# Antipatterns & Other Patterns

CSCI 4448/5448: Object-Oriented Analysis & Design

Lecture 40

# Acknowledgement & Materials Copyright

- I’d like to start by acknowledging Dr. Ken Anderson
- Ken is a Professor and the Chair of the Department of Computer Science
- Ken taught OOAD on several occasions, and has graciously allowed me to use his copyrighted material for this instance of the class
- Although I will modify the materials to update and personalize this class, the original materials this class is based on are all copyrighted © Kenneth M. Anderson; the materials are used with his consent; and this use in no way challenges his copyright

# Goals of the Lecture

- A look at Patterns outside of OOAD…
- Review basis and definitions for AntiPatterns
- Review common AntiPatterns
- Review other typical design pattern types and sources
- Review definition of and common examples of Dark Patterns

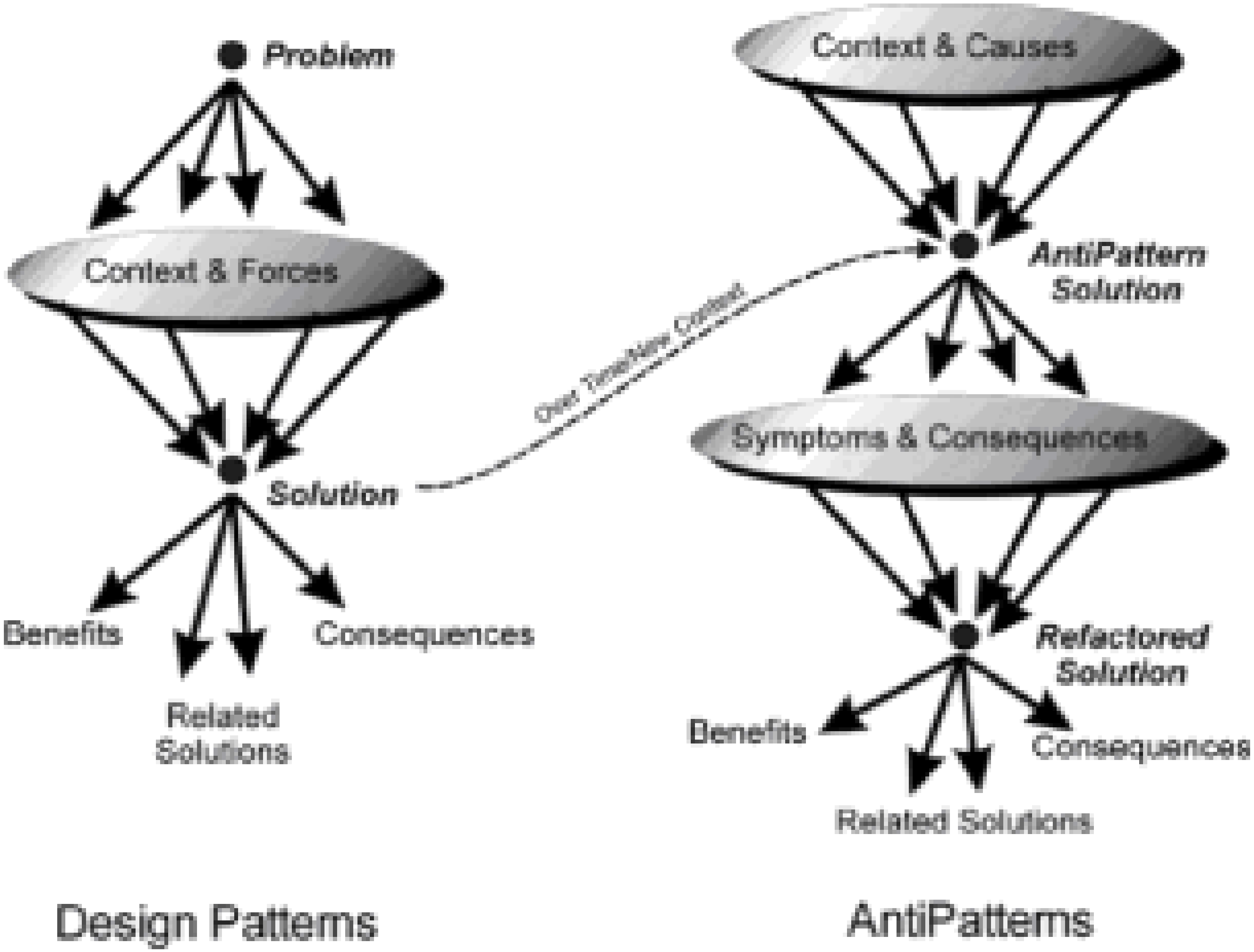
# AntiPatterns?

- The AntiPatterns book describes an AntiPattern as a common solution to a problem that generates decidedly negative consequences
  - Describes the general issue
  - Primary causes
  - Symptoms
  - Consequences
  - Refactored solutions

# AntiPatterns???

- All around you – bad designs, failed projects
- Most common software design mistakes…
- Truth about the software industry
- Reality of software projects
- Needed for change management
- Important method to describe why things go wrong
- More effective (!) than design patterns
- Stress release in the form of shared misery for the most common pitfalls in the software industry

# Patterns vs. AntiPatterns



- The AntiPattern “solution” is bad
- The suggested “refactored” solution is the better path
- Principle viewpoints on AntiPatterns are from architects, developers, and managers

# Three topics driving AntiPatterns

- Root Causes
- Primal Forces
- SDLM – software design-level model

# Root Causes

- Haste (leads to lack of test)
- Apathy (not solving known problems)
- Narrow-mindedness (refuse to use best practices)
- Sloth (using easy answers, ignoring long term impact)
- Avarice (excessive complexity)
- Ignorance (failing to seek understanding)
- Pride (not invented here)

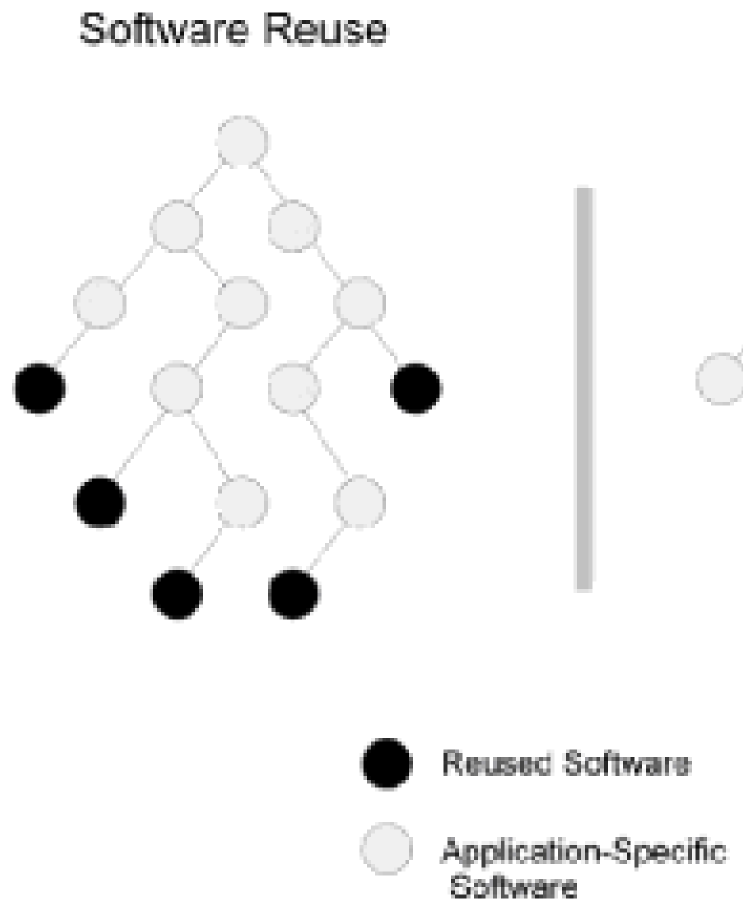
# Primal Forces

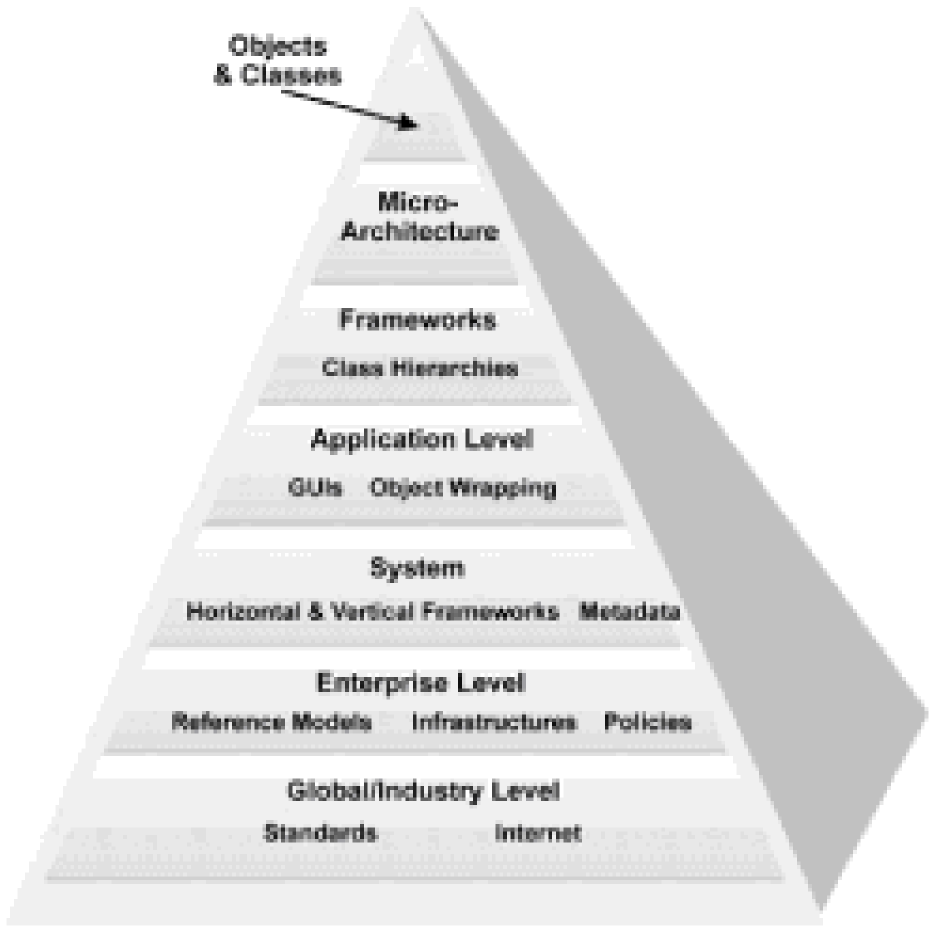
- Management of functionality: meeting the requirements
- Management of performance: meeting required speed of operation
- Management of complexity: defining abstractions
- Management of change: controlling evolution of software
- Management of IT resources: controlling use and implementation of people and IT artifacts
- Management of technology transfer: controlling technology change

- Different responsibilities for each – across developers, architects, project managers, CIOs
- Different impacts for – applications, systems, enterprises, industries

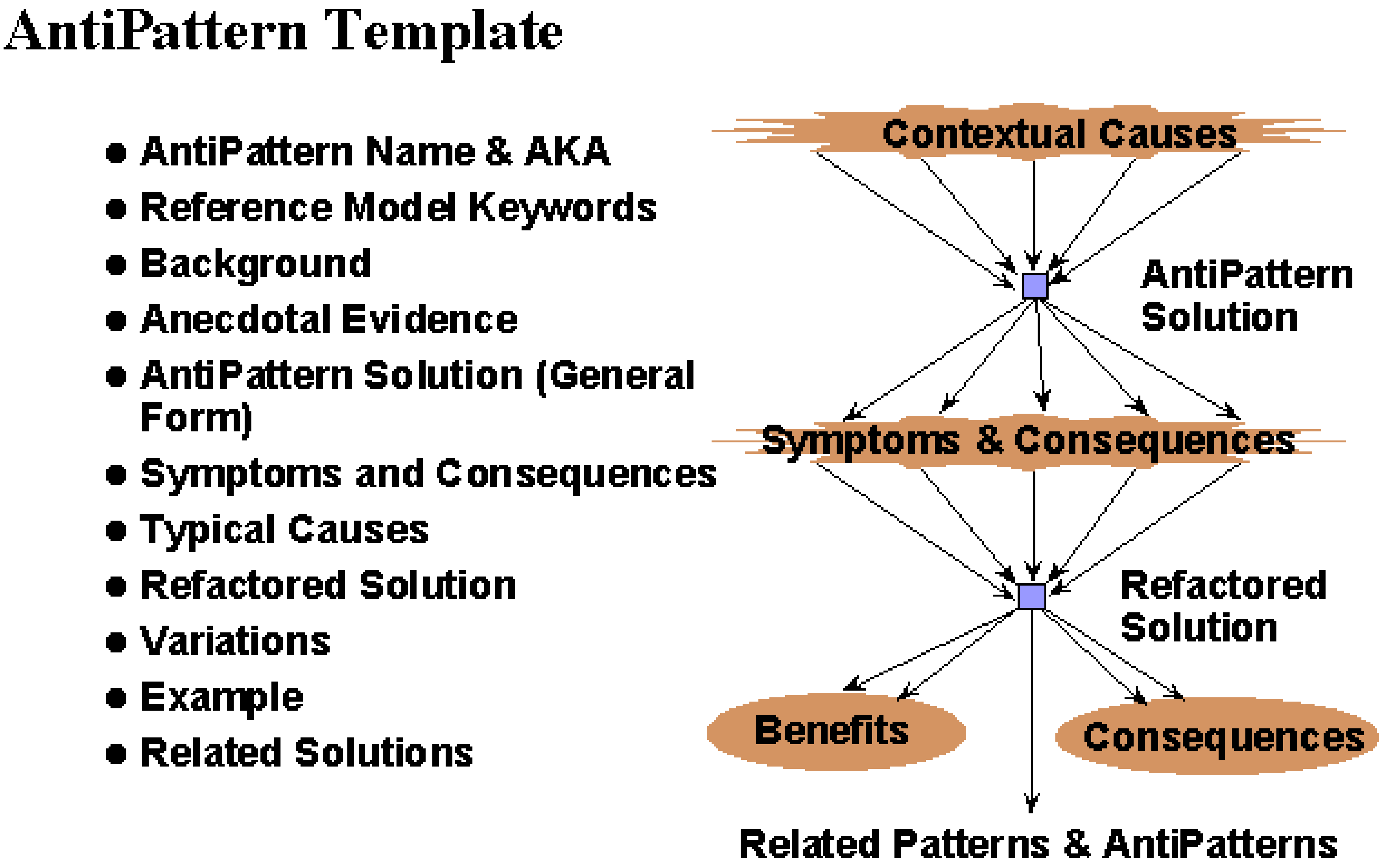
# Software Design-Level Model

- 7 levels: global, enterprise, system, application, macro-component (frameworks), micro-component, and object
- More focus on design reuse than software reuse needed





# AntiPattern Template



## AntiPattern Template

- AntiPattern Name & AKA
- Reference Model Keywords
- Background
- Anecdotal Evidence
- AntiPattern Solution (General Form)
- Symptoms and Consequences
- Typical Causes
- Refactored Solution
- Variations
- Example
- Related Solutions

Reference [2]

# Development AntiPatterns

- The Blob – God/central controlling object(s)
- Continuous Obsolescence – lagging behind in version/tech updates
- Lava Flow →
- Ambiguous Viewpoint – unclear modeling
- Functional Decomposition – Non-OO design in OO environment
- Poltergeists – Short-lived classes
- Boat Anchor – Costly unused technologies
- Golden Hammer – familiar tech applied to everything
- Dead End – Modifyng commercial software
- Spaghetti Code – Ad hoc and difficult structure
- Input Kludge – Custom UI with many evident bugs
- Walking through a Minefield – Pervasive bugs
- Cut-and-Paste Programming – Reuse by copying causes maintenance issues
- Mushroom Management – Developers kept in dark, away from users

AntiPattern Name: Lava Flow
Also Known As: Dead Code

Most Frequent Scale: Application
Refactored Solution Name: Architectural Configuration Management
Refactored Solution Type: Process
Root Causes: Avarice, Greed, Sloth
Unbalanced Forces: Management of Functionality, Performance, Complexity
Anecdotal Evidence: “Oh that! Well Ray and Emil (they’re no longer with the company) wrote that routine back when Jim (who left last month) was trying a workaround for Irene’s input processing code (she’s in another department now, too). I don’t think it’s used anywhere now, but I’m not really sure. Irene didn’t really document it very clearly, so we figured we would just leave well enough alone for now. After all, it works doesn’t it?!”

Background
…We gradually realized that between 30 and 50 percent of the actual code that comprised this complex system was not understood or documented by any one currently working on it… …At this point, we began calling these blobs of code “lava,” referring to the fluid nature in which they originated as compared to the basalt like hardness and difficulty in removing it once it had solidified…

# Architecture Antipatterns

- Autogenerated Stovepipe – auto generated interfaces interfere with subsystem design
- Stovepipe Enterprise – Ad hoc, brittle system
- Jumble – Poor/inconsistent UI designs
- Stovepipe System →
- Cover Your Assets – document choices not decisions
- Vendor Lock-In – proprietary architectures, highly complex, not maintainable
- Wolf Ticket – product claims/listings don't match actual quality
- Architecture by Implication – no documented architecture to guide development
- Warm Bodies – Large project teams, dependent on heroes
- Design by Committee – high complexity, no common vision
- Swiss Army Knife - Overdesign
- Reinvent the Wheel - Legacy systems don't interoperate, builds in isolation
- The Grand Old Duke of York – developers without architectural design capabilities

AntiPattern Name: Stovepipe System
Also Known As: Legacy System, Uncle Sam Special, Ad Hoc Integration
Most Frequent Scale: System
Refactored Solution Name: Architecture Framework

Refactored Solution Type: Software
Root Causes: Haste, Avarice, Ignorance, Sloth
Unbalanced Forces: Management of Complexity, Change

Anecdotal Evidence: "The software project is way over-budget; it has slipped its schedule repeatedly; my users still don't get the expected features; and I can't modify the system. Every component is a stovepipe."
Background
Stovepipe System is a widely used derogatory name for legacy software with undesirable qualities. In this AntiPattern, we attribute the cause of these negative qualities to the internal structure of the system. An improved system structure enables the evolution of the legacy system to meet new business needs and incorporate new technologies seamlessly. By applying the recommended solution, the system can gain new capabilities for adaptability that are uncharacteristic of Stovepipe Systems.

# Management AntiPatterns

- Blowhard Jamboree – hype released doesn’t match facts
- Analysis Paralysis – project gridlock during front end design
- Viewgraph Engineering – belief in company strength based on presentations or sales documents
- Death by Planning →
- Fear of Success – Errant behavior near releases
- Corncob – Difficult people obstruct development processes
- Intellectual Violence – Intimidation or personal gain from esoteric knowledge
- Irrational Management – Habitual indecisiveness, mistrust, or abuse
- Smoke and Mirrors – Demonstration leads to belief release is ready
- Project Mismanagement – Poor/misused development process
- Throw It Over the Wall – handoffs between entities mismanaged
- Fire Drill – demand for immediate results (esp. after management delays)
- The Feud – Management level conflicts impact teams
- E-mail Is Dangerous – E-mail used for sensitive or confrontational messages

AntiPattern Name: Death by Planning
Also Known As: Glass Case Plan, Detailitis Plan

Most Frequent Scale: Enterprise
Refactored Solution Name: Rational Planning
Refactored Solution Type: Process
Root Causes: Avarice, Ignorance, Haste
Unbalanced Forces: Management of Complexity
Anecdotal Evidence:
“We can’t get started until we have a complete program plan.”
“The plan is the only thing that will ensure our success.”
“As long as we follow the plan and don’t diverge from it, we will be successful.”
“We have a plan; we just need to follow it!”

Background
In many organizational cultures, detailed planning is an assumed activity for any project. This assumption is appropriate for manufacturing activities and many other types of projects, but not necessarily for many software projects, which contain many unknowns and chaotic activities by their very nature. Death by Planning occurs when detailed plans for software projects are taken too seriously.

# Other Patterns

- Additional OO Patterns (non-GoF)
- UI/UX Patterns
- Responsive Patterns
- Architectural Patterns
- Test/Automation Patterns
- Security Patterns
- Game Programming Patterns
- Machine Learning Design Patterns
- Microservices Patterns (covered elsewhere)
- Real-time Embedded Patterns

You can likely find others for most specific design related topics…

Always good to search for when you're starting a design…

Remember, patterns are providing experience reuse!

Take advantage of them, it doesn't mean you can't be creative, but it might help you avoid mistakes…

# Other OO Patterns (not Gang of Four)

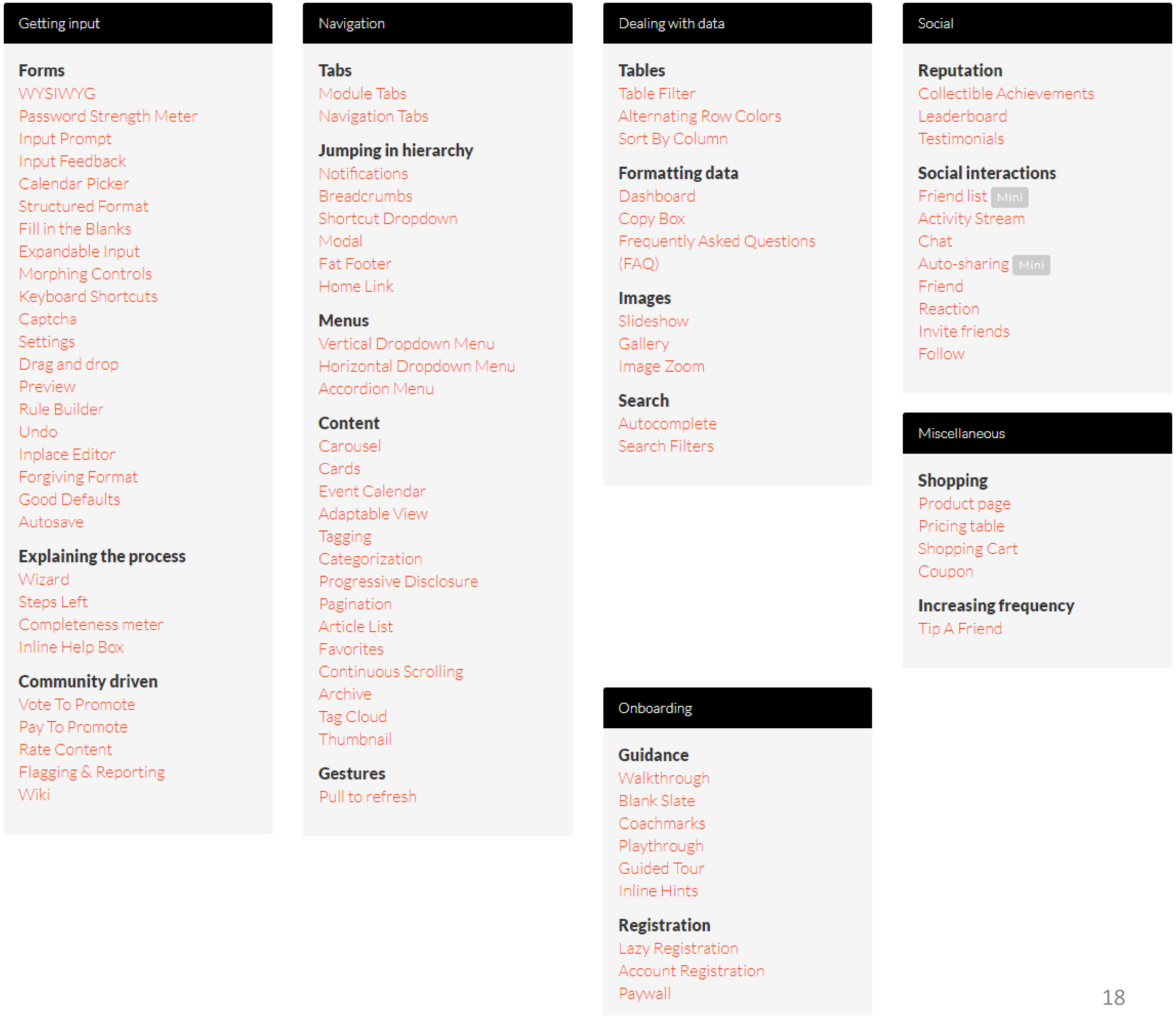
- Business Delegate – decouples presentation and business code – delegates business logic, lookups, and services for UI elements
- Composite Entity – ripples updates and persistence of graph objects
- Data Access Object – separates data access API from high level business objects (as with an ORM)
- Front Controller – centralizes response to requests for authentication, authorization, logging, request tracking – passes requests to handlers – creates a dispatcher for incoming requests
- Intercepting Filter – a filter for a request object, can be a chain of filters executing in order
- Service Locator – Caches high-cost service object requests
- Transfer (or Value) Object – Pass multi-attribute data sets from clients to servers
- From https://www.tutorialspoint.com/design_pattern/index.htm

# Other OO Patterns (not Gang of Four)

- Useful Wikipedia article listing OO design patterns and whether they are GoF or in Code Complete
  - Also some commentary on critical articles re pattern use
  - [https://en.wikipedia.org/wiki/Software_design_pattern](https://en.wikipedia.org/wiki/Software_design_pattern)
- Examples of non-Gang of Four “patterns”
  - Creational
    - Dependency Injection, Lazy Initialization, RAII (ensures resources are released when object lifespan ends)
  - Structural
    - Extension Object, Marker, Twin (allows multiple inheritance in languages that do not directly support it)
  - Behavioral
    - Servant, Specification (combine business logic using Boolean control)
  - Concurrency (Multithreading, Multiprocessing)
    - Double-checked locking, Join, Scheduler (control when threads can execute)

# UI/UX Patterns

- UI Patterns (shown)
  - Also includes Persuasive Design Patterns
- Interaction Pattern Library
- UI Design Examples
- Other UI Pattern Libraries



Getting input

**Forms**
WYSIWYG
Password Strength Meter
Input Prompt
Input Feedback
Calendar Picker
Structured Format
Fill in the Blanks
Expandable Input
Morphing Controls
Keyboard Shortcuts
Captcha
Settings
Drag and drop
Preview
Rule Builder
Undo
Inplace Editor
Forgiving Format
Good Defaults
Autosave

**Explaining the process**
Wizard
Steps Left
Completeness meter
Inline Help Box

**Community driven**
Vote To Promote
Pay To Promote
Rate Content
Flagging & Reporting
Wiki

Navigation

**Tabs**
Module Tabs
Navigation Tabs

**Jumping in hierarchy**
Notifications
Breadcrumbs
Shortcut Dropdown
Modal
Fat Footer
Home Link

**Menus**
Vertical Dropdown Menu
Horizontal Dropdown Menu
Accordion Menu

**Content**
Carousel
Cards
Event Calendar
Adaptable View
Tagging
Categorization
Progressive Disclosure
Pagination
Article List
Favorites
Continuous Scrolling
Archive
Tag Cloud
Thumbnail

**Gestures**
Pull to refresh

Dealing with data

**Tables**
Table Filter
Alternating Row Colors
Sort By Column

**Formatting data**
Dashboard
Copy Box
Frequently Asked Questions (FAQ)

**Images**
Slideshow
Gallery
Image Zoom

**Search**
Autocomplete
Search Filters

Onboarding

**Guidance**
Walkthrough
Blank Slate
Coachmarks
Playthrough
Guided Tour
Inline Hints

**Registration**
Lazy Registration
Account Registration
Paywall

Social

**Reputation**
Collectible Achievements
Leaderboard
Testimonials

**Social interactions**
Friend list Mini
Activity Stream
Chat
Auto-sharing Mini
Friend
Reaction
Invite friends
Follow

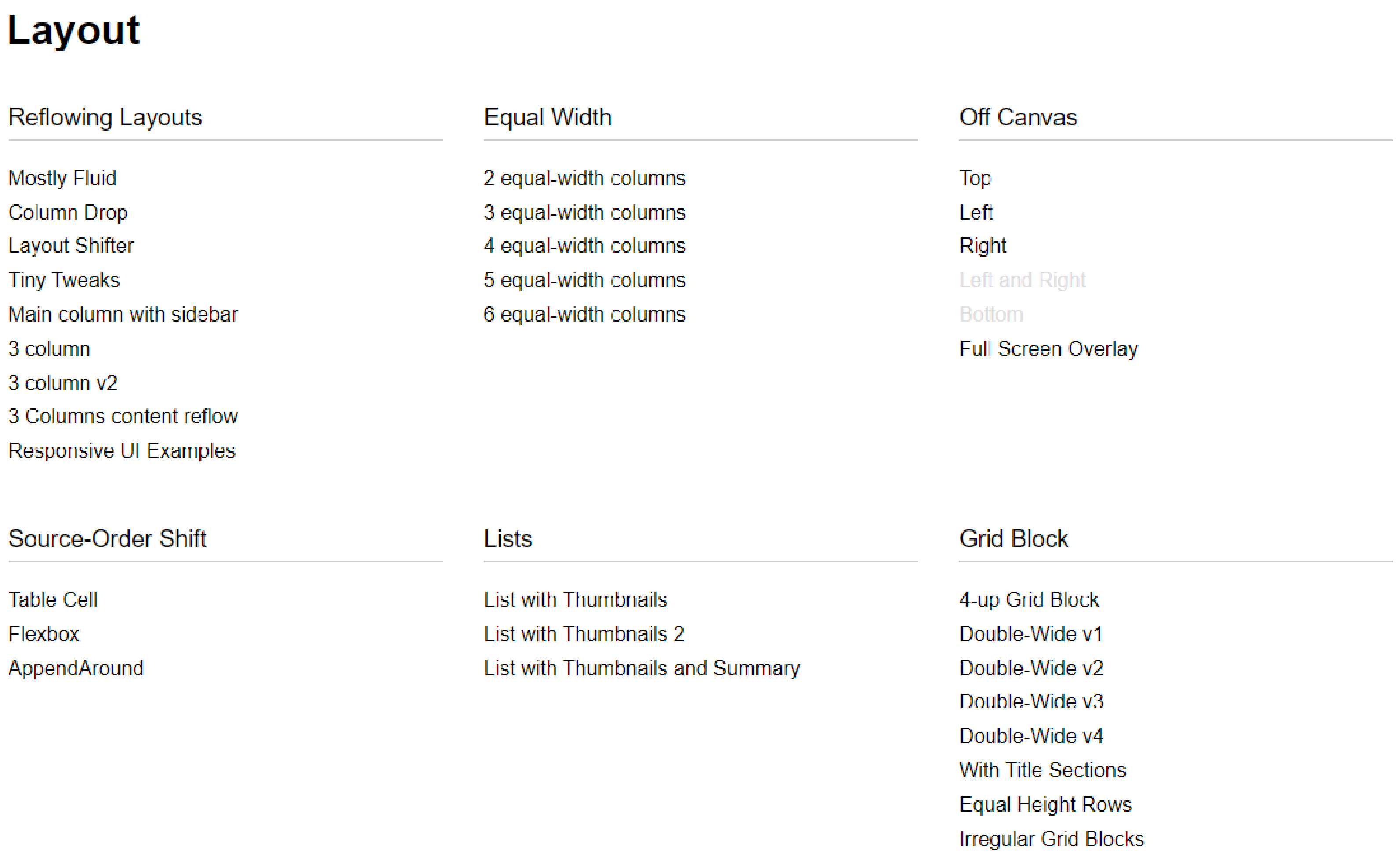
Miscellaneous

**Shopping**
Product page
Pricing table
Shopping Cart
Coupon

**Increasing frequency**
Tip A Friend

# Web and Responsive Design Patterns

- Responsive Design Patterns (shown)
- Javascript Design Patterns (mix of OO patterns and others)
- 579 Web Style Guides



## Layout

Reflowing Layouts

Mostly Fluid
Column Drop
Layout Shifter
Tiny Tweaks
Main column with sidebar
3 column
3 column v2
3 Columns content reflow
Responsive UI Examples

Source-Order Shift

Table Cell
Flexbox
AppendAround

Equal Width

2 equal-width columns
3 equal-width columns
4 equal-width columns
5 equal-width columns
6 equal-width columns

Lists

List with Thumbnails
List with Thumbnails 2
List with Thumbnails and Summary

Off Canvas

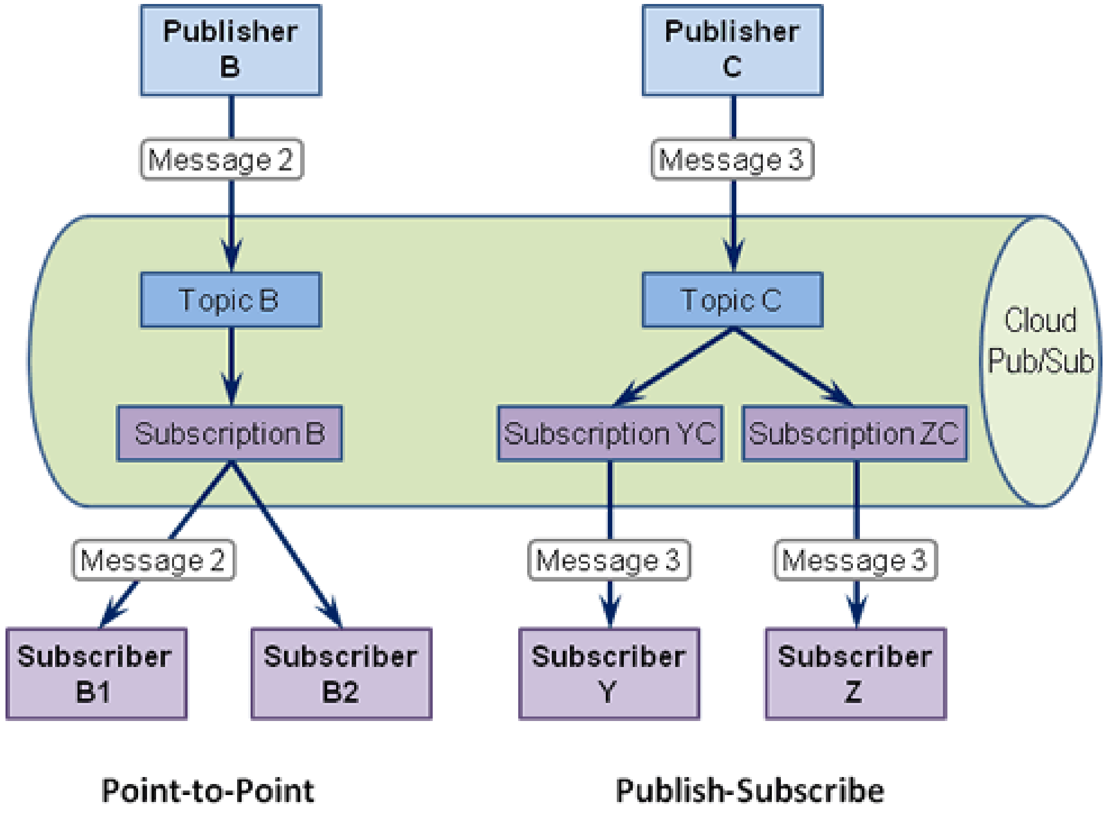
Top
Left
Right
Left and Right
Bottom
Full Screen Overlay

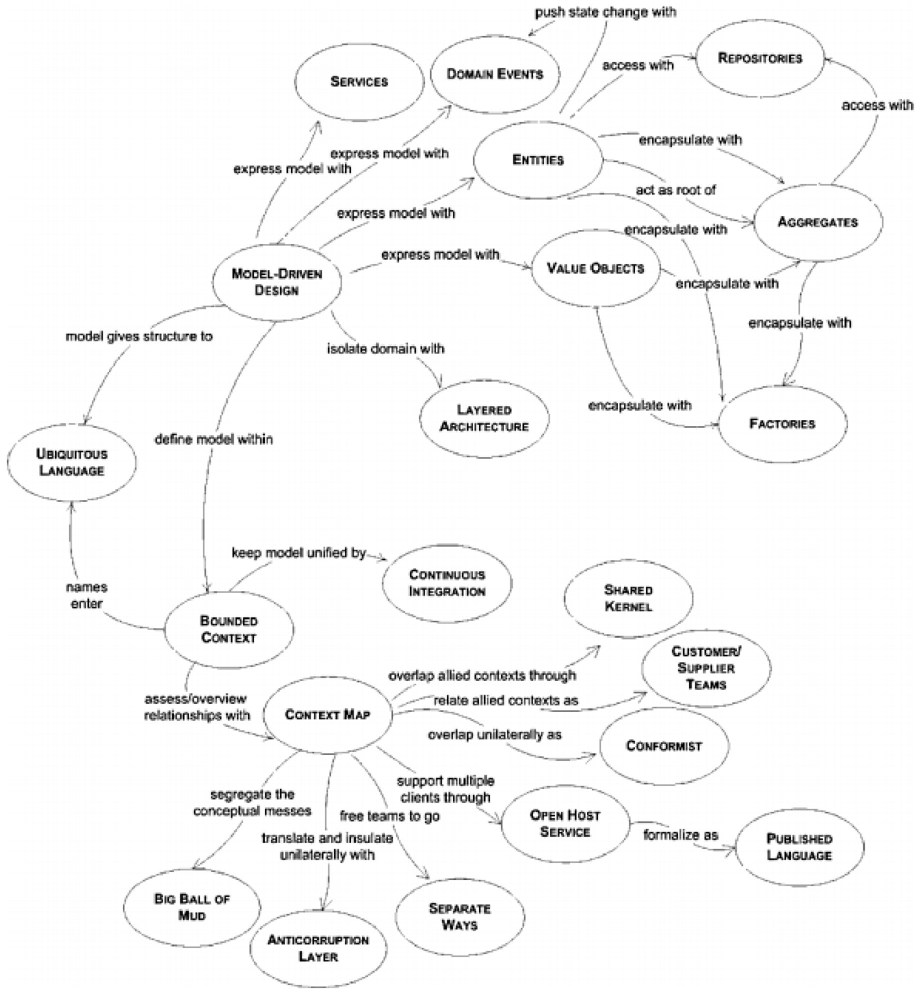
Grid Block

4-up Grid Block
Double-Wide v1
Double-Wide v2
Double-Wide v3
Double-Wide v4
With Title Sections
Equal Height Rows
Irregular Grid Blocks

# Architectural Patterns

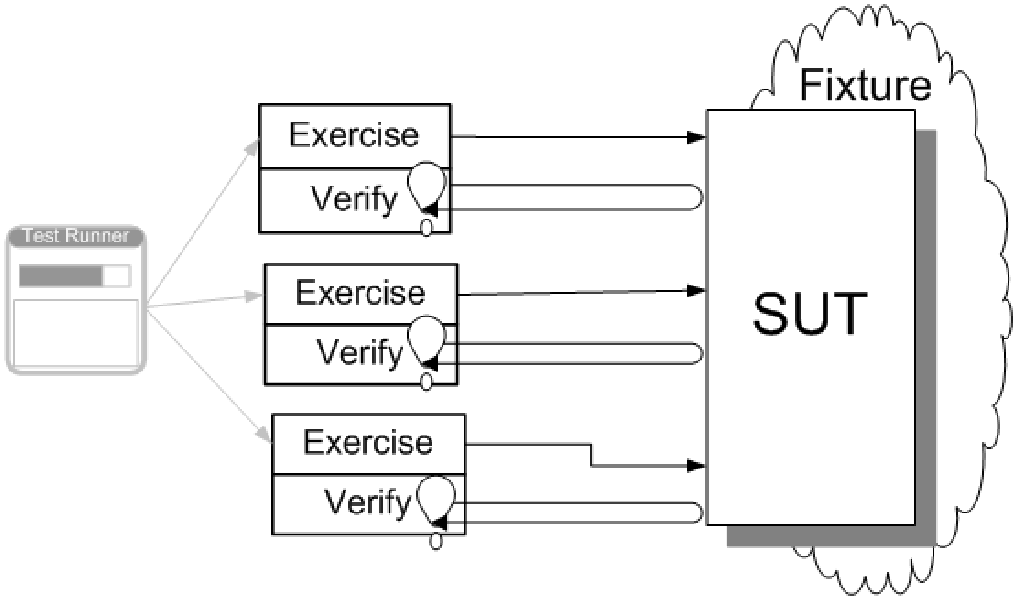
- Enterprise Integration Patterns (from the book by Fowler, a very complete web site for the book – typical diagram shown →)
- 10 Common Software Architectural Patterns in a nutshell
- Domain Driven Design Patterns (bottom image)

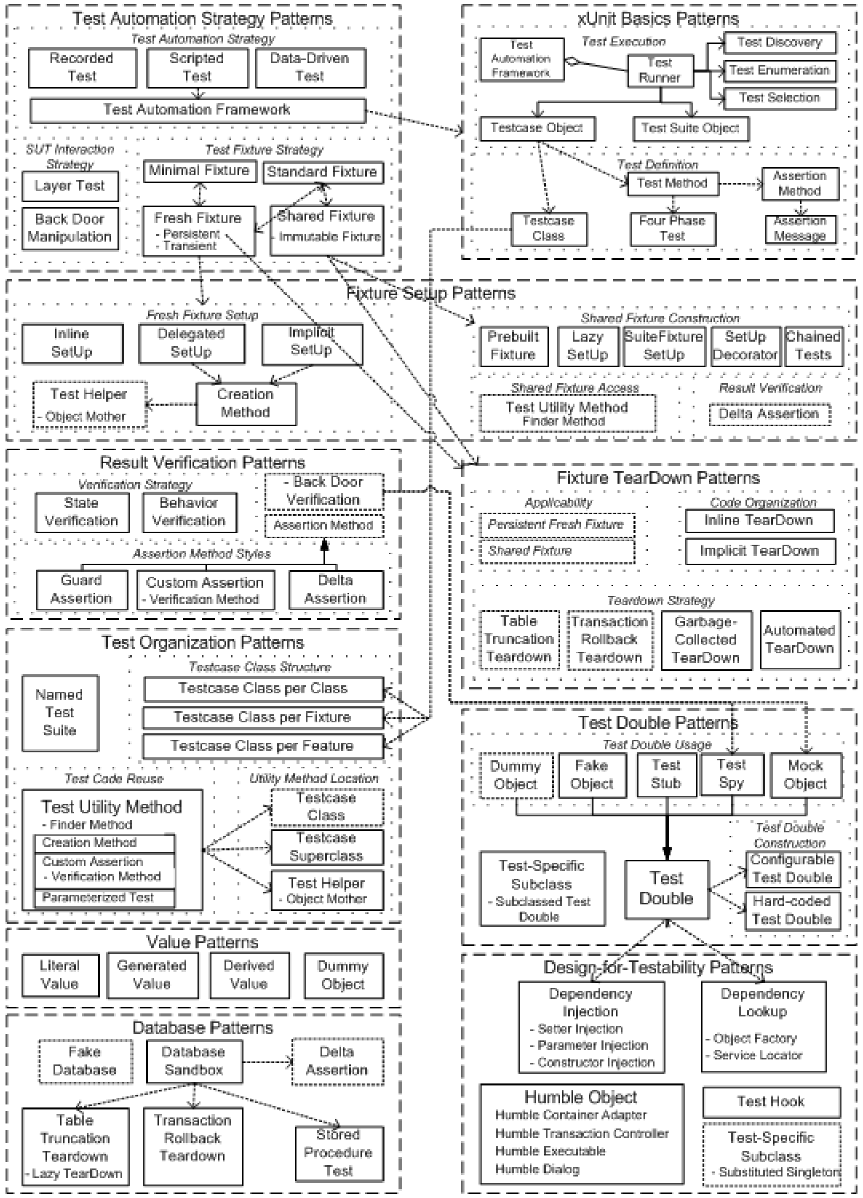




# Test/Automation Patterns

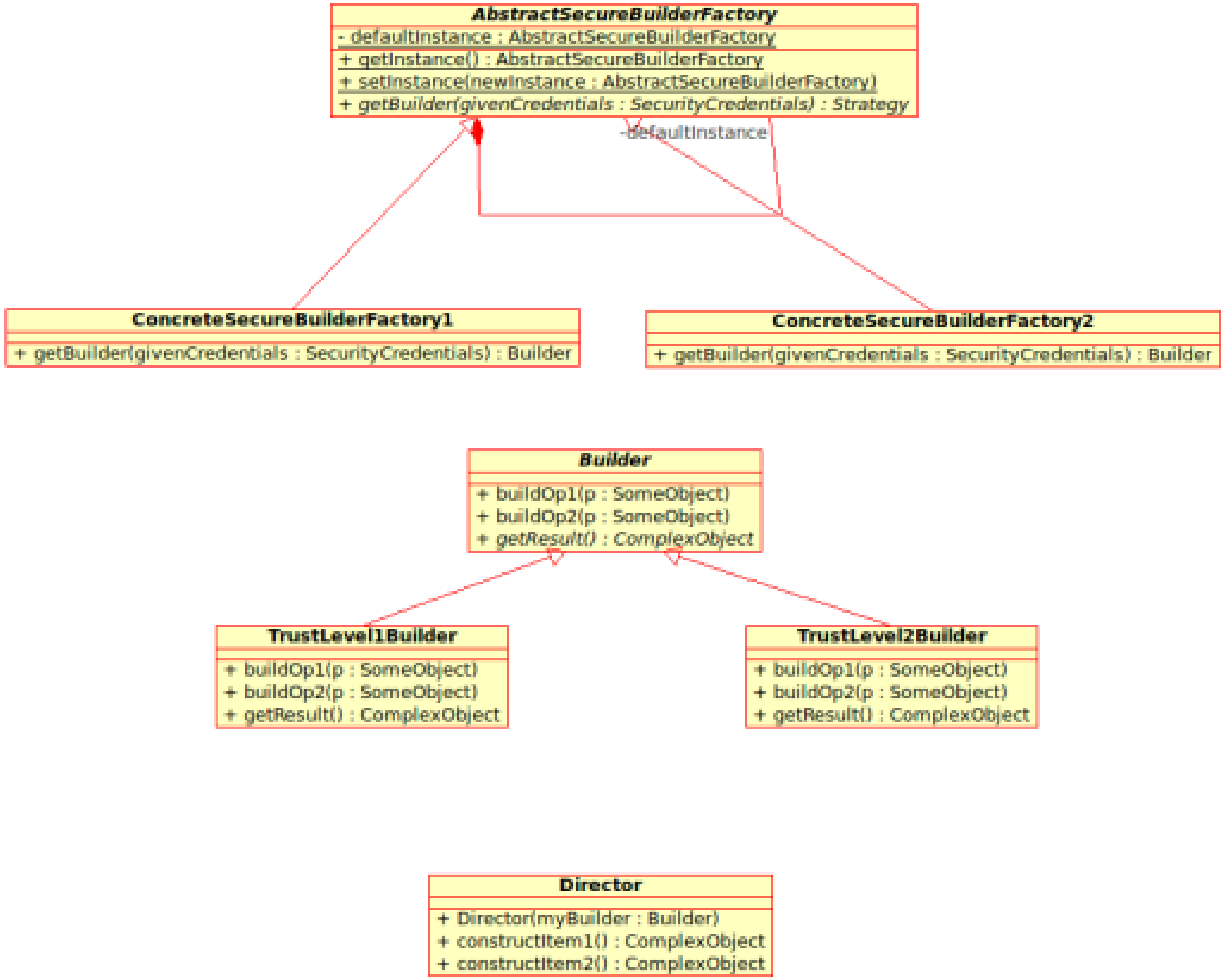
- [Test Automation Patterns Book (Axelrod)]()
- [XUnit Test Patterns]()
  - Test Automation
  - xUnit Basics
  - Fixture Setup
  - Result Verification
  - Test Organization
  - Values and Databases
  - Fixture Teardown
  - Test Doubles
  - Design for Testability





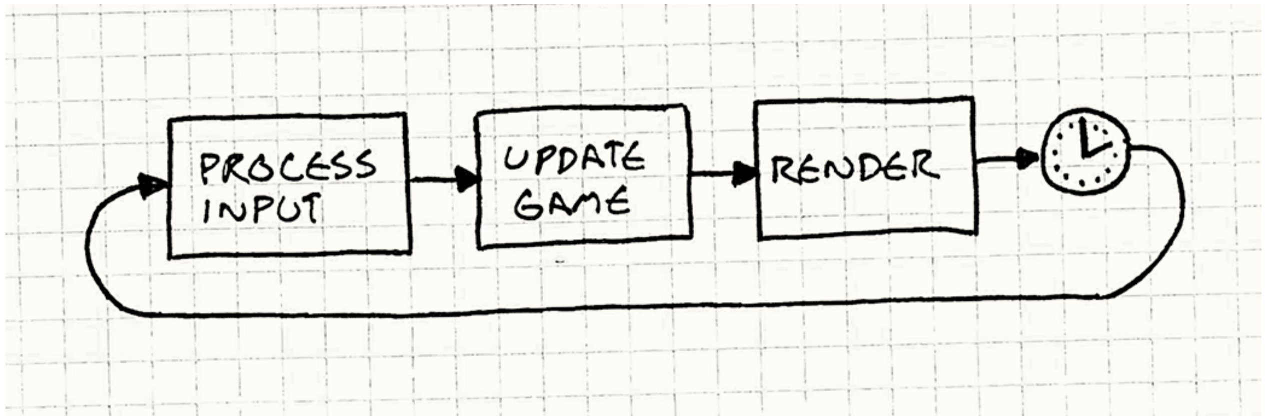
# Security Patterns

- Security Pattern Repository (UT San Antonio)
- Security Analysis and Patterns (OWASP)
- Secure Design Patterns (SEI)
    - Secure Builder Factory shown


AbstractSecureBuilderFactory
- defaultInstance : AbstractSecureBuilderFactory
+ getInstance() : AbstractSecureBuilderFactory
+ setInstance(newInstance : AbstractSecureBuilderFactory)
+ getBuilder(givenCredentials : SecurityCredentials) : Strategy
-defaultInstance
ConcreteSecureBuilderFactory1
+ getBuilder(givenCredentials : SecurityCredentials) : Builder
ConcreteSecureBuilderFactory2
+ getBuilder(givenCredentials : SecurityCredentials) : Builder
Builder
+ buildOp1(p : SomeObject)
+ buildOp2(p : SomeObject)
+ getResult() : ComplexObject
TrustLevel1Builder
+ buildOp1(p : SomeObject)
+ buildOp2(p : SomeObject)
+ getResult() : ComplexObject
TrustLevel2Builder
+ buildOp1(p : SomeObject)
+ buildOp2(p : SomeObject)
+ getResult() : ComplexObject
Director
+ Director(myBuilder : Builder)
+ constructItem1() : ComplexObject
+ constructItem2() : ComplexObject

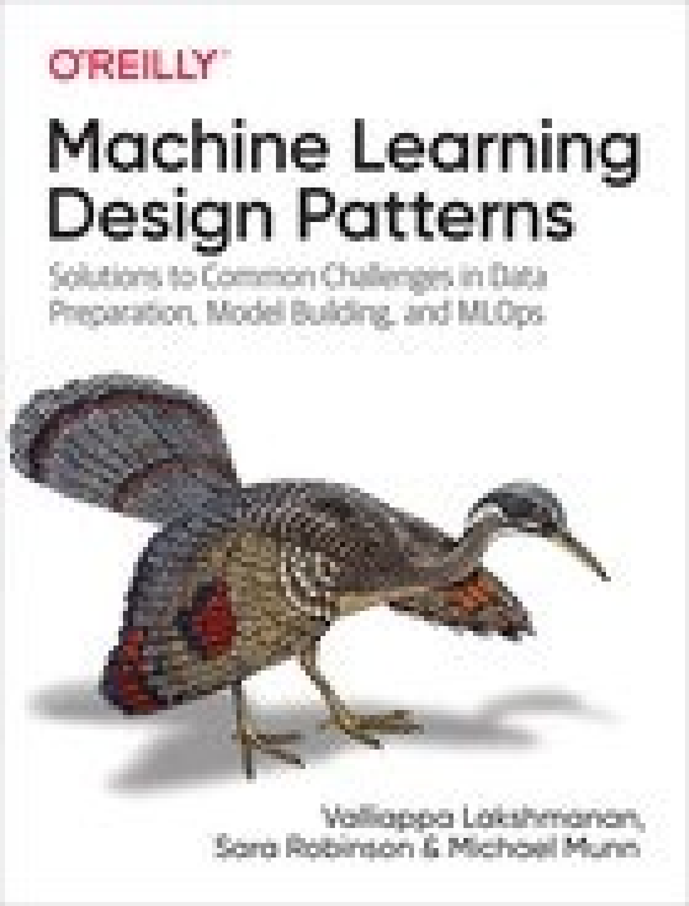
# Game Programming Patterns

- Patterns applied/developed for game applications!
- Discusses common OO patterns in games
- Adds game specific patterns for
  - Sequencing
    - Game Loop →
  - Behavior
  - Decoupling
  - Optimization
- Relates solutions to tools like Unity
- Robert Nystrom at [https://gameprogrammingpatterns.com/](https://gameprogrammingpatterns.com/)



# Machine Learning Design Patterns

- Book on ML Concerns and Design Patterns
- Data Representation, Problem Representation, Modifying the Training Loop, Reproducability, Resilience, Responsible AI

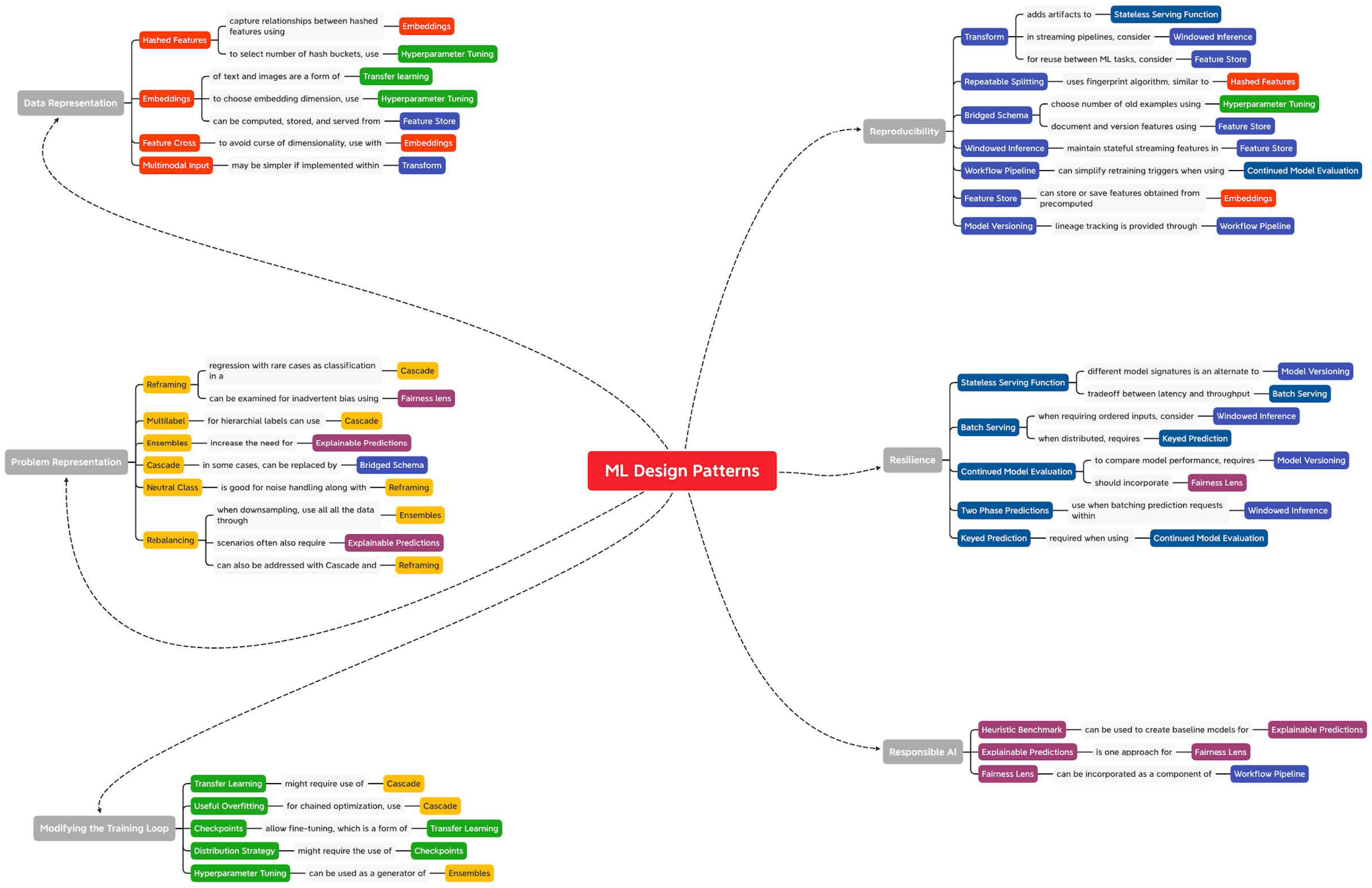


O'REILLY

Machine Learning Design Patterns

Solutions to Common Challenges in Data Preparation, Model Building, and MLOps

Valliappa Lakshmanan, Sara Robinson & Michael Munn



ML Design Patterns

Data Representation
- Hashed Features — capture relationships between hashed features using — Embeddings
- Hashed Features — to select number of hash buckets, use — Hyperparameter Tuning
- Embeddings — of text and images are a form of — Transfer learning
- Embeddings — to choose embedding dimension, use — Hyperparameter Tuning
- Embeddings — can be computed, stored, and served from — Feature Store
- Feature Cross — to avoid curse of dimensionality, use with — Embeddings
- Multimodal Input — may be simpler if implemented within — Transform

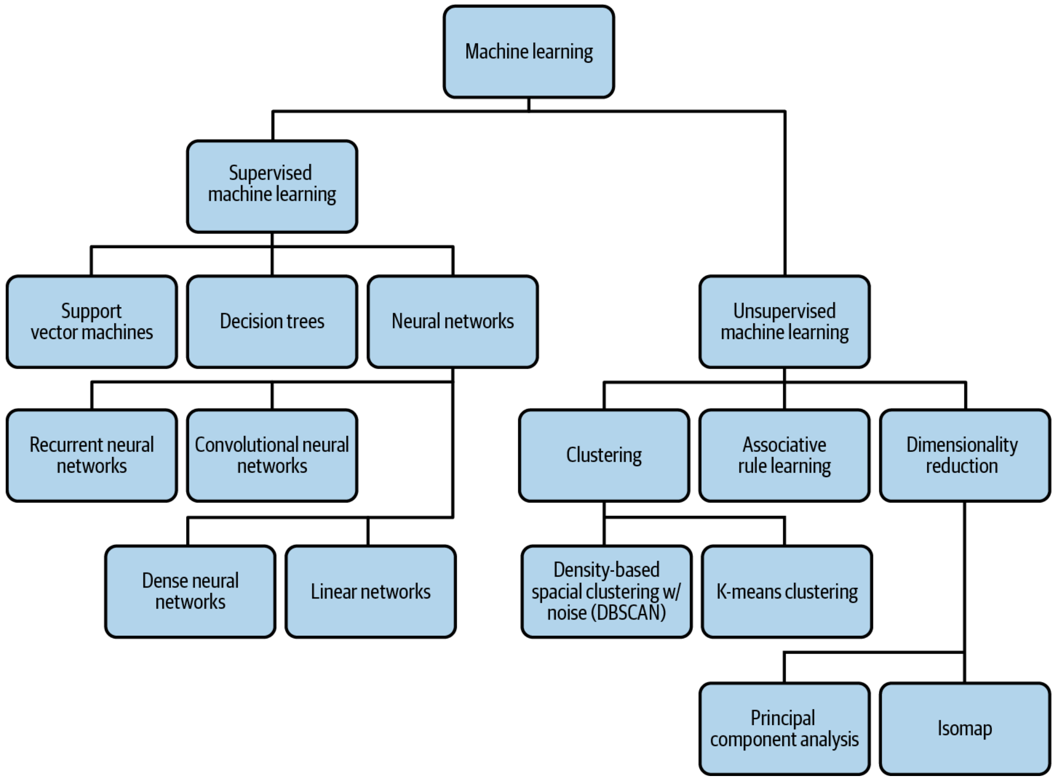
Reproducibility
- Transform — adds artifacts to — Stateless Serving Function
- Transform — in streaming pipelines, consider — Windowed Inference
- Transform — for reuse between ML tasks, consider — Feature Store
- Repeatable Splitting — uses fingerprint algorithm, similar to — Hashed Features
- Bridged Schema — choose number of old examples using — Hyperparameter Tuning
- Bridged Schema — document and version features using — Feature Store
- Windowed Inference — maintain stateful streaming features in — Feature Store
- Workflow Pipeline — can simplify retraining triggers when using — Continued Model Evaluation
- Feature Store — can store or save features obtained from precomputed — Embeddings
- Model Versioning — lineage tracking is provided through — Workflow Pipeline

Problem Representation
- Reframing — regression with rare cases as classification in a — Cascade
- Reframing — can be examined for inadvertent bias using — Fairness lens
- Multilabel — for hierarchial labels can use — Cascade
- Ensembles — increase the need for — Explainable Predictions
- Cascade — in some cases, can be replaced by — Bridged Schema
- Neutral Class — is good for noise handling along with — Reframing
- Rebalancing — when downsampling, use all all the data through — Ensembles
- Rebalancing — scenarios often also require — Explainable Predictions
- Rebalancing — can also be addressed with Cascade and — Reframing

Resilience
- Stateless Serving Function — different model signatures is an alternate to — Model Versioning
- Stateless Serving Function — tradeoff between latency and throughput — Batch Serving
- Batch Serving — when requiring ordered inputs, consider — Windowed Inference
- Batch Serving — when distributed, requires — Keyed Prediction
- Continued Model Evaluation — to compare model performance, requires — Model Versioning
- Continued Model Evaluation — should incorporate — Fairness Lens
- Two Phase Predictions — use when batching prediction requests within — Windowed Inference
- Keyed Prediction — required when using — Continued Model Evaluation

Modifying the Training Loop
- Transfer Learning — might require use of — Cascade
- Useful Overfitting — for chained optimization, use — Cascade
- Checkpoints — allow fine-tuning, which is a form of — Transfer Learning
- Distribution Strategy — might require the use of — Checkpoints
- Hyperparameter Tuning — can be used as a generator of — Ensembles

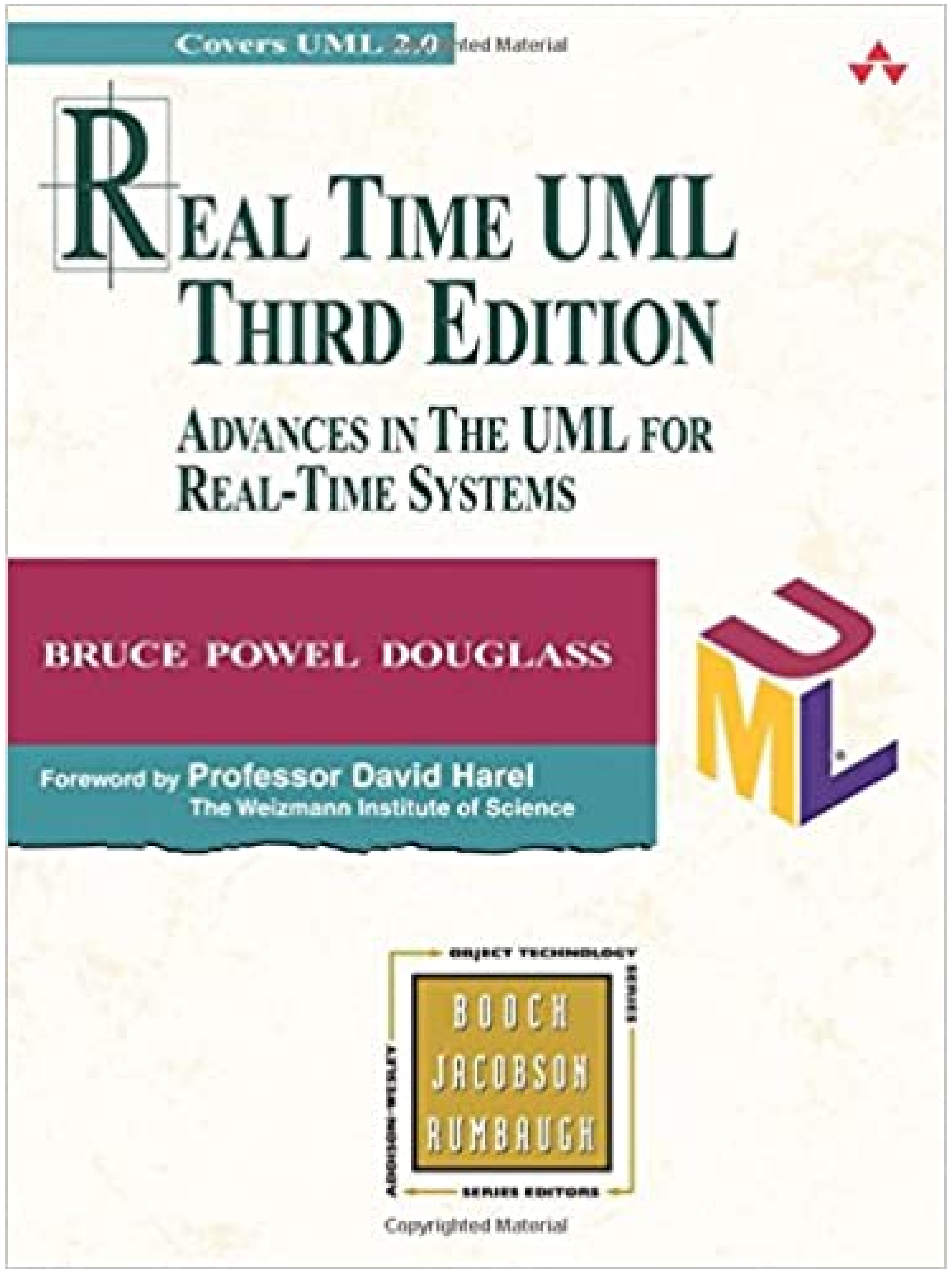
Responsible AI
- Heuristic Benchmark — can be used to create baseline models for — Explainable Predictions
- Explainable Predictions — is one approach for — Fairness Lens
- Fairness Lens — can be incorporated as a component of — Workflow Pipeline



Machine learning
- Supervised machine learning
  - Support vector machines
  - Decision trees
  - Neural networks
    - Recurrent neural networks
    - Convolutional neural networks
    - Dense neural networks
    - Linear networks
- Unsupervised machine learning
  - Clustering
    - Density-based spacial clustering w/ noise (DBSCAN)
    - K-means clustering
  - Associative rule learning
  - Dimensionality reduction
    - Principal component analysis
    - Isomap

# Real-time Embedded Patterns

- Real-Time Design Patterns (Douglas): http://www.uml.org.cn/UMLApplication/pdf/rtpatterns.pdf
- Book - Real Time UML: Advances in the UML for Real-Time Systems (3rd Edition) 3rd Edition (Douglass, 2004, Addison-Wesley)



Covers UML 2.0 ted Material

Real Time UML
Third Edition

Advances in The UML for
Real-Time Systems

Bruce Powel Douglass

Foreword by Professor David Harel
The Weizmann Institute of Science

Object Technology Series

Booch
Jacobson
Rumbaugh

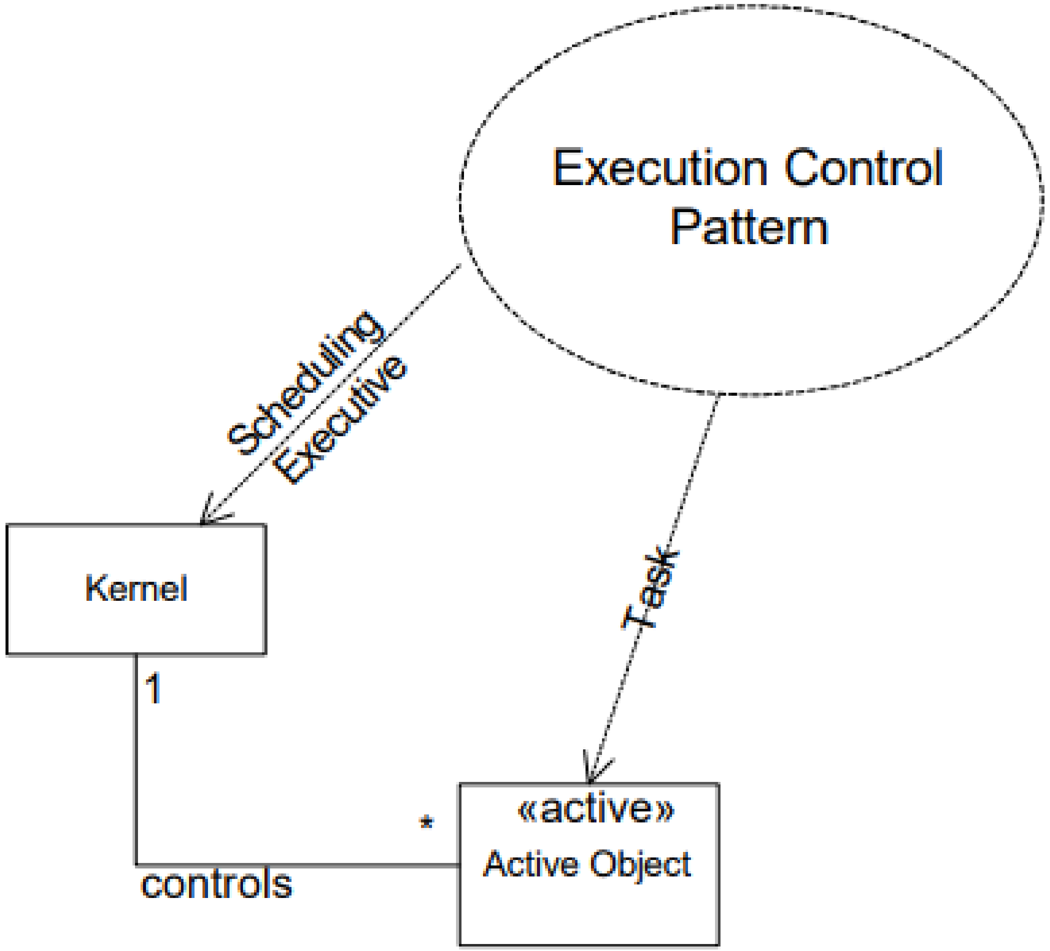
Series Editors

Addison-Wesley

Copyrighted Material

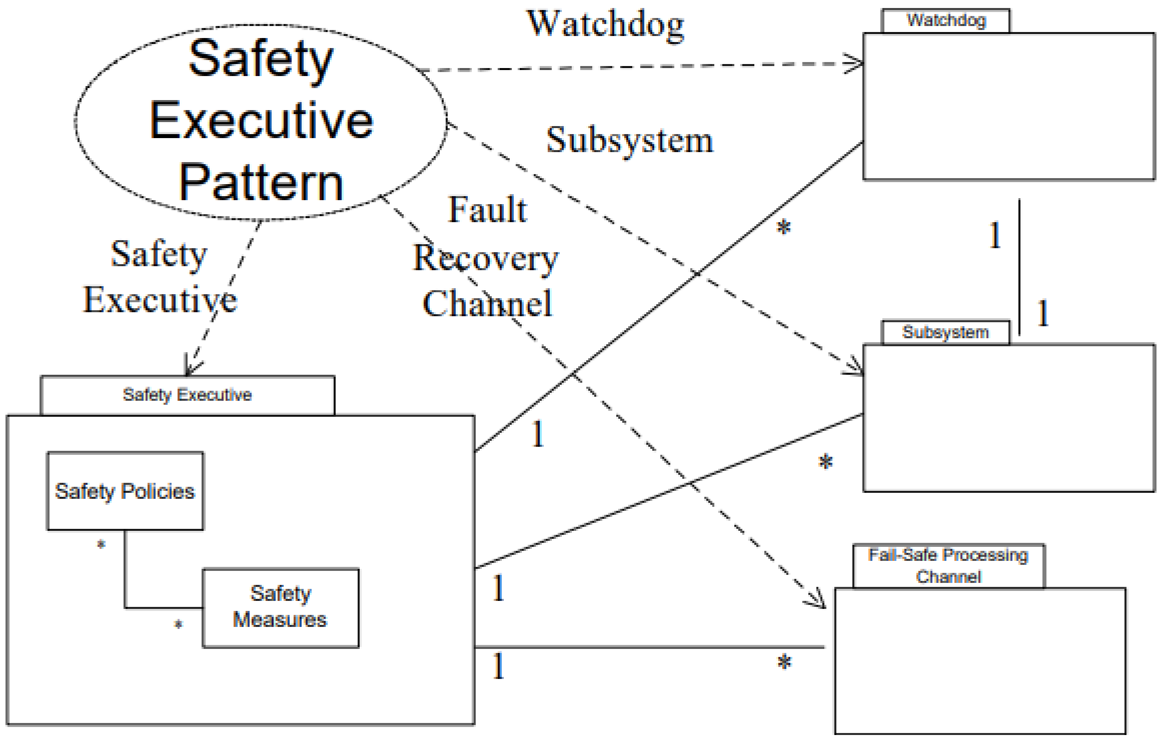
# Real-Time Embedded Patterns

- Execution Control
  - Preemptive Multitasking
  - Cyclic Executive
  - Time Slicing
  - Cooperative Multitasking
  - Communications
- Master-Slave
  - Time-division Multiplexing Access
  - Bus-mastered
- Reuse
  - Microkernel (layered system structure)
- Distributed Systems
  - Proxy (known connections)
  - Broker (unknown connections)
  - Asymmetric Processing – dedicated nodes
  - Symmetric Processing – dynamic load allocation
  - Semi-symmetric processing – as available



# Real-Time Embedded Patterns

- Resource
  - Static allocation
  - Fixed-size allocation
  - Priority ceiling – avoiding priority inversion
- Safety & Reliability
  - Homogeneous Redundancy – identical processing channels to avoid faults
  - Heterogeneous Redundancy – diverse processing channels
  - Sanity check – One channel processes, another checks it
  - Monitor-Actuator – One channel processes, another monitors performance
  - Watchdog
  - Safety Executive – Central safety monitor to id and recover from faults

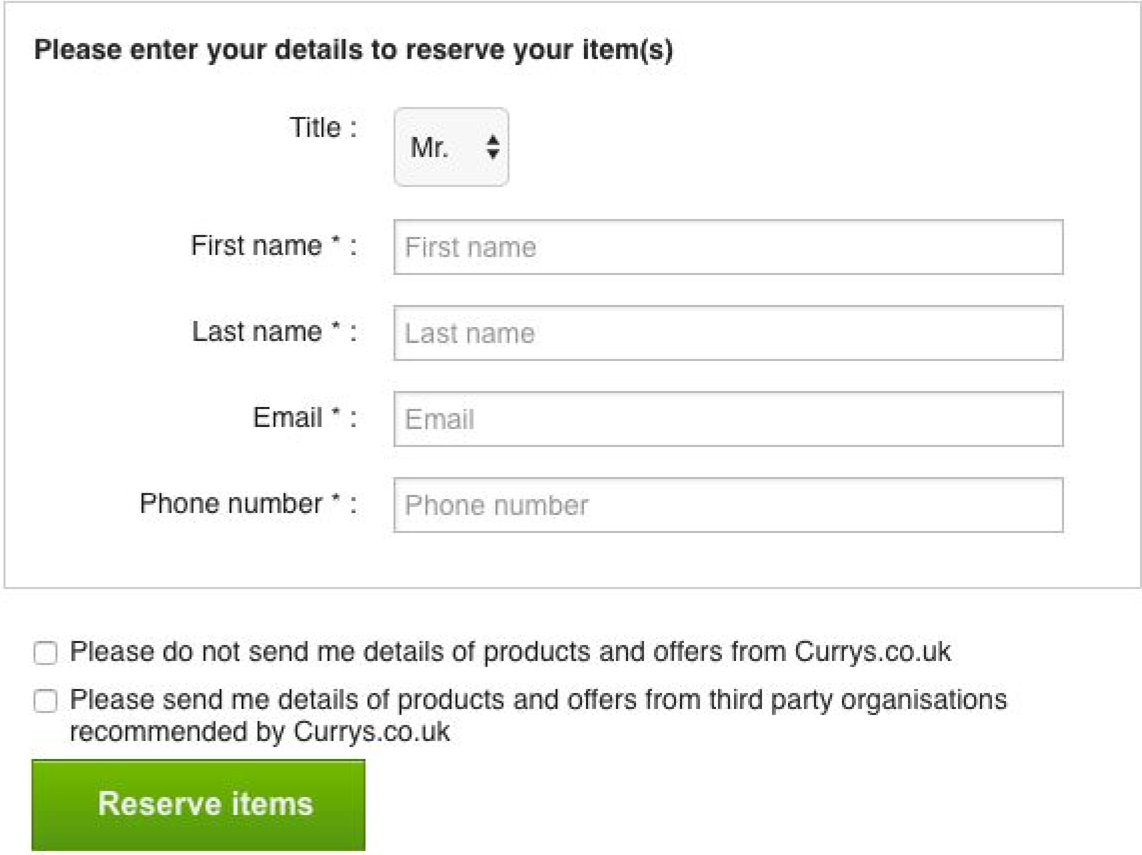


# Dark Patterns?

- (Not really patterns)
- Dark Patterns are tricks used in websites and apps that make you do things that you didn't mean to, like buying or signing up for something
- Darkpatterns.org wants to spread awareness and to shame companies that use them
- Came from an initial blog post by Harry Brignull, who runs Darkpatterns.com [4]

# Types of Dark Patterns

- Trick Questions →
  - While filling in a form you respond to a question that tricks you into giving an answer you didn't intend. When glanced upon quickly the question appears to ask one thing, but when read carefully it asks another thing entirely
- Sneak into Basket
  - You attempt to purchase something, but somewhere in the purchasing journey the site sneaks an additional item into your basket, often through the use of an opt-out radio button or checkbox on a prior page
- Roach Motel
  - You get into a situation very easily, but then you find it is hard to get out of it (e.g. a premium subscription)



**Please enter your details to reserve your item(s)**

Title : Mr.

First name * : First name

Last name * : Last name

Email * : Email

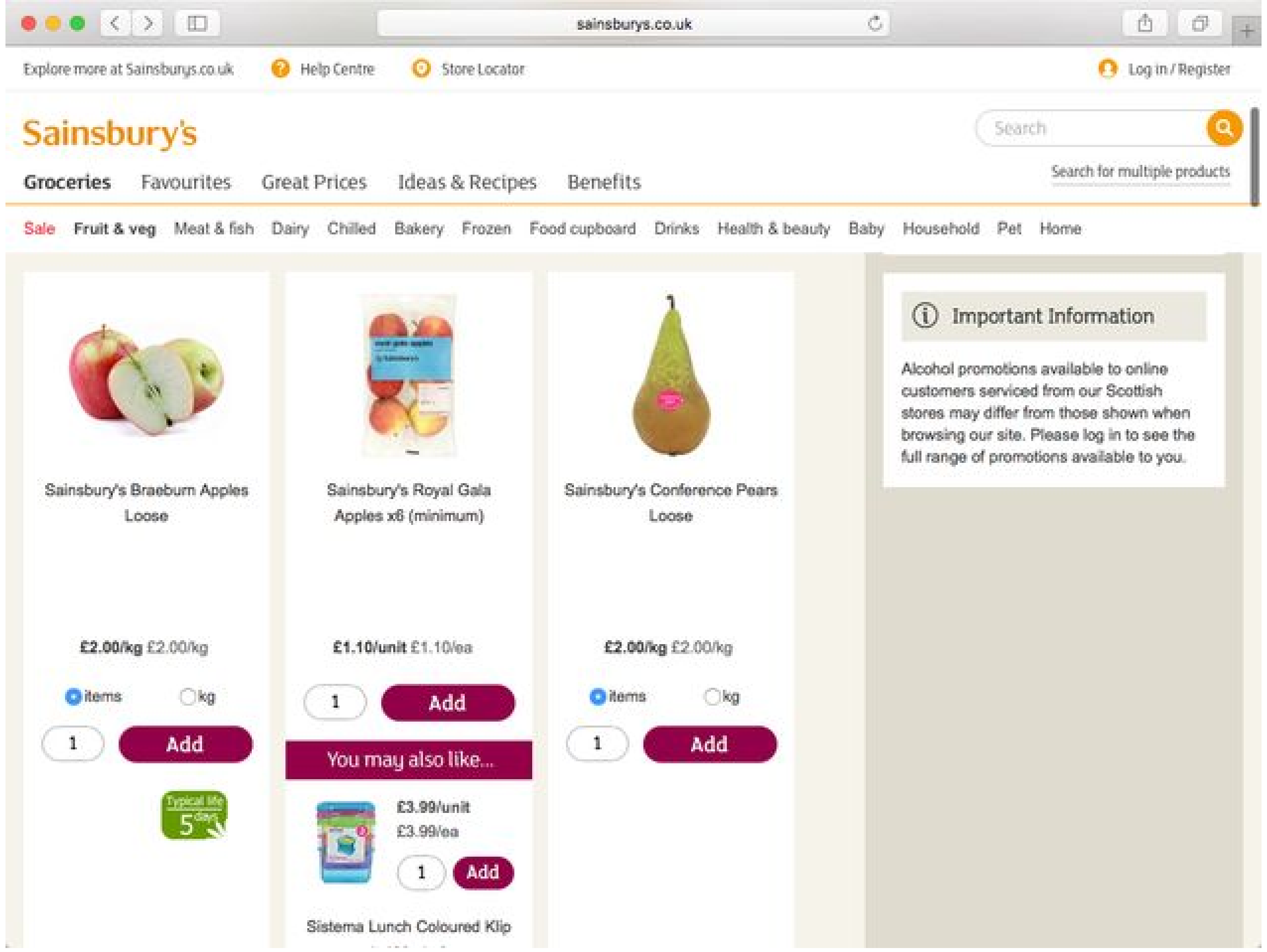
Phone number * : Phone number

□ Please do not send me details of products and offers from Currys.co.uk

□ Please send me details of products and offers from third party organisations recommended by Currys.co.uk

# Types of Dark Patterns

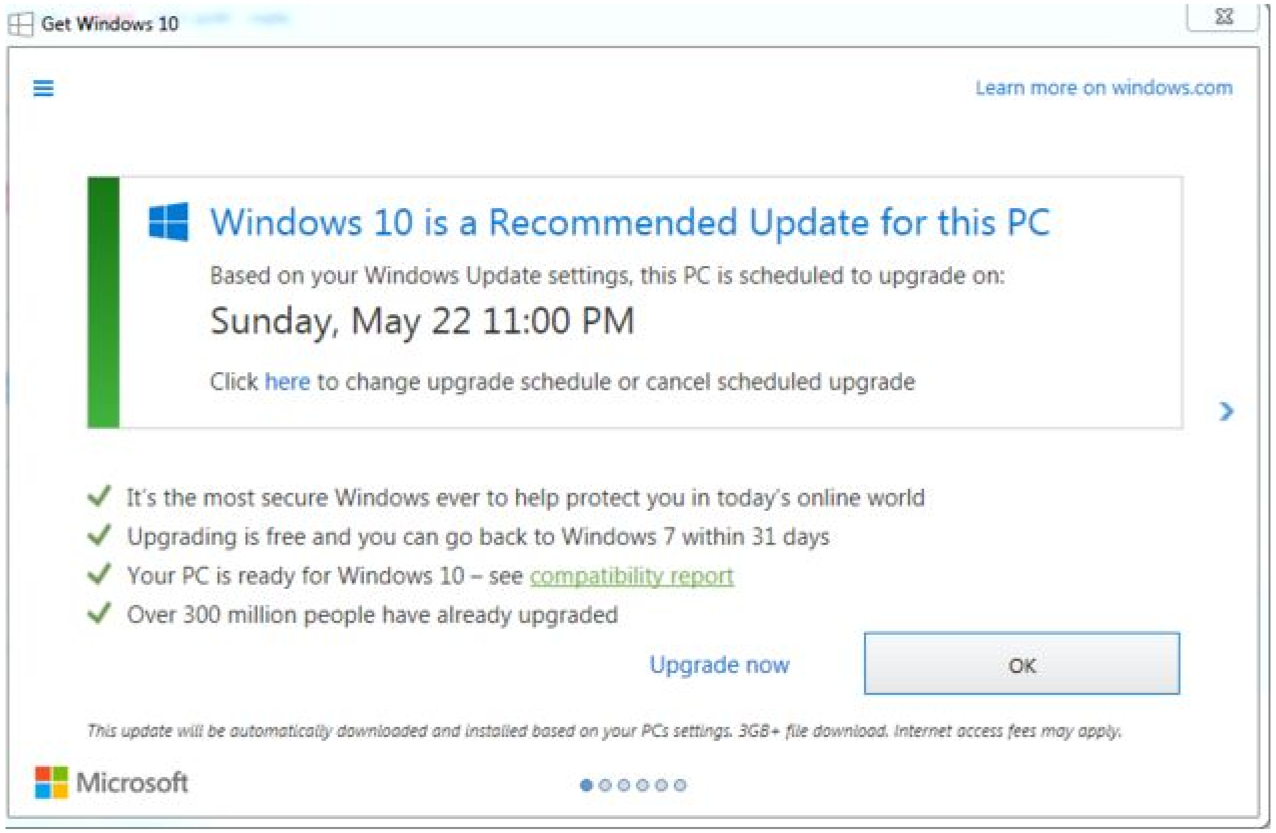
- Privacy Zuckering
  - You are tricked into publicly sharing more information about yourself than you really intended to; Named after Facebook CEO Mark Zuckerberg
- Price Comparison Prevention →
  - The retailer makes it hard for you to compare the price of an item with another item, so you cannot make an informed decision
- Misdirection
  - The design purposefully focuses your attention on one thing in order to distract your attention from another



Alcohol promotions available to online customers serviced from our Scottish stores may differ from those shown when browsing our site. Please log in to see the full range of promotions available to you.

# Types of Dark Patterns

- Hidden Costs
  - You get to the last step of the checkout process, only to discover some unexpected charges have appeared, e.g. delivery charges, tax, etc
- Bait and Switch →
  - You set out to do one thing, but a different, undesirable thing happens instead
- Confirmshaming
  - The act of guilting the user into opting into something; the option to decline is worded in such a way as to shame the user into compliance



Windows 10 is a Recommended Update for this PC

Based on your Windows Update settings, this PC is scheduled to upgrade on:

Sunday, May 22 11:00 PM

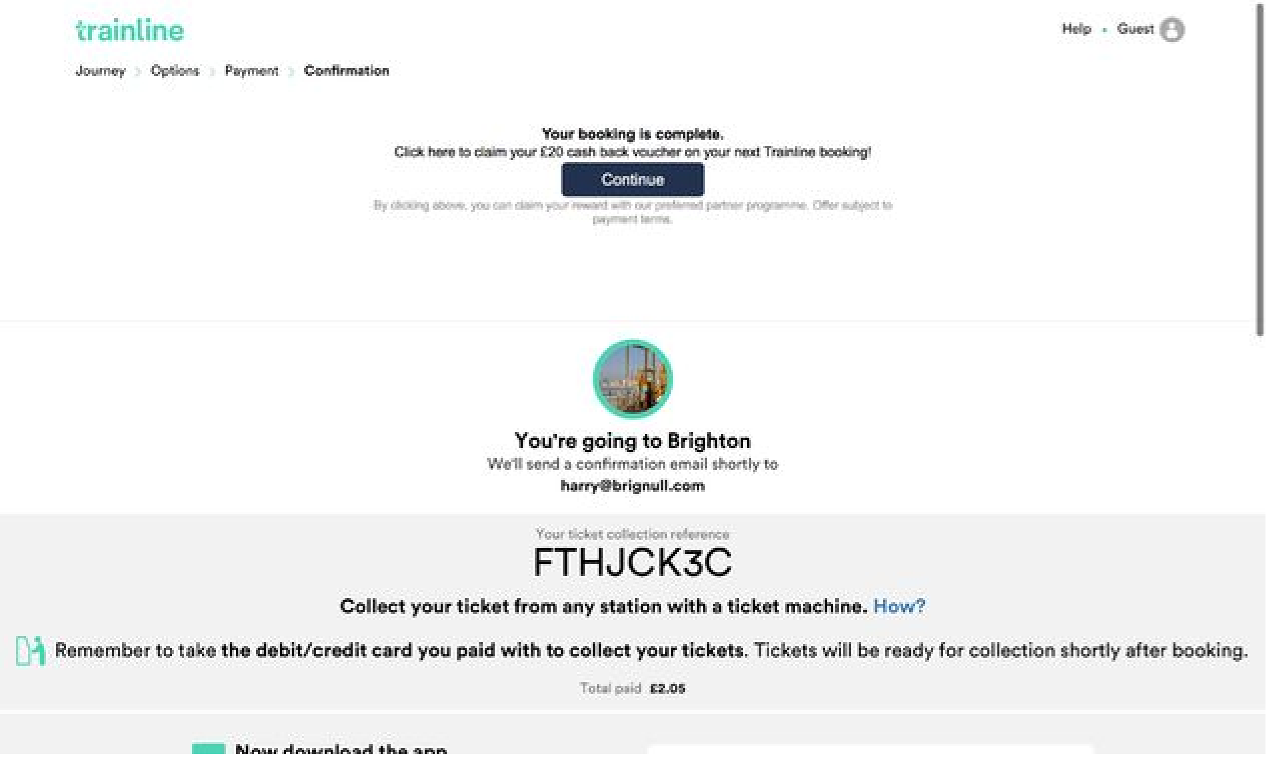
Click here to change upgrade schedule or cancel scheduled upgrade

- ✓ It's the most secure Windows ever to help protect you in today's online world
- ✓ Upgrading is free and you can go back to Windows 7 within 31 days
- ✓ Your PC is ready for Windows 10 – see compatibility report
- ✓ Over 300 million people have already upgraded

*This update will be automatically downloaded and installed based on your PCs settings. 3GB+ file download. Internet access fees may apply.*

# Types of Dark Patterns

- Disguised Ads →
  - Adverts that are disguised as other kinds of content or navigation, in order to get you to click on them
- Forced Continuity
  - When your free trial with a service comes to an end and your credit card silently starts getting charged without any warning; in some cases this is made even worse by making it difficult to cancel the membership
- Friend Spam
  - The product asks for your email or social media permissions under the pretense it will be used for a desirable outcome (e.g. finding friends), but then spams all your contacts in a message that claims to be from you



**Your booking is complete.**
Click here to claim your £20 cash back voucher on your next Trainline booking!

By clicking above, you can claim your reward with our preferred partner programme. Offer subject to payment terms.

**You're going to Brighton**
We'll send a confirmation email shortly to
**harry@brignull.com**

Your ticket collection reference
FTHJCK3C

**Collect your ticket from any station with a ticket machine.** How?

Remember to take **the debit/credit card you paid with to collect your tickets.** Tickets will be ready for collection shortly after booking.

Total paid **£2.05**

# References

[1] AntiPatterns, Brown et al., Wiley, 1998 (book on CU Skillsoft)

[2] http://antipatterns.com/briefing/sld007.htm

[3] https://www.darkpatterns.org/

[4] https://www.90percentofeverything.com/2010/07/08/dark-patterns-dirty-tricks-designers-use-to-make-people-do-stuff/

# Next Steps

- Graduate Pecha Kucha **due today**
  - **You must sign up for a Wed/Fri Pecha Kucha class presentation slot here:** https://docs.google.com/document/d/1EGvA3ZnKVhheJqdRUp7obqG7n3sLQPqCtV8mxwJhB0U/edit?usp=sharing
- Project 7 **due Wed 12/8 at 8 PM**
  - Includes report, code, and recorded demonstration
- Graduate Final Research Presentation **due Wed 12/8 at 8 PM**
  - Details on assignments in Canvas Files/Class Files
- New Quiz open now **due 12/1 –** opportunity for class feedback
- Bruce will be at normal office hours for the next two weeks

- Last two weeks of class
  - Week of 11/29
    - Anti- and Other Patterns
    - Graduate Pecha Kuchas (bonus points for attendance)
  - Week of 12/6
    - Final Review
    - (New) Comparative Software Design Philosophy
    - Other than OOAD
    - Class Wrap-up
- New Piazza topic this week for your comments for Participation Grade, last topic post soon
- Piazza article posts now available for extra bonus points (if you're not getting points in class); will add those bonus points in **after 12/8**
- Find a class staff member if you need anything!