parallels to software development
- What we can learn from product engineering?

Requirements & Stakeholders

- Jan 90 - Early involvement with 8 major airlines
- Mar 90 - Initial requirements
  - X-section ≈ B747
  - 325 passengers
  - Fly-by-wire controls
  - “Glass” cockpit
  - Flexible interior
  - 10% better seat-mile cost
- Oct 90 - United Airlines becomes launch “Beta” Customer
- UA extends the scenarios
  - Requirements change



Lessons:

- Create a coalition of stakeholders
- Gather input from key users
- Understand the business drivers
- Adapt to changing requirements

Development Team

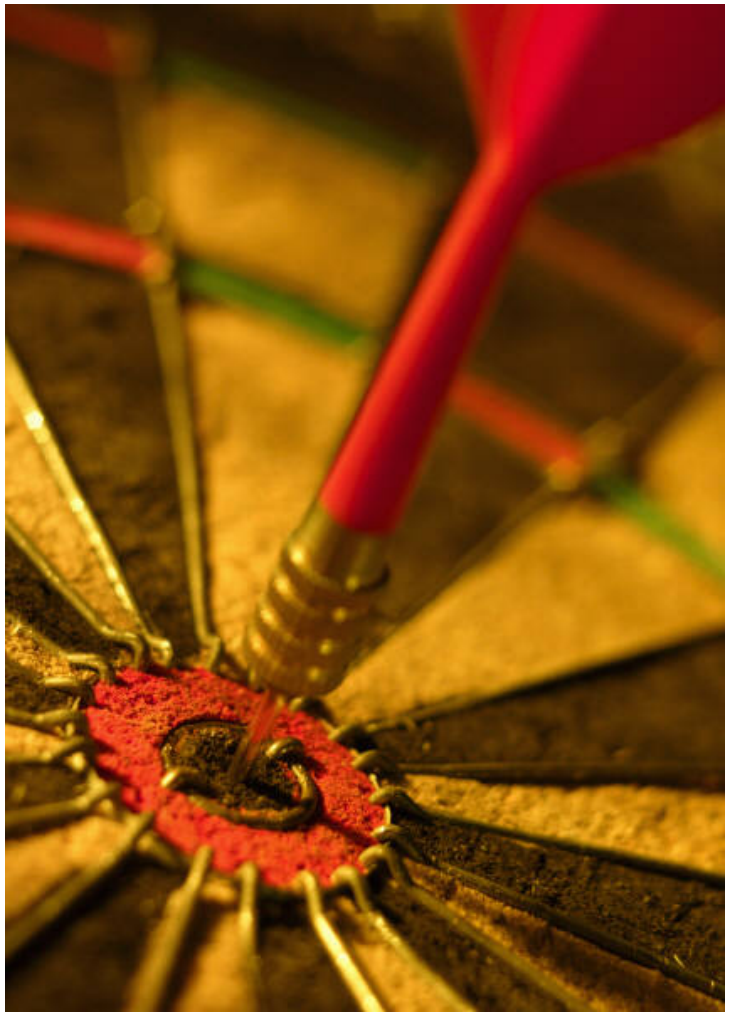
- Team Makeup
  - 240 Design teams
  - Up to 40 members each
- Assigned individual components
  - Pieces of the integrated whole
  - Clear picture of where it fits in
  - Clear feedback loops
  - Prominent time-based integration checkpoints
- Jan 93–Formally dubbed “B777”
- UA engineers join other airline “customer” teams

Lessons:

- Divide and conquer
- Maintain big picture, but allow creativity
- Include customers on development team
- Develop clear communication
- Let the computer models assist in that communication
- Have integration strategy to ensure no team strays (too far)

Trade-offs to Find "Sweet Spot"

- Weight *versus* Cost
- Teams allocated goals at outset
- Difficult and imprecise
- All systems interdependent
- How best to allow change?
  - Centralized authority?
  - Decentralization is best!
- Decision support rule
  - 1lb savings worth $300
- 5000 engineers able to make decisions



Lessons:

- Team empowered to be creative and responsive
- Decision still controlled within business parameters
- No decision-making delays due to hierarchical sign-off ceremonies for every little request
- Continuous build system enables gathering measurements by which goals are evaluated

- Regarding “cost metrics” for development, this is challenging for most software projects

Test the Process

- Risk mitigation is a key to development. For example:
- Test the modeling process
  - Risk: CAD methodology – will it work?
  - Built physical mock-up
  - Used all design techniques
  - Evaluated the end-product
- Result was an astounding success!
- Canceled all other component tests of the ***process***

Lessons:

- Test the process
- Prove you can deliver working results
- Gain faith in the automated “transforms” from the design “meta data” and the end results of a “build”
- Work in small batches
- Enable & consume rapid feedback
- Change the plan as needed based on findings

100% Computer-Aided Design

- First commercial aircraft example
- 3D CAD drawings for all design components
- Virtual 777 could be
  - Assembled
  - Simulated
  - Interference checked
- Reduced costly rework
- Provided early feedback

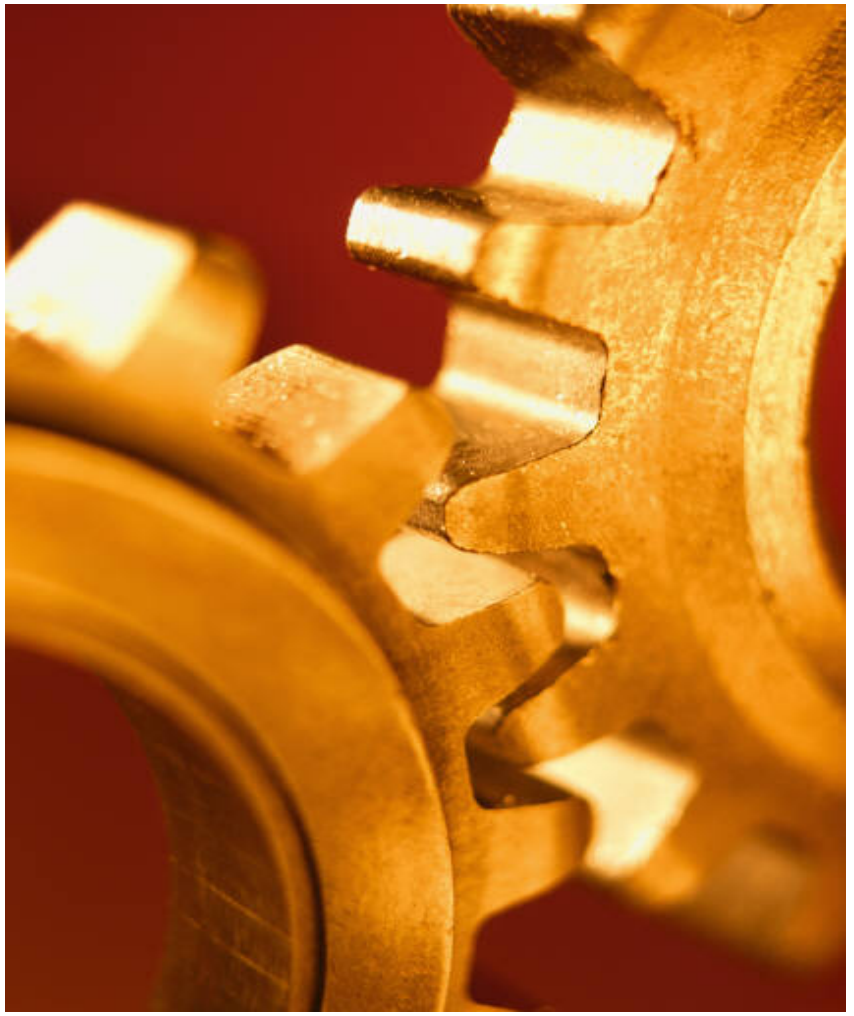
Lessons:

- Economics of past development efforts made it clear that modeling/testing was valuable
  - "Gambled" and built CAD tool as they went along
  - Up-front effort was expected to yield ultimate savings in time and money, and improved customer satisfaction
- Build "generators" from models to…
- The Virtual 777 – CI
  - Continuous builds
  - Integration tests
- Learn from the tests

- Do you have any costly processes that could be mitigated?

Production/QA Schedule

- Jan 93 - production begins (3yrs after requirements!)
- Apr 94 - prototype rolled out
- Jun 94 - first flight
- Apr 95 - airworthiness certifications received
- May 95 - first delivery
- Jun 95 - first commercial flight



The Parallels to Software Dev

- We can learn from the "Triple 7"
  - Committed clients!
  - Architecture is key
    - 240 Component teams
  - Empower teams
    - Use "cost" decision rules
  - Frequent results & feedback through continuous integration
    - Reduced queue wait times
  - Enable change at low levels
    - Reduced cost of delay
  - Wise use of process & tools
    - Thin slice through the entire process proven out
  - Trust but verify

The Business of Software

- Generally, we build software for a purpose!
- Typically we hope to make a profit/positive impact
- Purposes can take many forms:
  - Provide a commercial product
  - Software as a service
  - Support a business process
  - Create a marketing edge
  - All of the above

What About Your "Factory?"

- Do you…
  - Involve the client?
  - Have clear business purpose and goals?
  - Have clear and effective architecture?
  - Encourage a high-powered team?
  - Have continuous integration tests?
  - Test the process?
  - Deliver frequently?
  - Embrace change?
- Can you tie development action to "cost/benefits"?
- If your software were visible/physical, would it be pretty?

- FWIW: If you are like me, plenty of room to always improve!

Summary

- The engineering discipline has a lot to teach our software community
- Finding the "sweet spot" requires understanding the dynamics of your specific product's economics
- Look around at other professions for patterns and processes that might help you succeed
- Everyone has the responsibility to be a disciplined professional – including the client
- To move our profession closer to engineering, we must emerge from the dark recesses of the "cubicle"

- FWIW: These are the sort of techniques I have used successfully

Questions?

- Feel free to ask any questions
- Email: jonkern@comcast.net
- Website/Blog: http://technicaldebt.wetpaint.com