Help - I Was the CAD Manager: Now I'm a Software Developer? SD118887

Mark Perrott
Senior Solution Architect – Customer Success Services - Autodesk

Join the conversation #AU2017
Ideally, a dedicated software development team, skilled in the software development lifecycle, handles the task of customizing Autodesk products to meet your business needs.

But, somehow it ended up being you!
Key learning objectives

At the end of this class, you will be able to:

▪ Learn how to apply modern techniques to maintain and back up scripts and source code
▪ Understand the fundamentals of the software development lifecycle
▪ Learn how to deploy additional tools to enable an easier coding workflow
▪ Learn how to prepare code for cloud technologies
Key learning objectives

At the end of this class, you may not be able to:

- Understand the AutoCAD API any better
- Start coding from nothing
- Start your own software development firm
- Reach enlighten though LISP / VBA
Who am I?

- Mark Perrott
  - Autodesk Consulting
  - Solution Architect
  - 10 years Autodesk
Warning Opinions Approaching!

- My Opinions are just that
- Based on personal experience
- Cite supporters where possible
- Some lies to small children
- Learn when to break the rules, but do so knowingly
Theory: The Ivory Tower
Guiding Principles

- KISS
- Technical Debt
- YAGNI
- Standards & Conventions
- DRY
- Premature Optimization
- Do one thing well
- Enforce Boundaries
Keep It Super Simple

- “Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.”

- I don't know how it works, but it does. Should I use inheritance or polymorphism
Technical Debt

- “a debt that you incur every time you avoid doing the right thing... letting the code quality deteriorate over time” – Martin Fowler

- Not necessarily a bad thing
You Aren't Going to Need It

- Code built before required implementation adds to the burden of complexity.
- Dead code adds to the burden of complexity.
- But what if I need it in the future?
- Version 1 is never released, it escapes!
- Users don’t know what they want until they have it.
Standards & Conventions

- Chose a coding standard and stick with it
- Allows others to work with your code
Standards & Conventions

- Examples
  - Method (Pascal Case): .ToString()
  - Parameter (Camel Case): Array.Copy(Array sourceArray, ...)
  - Field (Camel Case usually prefixed): int _testNum, mTestNum

- Make it Meaningful!
  - du vs. defaultUser
  - foo vs. selectedShape
  - a1, a2, a3, a4 vs. Anything!
Guiding Principles

- If your organization doesn’t provide one go online

- C# Coding Conventions (C# Programming Guide)
  

- MSDN Naming Guidelines
  
Don’t Repeat Yourself

- “Every piece of knowledge must have a single, unambiguous, authoritative representation within a system.”
- – Andy Hunt and Dave Thomas, The Pragmatic Programmer

- Why we use functions and methods?
- When do we use functions and methods?
Premature Optimization

- “The root of all evil. Yet we should not pass up our opportunities in that critical 3% ... after that code has been identified” – Donald Knuth

- Focus on the solution first.
- Will this code be used?
- How optimized does it have to be?
- There is always room for improvement.
Each class, function, variable, etc. should define a single responsibility.
Enforce Boundaries (Encapsulation)

- Your code should only know what it needs to know
- Enables readability and understandability
- Not about code security
Enforce Boundaries (Encapsulation)

- Examples
  - Local Variable
  - Class Variable
  - Class Properties
  - Static Variable
    - Mutable Public Static Variables – Almost always a bad choice!
Implementation: The Shop Floor
Techniques & Tips

- Source Control
- Reinventing the wheel
- Code Smells
- Developer Tests
- Automation
- Help
Source Control

- Essential for collaboration
- Valuable even for single person projects
- Free solutions available

- Systems
  - Team Foundation Server
  - Subversion
  - Git
Source Control

- Demo
Reinventing the Wheel

- Don’t write a logger, a database connector an email client, a CSV importer/exporter
Reinventing the Wheel

- Third Part Library Examples
  - Logging – SeriLog
  - File format Support
    - JSON Newtonsoft.Json
    - CSV Helper
  - ORM – EF, Dapper
Reinventing the Wheel

- Package Managers
  - Ruby – Gems
  - Java – Maven
  - Node.js – NPM
  - Microsoft .Net – Nuget
Reinventing the Wheel

- Demo
Code Smells

- God Objects
- Duplication
- Global States
- Cyclomatic Complexity
- Exception squelch
Code Tests

- You are already testing
- Developer Tests vs. QA
- Tests vs. Test Harness
Code Tests

- Types of Developer Tests
  - Unit Test: Unit of code
  - Integration Test: Code and its Interactions
  - Functional Test: Application functionality
Code Tests

- Frameworks: MSTest, Nunit, Xunite.net
- Test Doubles: Dummy, Fake, Stub, Mocks
Code Tests

- Demo
Automation

- Automation reduces the room for error
- Automation leads to long term benefits
- An extension of don’t repeat yourself

“It builds for me”
Automation

- Automate
- Builds
- Tests
- Installers
- Deployments
Automation

- Systems
  - Team Foundation Server
  - Hudson
  - CruiseControl.Net
  - TeamCity
Automation

- Demo
Help

- Search Engines (Google, Bing etc.)
- Stack Overflow
- FDN (Forge Developer Network)
- Consulting