

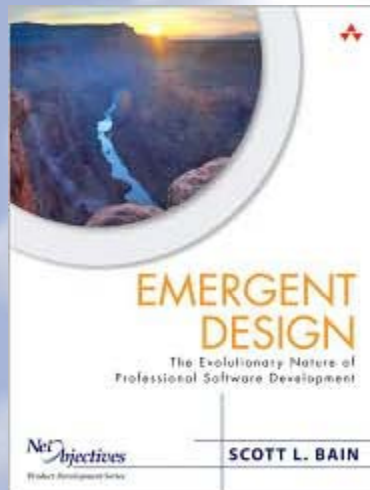


[info@netobjectives.com](mailto:info@netobjectives.com)

[www.netobjectives.com](http://www.netobjectives.com)

# Emergent Design

The Evolutionary Nature of  
Professional Software  
Development



Scott L. Bain  
Net Objectives

# Net Objectives: Who We Are



**Vision** Effective software development without suffering

**Mission** To assist companies in maximizing the business value returned from their efforts in software development and maintenance.

We do this by providing training, coaching, and consulting that directly assists and empowers our customers to create and sustain this ability

**Services** Training in sustainable product development  
Assessments  
Lean-Agile coaching and mentoring

**Expertise** Lean Software Development  
Agile Methods (Scrum, XP, RUP)  
Agile Analysis  
Design Patterns  
Test-Driven Development / Quality Assurance

# Scott Bain



slbain  
@netobjectives.com

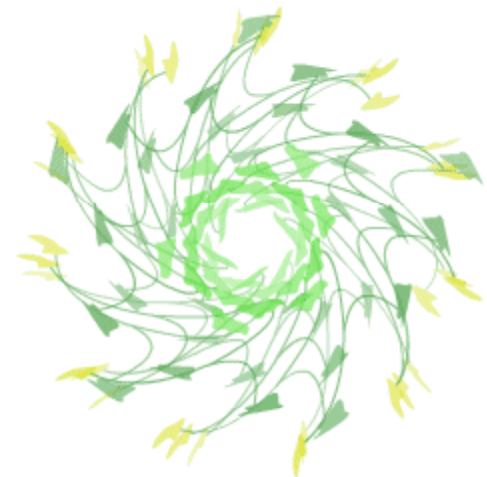
- Senior Consultant
- Certified Scrum Master
- Author, Emergent Design
- Instruction in Agile Development, Design Patterns, Test-Driven Development
- Consulting in quality coding and agile practices

*I have spent most of my professional life working for CBS television and radio.  
I am an avid enthusiast of the American and Russian space programs of the 1960's  
and an obsessive movie buff*

# What Is Design?



- Risk mitigation
- Opportunity to reduce waste
- Documentation of intent
- The thought process behind the software
- Exists at multiple levels:
  - Architecture
  - “Design”
  - Code



# When Do We Do Design?



- After Analysis?
- Before Implementation?
- Do we test our design?
- What if we get it wrong?
- What if we overdo it?
- How much design is enough design?

**Analysis**

**Design**

**Implementation**

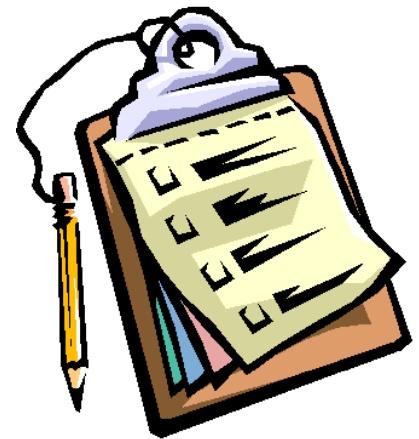
**Testing**

**Release**

# Agenda



- The natural flow of software development
- The traditional view
- A more realistic view
- Qualities, Principles, and Practices
- Disciplines: Testing, Refactoring, and Patterns
- Emergent Design (an example)
- Resources for further investigation



# The Nature of Software Development



“Let the water take you...”

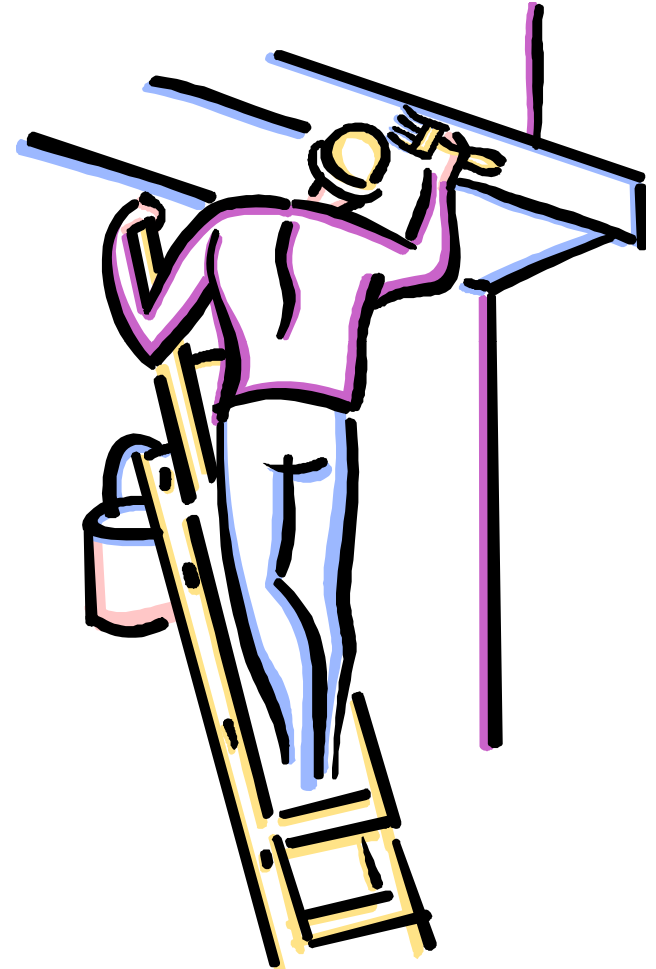


The water flows around the rocks, not through them.

# The Traditional View



- Analogous to Civil Engineering
- Plan, do, review
- Based on:
  - Concrete Analysis
  - Hi-Fidelity Communication
  - Slow Pace of Change
  - Planning around “knowns”

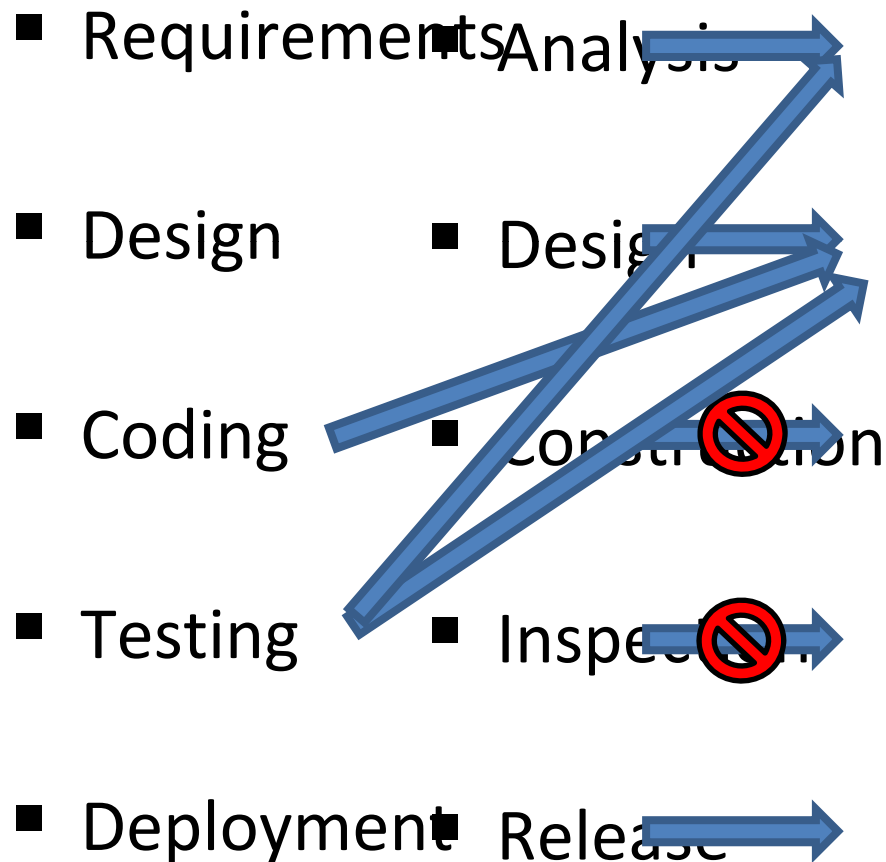




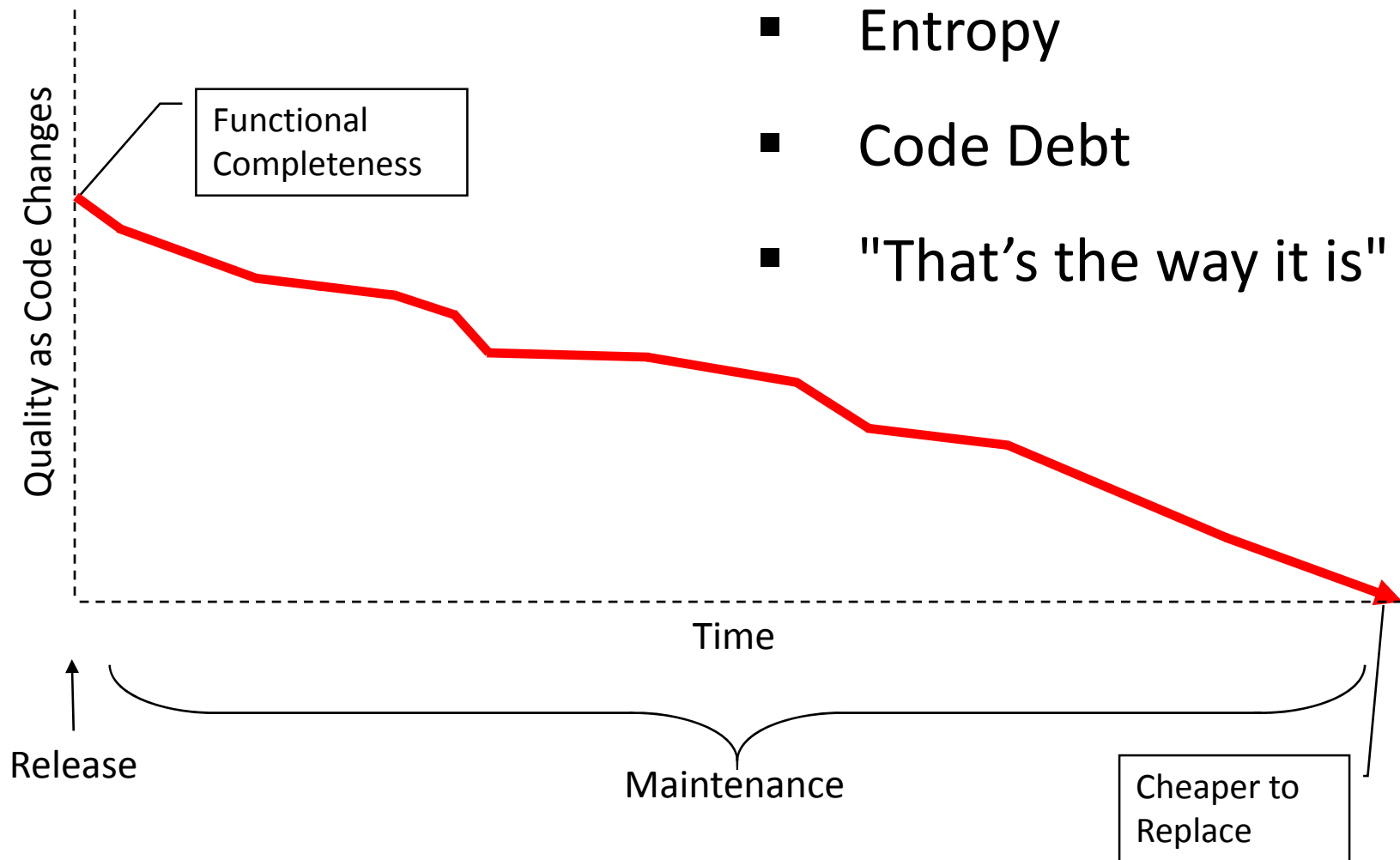
# A Realistic Fit?



Software Development Engineering



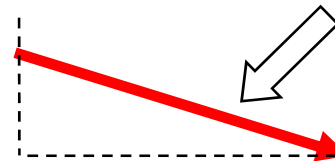
# “Inevitable” Decay



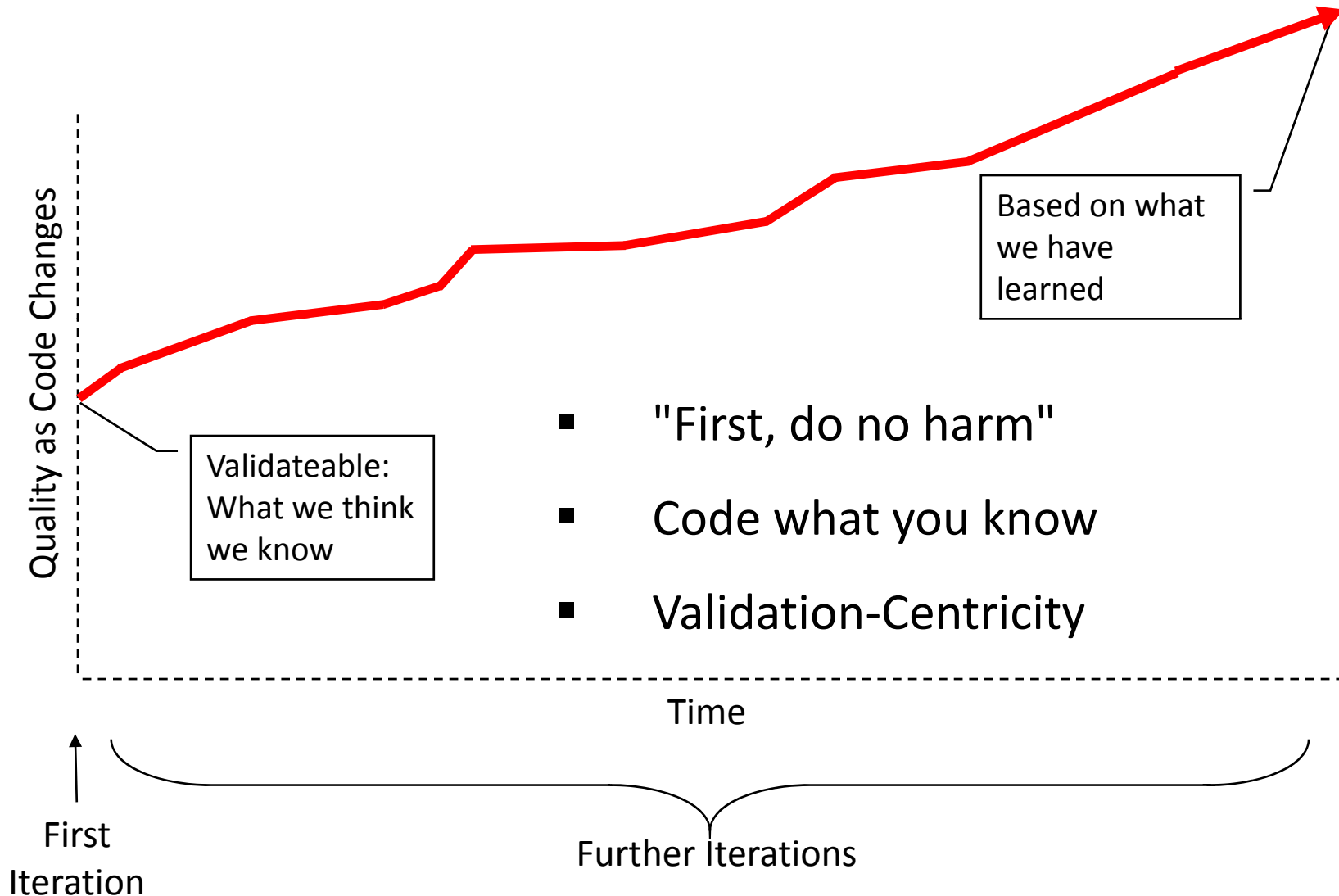
# The Importance of Software



- Software is permeating more and more of our businesses/lives/culture
- The quality of our software is having a greater influence on the quality of our lives
- Failures in software are now capable of doing greater and greater damage
- Other professions are becoming increasingly vulnerable to software



# Rejecting Decay: Changing Change

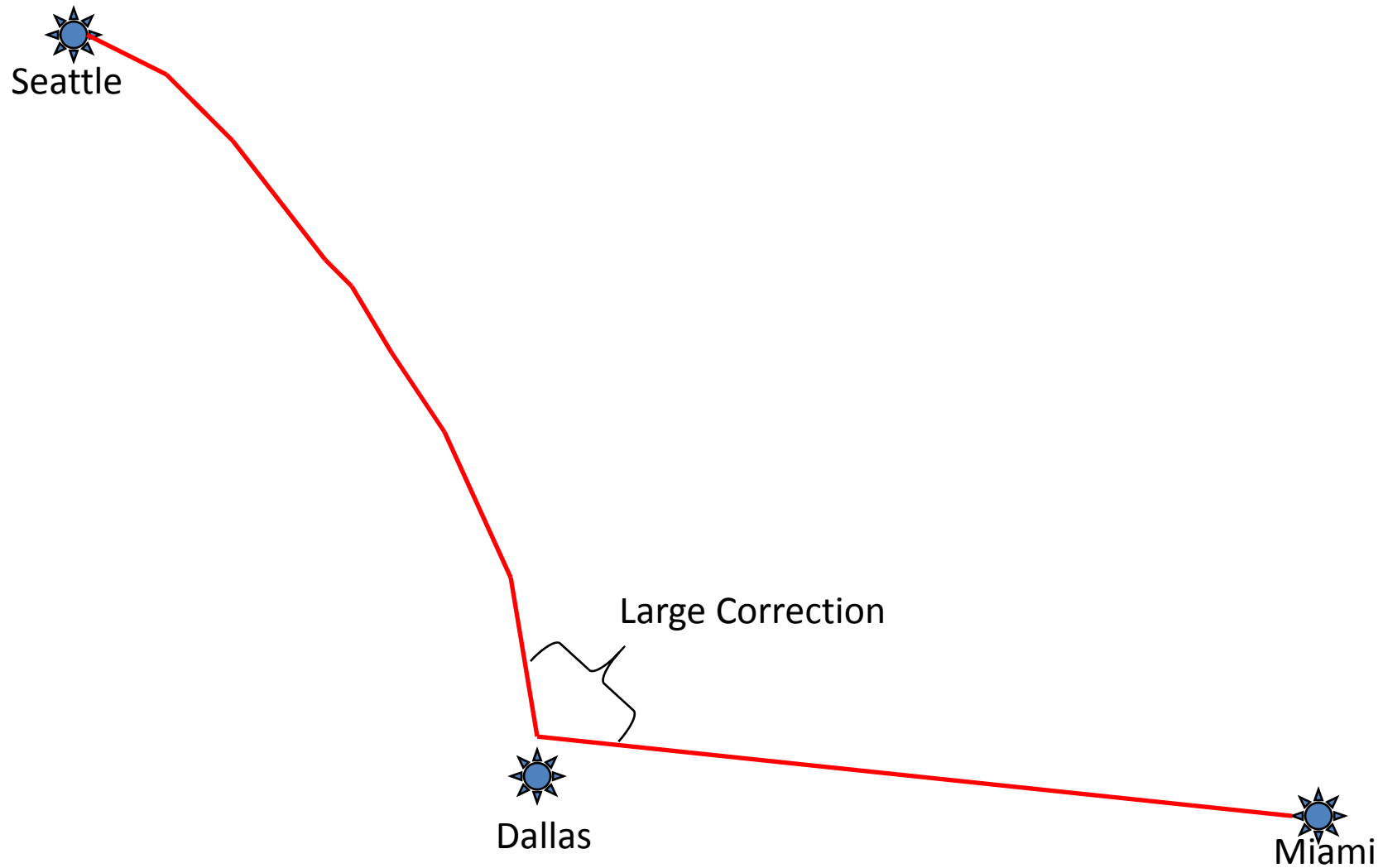


# Changing Expectations, Changing Design



- Let's compare the “old way” of looking at design with this idea of emerging a design
- While we believe we're “letting the water take us”, we also want to respond to any vulnerabilities or potential waste in an agile process
- So, as an analogy...

# Let's Fly to Miami: Plan, Do, Review



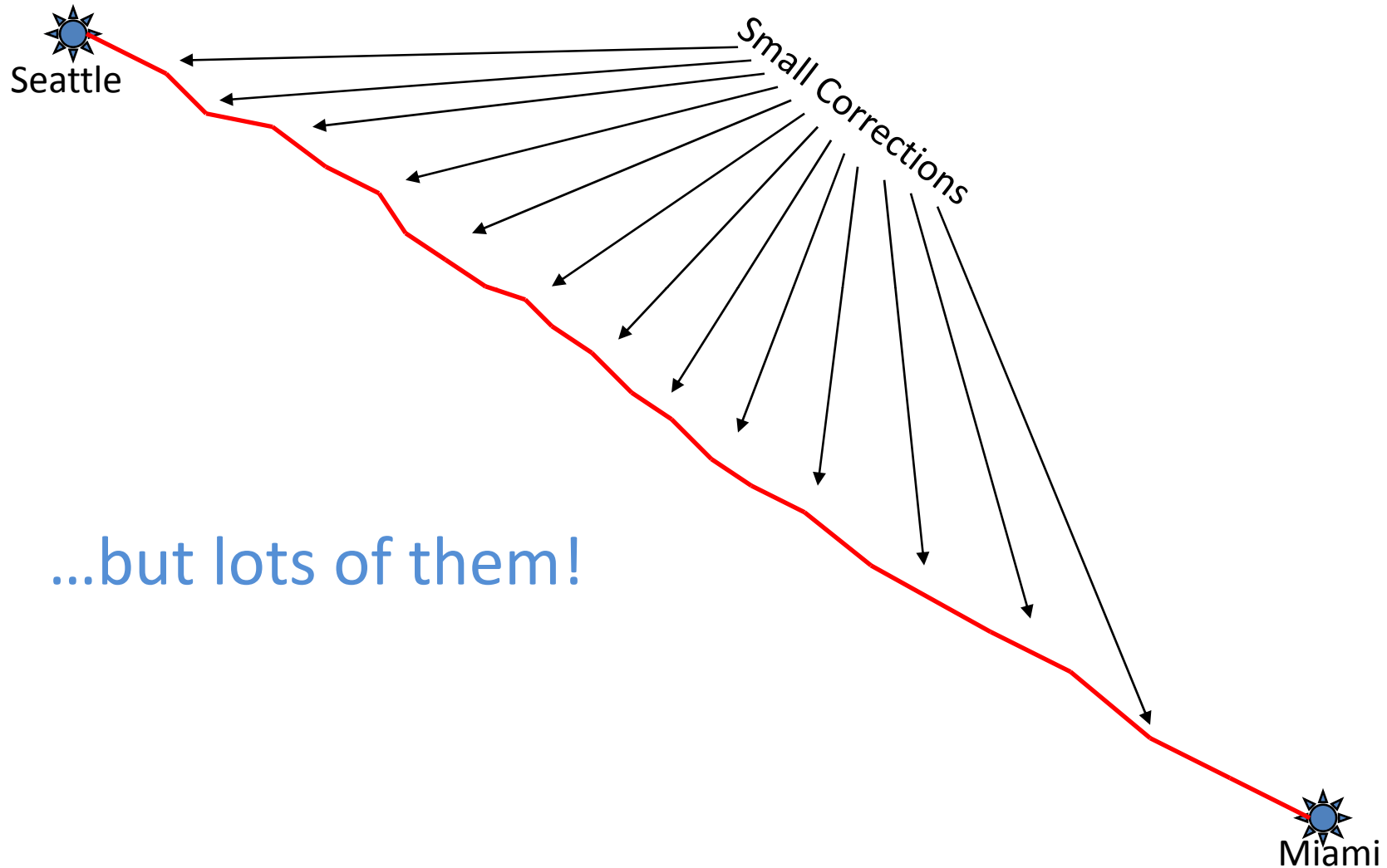
# The Essence of Lean-Agile Development



- Determine the most important thing to do next
  - Figure out how much of it you can do
  - Do it
  - Validate it
  - Repeat
- This step implies something...



# Let's Fly to Miami: Act, Validate, Adjust





# What Is Changeable Design?



- A design is changeable if:
  - In changing it, you do not increase risk or waste
  - It can be changed economically (ROI)
  - It can be changed without decaying
  
- We are guided by the Open-Closed Principle, which is manifested by Patterns...

# The Open-Closed Principle



- Based on the work of Bertrand Meyer, and ultimately a product of pre-OO thinking by Ivar Jacobsen
- Jacobsen: “All systems change during their life cycles. This must be borne in mind when developing systems expected to last longer than the first version”
- Meyer: “Software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification”

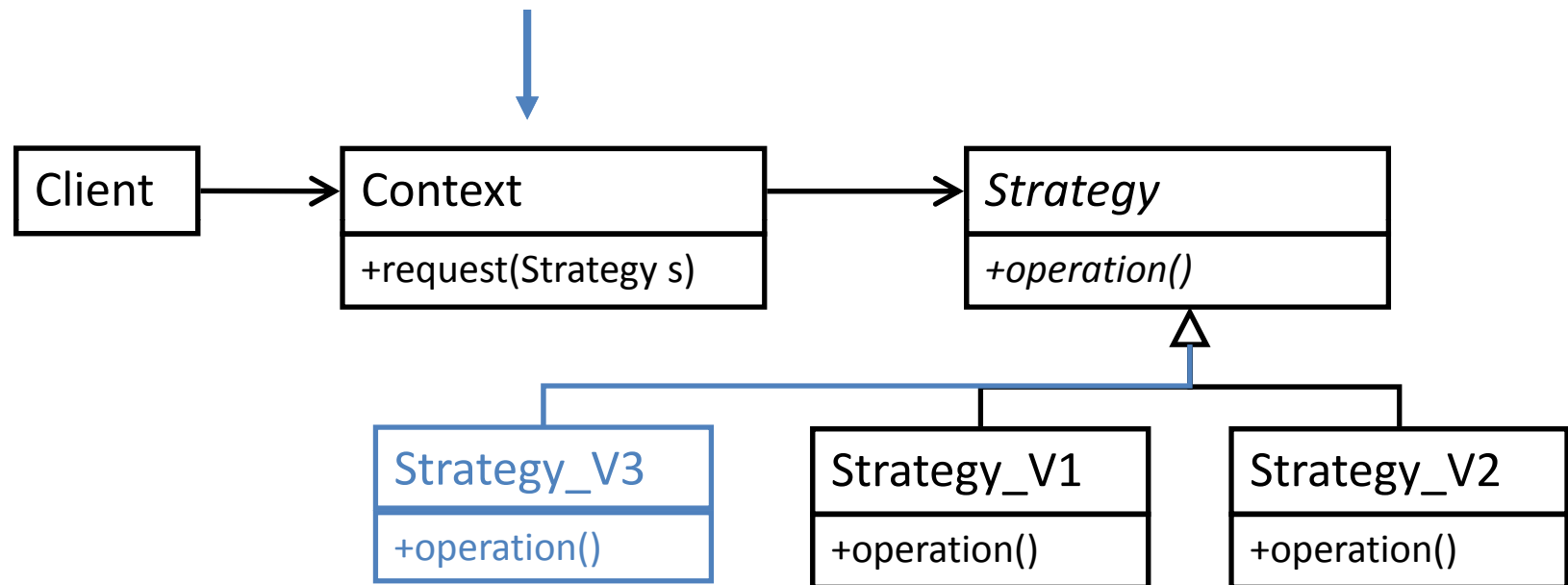


[http://en.wikipedia.org/wiki/Open\\_Closed\\_Principle](http://en.wikipedia.org/wiki/Open_Closed_Principle)

# Applied to Classes: The Strategy Pattern



Do not *change the code* in the context



New Requirement: *add a class*

# The Kobayashi Maru



- From “Star Trek II: The Wrath Of Khan”
- The “un-winnable scenario”
- Kirk won anyway
- By cheating!



KOBAYASHI MARU	
CLASSIFICATION:	Class III Neutronic Fuel Carrier
REGISTRY:	Amber, Tau Ceti IV
MASTER:	Kojiro Vance
CREW:	81
PASSENGERS:	300
DEAD WEIGHT TONNAGE:	147,943 M.T.
CARGO CAPACITY:	97,000 M.T.
LENGTH:	237 m.
BEAM:	111 m.
HEIGHT:	70 m.
MAX CRUISE SPEED:	wf 3
MAX EMERGENCY SPEED:	wf 6

It's okay to cheat if it really helps you...

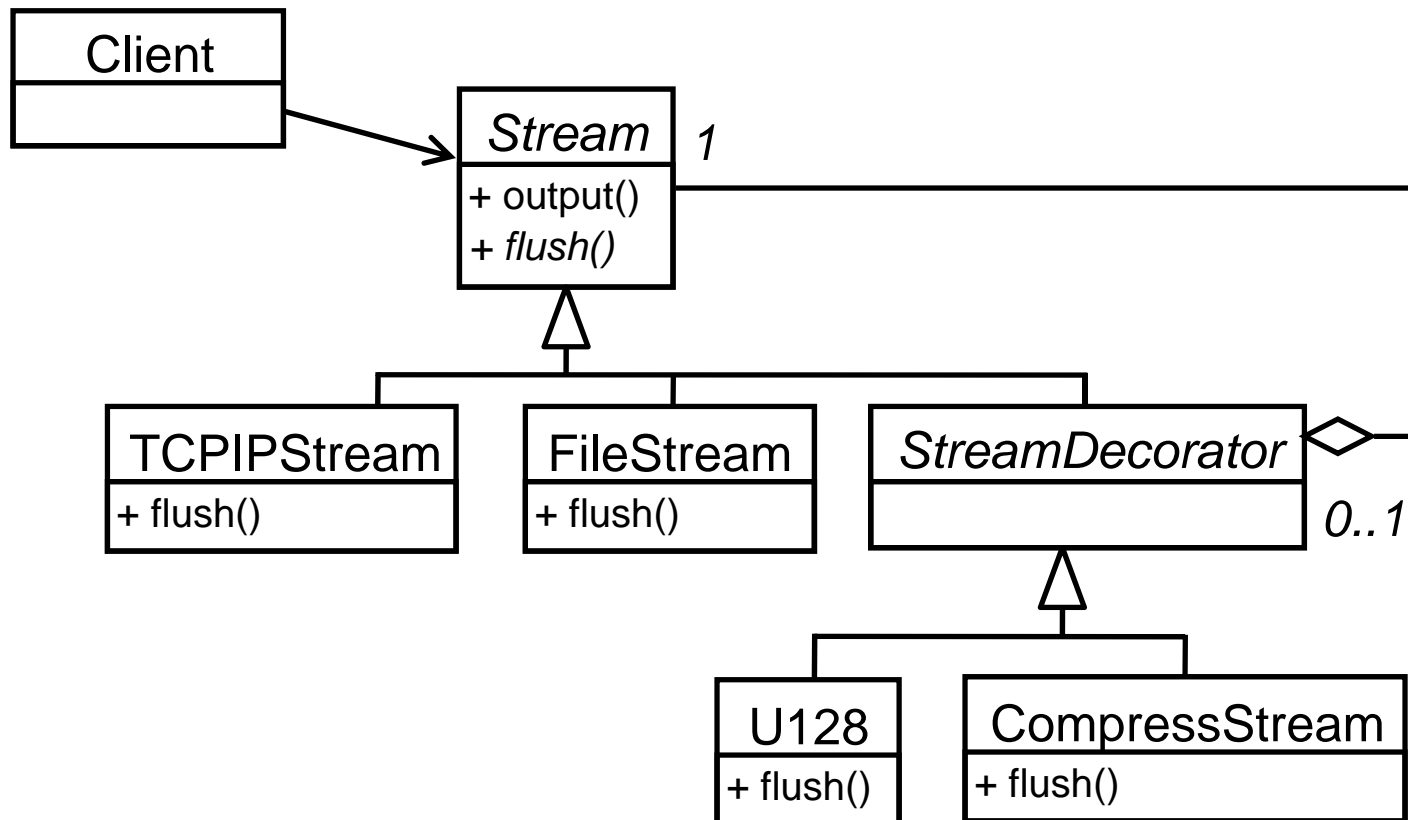
[http://en.wikipedia.org/wiki/Kobayashi\\_Maru](http://en.wikipedia.org/wiki/Kobayashi_Maru)

# Open Closed to Other Things: The Decorator Pattern



- The Decorator Pattern is about adding one, two, or many optional behaviors on top of an existing behavior
- For flexibility, we often accomplish this by placing the behaviors in a linked list (although this is not the pattern, only an example implementation)
- If you've done streaming IO in Java or .Net, you've already used a decorator

# Decorator Pattern for Streaming IO

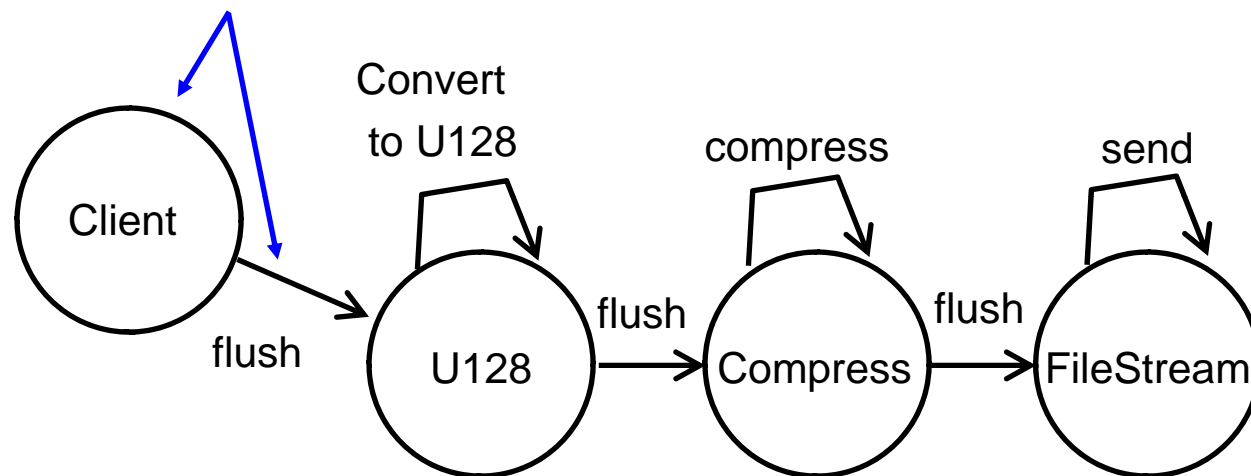


```
Stream myStream = new U128(  
    new CompressStream(  
        new FileStream(filename))) ;
```

# Decorator's Structure and Behavior

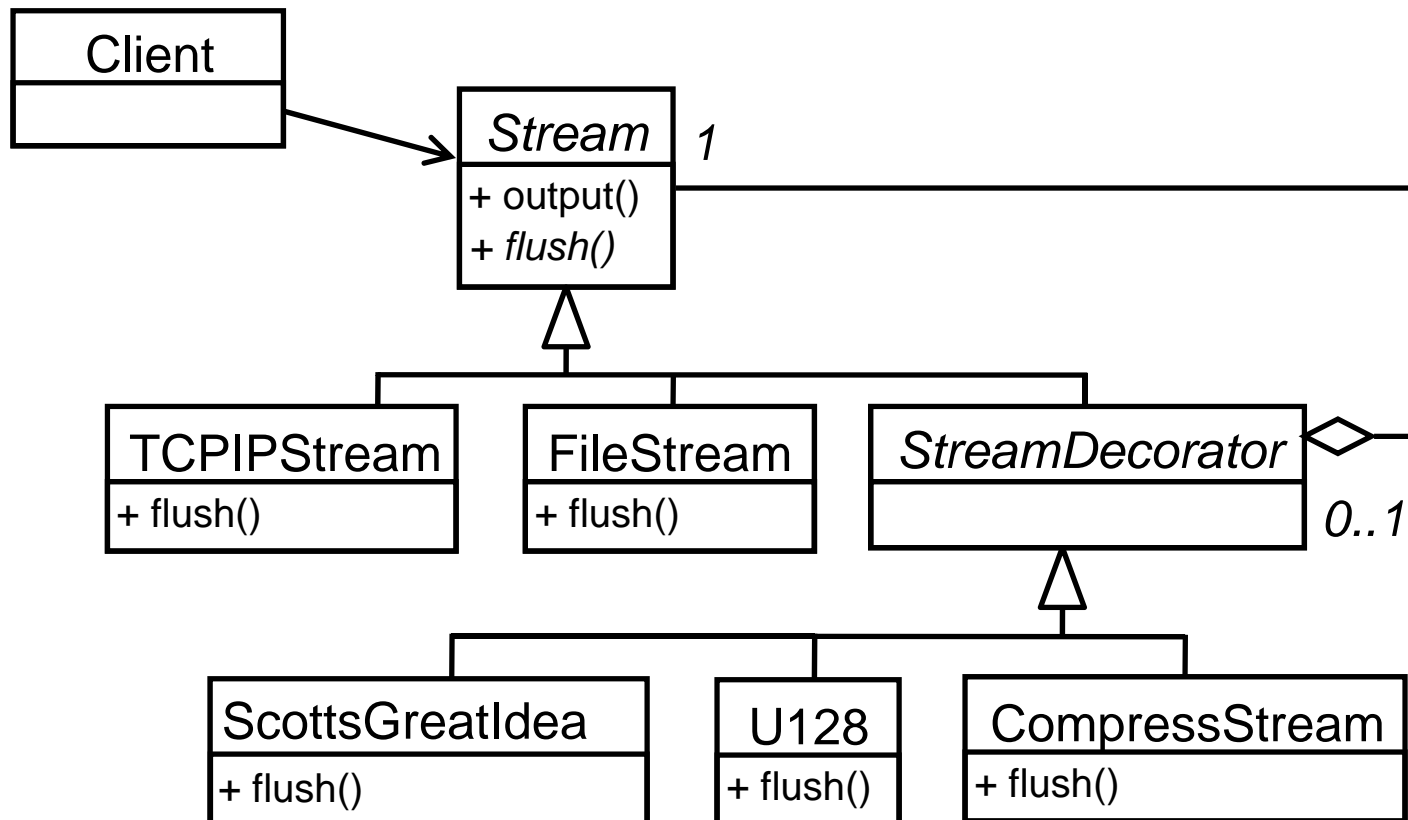


Single Behavior, sees  
only "Stream"



Case: Convert to Unicode, Compress, Flush

# Open-Closed to a New Decorator

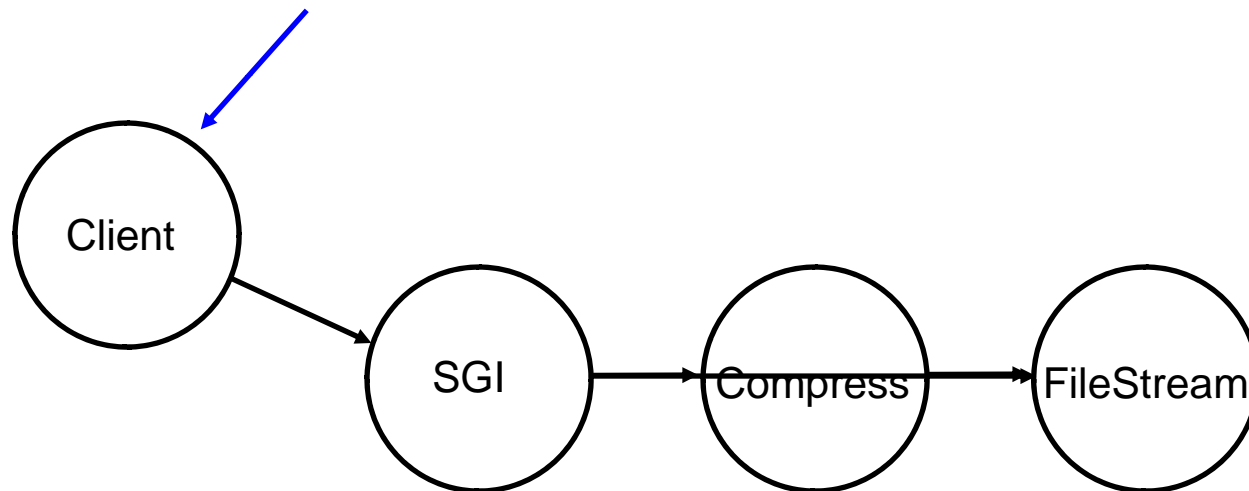




# Open-Closed to More Things...



No change to the  
Client's code...



# Open-Closed is a “Principle”



- No matter how “Open-Closed” your system is, it could be moreso
- Sometimes it’s overkill, especially if you know you can add it later
- If we don’t go so far as to introduce it at the class or design level, we can still be guided by it...

# What Is Changeable *Code*?



- Code can be open-closed too
- It has to do with:
  - Cohesion (each class or method is “single minded”)
  - Coupling (dependencies and side-effects are minimized)
  - Non-duplication (one rule in one place)
- ...and these ***qualities*** are manifested by Patterns too

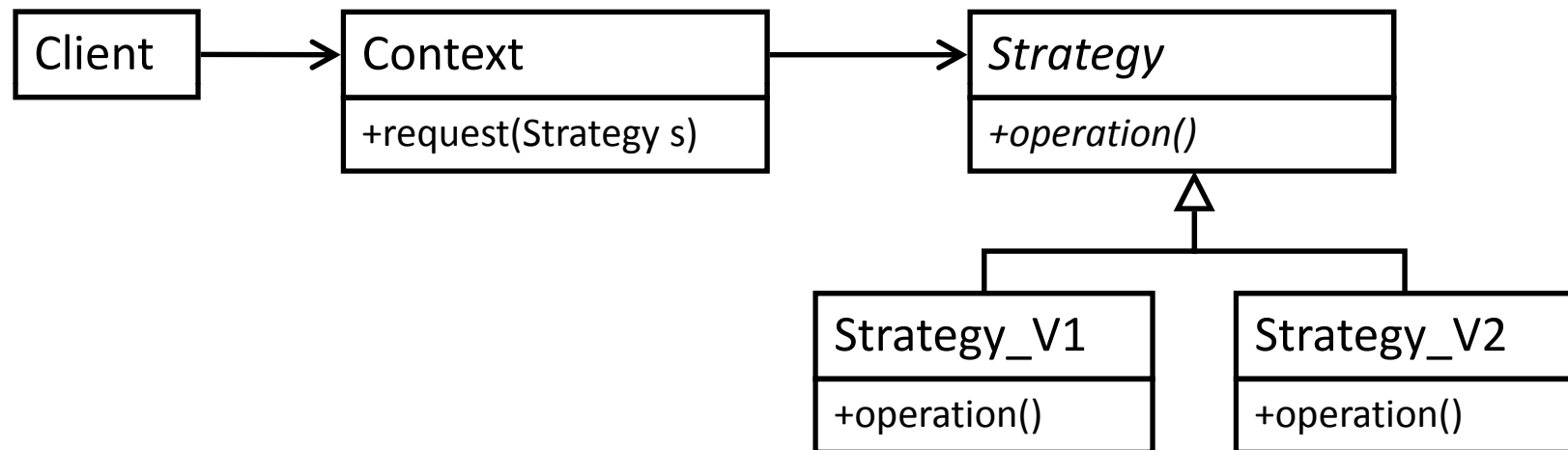
# Patterns Also Promote Code Quality



Cohesive: Delegates an operation

Decoupled: No commitment to specific implementation

Any duplication pushed up here



Cohesive: Implementation is separate

Decoupled: Implementations have no side-effects

# Open-Closed at the Code Level

## Method Cohesion



```
int BigOlHonkinAlgorithm(String parm) {  
    int rval = doStepOne(parm);  
    rval = doStepTwo(rval);  
    if (checkForStep3()) {  
        rval = doStepThree(rval);  
    }  
    return rval;  
}  
  
int doStepOne(String parm) {  
    // Little piece of the overall algorithm, sets myRval  
    return myRval;  
}  
  
int doStepTwo(int parm) {  
    // Little piece of the overall algorithm, sets myRval  
    return myRval;  
}  
  
bool checkForStep3(){  
    // business rule, sets rval  
    return rval;  
}  
  
int doStepThree(int parm) {  
    // Little piece of the overall algorithm, sets myRval  
    return myRval;  
}
```

# Principles and Practices



## Principles:

- General wisdom about design
- Can be difficult to follow in some
- Are compromised with high to follow
- Helps us to think about what we are doing
- Example: OCP



## Practices:

- Are always
- Are other
- Are becoming high circ
- Help we
- Example: let's look at two...



,  
ways"  
st and  
hings

# Practice: Programming By Intention



```
class Transaction
{
    public Boolean commit(String command)
    {
        Boolean result = true;

        String[] tokens = tokenize(command);

        normalizeTokens(tokens);

        if(tokens.Length > getMaximumLength()){
            result = processLargeTransaction(tokens);
        }else{
            result = processSmallTransaction(tokens);
        }
        return result;
    }
}
```

# Practice: Encapsulate the Constructor



- A simple practice that promotes the separation of use from construction
- Is the topic of a Streamzine
- I'll show it simply here...



# Instead of This...



```
class Service {
    public String doOperation(int x) {
        //Whatever this does
        return rVal;
    }
}

class Client {
    Service myService;
    Client() {
        myService = new Service();
    }

    public void m() {
        // Other stuff
        myService.doOperation(10);
        //
    }
}
```

# We Do This...\*



```
class Service {
    private Service(){}
    public static Service getInstance() {
        return new Service();
    }
    public String doOperation(int x) {
        //Whatever this does
        return rVal;
    }
}

class Client {
    Service myService;
    Client() {
        myService = Service.getInstance();
    }

    public void m() {
        // Other stuff
        myService.doOperation(10);
        //
    }
}
```

\*Based on a recommendation by Joshua Bloch in *Effective Java*

# So Later We Can Do This...



```
abstract class Service {

    public static Service getInstance() {
        if(whateverCondition()) {
            return new Service_V1();
        }else{
            return new Service_V2();
        }
    }
    public abstract String doOperation(int x);
}

class Service_V1 : Service {
    // impl1
}

class Service_V2 : Service {
    // impl1
}

//... with little or no change to the client(s)
```

# Disciplines



- A “Discipline” is similar to a practice, in that it is highly valuable, and should be understood by all
- They are not, however, without cost
- Therefore, for a Discipline to become a standard in our business, it must be worth the effort it requires
- Three we’re sold on:
  - Pattern-Oriented Development, which we’ve touched on
  - Test-Driven Development
  - Refactoring (in two different senses of the word)

# Testability and Quality

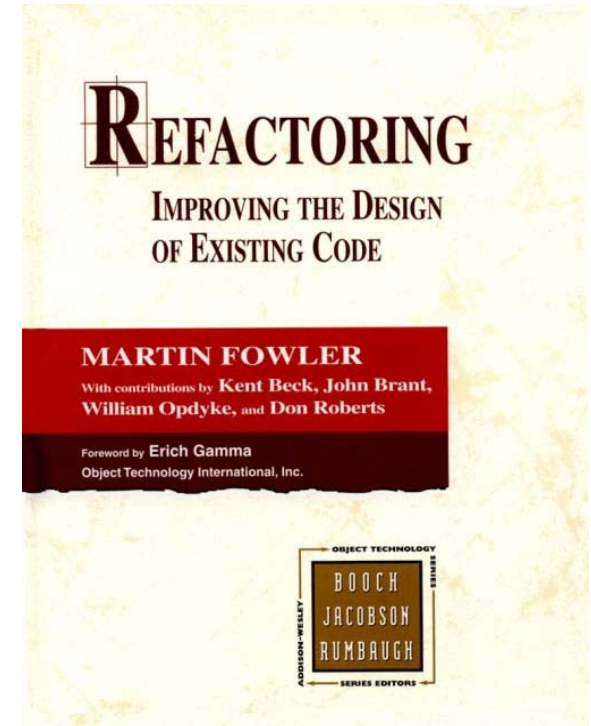


- One big virtue of Test-Driven Development /Design is that it drives the “testability” issue to the forefront in design
  - A class that is too tightly coupled will be hard to test, because all of its dependencies will have to be tested with it
  - A class that is weakly cohesive will be hard to test, because all possible combinations of its multiple responsibilities will have to be tested, due to lack of encapsulation
  - Redundancies in the system will produce redundancies in the test(s)

# Refactoring



- “Improving the Design of Existing Code”
- Behavior-Preserving Change of code
  - Same outward effect as before
  - Same tests pass as before
  - Code Quality is Improved
- Often seen as “Developer Obsession” with code cleanliness, because it does not produce new business value



*Refactoring: Improving the Design of Existing Code* by Martin Fowler, Kent Beck, John Brant, William Opdyke, Don Roberts

# Refactoring Moves for Improving Cohesion



## ■ Extract Method

- ❖ Very common need: take a method that *has become* poorly cohesive and pull details out into a new method.

## ■ Move Method

- ❖ Very common need: take a class that *has become* poorly cohesive and pull details out into another class.
- One often leads to the other.



# Two Kinds of Refactoring



- Classic “Fowlerian” Refactoring:
  - Taking working code and making it better
  - Improving its clarity
  - Improving its design
  
- Refactoring to the Open Closed
  - Taking working code that was fine when written
    - But is not open-closed to adding a new requirement
  - Making it open closed (in a particular sense)
  - ...so you can enhance (add a feature) with less risk and less waste
  - Eliminating risk and waste are business values

(Fowler understands this too...)



# Design as Evolution: Emergent Design

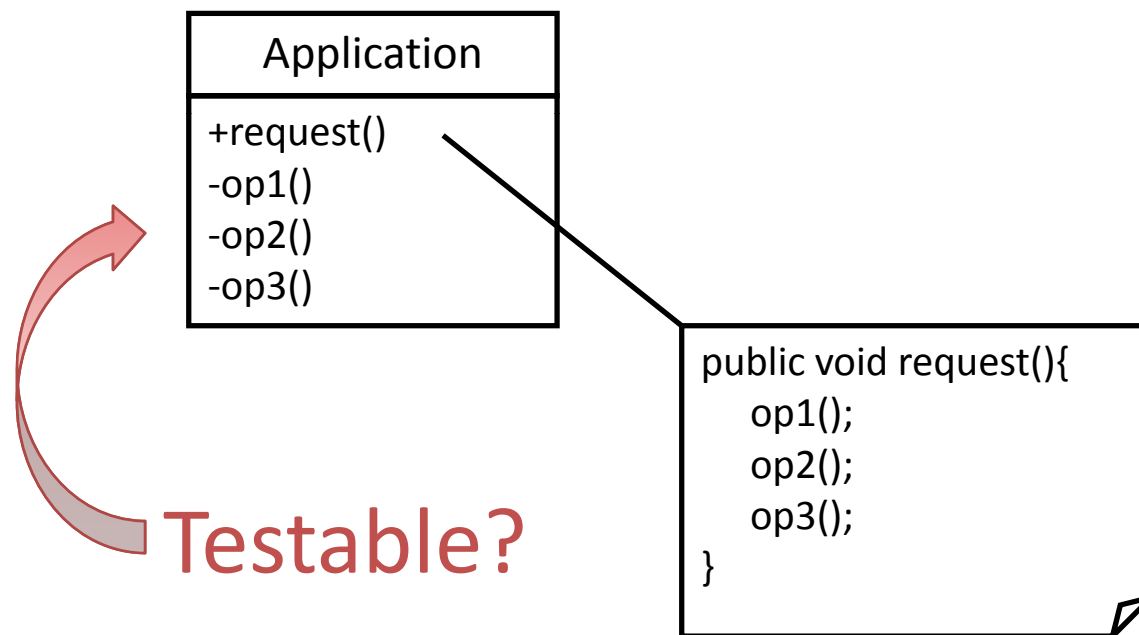


- Software is valuable relative to the world around us, and the needs it has...
- The world, and its needs, change
- For software to retain its value, it must change
- The pace of change is increasing
- Software must always be in a state of evolution
- Disciplines, Principles, and Practices make it possible...

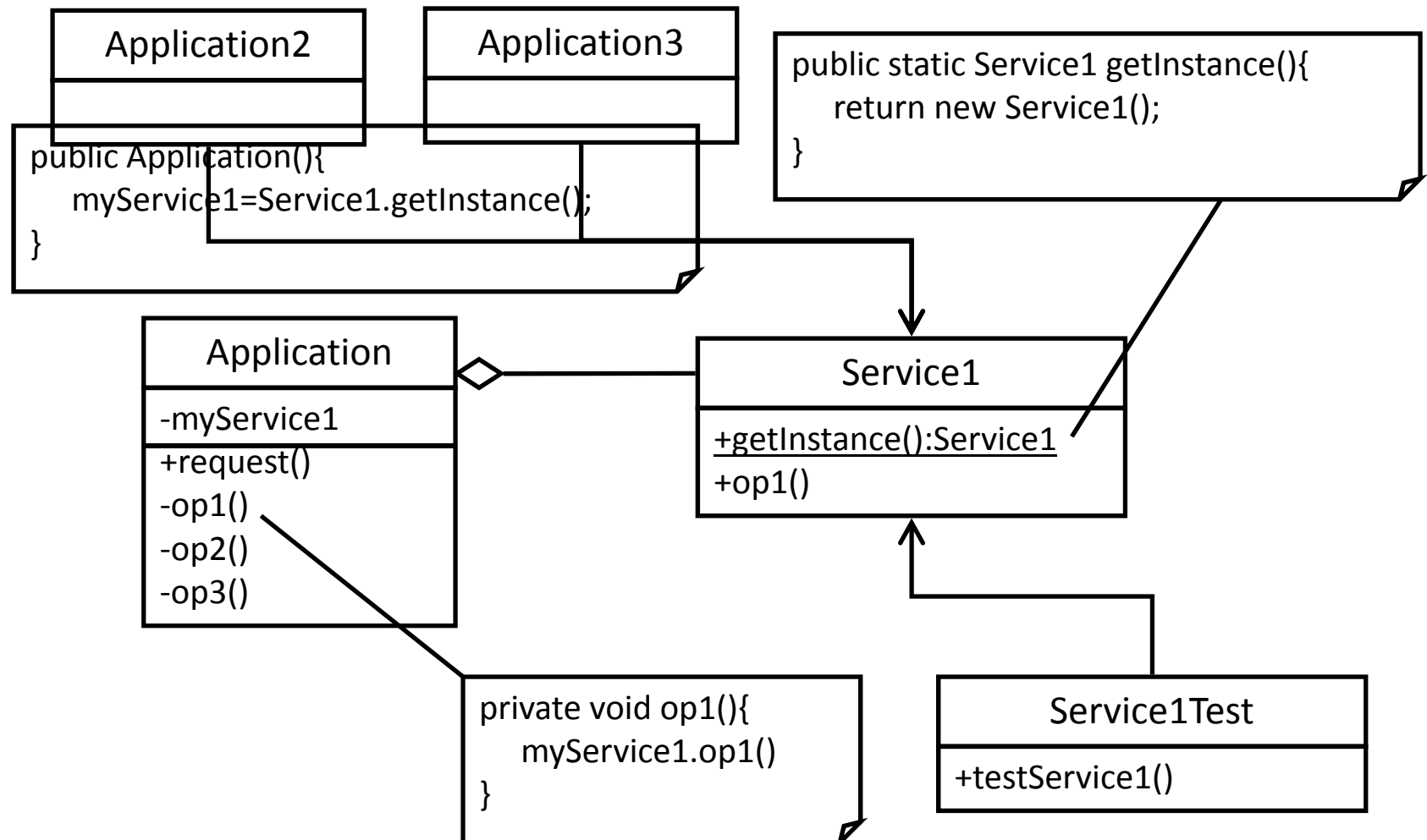
# Practice: Programming By Intention



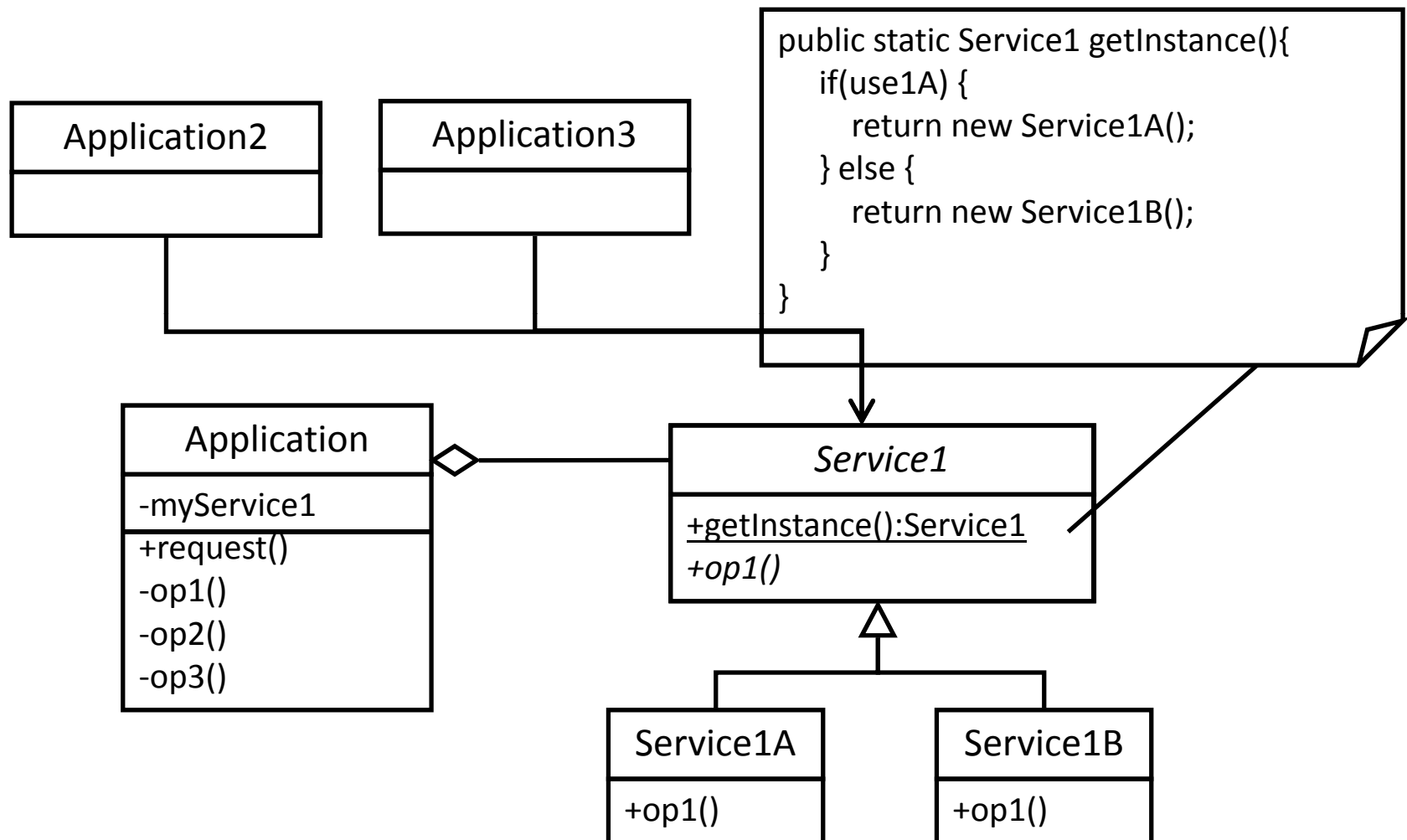
If we'd neglected to do this (or someone else neglected to do this) the refactor would be *extract method*.



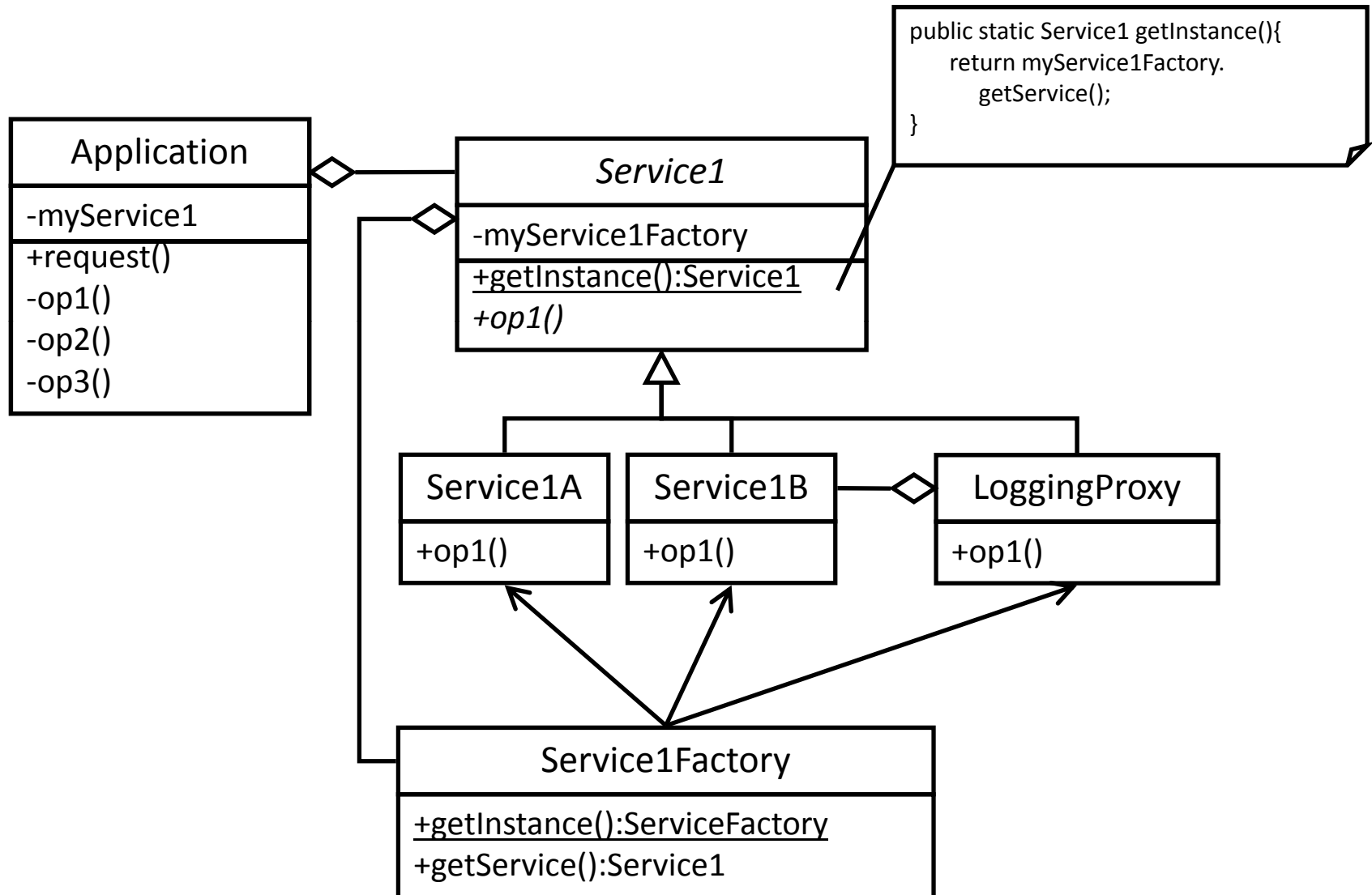
# Testability: Refactor - Extract Class



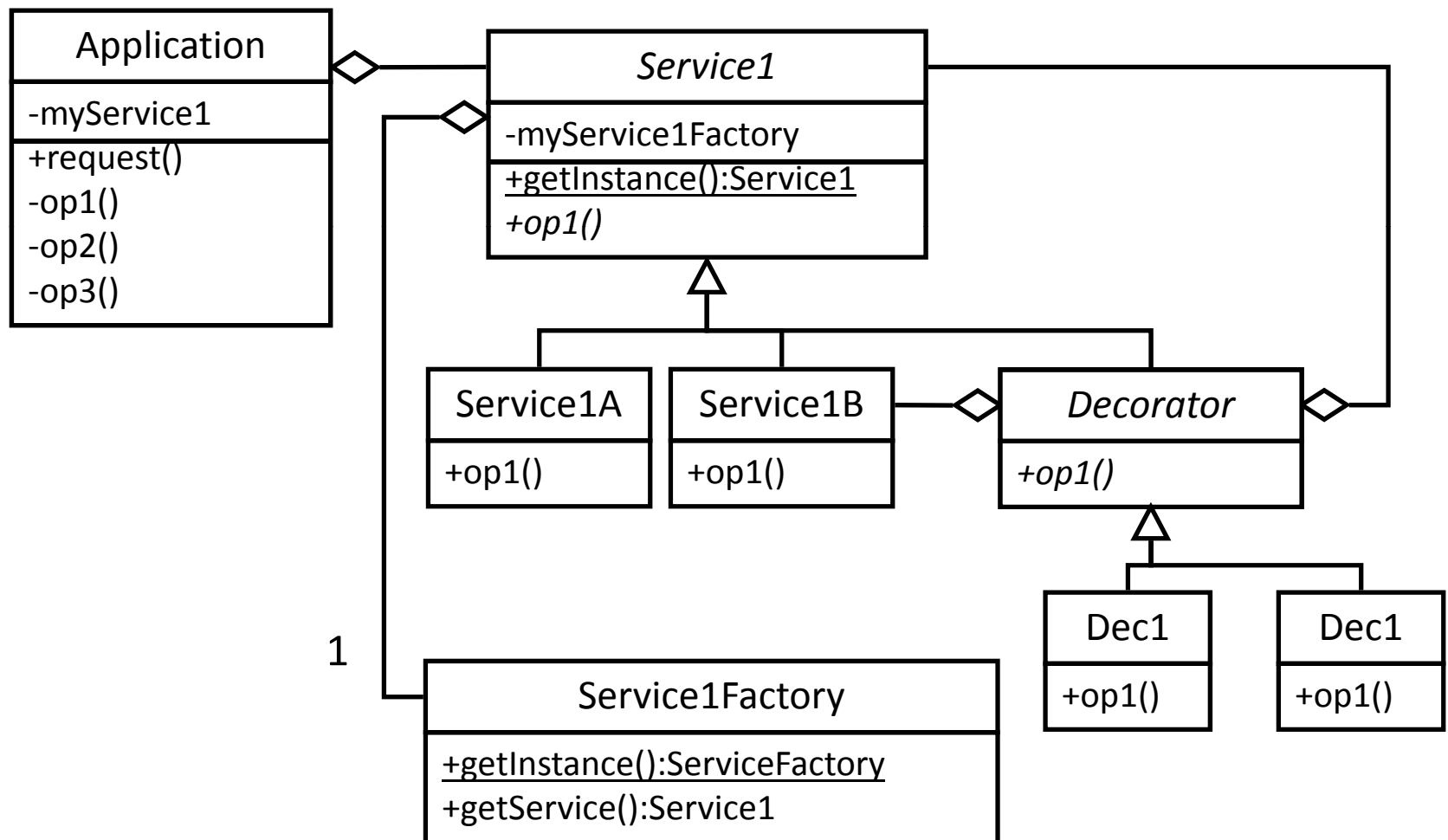
# Open-Closed Principle: Refactor - Extract Interface



# Patterns: Refactor – Introduce Proxy



# Patterns: Refactor – Evolve Decorator



# Remember Where We Started?



Application
+request() -op1() -op2() -op3()

# Avoiding Over-Design



- Complexity When You Need it
  - But not before you need it
- Good practices reduce risk
- Risk is wasteful
- Over-Design is wasteful
  
- Emergent Design uses refactoring to reduce over-design and enable low-risk change
- Principles can guide us...
- Practices can protect us...
- Disciplines can empower us...
- You cannot stop thinking!




# Resources for Further Investigation



- Design Patterns:
  - [www.netobjectivesrepository.com](http://www.netobjectivesrepository.com)
  
- Testing, Refactoring, Qualities
  - [www.netobjectives.com/resources](http://www.netobjectives.com/resources)


# Pattern Repository





slbain · [My Account](#) · [Help](#) · [Sign Out](#)

☆ home [Edit This Page](#)



## The Net Objectives Pattern Repository

This repository is sponsored by [Net Objectives](#), a Seattle-based organization dedicated to training, coaching, and consulting on software design, agile methodologies, test-driven development, lean software process, and scrum.

You are free to use this material for your edification and study. Please ask any questions or contribute your views on patterns at our Yahoo Lean Programming Group: <http://tech.groups.yahoo.com/group/leanprogramming/>






The authors of this repository lurk on that discussion group and will answer your questions, comments, suggestions and make changes to this repository as needed.


For other online resources provided by Net Objectives, please visit the [Net Objectives Main Site](#). The site manager is [Scott Bain](#)

### Site Information:

- [Structure of the Pattern Pages](#)
- [Why Membership is Required](#) (for editing only)
- [Acknowledgements](#)

**Actions**

-  [Make a New Space](#)
-  [New Page](#)
-  [Recent Changes](#)
-  [Manage Space](#)
-  [Site Administration](#)




**Navigation**

- [Home](#)

**Quick Links to Patterns:**

- [Abstract Factory](#)
- [Adapter](#)
- [Bridge](#)
- [Chain of Responsibility](#)
- [Command](#)
- [Composite](#)
- [Decorator](#)
- [Facade](#)





*info@netobjectives.com*  
*www.netobjectives.com*

# Thank You!

... and following is more to help you  
plan your next steps

# Resources



Lean Software Development

Agile / Scrum

Design / Testing /  
Programming Skills for Agile  
Developers

Lean-Agile Testing

Agile VSTS

Certification

Bibliography by Topic

Tools

Book: Design Patterns  
Explained

Book: Emergent Design

Book: Prefactoring

Book: Scaling Scrum to the  
Enterprise with Scrum#

Blog: Net Objectives  
Thoughts

User Groups

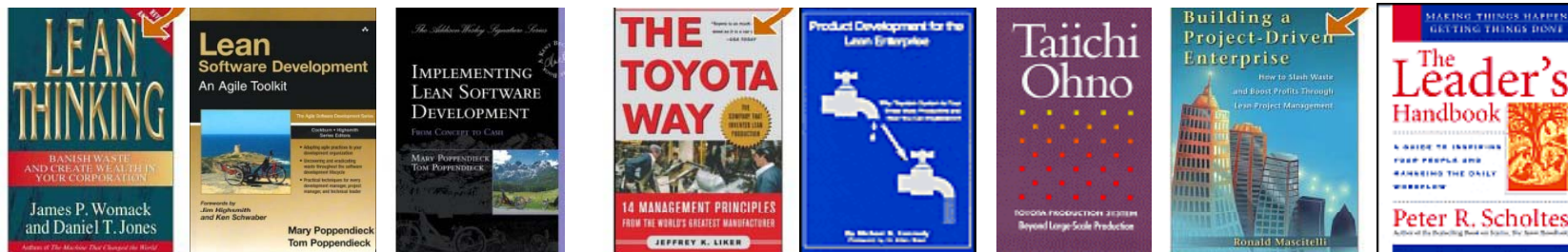
- Resources: [www.netobjectives.com/resources](http://www.netobjectives.com/resources)
  - Webinars/Training Videos (PowerPoint with audio)
  - Articles and whitepapers
  - Pre/post course support Supporting materials
  - Quizzes
  - Recommended reading paths
- Blogs and podcasts: [blogs.netobjectives.com](http://blogs.netobjectives.com)
- Annotated Bibliography
- After-Course Support (students only)
- Additional Training
- Two User Groups
  - <http://tech.groups.yahoo.com/group/leanagilescrum>
  - <http://tech.groups.yahoo.com/group/leanprogramming>

Join our e-mail list to receive regular updates and information  
about our resources and training of interest to you

# A Short List of Books - Lean Related



- Womack and Jones: *Lean-Thinking*
- Mary & Tom Poppendieck
  - *Lean Software Development*
  - *Implementing Lean Software Development: From Concept to Cash*
- Jeff Liker: *The Toyota Way*
- Michael Kennedy: *Product Development in the Lean Enterprise*
- Taiichi Ohno: *Toyota Production System*
- Ronald Mascitelli: *Building a Project-Driven Enterprise: How to Slash Waste and Boost Profits Through Lean Project Management*
- Peter Scholtes: *The Leader's Handbook: Making Things Happen, Getting Things Done*



See <http://www.netobjectives.com/resources/bibliography> for a full bibliography

## Other Relevant Books



- William Bridges: *Managing Transitions*
- Weick and Sutcliffe: *Managing the Unexpected: Assuring High Performance in an Age of Complexity*



See <http://www.netobjectives.com/resources/bibliography> for a full bibliography



# A Short List of Books - Technical



- Mugridge & Cunningham: *Fit for Developing Software*
- Michael Feathers: *Working Effectively with Legacy Code*
- Shalloway & Trott: *Design Patterns Explained, A New Perspective on Object-Oriented Design*
- Bob Martin: *Agile Software Development: Principles, Patterns and Practices*
- Freeman, Freeman, Bates, Sierra: *Head First Design Patterns*
- Martin Fowler, *Refactoring: Improving the Design of Existing Code*
- Ken Pugh, *Prefactoring*
- Scott Bain, *Emergent Design: The Evolutionary Nature of Professional Software Development*



See <http://www.netobjectives.com/resources/bibliography> for a full bibliography

# Net Objectives Services



## Training in Sustainable Product Development

Net Objectives offers the most comprehensive Lean-Agile training in the world. Our offerings include Lean, Agile Analysis, Scrum, Design Patterns, Test-Driven Development, and Lean-Agile Testing.

Our approach is a blend of principles and practices to provide a complete team and/or enterprise wide training solution.

## Assessment Services

An effective way to embark on an enterprise level transition to Lean-Agile methods is to start with an assessment of where you are, where you want to go and options on how to get there that are right for you and your budget.

## Certification Programs by Net Objectives

Net Objectives offers certification programs that provides a road-map of knowledge as well as resources to get there.

- Scrum Certification
- Scrum Master Certification
- Product Owner Certification

Net Objectives is not affiliated with the Scrum Alliance

## Lean-Agile Coaching

While training provides foundational knowledge and is a great jump start, coaching is another effective way to increase the abilities of teams.

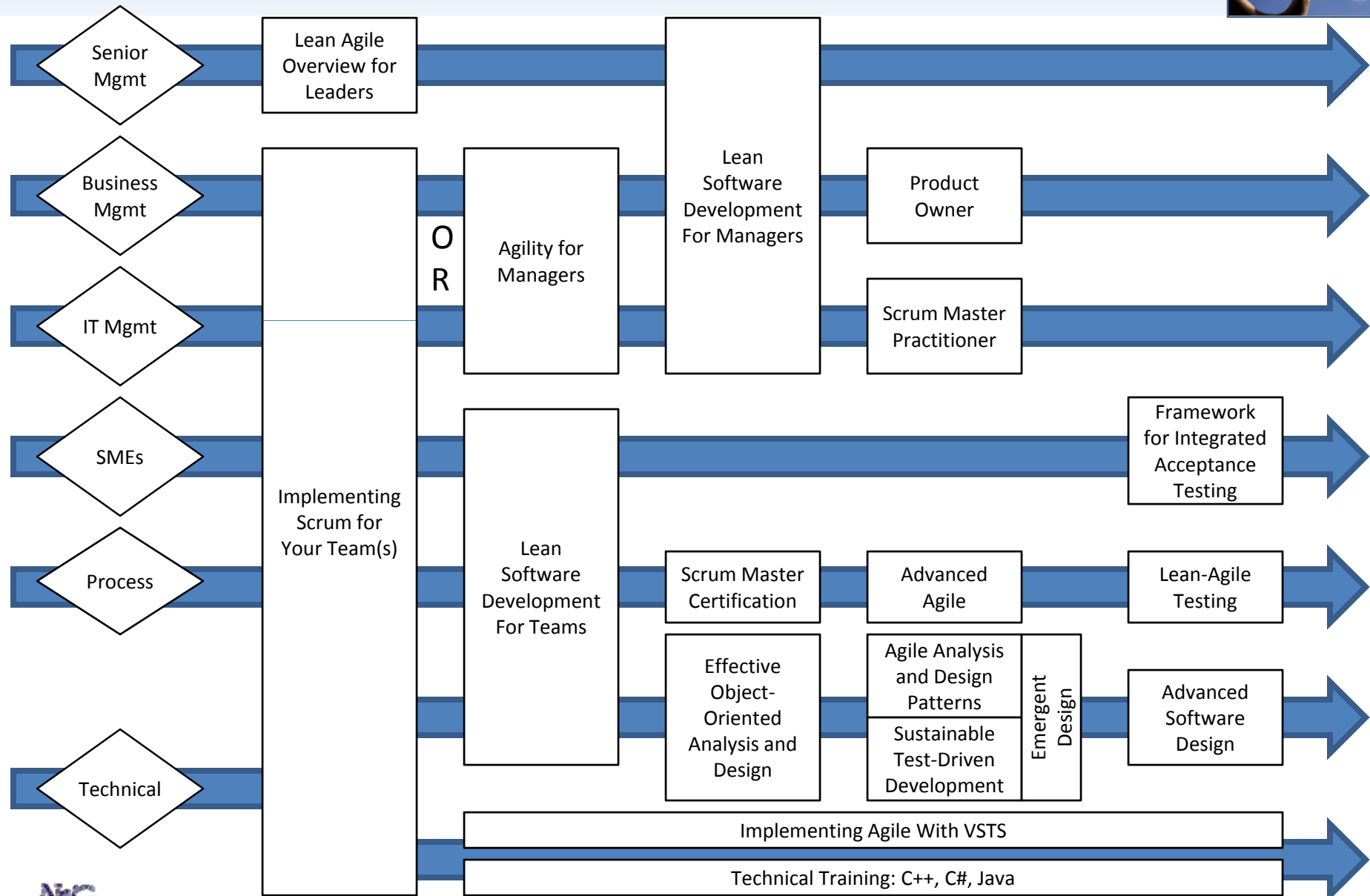
Our coaches work with your teams to provide guidance in both the direction your teams need to go and in how to get there.

Coaching provides the knowledge transfer while working on your own problem domain.



# Achieving Enterprise and Team Agility

## Best Practices Curriculum



# Net Objectives Courses



## ■ Lean Software Development

- Lean Software Development for Management
- ★ Lean Software Development
- Lean-Agile Software Development
- Lean-Agile Testing Practices

## ■ Agile/Scrum

- ★ Implementing Scrum for Your Team
- ➡ Implementing Scrum for Multiple Teams
- ★ Scrum Master Certification by Net Objectives
- Agile Estimation and Analysis for Developers and Product Owners
- Product Owner Certification by Net Objectives
- ➡ Implementing Agile Development with Microsoft™ Visual Studio Team System™
- Agile Life-Cycle Management with VersionOne

## ■ Agile Software Development

- ★ Design Patterns Explained
- ➡ Emergent Design: Effective Agile Software Development
- ➡ Design Patterns for Agile Developers
- ★ Sustainable Test-Driven Development
- Advanced Software Design
- Test-Driven ASP.NET
- Effective Object-Oriented Analysis and Design



A Top 5 Course  
A New Course

For more information, see: [www.netobjectives.com/training](http://www.netobjectives.com/training)



*info@netobjectives.com*  
*www.netobjectives.com*

# Net Objectives Course Catalogue

Following is a list of all of the  
courses we offer

# Lean Software Development



*1 day and 3 and 5 day courses*

- Specific versions for management and for general teams
- Provides complete coverage of Lean Software Development from a management, process and QA point of view
- Covers the material in the Management course above as well as going into greater depth about how a software development team manages Lean-Agile projects
- Includes a future value stream mapping exercise, an overview of the Lean-Agile Process Management method and the new relationship QA has in the development team

For more information, see: [www.netobjectives.com/training/lean-software-development](http://www.netobjectives.com/training/lean-software-development)

# Lean-Agile Testing Practices



*2 day course*

- Lean-Agile methods promote the rapid delivery of value to the customer. One way they do this is to defer detailed definition and design of system features until the “last responsible moment.” This challenges the whole team to stay continuously synchronized within very short Iteration cycles. The team must be creative, smart, and efficient with their verification and validation testing activities.
- How Lean principles can add value to your organization and how they apply to Quality Assurance goals and activities.
- Learn Agile Testing Practices needed to quickly deliver the highest value to your customers
- Discover solutions to your Lean-Agile testing challenges
- Recognize Lean opportunities for change in your organization
- Respond and adapt to Agile development changes effecting your team
- Evaluate and implement Lean-Agile testing practices for your team across the whole release cycle
- Evaluate FitNesse as an automated acceptance testing tool solution for your team
- Recommend Agile Testing transitioning solutions

For more information, see: [www.netobjectives.com/training/lean-agile-testing](http://www.netobjectives.com/training/lean-agile-testing)

# Implementing Scrum for Your Team



*2-3 day course*

- This course is a team-centered offering that teaches a development team how to implement Scrum. It is a combination of interactive lecture with a significant amount of time spent on hands-on exercises.
- If at all possible, the entire team should attend the course together.
  - While only one or two team-members need to know how to play the role of the Scrum Master, all the members need to know what Scrum is and what is expected of them.
- This course teaches:
  - What Scrum is
  - How to manage Scrum projects
  - How to manage requirements in Scrum projects
  - How to use Planning Poker to do story estimation
  - The roles of a Scrum team
  - The role of the Scrum Master
  - The role of the Product Owner
  - The limitations of Scrum
  - How to Scale Scrum

For more information, see: [www.netobjectives.com/training/agile-scrum](http://www.netobjectives.com/training/agile-scrum)

# Implementing Scrum for Multiple Teams



*3 day course*

- We offer an extended 3-day version of our Scrum course which includes the core course but adds:
  - An advanced visual control for managing the work across teams
  - How teams need to integrate and work together to be effective
  - How to create an integration team that goes well beyond Scrum of Scrums
  - How to minimize technical dependencies between teams

For more information, see: [www.netobjectives.com/training/agile-scrum](http://www.netobjectives.com/training/agile-scrum)

# Scrum Master Certification

by Net Objectives



*3 day course*

- This course teaches how to be a successful Scrum Master. The Scrum Master course focuses on the role and responsibilities of the Agile project manager, the Scrum Master. Learn how to ensure your team is fully functional and productive and act as the facilitative team lead working closely with the Product Owner. You will learn how to make your development team, your project, and your organization Agile. Our course is different from other CSM classes in that it teaches the principles on which Scrum is based as well as all of the necessary Scrum practices. At course completion, you are eligible to get your Professional Scrum Master Certification from Net Objectives. Net Objectives is not affiliated with the Scrum Alliance.

For more information, see: [www.netobjectives.com/training/certification](http://www.netobjectives.com/training/certification)



# Agile Estimation and Analysis for Developers and Product Owners



*2 day course*

- This course blends several technologies in a breakthrough course that equips the entire team to uncover and manage the story definition/discovery process.
- It focuses on uncovering and managing customers' needs of the product being built and teaches how to discover the stories in an Agile manner.
- It goes beyond the process of merely pulling out stories as they are encountered, to illustrate how to organize stories so they can be more easily implemented in a consistent manner.
- Techniques on how to organize requirements to help insure consistent and complete information from your customers and/or subject matter experts (SMEs) are also presented

For more information, see: [www.netobjectives.com/training/agile-scrum](http://www.netobjectives.com/training/agile-scrum)

# Product Owner Certification

## by Net Objectives



*1 day course*

- This course can be delivered as either a stand-alone follow up course to any of the Scrum courses that include analysis or it can be integrated into any of these courses adding one additional day of training to the original course. Product Owner training focuses on both the management of the stories on the product backlog and on how to lead a development team in discovering what the best product is for the customer(s). At course completion, you are eligible to get your Professional Product Owner Certification from Net Objectives. Net Objectives is not affiliated with the Scrum Alliance.

For more information, see: [www.netobjectives.com/training/certification](http://www.netobjectives.com/training/certification)

# Implementing Agile Development with Microsoft™ Visual Studio Team System™



*5 day course*

- This course equips students with the knowledge and skills needed to participate effectively on an Agile team that is using Microsoft Visual Studio Team System (VSTS). Participants will learn the roles of Agile, the principles of Agile planning, analysis, and development. VSTS will be used to illustrate the practices taught. There also is a section on how to customize VSTS to support the team's Agile methods. This course can be thought of as an integration of:
  - How to implement Agile for your team
  - How to do Agile Estimation and Analysis
  - How to use VSTS in an Agile team
  - How to customize VSTS for your team
  - deployment and can be customized for teams utilizing hybrid methodologies.

For more information, see: [www.netobjectives.com/training/agile-scrum](http://www.netobjectives.com/training/agile-scrum)

# Agile Life-Cycle Management with VersionOne



2 day course

- Provides the necessary understanding of how Agile projects are managed and how to use VersionOne's flagship product **V1: Agile Enterprise** to manage one's work.
- VersionOne's 100% web-based management platform incorporates a simple, intuitive framework for organizations introducing and scaling their agile development efforts.
- Using VersionOne, all project stakeholders - developers, testers, managers, product managers, customers, and software executives - work together to easily coordinate project plans, priorities, and progress.
- Deployable in minutes, VersionOne enables development teams to accelerate the rollout of today's leading agile methodologies across multiple projects, releases, teams, and locations. Configurable, methodology-specific templates for Scrum, Extreme Programming (XP), DSDM, and Agile UP accelerate internal deployment and can be customized for teams utilizing hybrid methodologies.



For more information, see: [www.netobjectives.com/training/agile-scrum](http://www.netobjectives.com/training/agile-scrum)

# Design Patterns Explained



*2 day or 3 day course*

- This course goes beyond merely teaching several design patterns. It also teaches the principles and strategies that make design patterns good. This enables students to use advanced design techniques in solving their problems whether design patterns are present or not.
- After teaching several patterns and the principles underneath them, the course goes further by showing how patterns can work together to create robust, flexible, maintainable designs.
- Design patterns are about using existing quality solutions to solve recurring problems. They are valuable to learn, because knowing them:
  - provides quality solutions that might not otherwise be thought of
  - gives a common set of terminology to be used amongst team members
  - improves the team-wide quality of design and code

For more information, see: [www.netobjectives.com/courses/design-patterns-explained](http://www.netobjectives.com/courses/design-patterns-explained)

# Emergent Design: Effective Agile Software Development



*5 day course*

- This 5-day course focuses on teaching developers how to:
  - Minimize complexity
  - Maximize Flexibility and Scalability
- It does it with a combination of:
  - Code qualities
  - Handling Variations with Design Patterns
  - Test-Driven Development
  - Refactoring (legacy and agile techniques)
  - Emergent Design
- This course covers the material in both our Design Patterns for Agile Developers and our Test-Driven Development Courses.

# Design Patterns for Agile Developers



*3 day course*

- This course teaches how work effectively in an agile environment.
  - Agile environments embrace change, and therefore require changeable systems.
  - This requires understanding how code quality, design patterns, refactoring, and object management with factories can make systems easier to change.
- Many design patterns are taught to illustrate how to handle variations without adding un-needed complexity.
  - We teach you how to remove the need for overdesign, creating confidence that you can add what you need when you need it.

For more information, see: [www.netobjectives.com/courses/agile-analysis-design-patterns](http://www.netobjectives.com/courses/agile-analysis-design-patterns)

# Sustainable Test-Driven Development



3 day course

- The practice of Agile Software Development requires, among other things, a high degree of flexibility in the coding process. As we get feedback from clients, stakeholders, and end users, we want to be able to evolve our design and functionality to meet their needs and expectations.
- This implies an incremental process, with frequent (almost constant) change to the code we're working on. Each change is an opportunity to make the product more appropriate to the needs it is intended to address.
- Traditionally, changing working code is a stressful prospect, one which we have tended to shy away from. No matter how hard we try, we're almost always faced with making changes. Because of this, many developers have decided to embrace change as their primary working mode.
- However, the reasons we feared change in the first place have not disappeared. Therefore, we need new tools and techniques to ameliorate the problems that change creates.
- Refactoring, the discipline of changing code without harming it, is one such technique. Unit testing, which ensures that a given change has not caused an unforeseen ripple effect in the system, is another.

For more information, see: [www.netobjectives.com/courses/test-driven-development](http://www.netobjectives.com/courses/test-driven-development)



# Advanced Software Design



*2 day course*

- This course that continues the exploration of agile design and analysis techniques begun in the Design Patterns for Agile Developers course.
- We provide an in-depth examination of the critical forces that drive design decisions, and enable a more reliable cost-benefit analysis as part of this.
- We Present a detailed case study, and add additional patterns, including:
  - Visitor
  - Mediator
  - Composite
  - State
  - Observer
  - Command
- We also show how patterns often show up together in repeated ways (often termed “compound patterns”) due to the use of Rapid-Application Development tools and the desire to accommodate changing technology.

For more information, see: [www.netobjectives.com/courses/advanced-software-design](http://www.netobjectives.com/courses/advanced-software-design)

# Test-Driven ASP.NET



*2 day course*

- Test-Driven Development (TDD) is a powerful tool for combining software design, testing, and coding to increase reliability and productivity. With ASP.NET, however, Microsoft has created a tool that doesn't easily lend itself to test-driven development.
- Furthermore, the ease of creating ASP and ASP.NET applications has led to established applications that don't have any tests, making changes difficult and risky.
- This course teaches you how to use NUnitAsp 2.0 to perform test-driven ASP.NET. We show you "best practices" for performing test-driven development of new ASP.NET and then dive into the challenges and solutions of adding tests to existing code. We provide guidance in test-driven development, NUnitAsp, and throw in lots of pithy comments derived from years of experience with architecting web applications.

For more information, see: [www.netobjectives.com/courses/test-driven-development-asp](http://www.netobjectives.com/courses/test-driven-development-asp)

For a taste of this course, see: [nunitasp.sourceforge.net/AdvancedNUnitAsp.html](http://nunitasp.sourceforge.net/AdvancedNUnitAsp.html)

# Effective Object-Oriented Analysis and Design



5 day course

- Object Orientation was and is primarily about the needs of the developer. It came from the best practices of traditional, procedural code, and from all the clever things programmers tried to do in FORTRAN, C, RPG, etc... But where those older languages sometimes hindered object orientation, new OO languages promote and enable OO design, if used properly.
- However, learning the syntax and supporting library API's for an OO language like C++, Java, C#, or VB.NET is really just the first step toward making effective use of Object Orientation. Without a true understanding of the principles that comprise good OO design, and the real benefits they provide, the software produced with an OO language can be just as brittle, inflexible, and hard-to-maintain as ever.
- This course teaches developers and development teams how they can get the maximum benefit from working in an OO language and platform. Using this knowledge, they will produce code more efficiently, with fewer defects, in a more predictable period of time, and which is far easier to maintain.

Available in C++, Java, C#, or VB.NET

For more information, see: [www.netobjectives.com/courses/object-oriented-analysis-design](http://www.netobjectives.com/courses/object-oriented-analysis-design)