Antipatterns & Other Patterns

CSCI 4448/5448: Object-Oriented Analysis & Design

Lecture 38

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

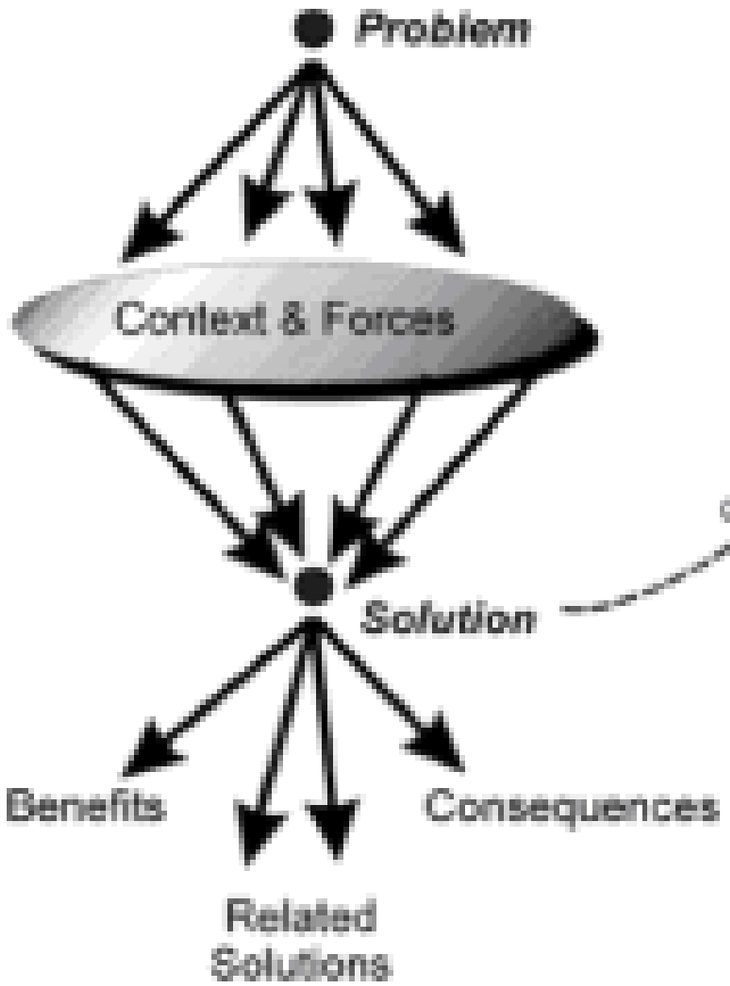
AntiPatterns?

- 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

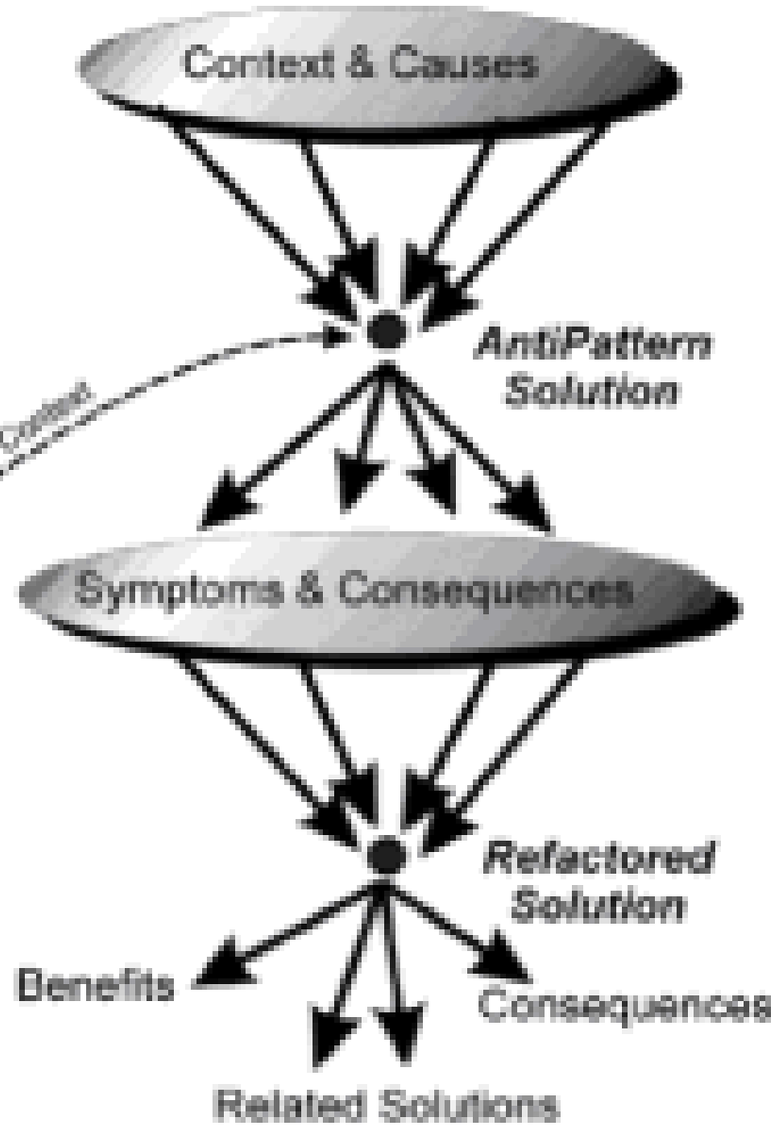
AntiPatterns???

- 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



Design Patterns



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)

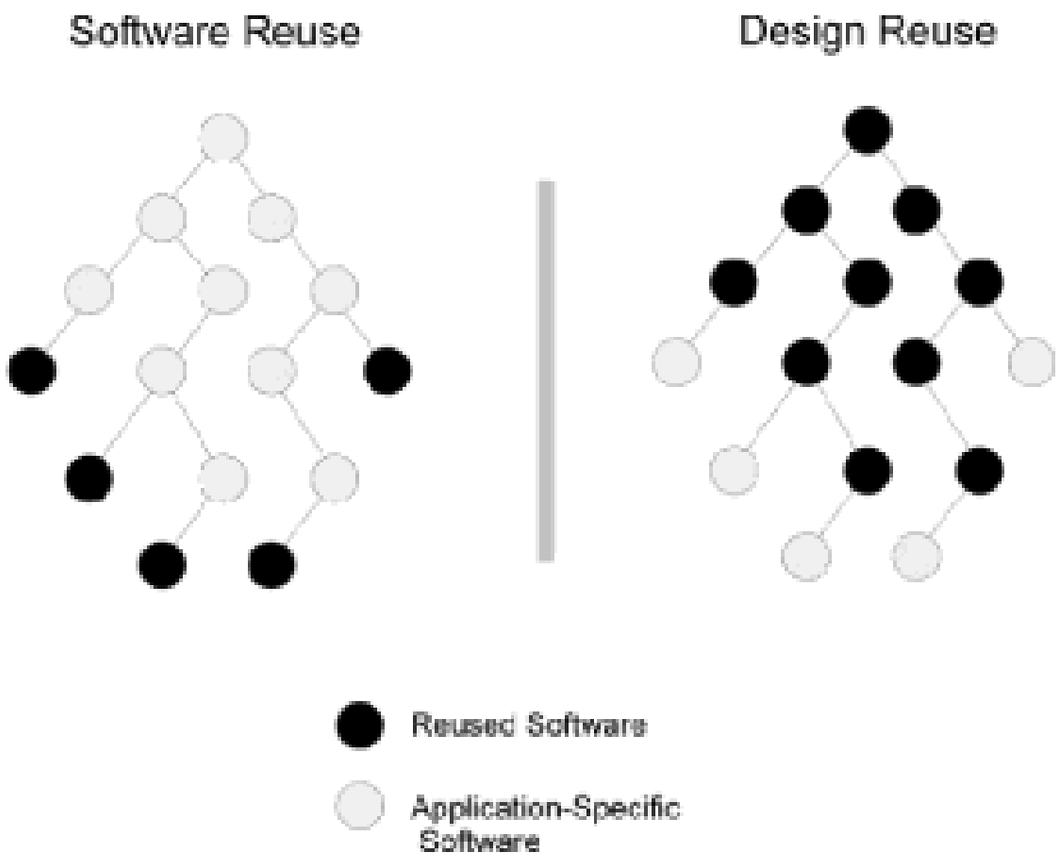
Primal Forces

- 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

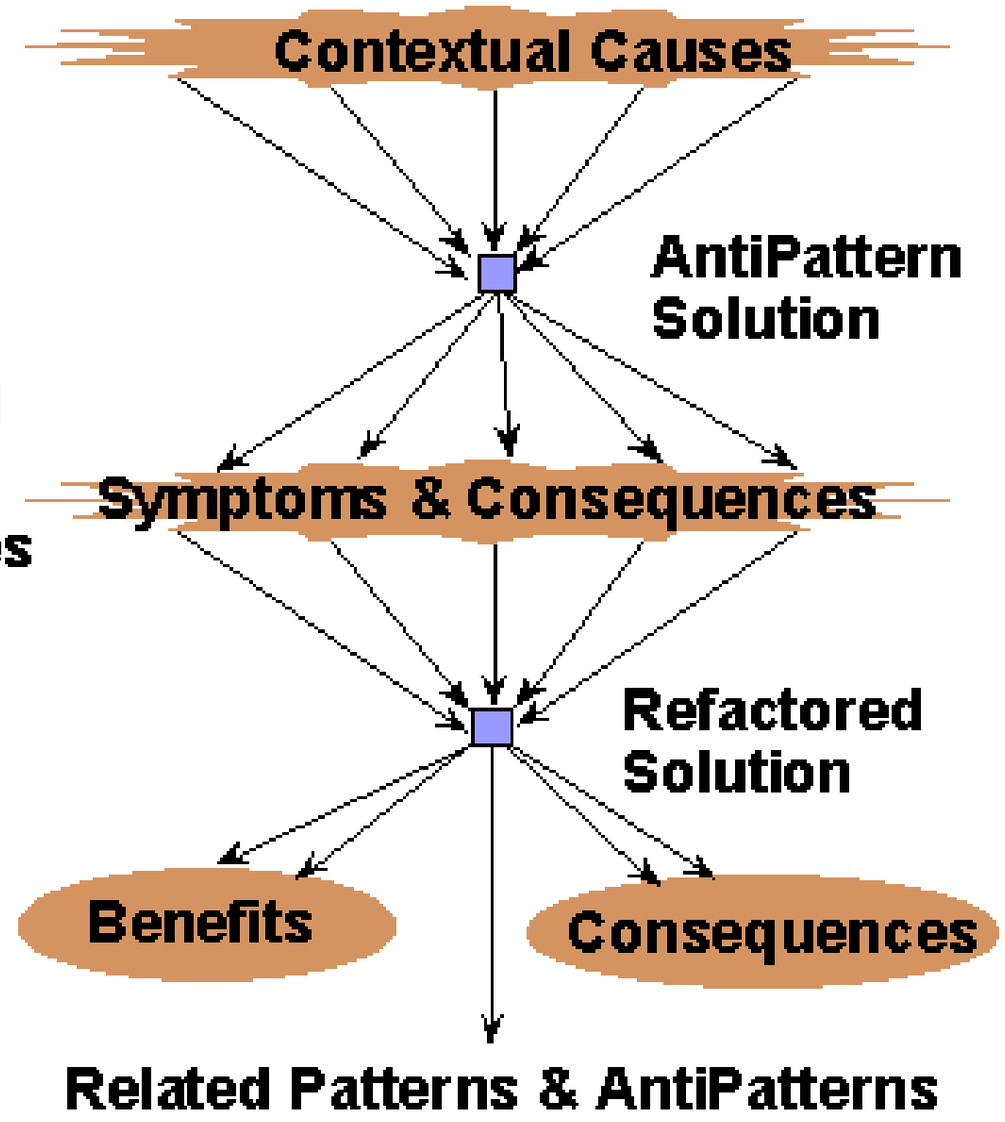
- 7 levels: global, enterprise, system, application, macro-component (frameworks), micro-component, and object
- More focus on design reuse than software reuse needed



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

- UI/UX Patterns
- Responsive Patterns
- Architectural Patterns
- Test/Automation Patterns
- Security Patterns
- Real-time Embedded Patterns

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

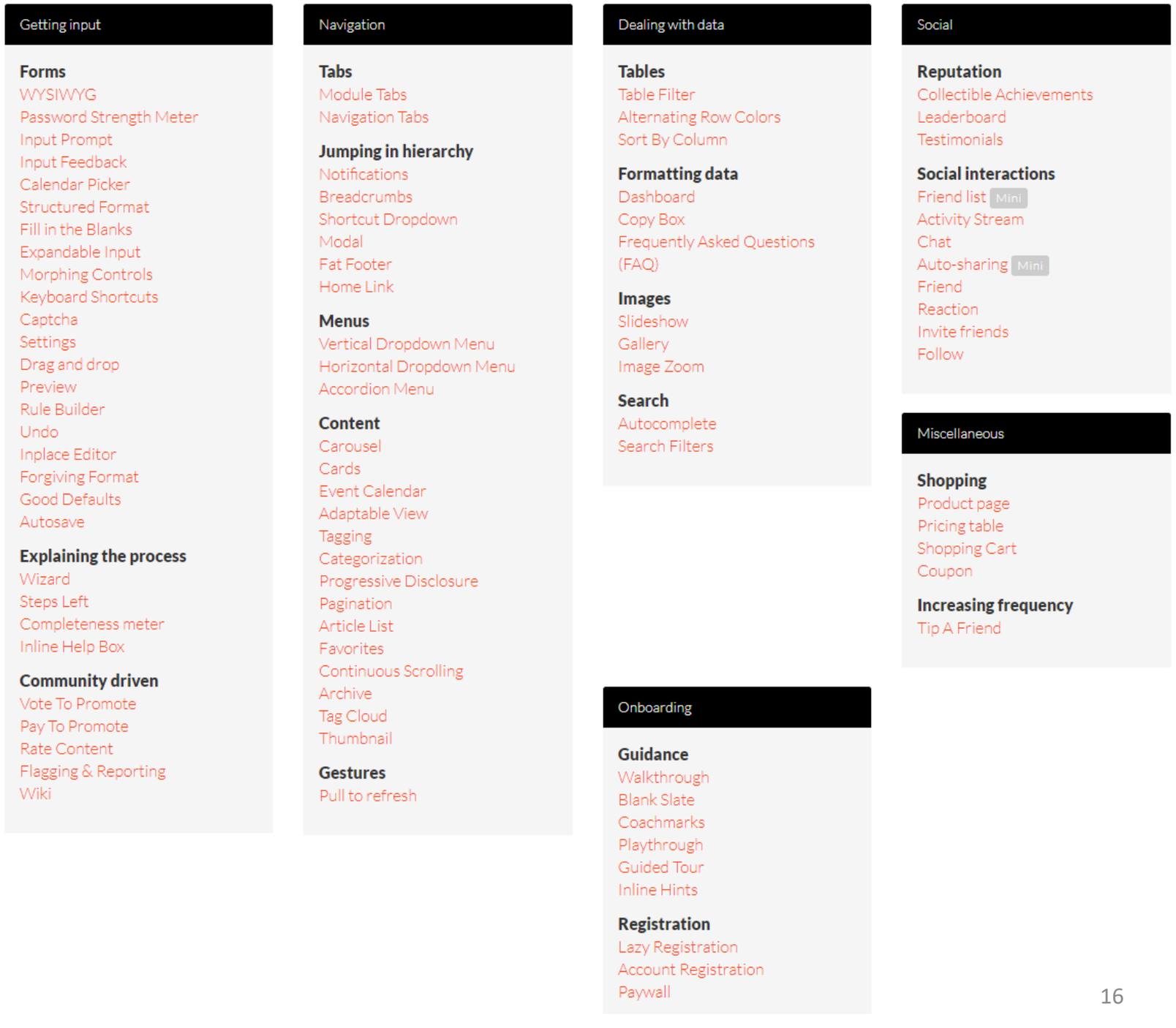
Always good to search for when you're starting a design…

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…

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

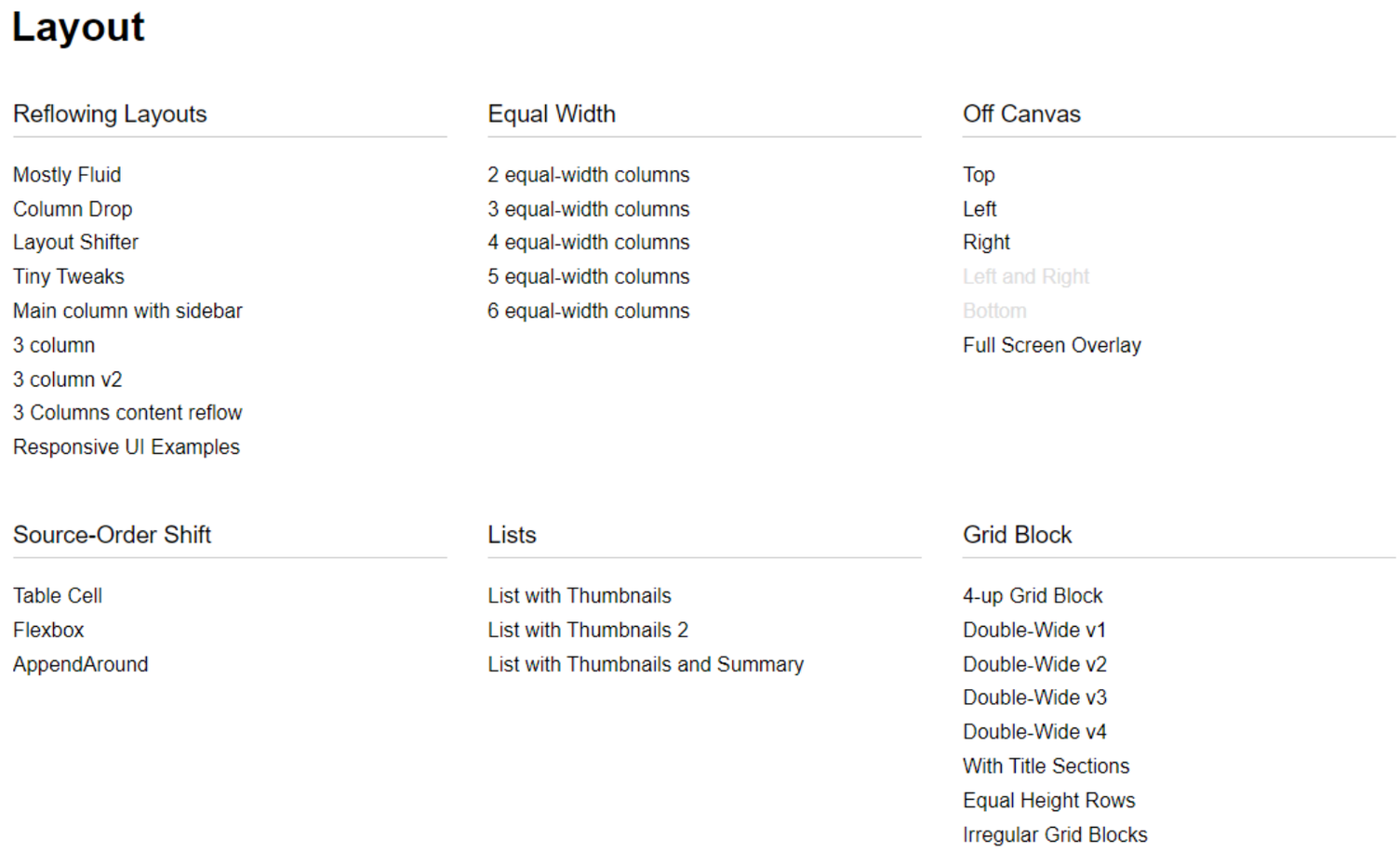
Miscellaneous

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

Equal Width

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

Off Canvas

Top
Left
Right
Left and Right
Bottom
Full Screen Overlay

Source-Order Shift

Table Cell
Flexbox
AppendAround

Lists

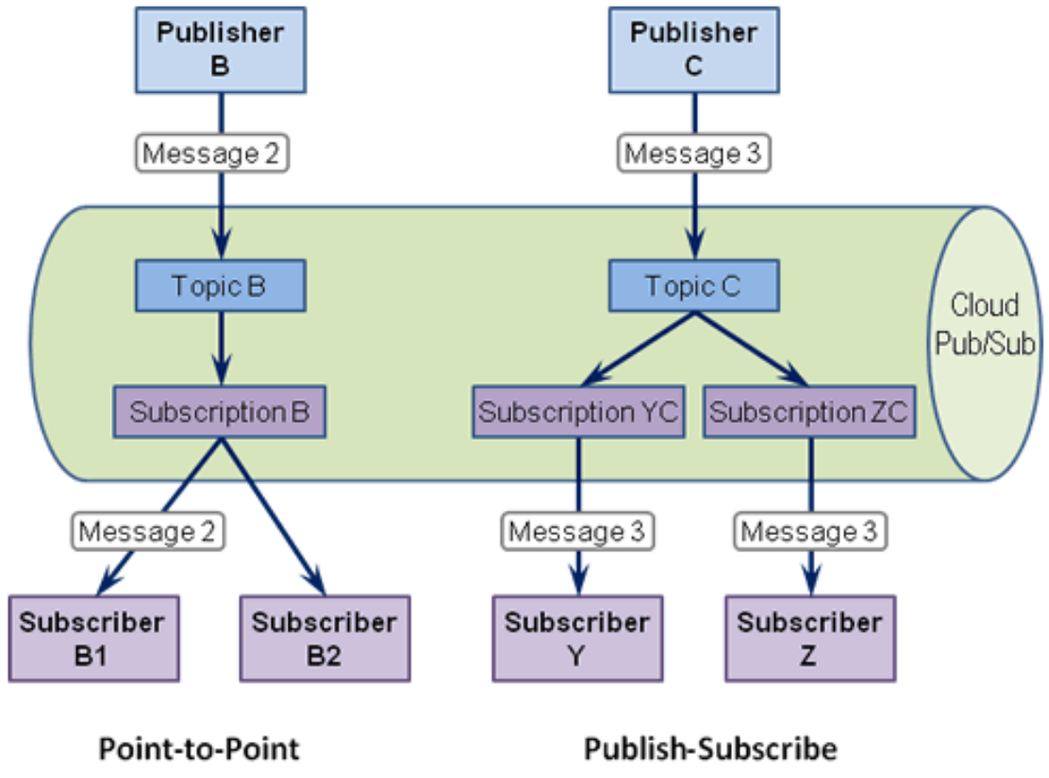
List with Thumbnails
List with Thumbnails 2
List with Thumbnails and Summary

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

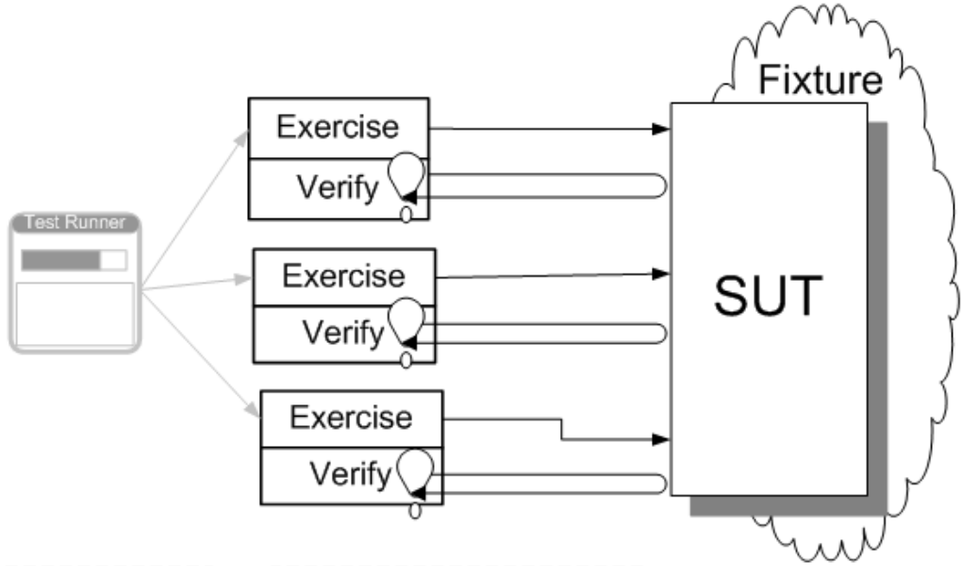
Architectural Patterns

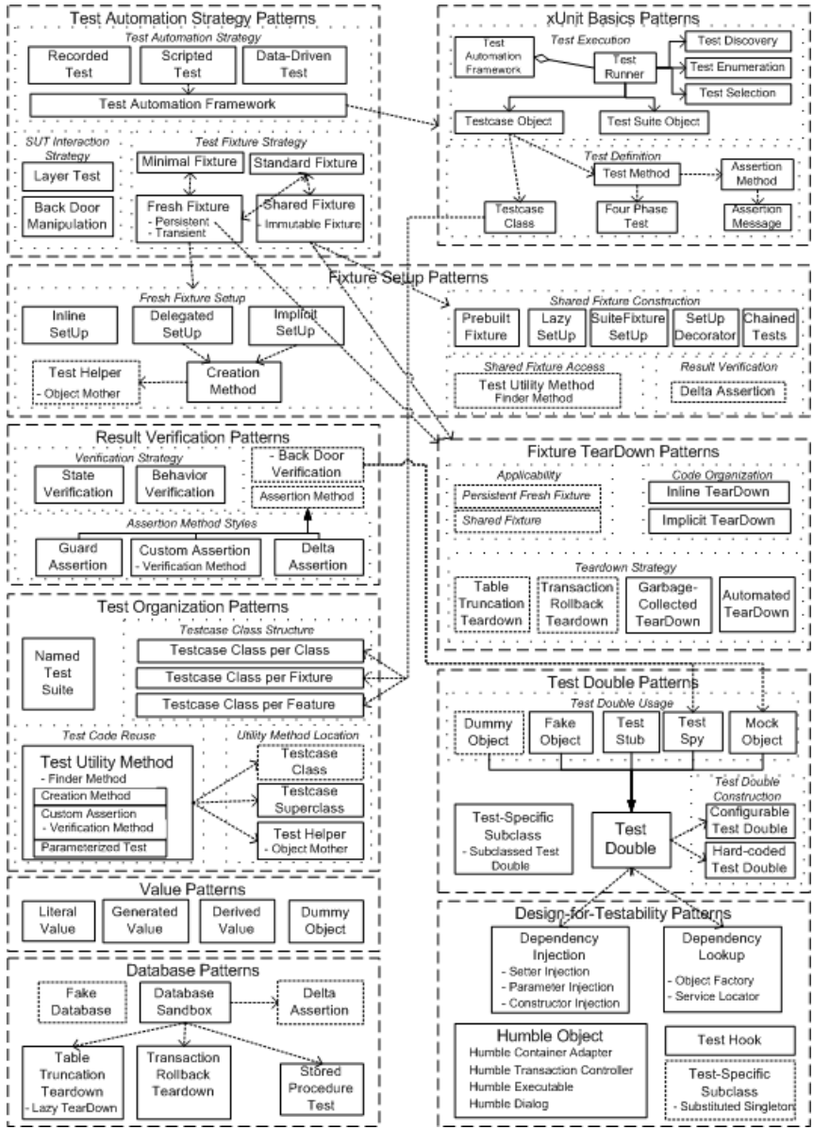
- Enterprise Integration Patterns (from the book by Fowler, a very complete web site for the book – typical diagram shown →)
- 10 Common Software Architectural Patterns in a nutshell
- Domain Driven Design Patterns (bottom image)



Test/Automation Patterns

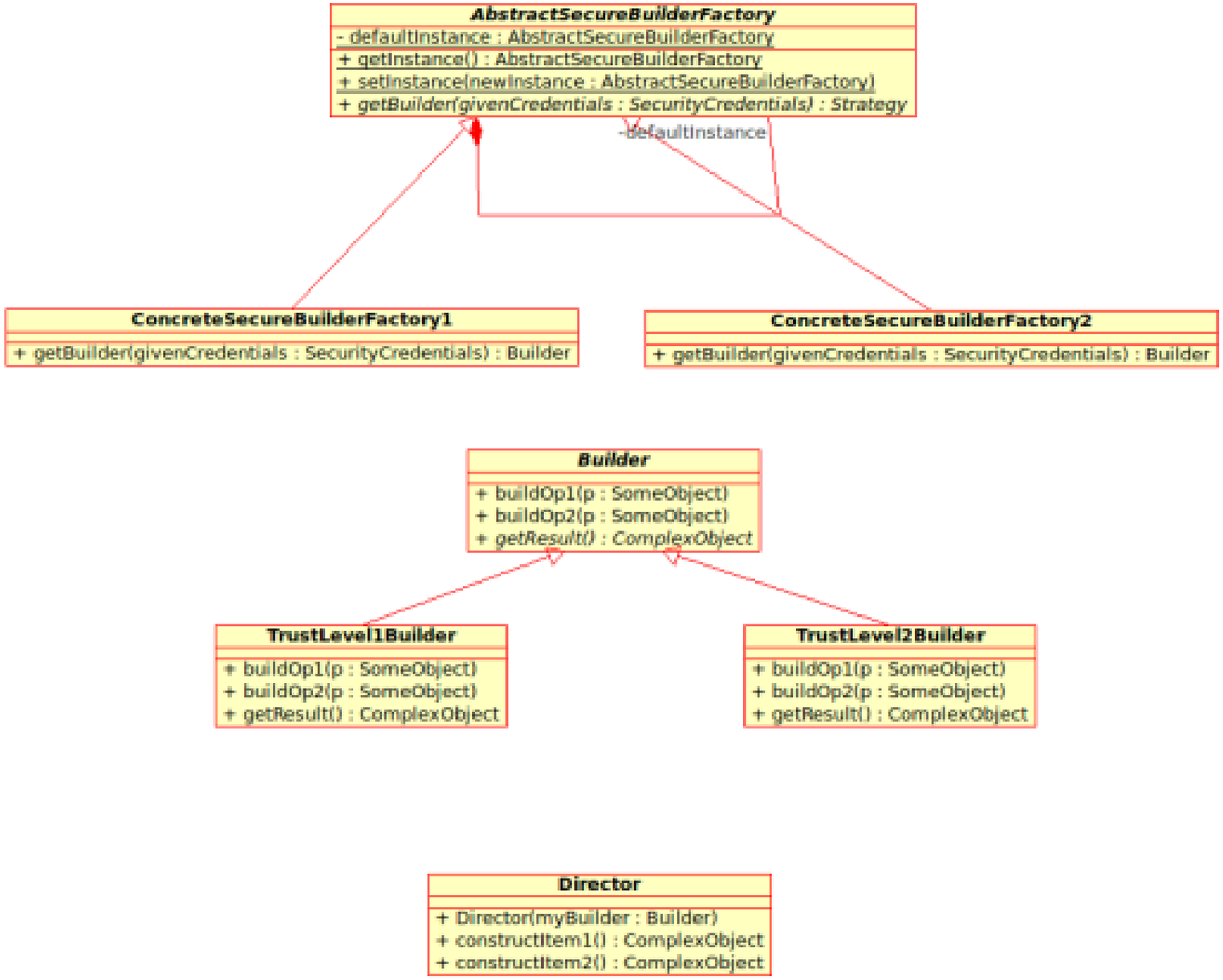
- [Test Automation Patterns Book (Axelrod)]
- [XUnit Test Patterns]
  - Test Automation
  - xUnit Basics
  - Fixture Setup
  - Result Verification
  - Test Organization
  - Values and Databases
  - Fixture Teardown
  - Test Doubles
  - Design for Testability





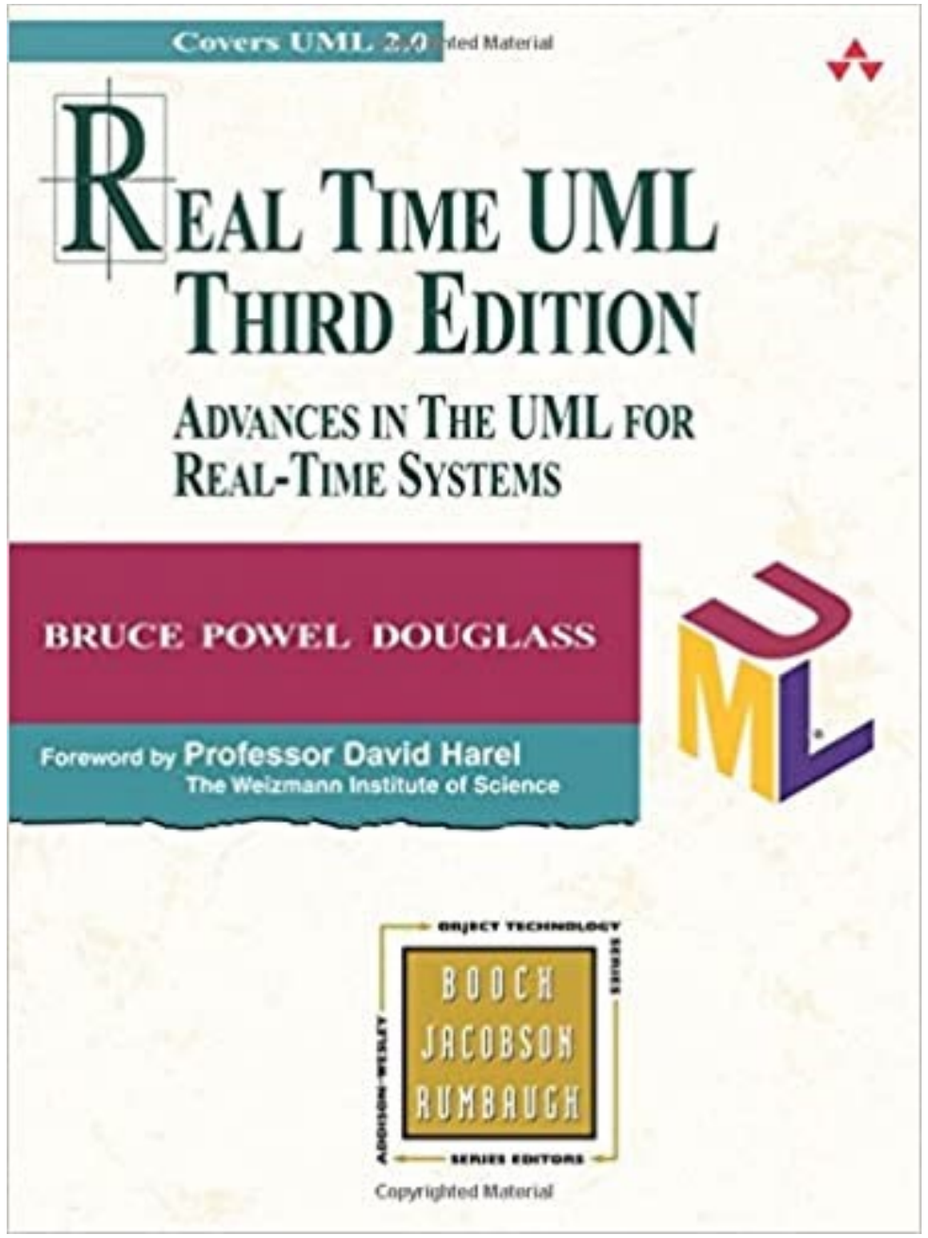
Security Patterns

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



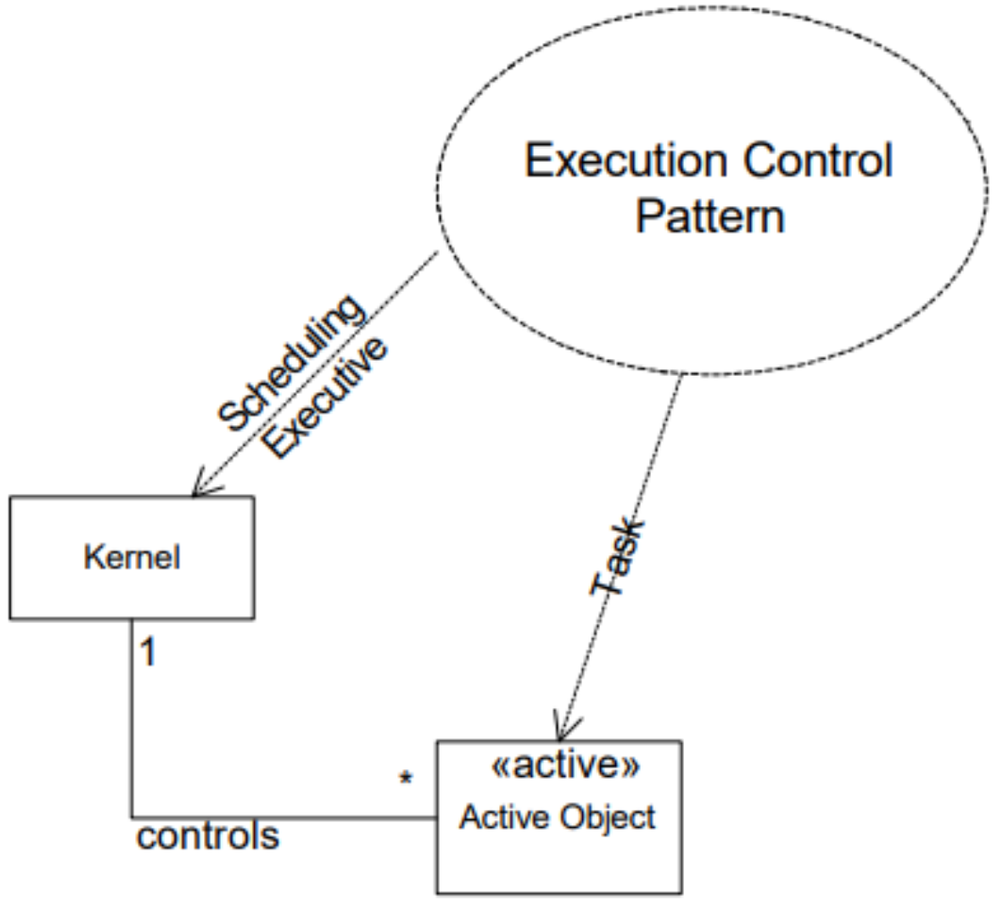
Real-time Embedded Patterns

- 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)



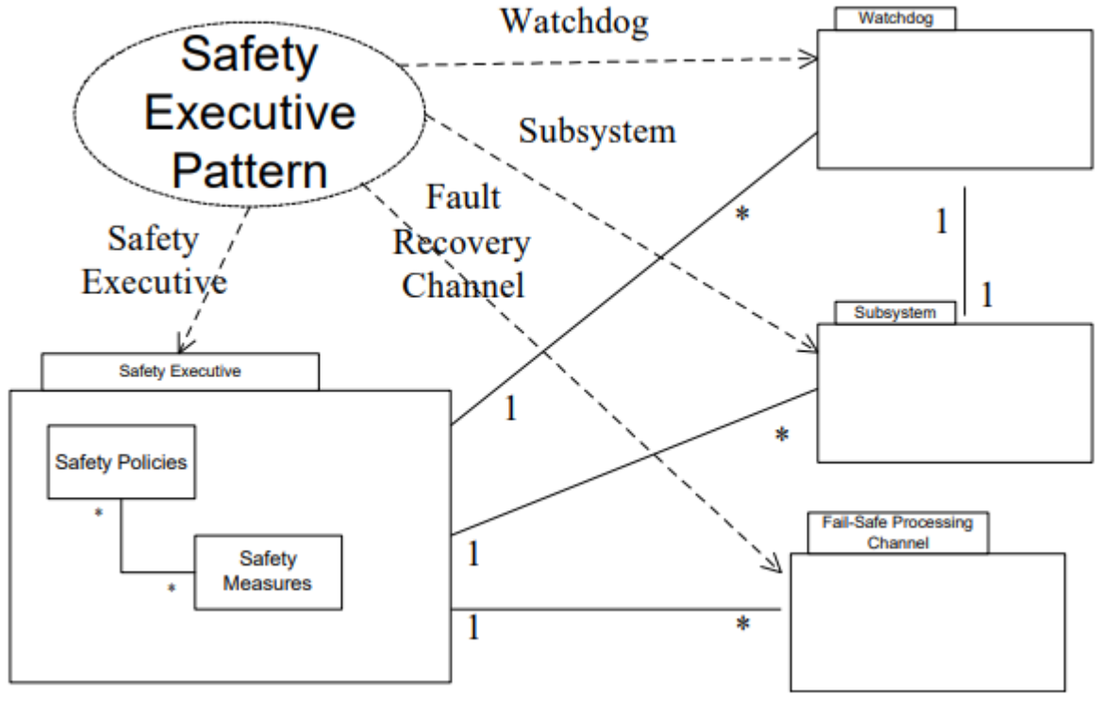
Real-Time Embedded Patterns

- 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

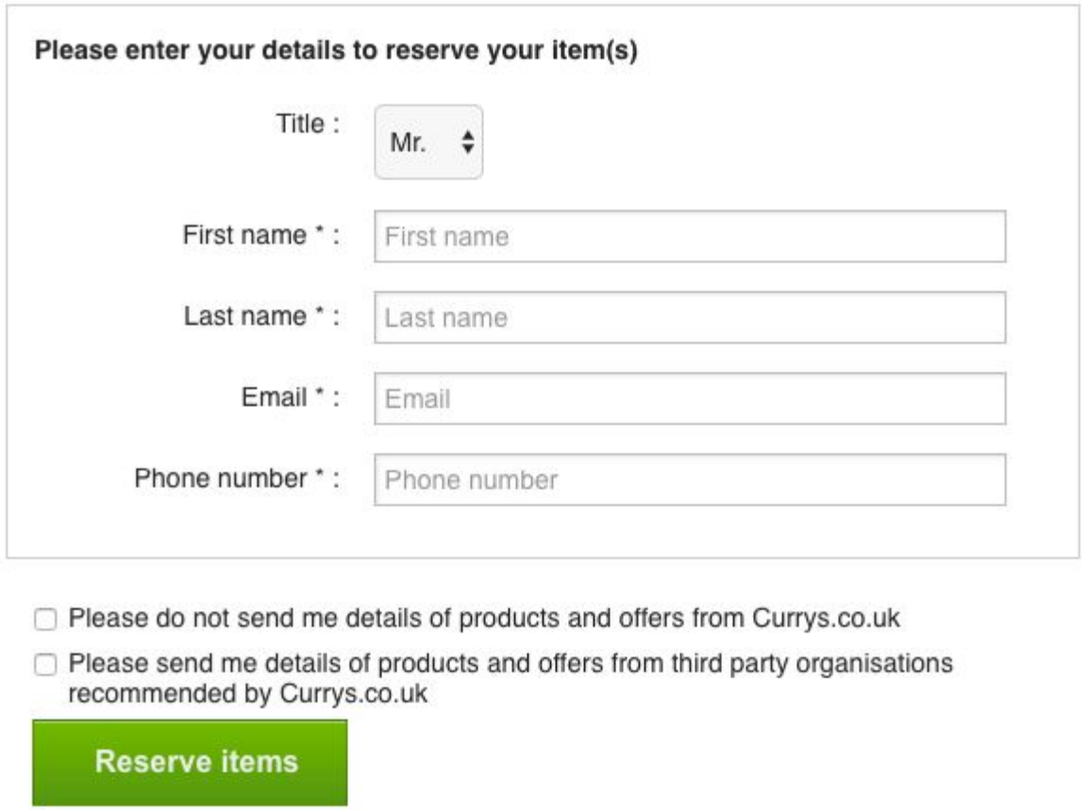


Dark Patterns?

- 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]

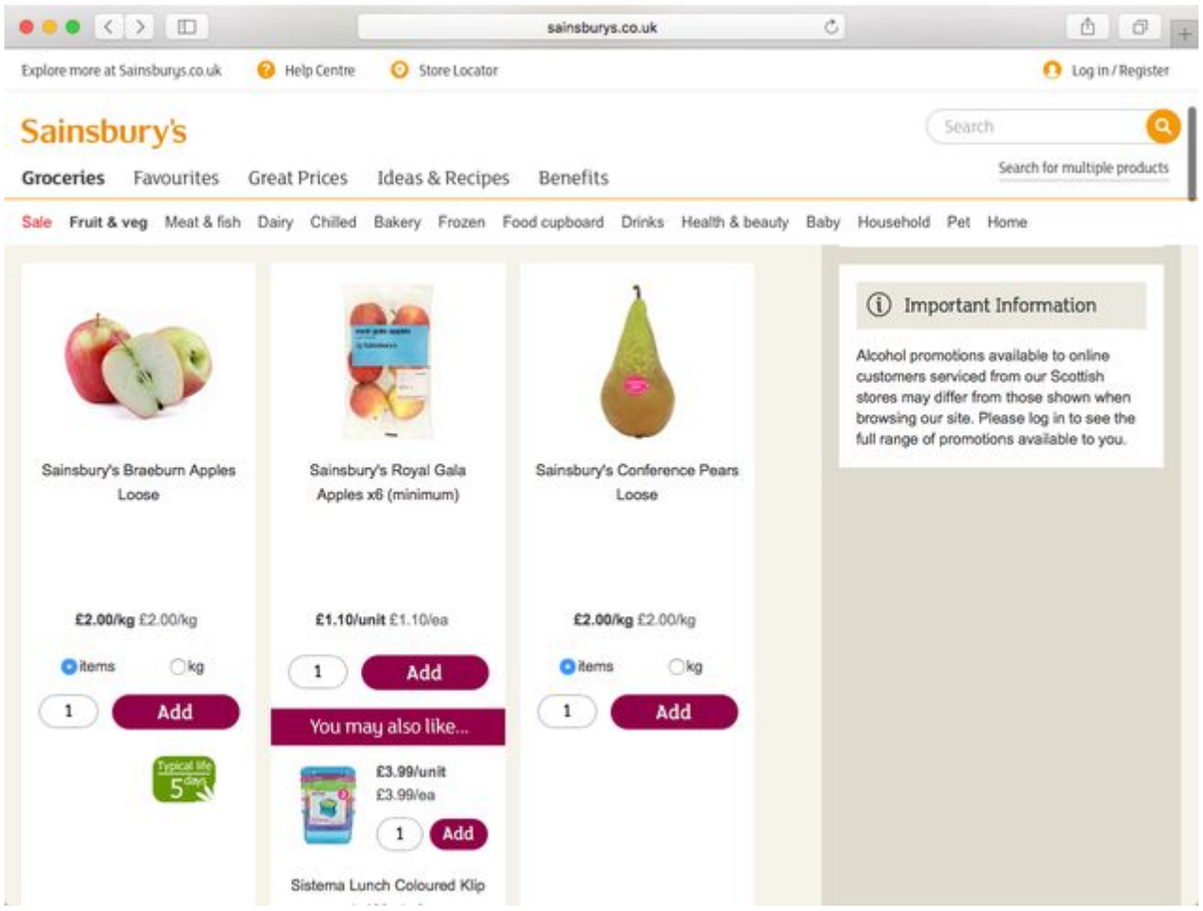
Types of Dark Patterns

- 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)



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

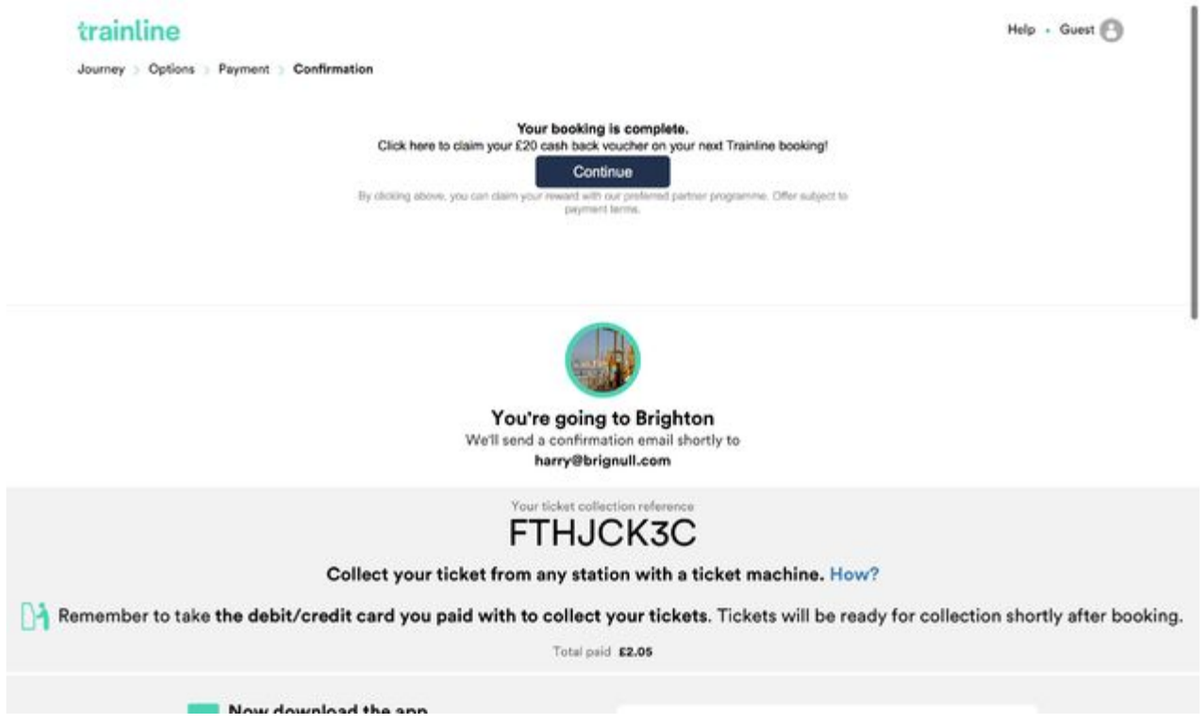


Types of Dark Patterns

- 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

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



Class Feedback

- I really appreciate your feedback on things I can do better
- FCQs (should be in your e-mail)
  - http://colorado.campuslabs.com/courseeval
  - Open 11/23 12 AM, close Wed 12/2 11:59 PM
  - How I'm "graded" for performance reviews as an instructor
- Quiz 11 – a class survey
  - The quiz is required, but you may "No Comment" answers if you wish
  - A completed quiz 11 will be worth 20 points
  - I use this feedback to tweak the class for the next time it's taught
- Post anonymously (public or private) on Piazza
- Come by and see me, e-mail, etc.

Reminder: the Final Exam

- Our last class is Monday 12/7, Tuesday 12/8 is reading day – no classes
- The Final for the OOAD class is scheduled for Wed 12/9 from 1:30 to 4 PM
- I am going to open the Final on Canvas the morning of 12/9 and let it run until the evening of Thur 12/10, so you'll have approximately 2 days to choose a time to take it
  - I have to close on Thursday to complete final class grading
- Similar to the Midterm, the Final will be on Canvas and will be open notes/book/etc.; It will be available for 3 hours after you start it, but is targeted at about an hour in length
- It will only cover material reviewed after what was covered on the Midterm (I'll review the specific coverage in class when it's closer)
- The Final exam is optional
  - If you do not take the Final exam, you will get the same grade for the Final that you received for your Midterm
  - If you take the Final exam, your Final exam grade will be the larger of:
    - your Midterm grade and
    - the grade you make on the Final
  - Therefore, you cannot score less on the Final than you did on the Midterm
  - Note that taking the Final does not affect your Midterm grade
- Let me know if you have questions about your situation

Next Steps

- Project 6 due 12/4 Noon
  - Final part of three for the semester project
  - Get started sooner than later!
  - Be aware of how your travel plans may impact your team and turning in your assignments!
- Quiz 10 is due this Wed 11/25
  - Quiz 10 is last topic quiz, Quiz 11 will be a class survey for your feedback
- Graduate Pecha Kuchas are due Tuesday Dec 1
  - **Sign up for a presentation slot here:**
    - https://docs.google.com/document/d/1BYxUTTlh66DLLhVU8GbDZm94H2JtdMYXyIrN7vPaWh8/edit?usp=sharing
- Graduate Final Presentations are due Monday Dec 7
- Article Reviews are available for extra bonus points…
- Grading continues… We'll post Project 5 results after our interviews are completed.
- I will be posting bonus points on Canvas after the coding exercise closes tonight,
  - Maximum of 10 bonus points per student (from class activities, coding exercise, article posts)
- New discussion topic coming up… Visit Piazza often – it is for your participation grade, so participate!
- Coming up: Reflections on Discussions, What is Not OOAD, graduate Pecha Kuchas, more…
- If you need help – Office hours, Piazza, e-mail – we are here for you!

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/