Stepping Stones In Refactoring to Microservices

Amr Noaman
Agile Academy
Amr Noaman Abdel-Hamid

Agile practitioner, coach, trainer, consultant, writer and lecturer

Co-founder of
Agile Academy
Egypt Lean and Agile Network
Agile Egypt

Email: amr@agileacademy.co
Blog: amr.agileegypt.org
Twitter: @AmrNoaman

Co-initiator of Egypt’s GoAgile program at 2011, to boost lean & agile software development in Egypt

A step-by-step guide to clean coding techniques and professional software development habits.

Member of the Conference Program Team – Development Practices and Craftsmanship – Agile 2017 & Agile 2018
Agenda

• Preliminary Steps Before You Start
• Define: Subsystem, Module, Component, Service, and Microservice
• Strategies for breaking code apart
• Techniques for breaking circular dependencies
• Should we move to Service-based architecture? Or Microservices?
• Stations not Stages!
Stepping Stones In Refactoring to Microservices

**BEFORE YOU START**
If this is your garage, what’s the first step in preparing it?
If this is your room, what's step one in preparing it?

Yes! Removing clutter
Think ...

A man who was stressed out, recently divorced, depressed, drinking way too much alcohol and using drugs,

felt dramatically better after he got rid of 80% of his belongings.

Similarly, if this is your code; first, remove the clutter.
Quick Wins
- Remove dead code
- Remove code duplicates
- Reduce method size
- Enhance identifier naming

Divide & Conquer
- Split code into modules and reduce coupling
- Discover and grow physical and replaceable components
- Promote components to services or microservices

Inject Quality In
- Cover components with automated tests
- Enhance components internal design

Sustainable Refactoring Roadmap v4.0

Continuous Inspection

©2014-2018 – Agile Academy – www.agileacademy.co
Quick Wins

- Remove dead code
- Remove code duplicates
- Reduce method size
- Enhance identifier naming

Divide & Conquer

- Split code into modules and reduce coupling
- Discover and grow physical and replaceable components
- Promote components to services or microservices

Inject Quality In

- Cover components with automated tests
- Enhance components internal design

Sustainable Refactoring
Roadmap v4.0

Continuous Inspection

©2014-2018 – Agile Academy – www.agileacademy.co
Before quick wins

After quick wins
After quick wins

Introduce Structure
Quick Wins
- Remove dead code
- Remove code duplicates
- Reduce method size
- Enhance identifier naming

Divide & Conquer
- Split code into modules and reduce coupling
- Discover and grow physical and replaceable components
- Promote components to services or microservices

Inject Quality In
- Cover components with automated tests
- Enhance components internal design

Sustainable Refactoring
Roadmap v3.0

Continuous Inspection
Divide & Conquer

- Split code into **modules** and reduce coupling
- Discover and grow physical and replaceable **components**
- Promote components to **services** or **microservices**
What is Software Architecture

• “The fundamental organization of a system, embodied in its components, their relationships to each other and the environment”

- ANSI/IEEE Std 1471-2000, Recommended Practice for Architectural Description of Software-Intensive Systems
Definitions

**Module**: any logical grouping of cohesive code functions; can be as big as a sub-system, like an accounting or HR module, or as small as a class, like a calculator or an xml parser.

**Component**: a physical replaceable part of the system which can be accessed through a defined set of interfaces. This can be a jar, war, ear, gem, dll, etc.
Subsystem – Component – Module
Stepping Stones In Refactoring to Microservices

STRATEGIES FOR BREAKING CODE APART
What Factors Drive Code Decompositions?

Code parts which:
1. Change together
2. Released together
Some Useful Grouping Themes

• Business or domain
• Utility
• Port
• Model Archtype
• Architectural

More discussion about types of components at:
Honor Existing Architectural Components

Components of a Transaction Processing architectural style

- Transaction Port or View
- Transaction Dispatcher
- Cash Deposit Processor
- Cash Withdrawal Processor
- Simple Transfer Processor
- Oversees Transfer Processor
Tangled code with no clear boundaries between code modules
Discover and Split Utility & Port Modules

- calculatePercentage
- divideFullNameIntoParts
- multiplyByPercent
- updateXml
- writeToExcel

**Business Code**

- parseJson
- calculatePercentage
- divideFullNameIntoParts
- multiplyByPercent
- updateXml
- writeToExcel

**Non-cohesive all-in-one utility module or class**

**Specialized and more cohesive utility classes**

- JsonParser
- ExcelWriter
- StringUtilities
- MathUtilities
And, Business and Domain Modules

- Cohesive business module
  - JsonParser
  - StringUtilities
  - MathUtilities
  - ExcelWriter

- Recruitment
  - ExcelWriter
  - StringUtilities
  - MathUtilities

- Payroll
  - JsonParser
  - StringUtilities
  - MathUtilities

- TimeManagement
Now, Sprout Components

Cohesive business module
- JsonParser
- MathUtilities
- StringUtilities
- ExcelWriter

Recruitment
- ExcelWriter
- StringUtilities

Payroll
- TimeManagement
- JsonParser
- MathUtilities
Code modules are separated (may be by namespaces), but still they are deployed as one package.
Each code module is a standalone component (jar, war, gem, dll, etc.) which is deployable on its own.
Service Interface (SI):

An exported interface through which a web services can be

Standardize communication

Better structure your teams!

Each component is a standalone service with clear and defined service interface
Stepping Stones In Refactoring to Microservices

**SERVICE-BASED ARCHITECTURE OR MICROSERVICES**
Divide & Conquer

- Split code into modules and reduce coupling
- Discover and grow physical and replaceable components
- Promote components to services or microservices
- Cover components with automated tests
- Enhance components internal design

Inject Quality In
Components vs. Services

“A service is similar to a component in that it’s used by foreign applications. The main difference is that I expect a component to be used locally (think jar file, assembly, dll, or a source import). A service will be used remotely through some remote interface, either synchronous or asynchronous (eg web service, messaging system, RPC, or socket).”

— Martin Fowler
Components vs. Services
Services vs. Microservices

Services share a common datastore, whereas each microservice has a separate standalone datastore.
• Three advantages
• Three disadvantages

• Three advantages
• Three disadvantages

• Three advantages
• Three disadvantages
Now, Microservices .. Right?
It depends ...

• What’s your objective out of refactoring?
  – Better quality, more structured code? ✔
  – Better maintainability? ✔
  – Disjoined parts, updating one won’t blow-up others? ✔
  – Faster time to develop new features? ✔
Next steps may be over-engineering and less-rewarding
Have you thought of:

- Risks of splitting the DB
- Synchronization and aggregation of data
- Hosting costs
- Required skills and calibers
- Tools needed
- Security issues
- ...
Stations not Stages!

- Different Subsystems may embark at different stations
- Moving to the next station is an investment, do it if it worth investing!
Strangulating your Legacy App

1. Identify your sub-systems

2. Prioritize and select one to start with

3. Go through the quick-wins stage

4. Refactor it till it embarks at one station!
Published December 2017!

A step by step guide to clean coding techniques and professional development habits

https://leanpub.com/RefactoringToCleanCode
Keep In Touch!

amr@agileacademy.co
amr.noaman@gmail.com

www.agileacademy.co
Blog: amr.agileegypt.org
@amrnoaman

Agile Academy
Inspiring Minds. Enabling Agility