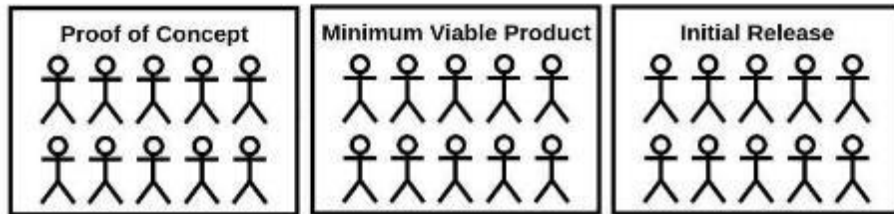
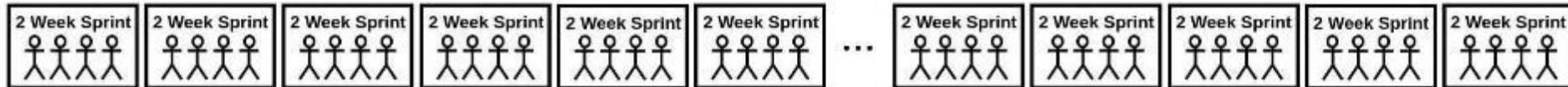




PARALLEL AGILE

Sequential Agile

Sequential Agile proceeds in a series of 2-week sprints. Bigger projects require more sprints and take longer to complete.



Parallel Agile

Parallel Agile proceeds in a series of three (roughly) month-long development efforts. Bigger projects require more developers but development time remains at approximately 3 months total.

Systems Engineering for Highly Rapid Development

The Parallel Agile Process



PARALLEL AGILE

What if we could easily integrate
the work of many developers
writing software in parallel?



PARALLEL AGILE

What if we could easily integrate the work of many developers writing software in parallel?

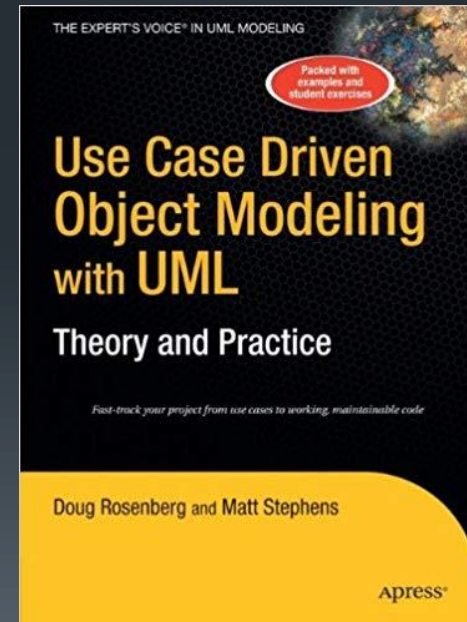
- Accelerate projects by increasing the number of developers
- Get to market faster without sacrificing quality
- Scale by adding developers, not stretching the calendar
- Significant cost savings, radical schedule compression



PARALLEL AGILE

Accelerate projects by increasing the number of developers
each developer gets a use case

- We've known how to split systems into use cases for a long time
 - It's putting the work back together that's the trick
- Default partitioning for parallel development is **one developer per use case**
 - Complex use cases split into sub-tasks as necessary
 - More use cases require more developers
- Usually **one use case can be completed in under a month**
 - Write narrative description
 - Decompose into Models Views Controllers (MVC)
 - OOD using sequence diagrams, state machines
 - Code, then Acceptance Test
- Different use cases can share the same domain objects
 - Integration across use cases by API to access domain object





PARALLEL AGILE

Get to market faster without sacrificing quality

- 3 phases: Proof of concept, MVP, Initial Release
 - Each phase approximately a month long
 - Proof of concept uses **prototyping** to discover requirements, reduce risk
 - MVP uses **UML modeling**, details sunny/rainy day scenarios, reduce technical debt
 - Initial Release focuses on **acceptance testing**, performance tuning, optimization, reduce hotfixes





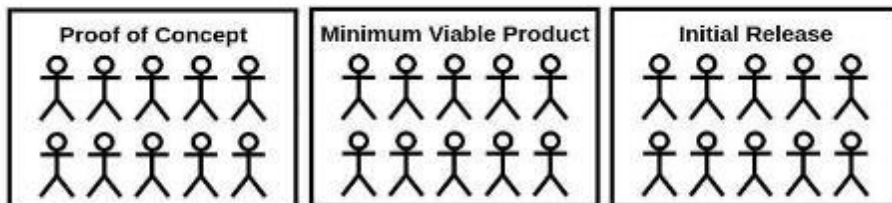
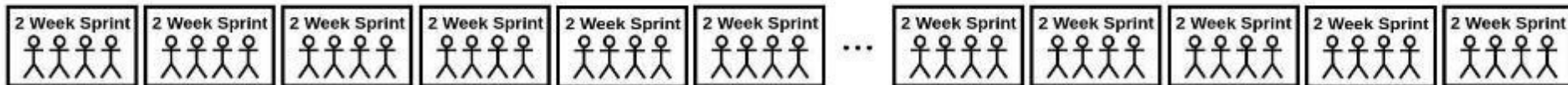
PARALLEL AGILE

Scale by adding developers, not stretching the calendar

- Big projects don't add more sprints, they add more developers working in parallel
- Small, medium, (reasonably) large projects take the same amount of time (roughly 3 months) if enough developers are available
 - *No, we can't do an entire ballistic missile defense system in 3 months*
- Merge and integrate at the end of each phase
- Test team works concurrently with developers for each phase

Sequential Agile

Sequential Agile proceeds in a series of 2-week sprints. Bigger projects require more sprints and take longer to complete.



Parallel Agile

Parallel Agile proceeds in a series of three (roughly) month-long development efforts. Bigger projects require more developers but development time remains at approximately 3 months total.



PARALLEL AGILE

Significant cost savings, radical schedule compression

- Reduced need for refactoring, fewer hotfixes minimizes cost
- Schedule compression analogous to parallel processing in HW
- *Agility* - get to code early to discover requirements
- *Discipline* - don't ship the prototype and refactor later, but design carefully
- Design is done one use case at a time, doesn't take much time
- Prototype first then design - not big design up front



PARALLEL AGILE

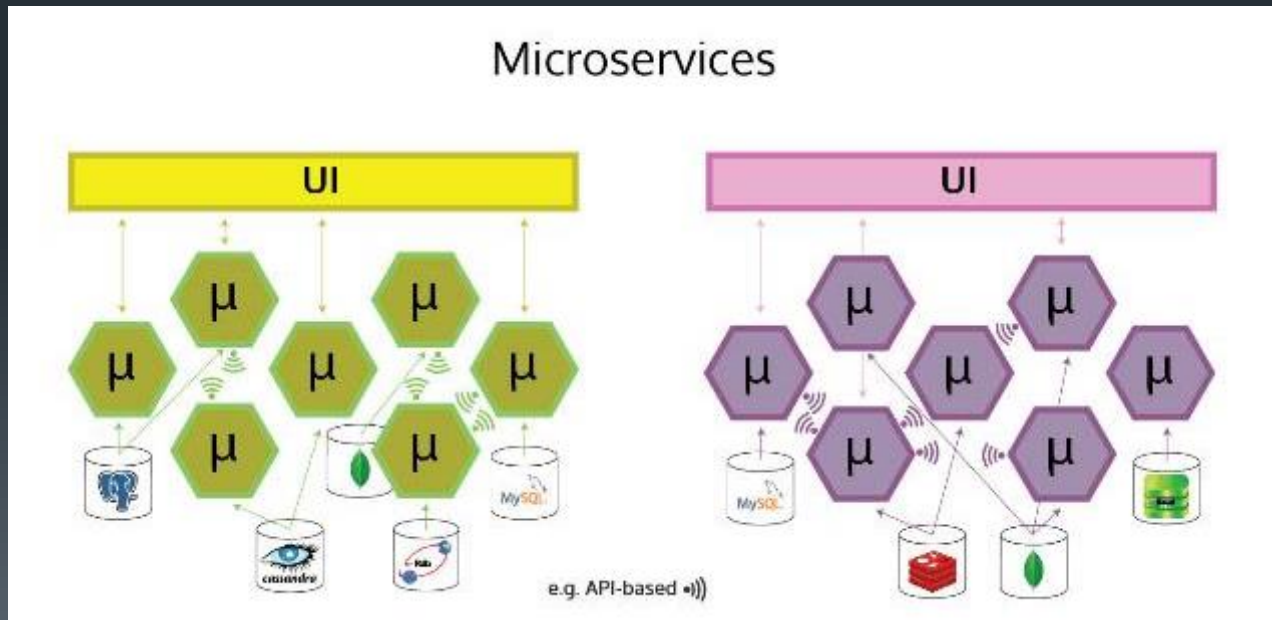
How can we easily integrate the work of many developers writing software in parallel?



PARALLEL AGILE

Q: How can we easily integrate the work of many developers writing software in parallel?

A: Apply B2B integration strategies at the developer level





PARALLEL AGILE

Apply B2B integration strategies at the developer level

Code generate NoSQL databases and REST APIs at project inception



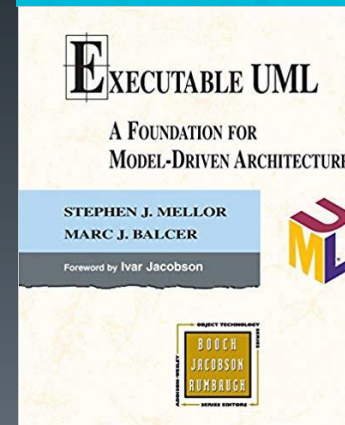
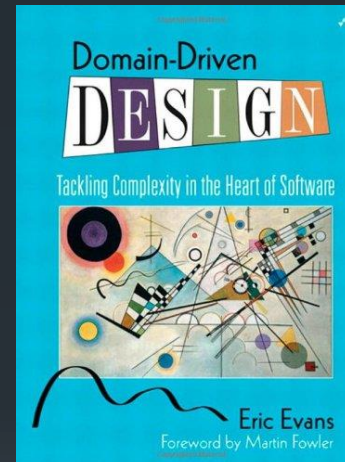


PARALLEL AGILE

Code generate NoSQL DB and REST APIs at project inception

▪ *Executable Domain Models*

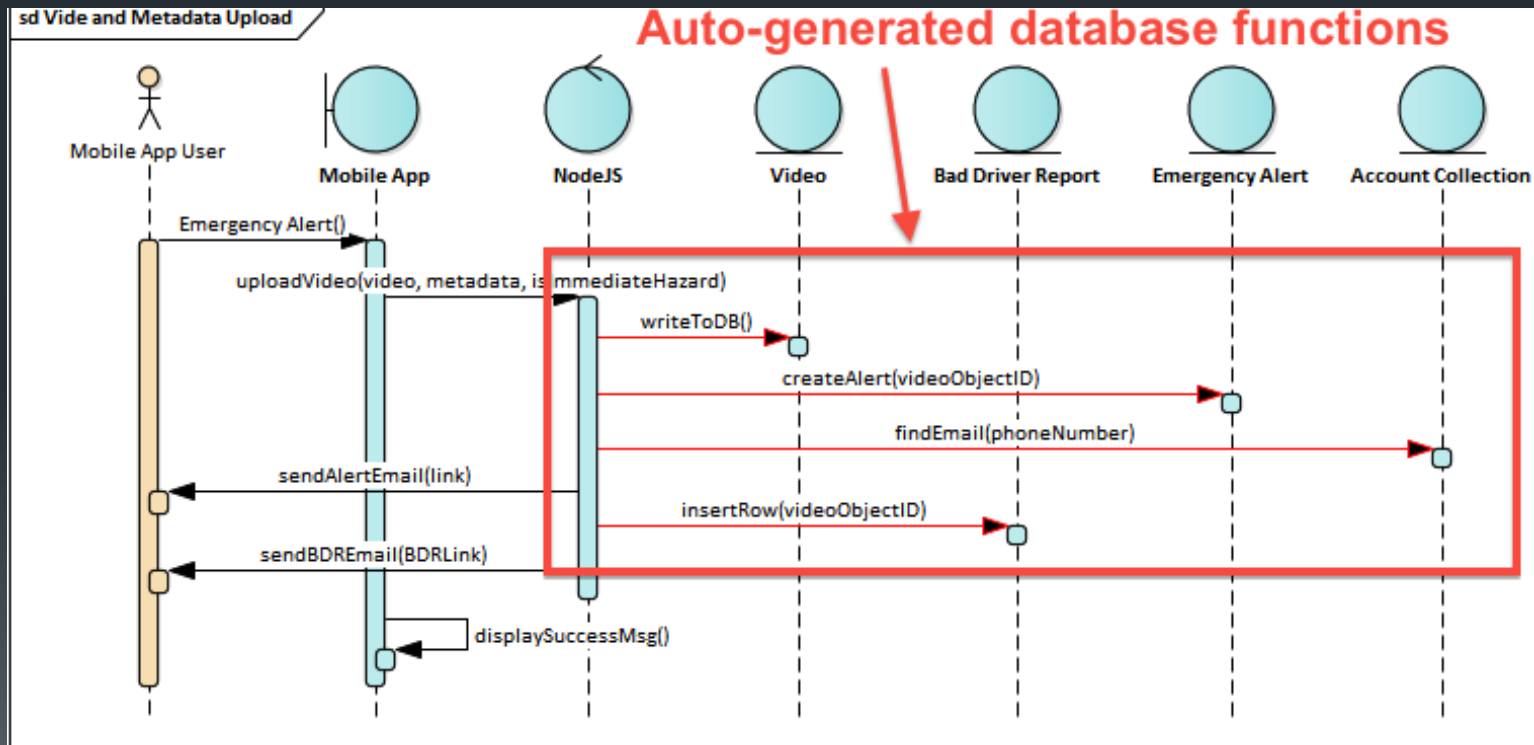
- Model the problem domain
- Each domain object becomes a NoSQL collection
- Database Access functions code generated
- Database Access functions wrapped in REST API
- DB+API generated at the beginning of the project
- Prototypes built against live DB using API
- Database schema evolved from prototypes
- Developer-to-developer integration via shared domain objects





PARALLEL AGILE

Database access code doesn't get written manually

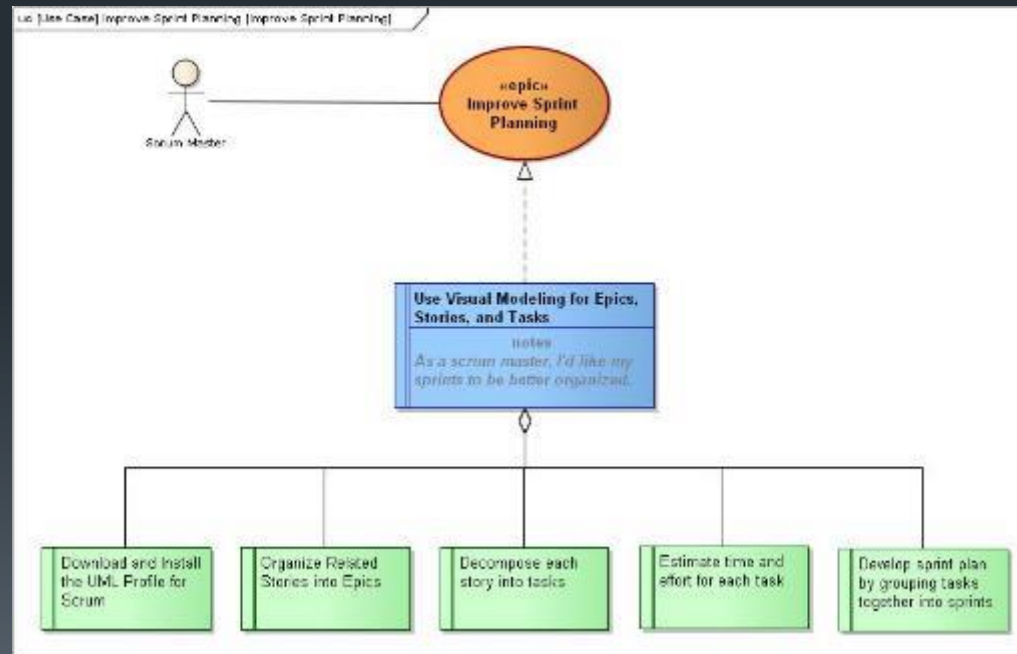


in round numbers this might be 20-40% of your code



Agile project management using visual models

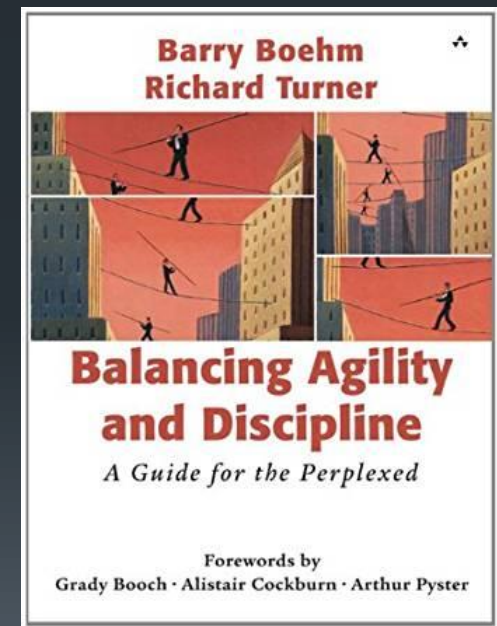
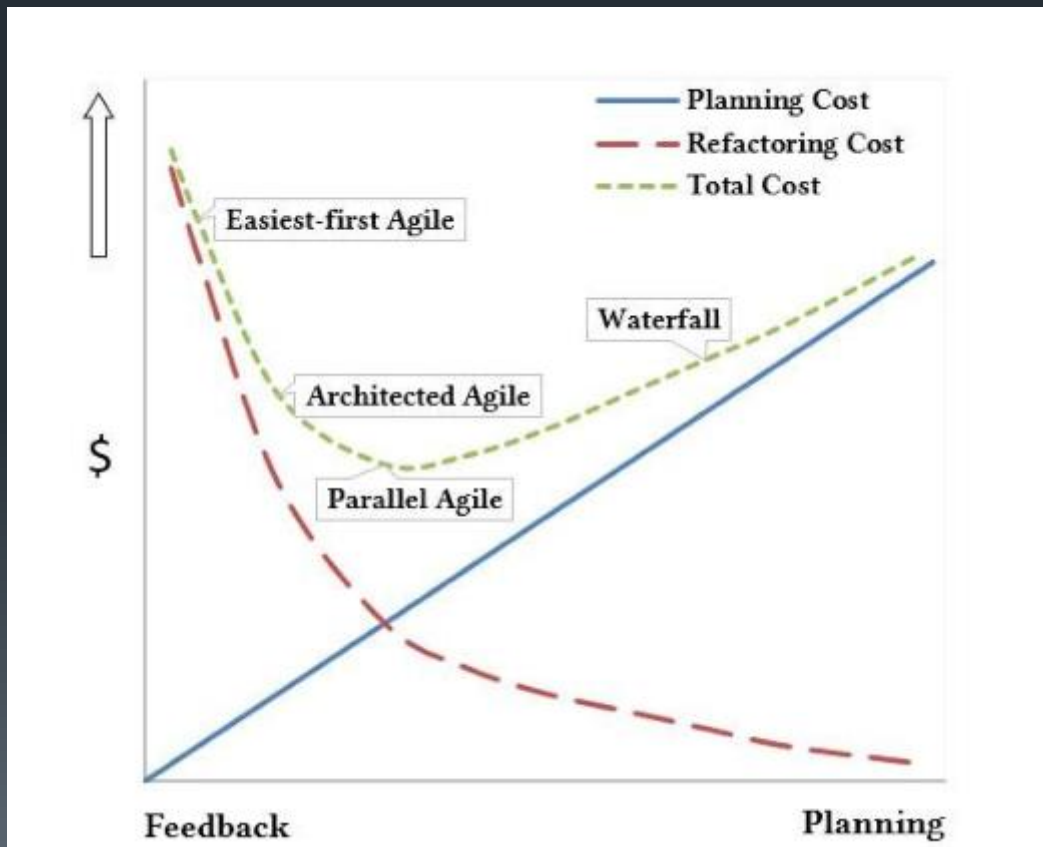
- Work is organized into Epics, User Stories, and Tasks
- Bottom-up estimation similar to story points
- Top-down estimation from use cases, MVC decompositions





PARALLEL AGILE

Balancing Agility and Discipline targets the cost minimum

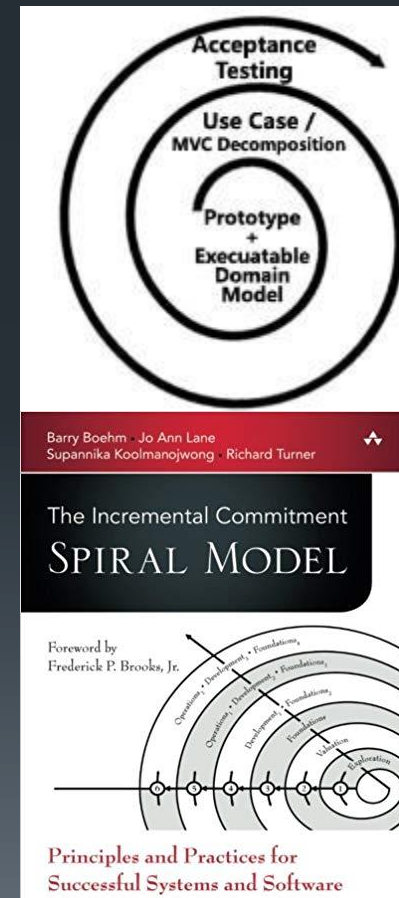




PARALLEL AGILE

3 phase spiral model development: each phase takes a month

- Inception – Model the problem domain and make it executable
- Proof of Concept Phase – Prototype to discover requirements
- MVP Phase – Model behavior to elaborate requirements
- Release Phase – Acceptance test against requirements

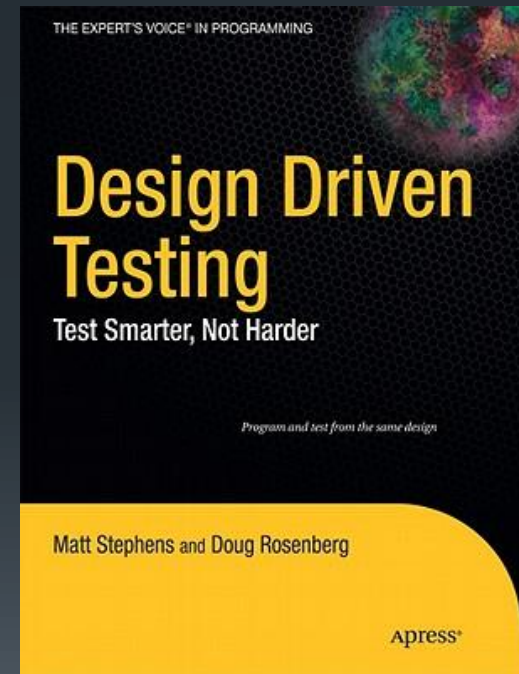




PARALLEL AGILE

Continuous Acceptance Testing

- Test all sunny-day/rainy-day scenarios
- Test all requirements
- Test team operates concurrently with development





PARALLEL AGILE

Everything goes faster when
you work in parallel



PARALLEL AGILE

Everything goes faster when you work in parallel

- *Develop in parallel*
 - usually one use case per developer
 - split complex use cases as needed (e.g. server-side logic)
- *Integrate in parallel*
 - multiple use cases access shared domain objects using REST API
 - even early prototypes access live database
 - language and platform neutral (iOS, Android, Angular, Unity...)
- *Test in parallel*
 - test team works concurrently with dev team
 - it's never too early to start acceptance testing



PARALLEL AGILE

Does it work?

The difference between theory and practice is that in theory there's no difference between theory and practice but in practice there is.



PARALLEL AGILE

Does it work?

- Four test projects involving around 200 graduate students
 - 2014-2015 Location Based Advertising (75 students)
 - 2015 Picture Sharing (12 students)
 - 2016-2018 CarmaCam (75 students)
 - 2017-2018 TikiMan Go Game project (25 students)



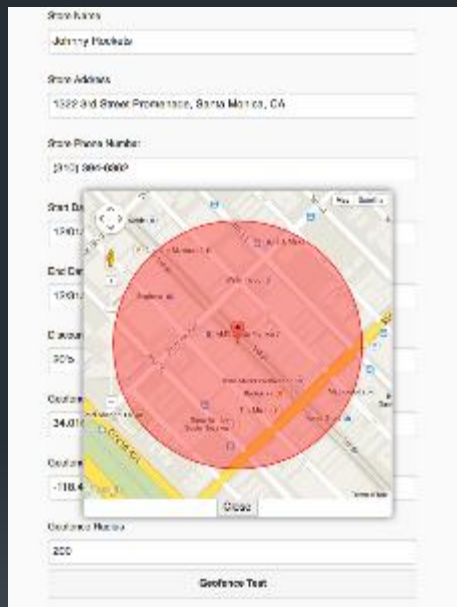
PARALLEL AGILE

Project 1: Location Based Advertising

- Originally a couple of CS577 homework assignments
- 47 students wrote use cases, 29 wrote prototype code
- 75 students over 3 semesters
- Total development effort around 2 person-years
- Developed concept of Executable Domain Models



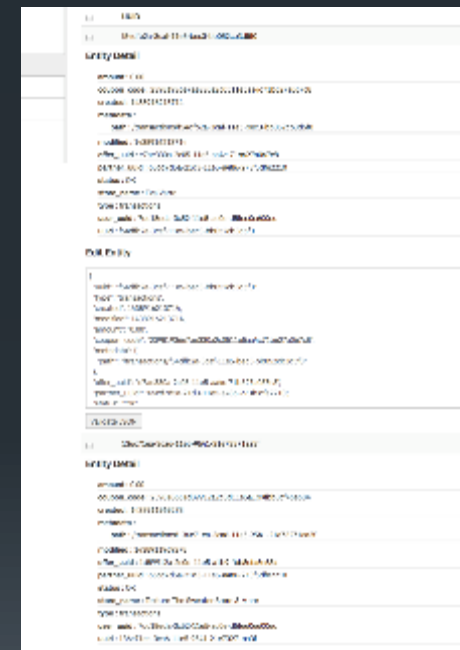
Concept: Geofenced coupon delivery



Store publishes coupon and specifies geofence using web app



Mobile app receives coupon when user arrives at store



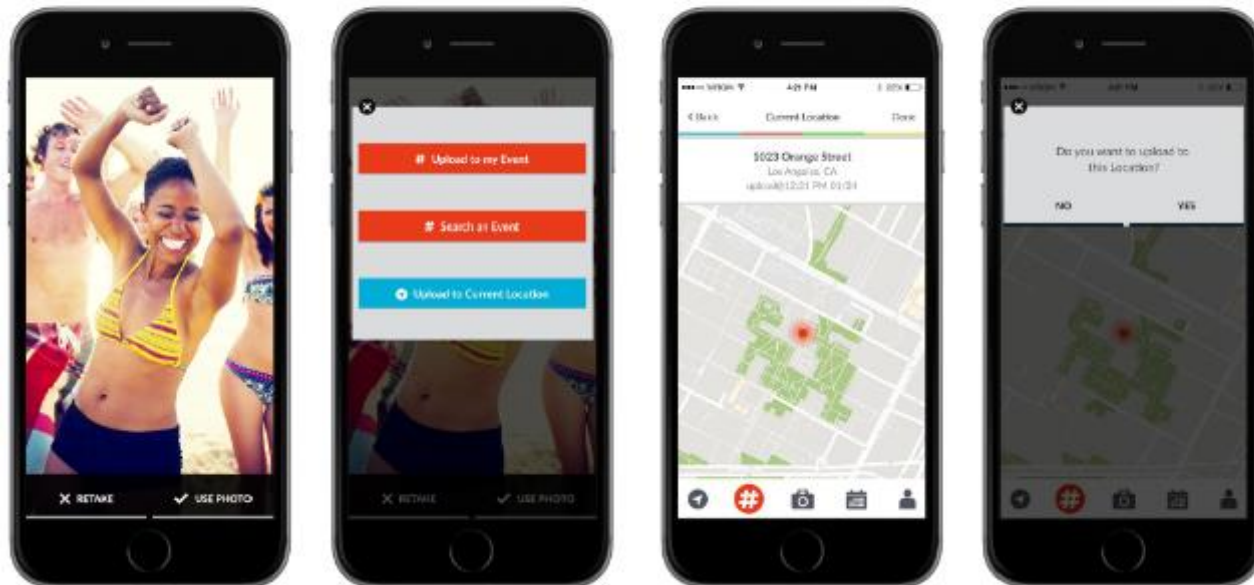
Redeeming coupon generates transaction



PARALLEL AGILE

Photo sharing app for events


Upload photo to location







PARALLEL AGILE


Project 3: CarmaCam



Home


0:18





Date: 07/13/2018 Time: 17:58:15

Capture

License Plate #

Vehicle description

Category

License State:

Incident description

Severity

Uploading media files to the cloud is interesting...



PARALLEL AGILE

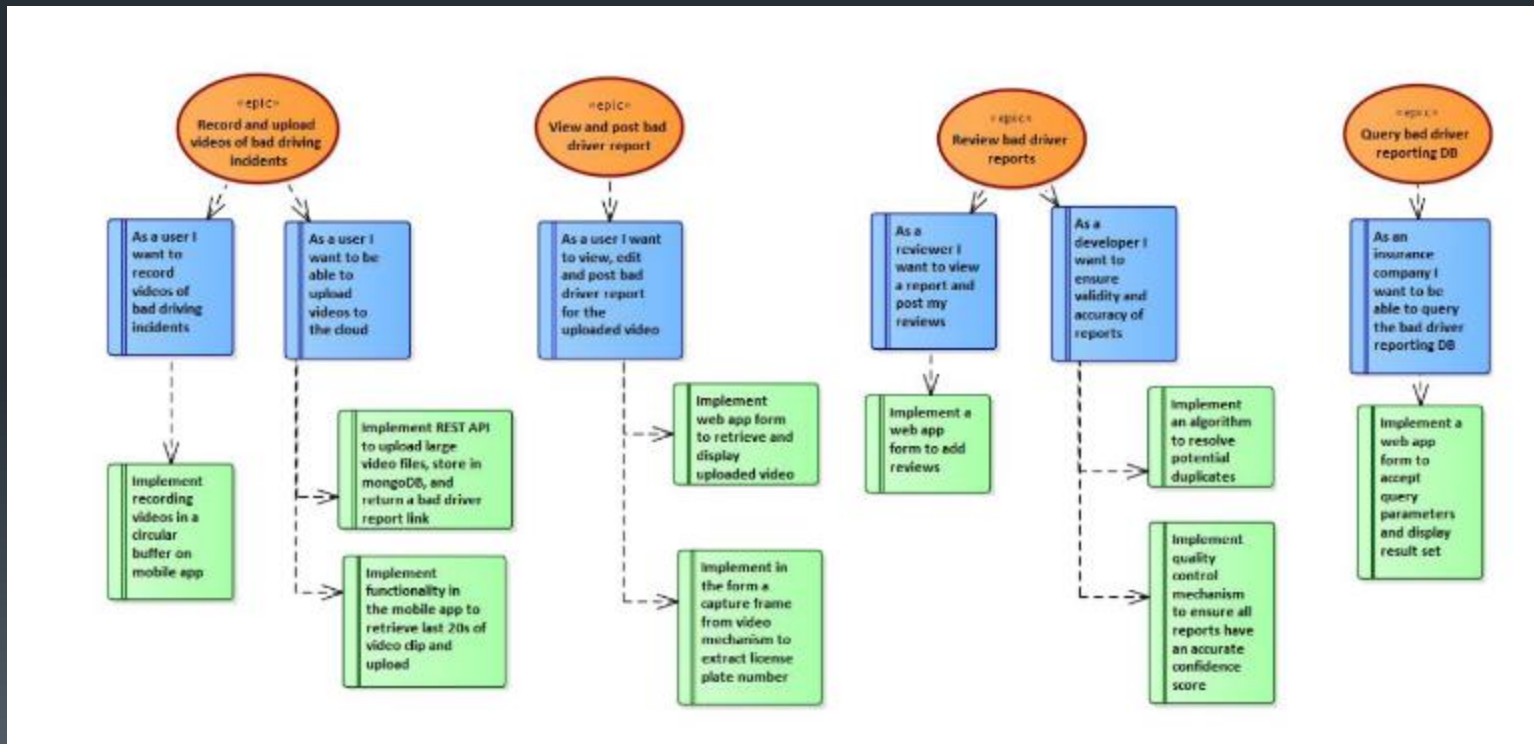
CarmaCam tests the full Parallel Agile process

- MEAN Stack, Android, iOS, Mongo DB, Node JS, AWS EC2
- Around 75 students over multiple semesters
- 95% staff turnover across semesters (!)
- 25 students in CS590 Fall 2018
- Pilot project for Executable Domain Model code generator
- Evolved 3 phase development pattern
- Visual model sprint plans
- Now entering commercial use for bus lane parking violations
- Adding Machine Learning to recognize dangerous driving from video



PARALLEL AGILE

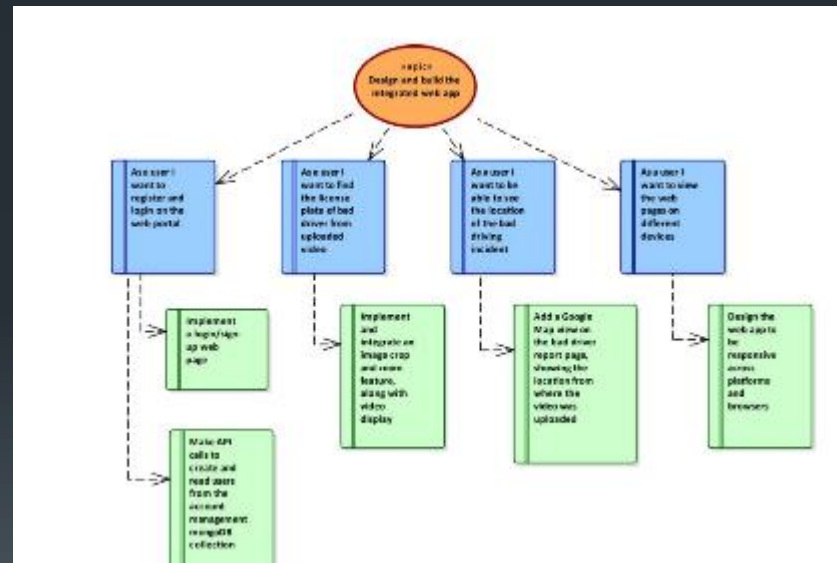
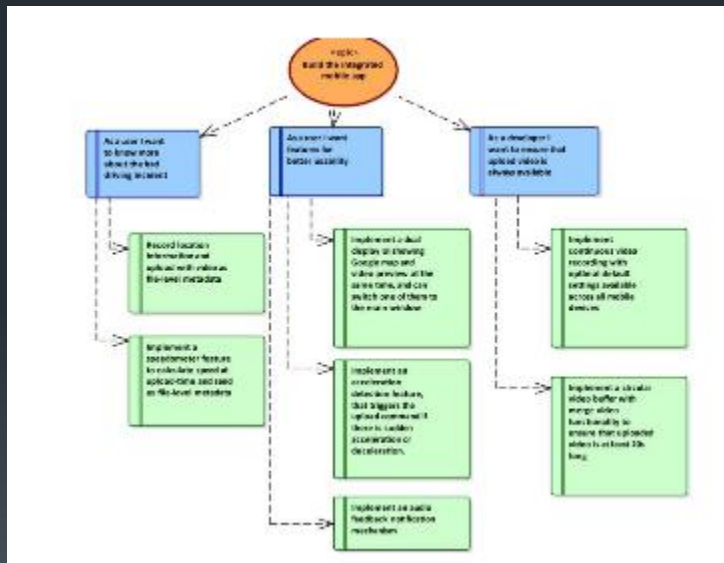
Proof of Concept (prototyping, 15 students)





PARALLEL AGILE

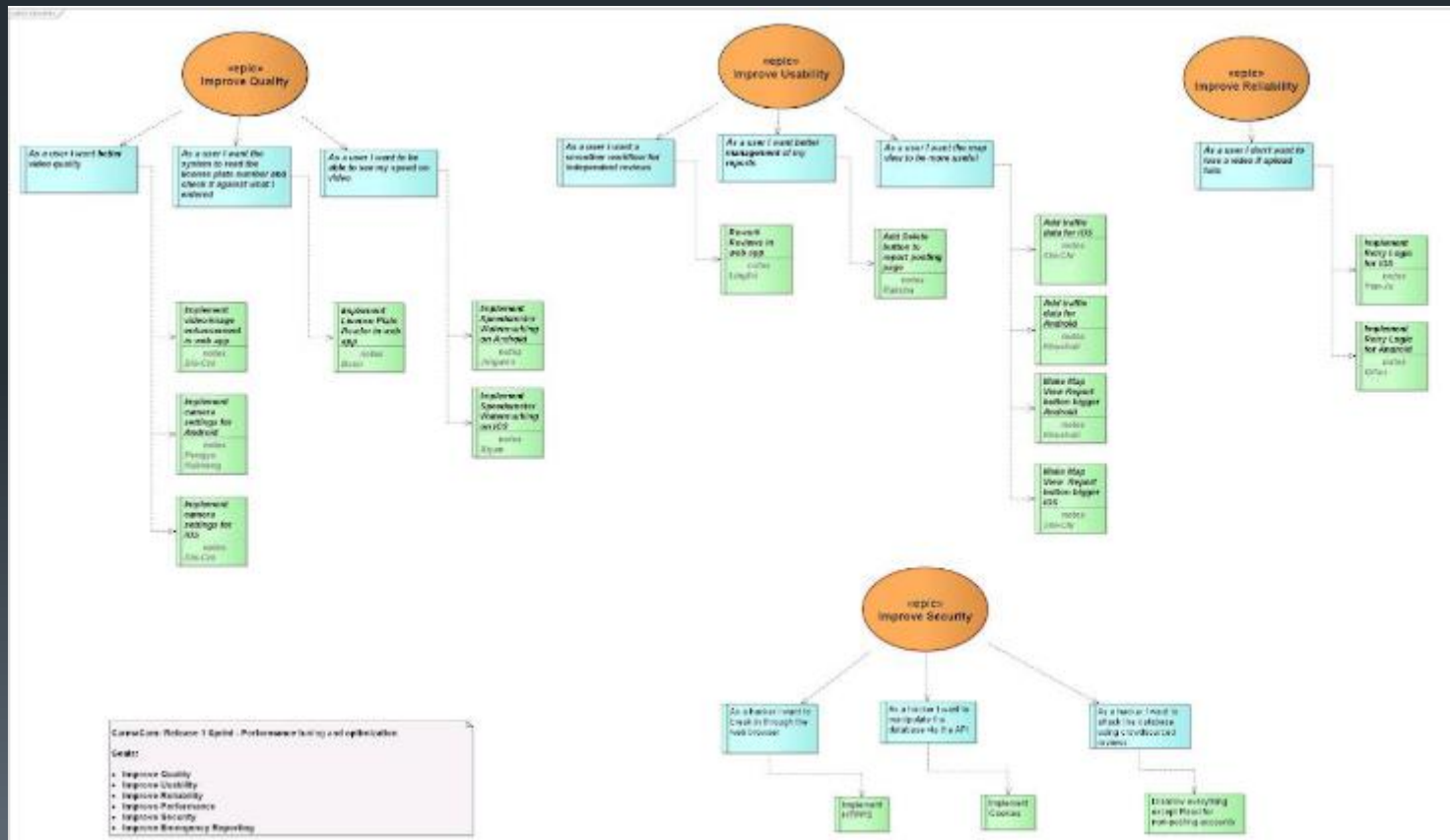
MVP (3 returning students, 12 new ones)





PARALLEL AGILE

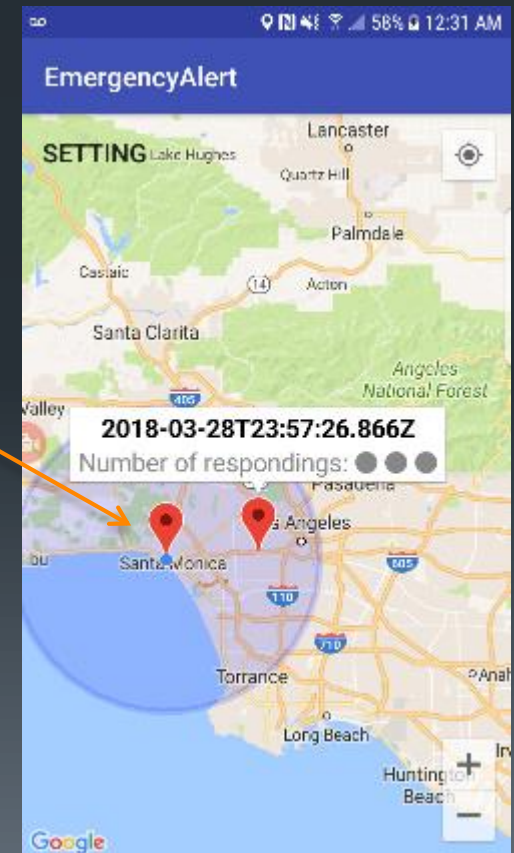
Initial Release (20 new students 0 returning, then 10 more, 2 returning)





PARALLEL AGILE

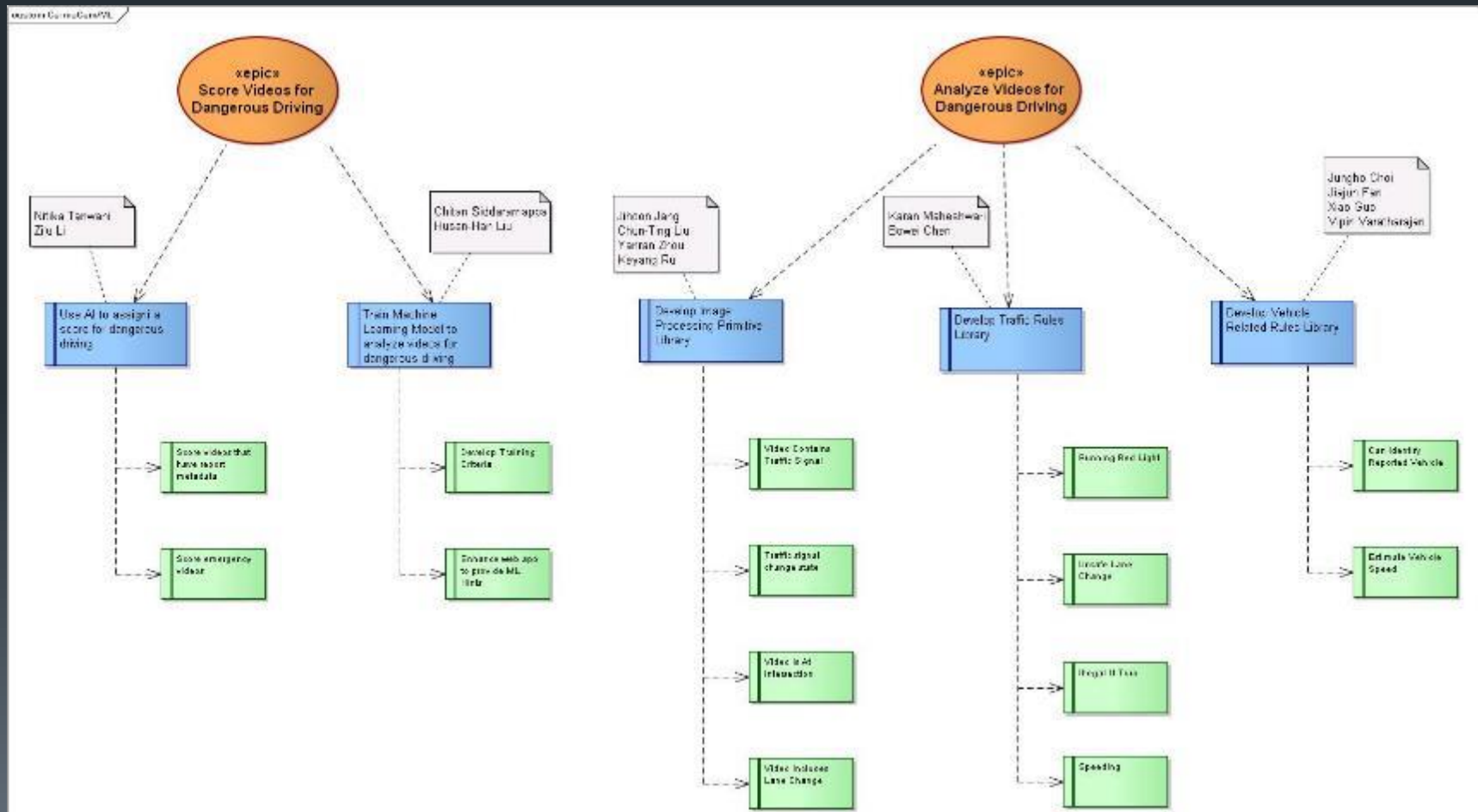
Emergency Alert receiver app uses geospatial query to monitor DUI videos



Need AI verification of DUI before broadcasting emergency video



Train AI models to recognize dangerous driving (15 students)





PARALLEL AGILE

Train AI models to recognize dangerous driving from video

- Machine Learning Proof of Concept
 - Use AI models to filter out spurious videos
 - Route DUI videos to field or to dispatcher for review
- Python, Tensorflow, OpenCV, Convolutional Neural Networks





PARALLEL AGILE

Project 4: TikiMan Go

- VR/AR Game project
- Battle animated tiki men for control of Hawaii
- Built in Unity3D, C#, Mongo, Node
- 25 students to date
- Now partnering with well-known tiki artist





PARALLEL AGILE

Virtual Reality off-island, Augmented Reality on-island





PARALLEL AGILE

So...Does it work?

Yep!



PARALLEL AGILE

It works for these technologies

- **LBA**: Cassandra, Node JS, JQuery Mobile, Cordova/PhoneGap
- **PicShare**: iOS, Parse Framework
- **CarmaCam**: Android/Java, iOS/Swift, MEAN Stack (Mongo DB, Express JS, Angular JS, Node JS), Python, TensorFlow, Open CV
- **TikiMan Go**: Unity 3D, Blender, Kudan, World Composer, Mongo DB, Node JS



PARALLEL AGILE

Student laboratory has allowed us to experiment

- Not many businesses could spare 200 developers to experiment with
- After 4 years of student projects it's time to graduate!
- Student projects average 95% staff turnover every semester
 - *Don't try this at home!*



PARALLEL AGILE

Industry testing beginning now

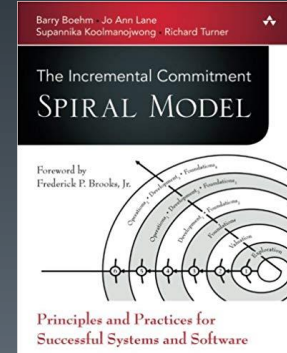
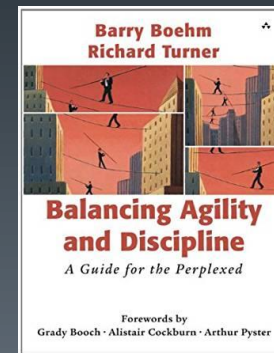
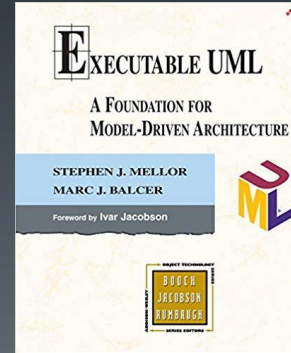
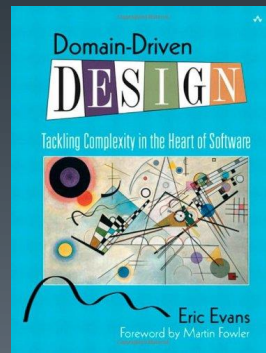
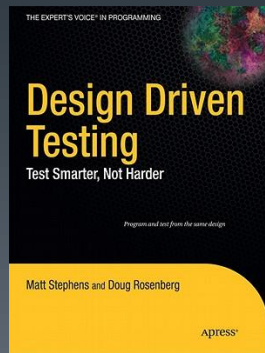
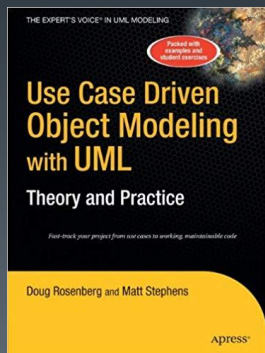
- Mongo/Node replacement for 30 year old legacy database system
- Both CarmaCam and TikiMan becoming commercial ventures
- 30+ years of ICONIX experience behind the UML approach
- Vast experience with mega-projects from TRW behind ICSM
- Cutting edge PhD research behind code generator
- *Your project welcome here!*



PARALLEL AGILE

Solid Foundation

- Parallel Agile is deeply grounded in proven techniques
- Combining these techniques enables large-scale parallel development
- Experiments with student projects ongoing
- Industry trials beginning





PARALLEL AGILE

Parallel Agile lets us integrate the work of many developers

- Enabling parallel development has great potential for reducing cost and accelerating schedules
- Book in progress (**Parallel Agile – Boehm, Rosenberg, Wang, Qi**)
- www.parallelagile.com
- doug@parallelagile.com



PARALLEL AGILE

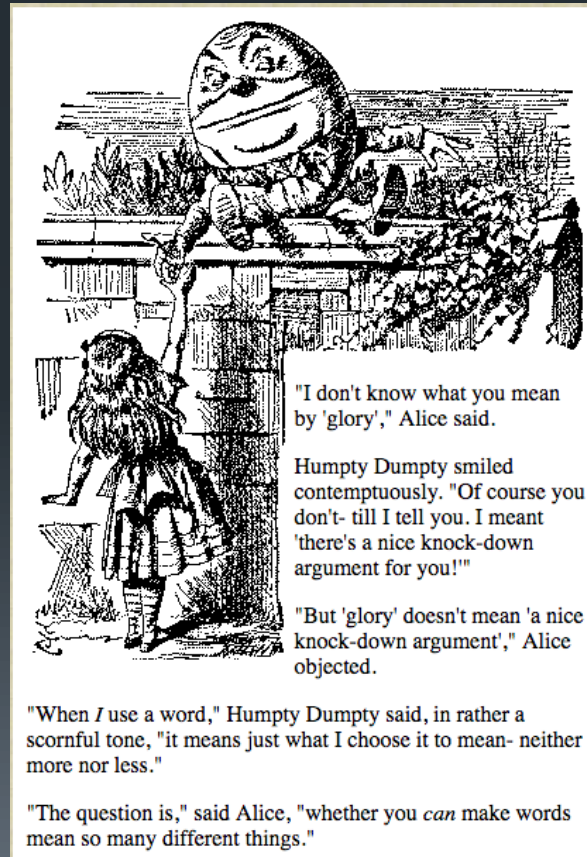
Backup Slides



PARALLEL AGILE

It's easier to take Humpty Dumpty apart than to put him back together

Oddly enough, having words (domain objects) only mean a single thing is the key to integrating the work of many developers.

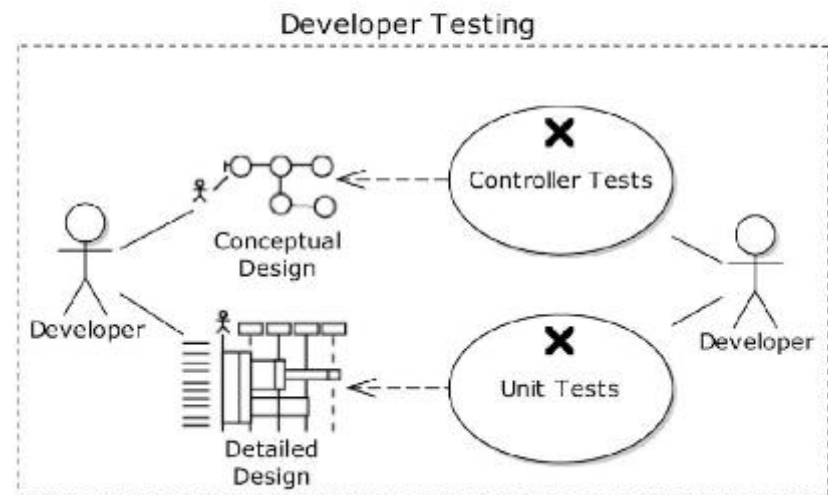
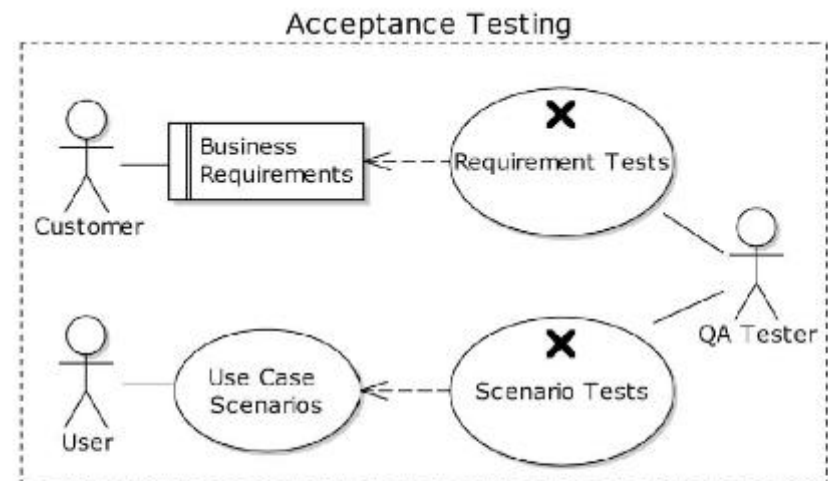




PARALLEL AGILE

Test early, test often

- Main focus is on acceptance testing
- Generate test cases where possible
- Expansion of use case sunny/rainy day threads especially useful
- Test team works in parallel with developers

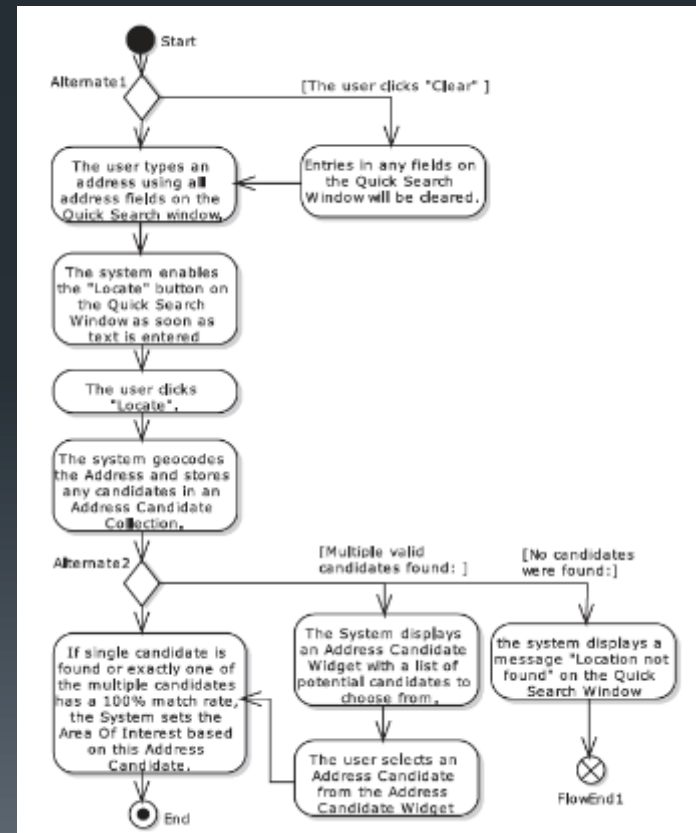




PARALLEL AGILE

Scenario tests exercise all usage paths

- For a use case with one sunny day path and three rainy day paths we need at least four test threads
- Scenario tests are generated automatically





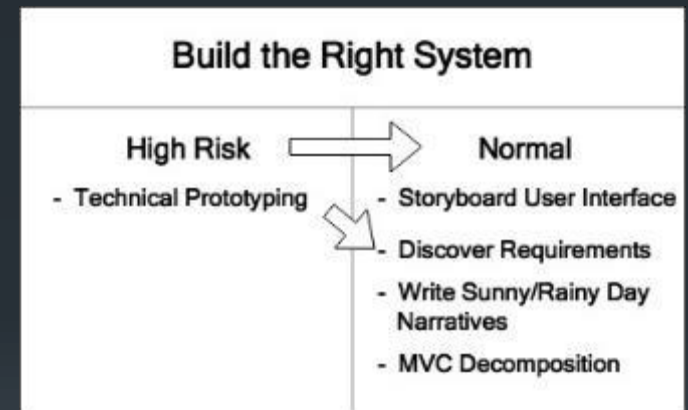
PARALLEL AGILE

Parallel development, simplified

Develop in parallel
One developer per use case

Set up for parallelism

Inception
- Model the Problem Domain
- Identify Use Cases
- Generate Code + APIs from Domain Model
- Organize Use Cases into Packages
- Assign Developers to Use Cases



Build the System Right	
Model Driven	Feedback Driven
- Sequence Diagram	- Write a Unit Test that Fails
- State Machine	- Write Code to Make the Test Pass
- Class Diagram	
- DB Schema	



PARALLEL AGILE

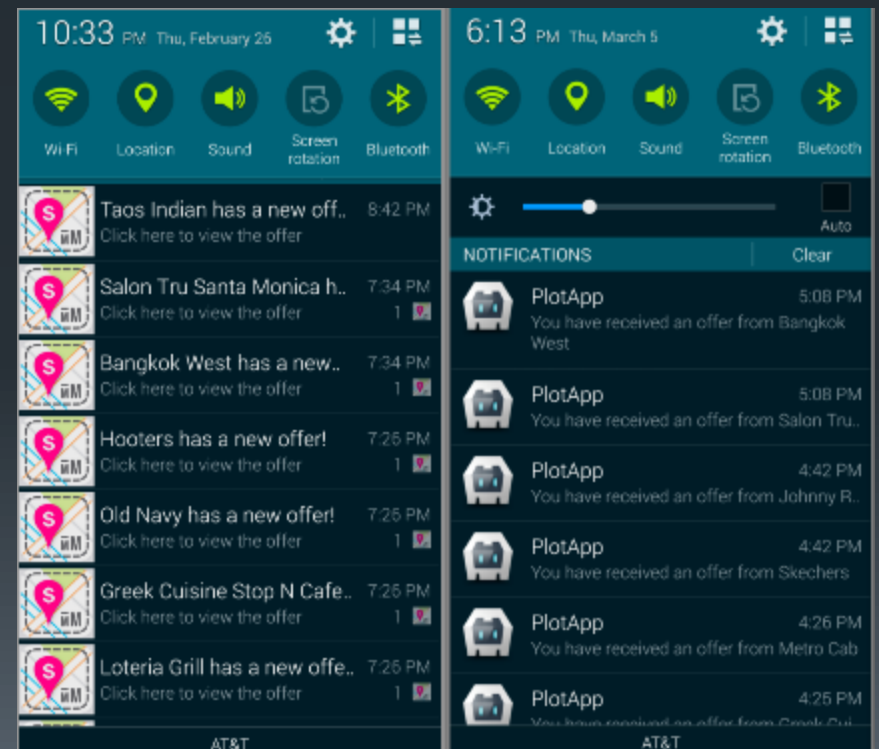
We prototyped multiple geofencing solutions then built our own

We built power-safe, background capable, cross-platform geofencing as a Cordova/PhoneGap plug in

Wi-Fi power consumption with GPS accuracy

+95% on-time coupon delivery

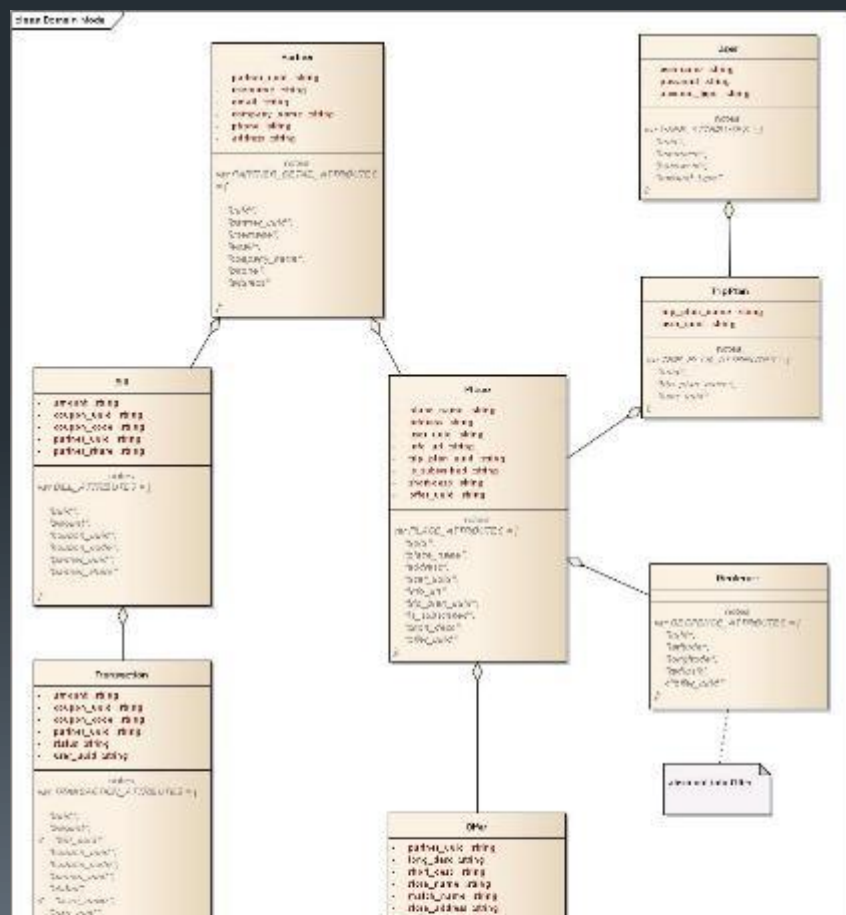
Cassandra, JQuery Mobile, Node JS





PARALLEL AGILE

Design pattern for executable domain models developed



Each class becomes a NoSQL Collection

Create
Read
Update
Delete



PARALLEL AGILE

Project was technical success but marketing failure

mMAPS
MILLION MAPS SANTA MONICA

BIG SAVINGS ARE HERE!
Million Maps App has launched. Get the App below!

AVAILABLE NOW!

Download on the
App Store

GET IT ON
Google play

View Online Map



PARALLEL AGILE

Lessons from LBA

- Successful failure because we learned from it
 - Massively parallel development is possible
 - REST APIs enabled work to integrate
 - Total effort numbers around 2 person years for 75 students
 - Possible to code generate DB access functions and API
 - Two students entered PhD program



PARALLEL AGILE

Hang on a second...

- We just integrated the work of 29 developers who were working independently
 - *That's not supposed to be possible!*
- Maybe we should try it again?



Model sunny and rainy day scenarios

Upload to Location after Taking Photo

Upload to location after taken photo

Primary course of action:

- System displays Uploading Choice menu screen.
- User taps "Upload to Current Location" button.
- System get the current location from the device.
- System display the current location screen.
- Taps "Done" button.
- System pop up the alert and ask user to confirm.
- User taps "Yes" button.
- System display Uploading screen.
- System set location for the photo;
- System send the photo to server and get successful feedback.
- System show "Upload successfully" screen.
- User taps "OK" button.
- System displays the photo uploaded.

Alternate course of action (System can't get current location from device):

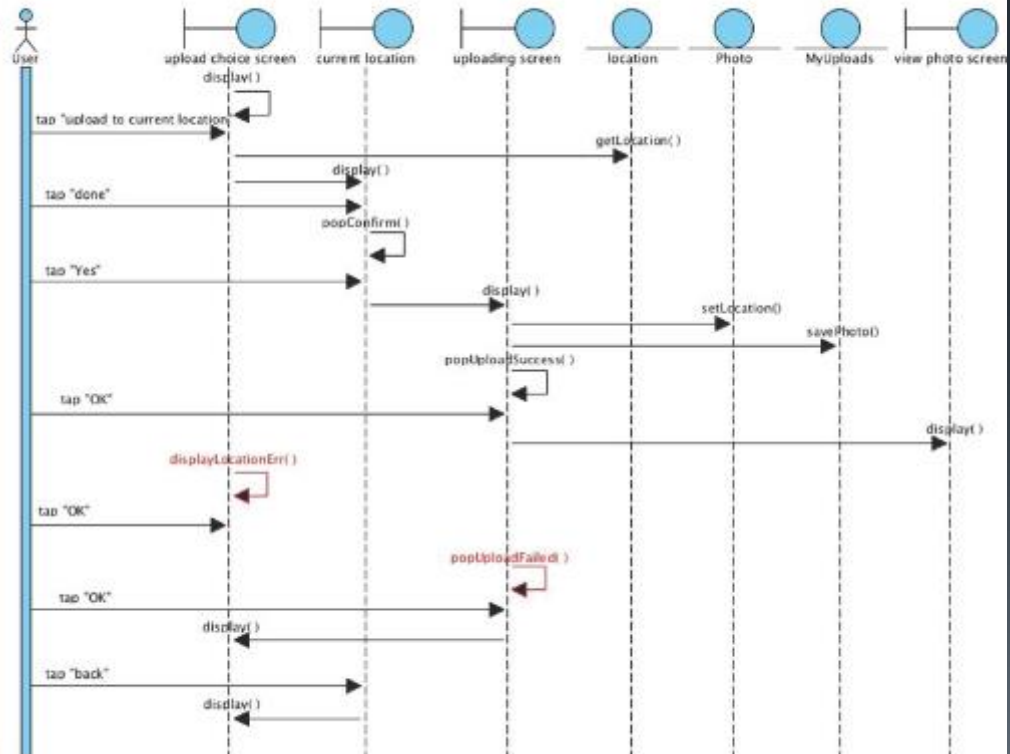
- System displays cannot get current location error message.
- User taps "OK" button.
- System displays Uploading Choice menu screen.

Alternate course of action (System can't get access to server OR get failed feedback):

- System displays "Failed to upload, try again later" message.
- User taps "OK" button.
- System displays the photo uploaded.

Alternate course of action (User taps "back" button at any screen):

- System displays the last screen before the current one.





PARALLEL AGILE

40 days to code iOS app

Task Name	Duration	Start	Finish
iOS Mobile App			
▣ Development	70 days	Tue 1/19/16	Mon 4/25/16
Research Parse Alternatives	5 days	Tue 1/26/16	Mon 2/1/16
Research Parse Open Source Server Code	5 days	Tue 1/26/16	Mon 2/1/16
▣ iOS Mobile App	40 days	Tue 1/19/16	Mon 3/14/16
Fix Upload Photo to Location	5 days	Tue 1/19/16	Mon 1/25/16
Implement Account Profile Photo	5 days	Tue 1/19/16	Mon 1/25/16
Research Front-End JS for Parse	5 days	Tue 1/19/16	Mon 1/25/16
Research Cloud Code JS for Parse	5 days	Tue 1/19/16	Mon 1/25/16
Create Event	10 days	Tue 1/19/16	Mon 2/1/16
Browse Photo By Location	10 days	Tue 1/19/16	Mon 2/1/16
Build My Uploads Screen	5 days	Tue 1/26/16	Mon 2/1/16
Build View Photos of Event Screen	5 days	Tue 1/26/16	Mon 2/1/16
Implement Upload Photo to Event	5 days	Tue 2/16/16	Mon 2/22/16
Implement Search Events	5 days	Tue 2/16/16	Mon 2/22/16
Implement Join Events	5 days	Tue 2/16/16	Mon 2/22/16
Implement Delete Event Photos	5 days	Tue 2/23/16	Mon 2/29/16
Implement Event List	5 days	Tue 2/23/16	Mon 2/29/16
Implement Download Photo to Device	5 days	Tue 2/23/16	Mon 2/29/16
Implement Designs	10 days	Tue 3/1/16	Mon 3/14/16

Students averaged 10-15 hours a week so full time equivalent around 2 weeks



Emphasis on acceptance testing

TC-04: Delete Uploaded Photo

Test Level: **Software Item**

Test Class: **Erroneous**

Test Completion Criteria:

1. User is able to delete photos for an event
2. Event owner able to delete photos tagged to their event
3. Errors displayed as appropriate

7 Subtest Cases:

TC-04-01: Check deleting a photo when every part of the system works well

TC-04-02: Check deleting a photo without Internet access (off-nominal)

TC-04-03: Check deleting a photo when server is unavailable (off-nominal)

TC-04-04: Check event owner able to delete a photo uploaded to their event

TC-04-05: Check event owner able to cancel deleting a photo (off-nominal)

TC-04-06: Check event owner deleting a photo without Internet access (off-nominal)

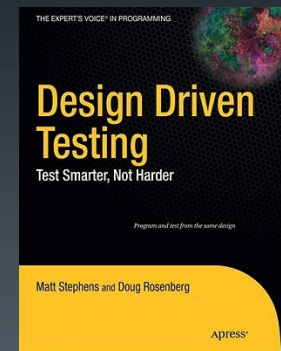
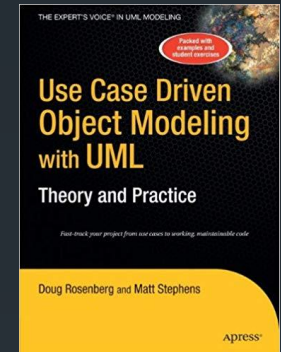
TC-04-07: Check event owner deleting a photo when server is unavailable (off-nominal)



PARALLEL AGILE

Lessons from PicShare

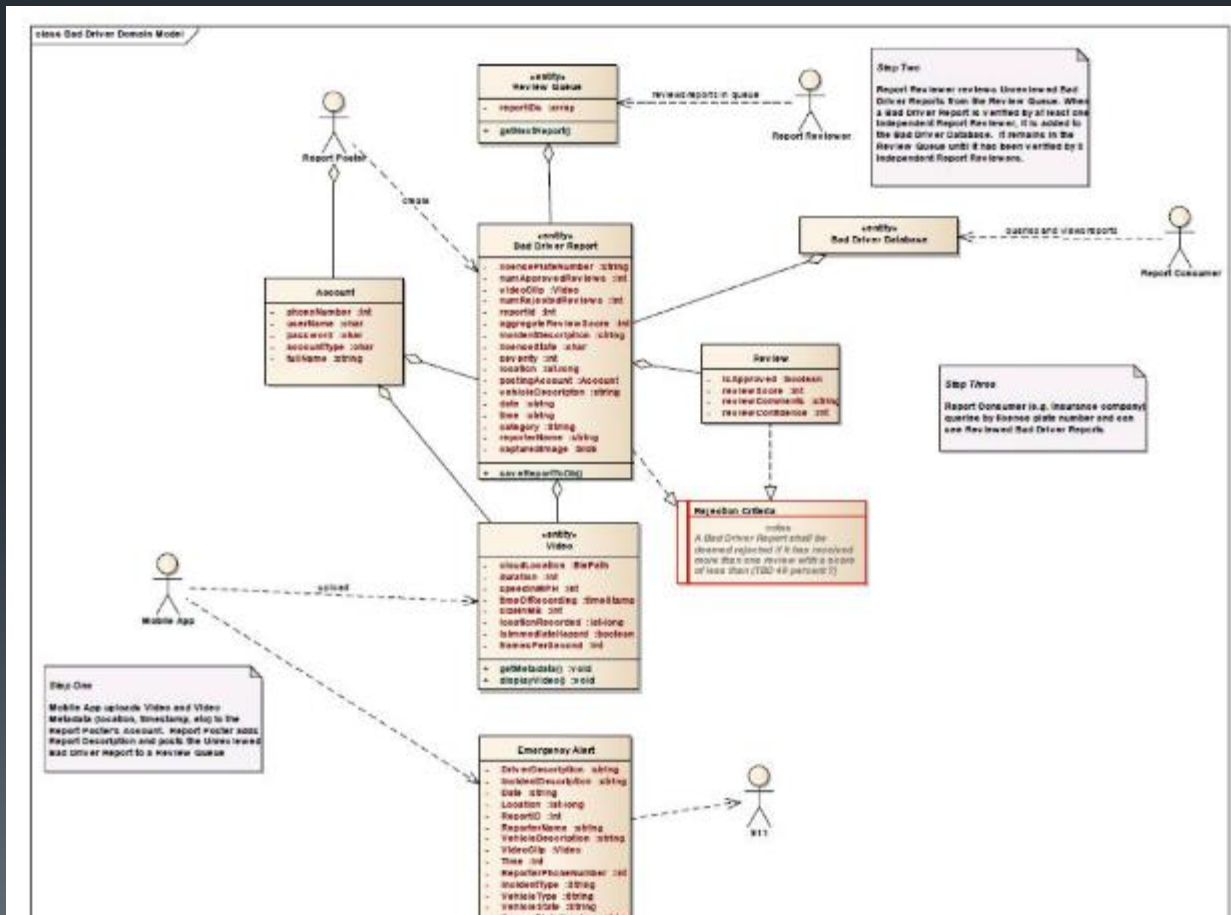
- Use of Parse Framework for back end validated concept of Executable Domain Models (code generator was under development in 2015)
- Model View Controller (MVC) decomposition of use cases critically important and not generally well understood
- Testing rainy day scenarios very important
- Small-scale parallel development test but successful
- Productivity and reliability numbers were good
- Second semester of project done without any mentoring but still succeeded
- *Uploading media files to the cloud is interesting...*





PARALLEL AGILE

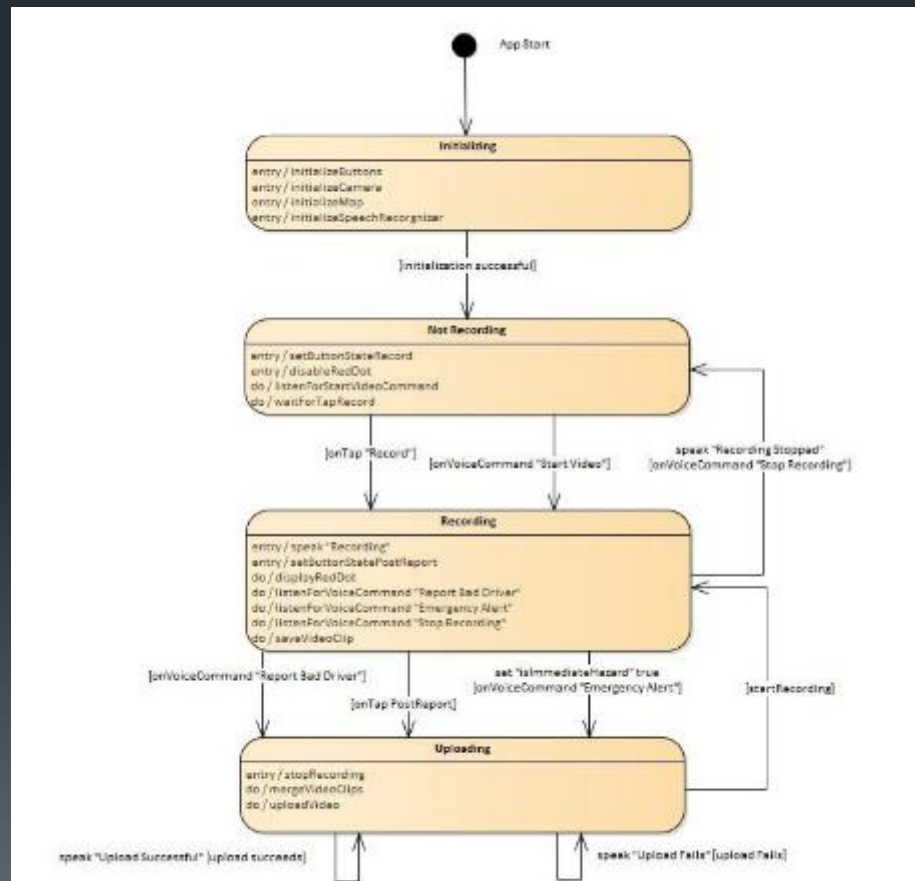
Inception (generate database and API code)





PARALLEL AGILE

MVP – careful design

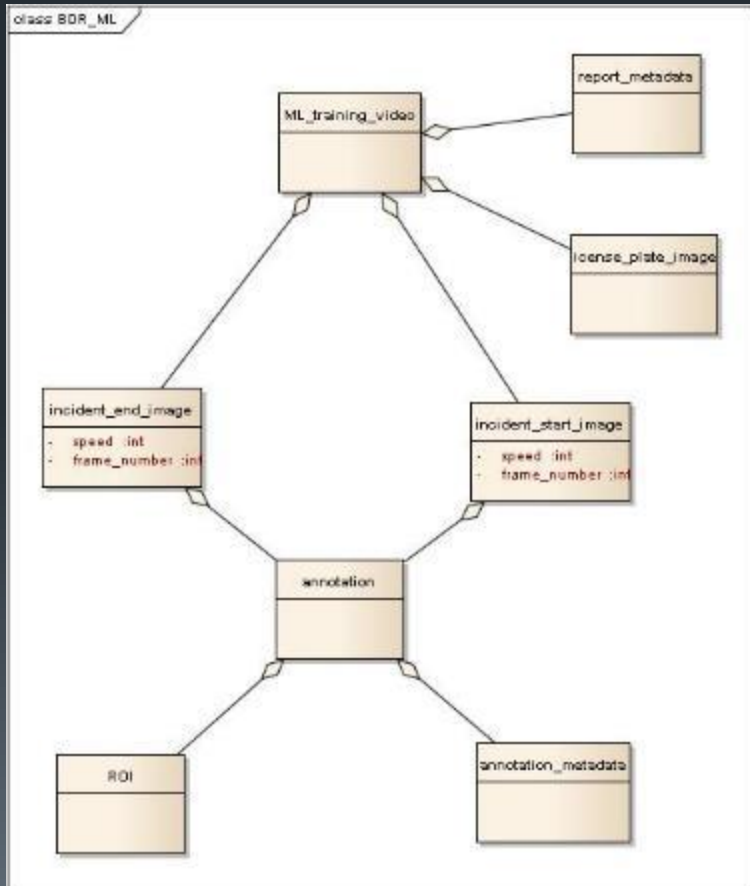




PARALLEL AGILE

Code generate database, API for machine learning training

As a ML model trainer I'd like to be able to annotate key frames from an incident video



Whiteboard sketch becomes executable



PARALLEL AGILE

Mongo, Node and client-side code auto-generated

Allows us to prototype
the video editor page

The screenshot shows a Mac OS desktop environment. In the background, a file explorer window displays a directory structure with folders for 'JavaScript' and 'Server', and files for 'BDR_ML.json', 'BDR_ML.xml', and 'Swift'. In the foreground, two code editors are open. The top editor shows a Swift class definition for 'annotation' with methods 'createOne', 'successCB', and 'errorCB'. The bottom editor shows a JavaScript code snippet for 'annotation' with similar methods and comments.

```
import com.google.gson.JsonObject;

public class annotation {

    public static String className = "annotation";
    public static String[] attributes = {
        "incident_start_image_id", "incident_end_image_id"
    };

    public static void createOne(JsonObject data) {
        class createOneCB {
            public void successCB(String result) {
                System.out.println("successCB: " + result);
            }
            public void errorCB(String message) {
                System.out.println("errorCB: " + message);
            }
        }
        adapter.createOne(className, data, successCB, errorCB);
    }
}

class annotation {
// attributes: ["incident_start_image_id", "incident_end_image_id"]
var tableName = "annotation"

func createOne(data: JSON) {
    func successCB(_ result: Any) -> Void {
        print(result)
    }
    func errorCB(_ result:Any) -> Void {
        print(result)
    }
    adapter.createOne(tableName, data, successCB, errorCB)
};

My {
    result: Any) -> Void {
        print(result)
    }
    func errorCB(_ result:Any) -> Void {
        print(result)
    }
}

adapter.createOne(tableName, data, successCB, errorCB)

[
    data: Any) -> Void {
        print(result)
    }
    func errorCB(_ result:Any) -> Void {
        print(result)
    }
}

adapter.createOne(tableName, data, successCB, errorCB)
```



PARALLEL AGILE

Lessons from CarmaCam

- Domain-driven microservice architecture has allowed 75 developers to collaborate over multiple semesters with very few issues
- 3 phase approach (POC, MVP, R1) works well
- Visual model sprint plans work well
- *95% staff turnover every semester is really hard*
- Student projects can graduate into real products
- Productivity numbers are very high with parallel development



PARALLEL AGILE

Lessons learned from TikiMan Go

- Auto-generated DB and API code works for game development
- Prototype first then design works well
- 1 developer per use case partitioning works well
- Matching developer skills to tasks hugely important
 - recruiting has been the biggest problem
- 100% staff turnover every semester is *really hard*



PARALLEL AGILE

Summing up

- *If we could* integrate the work of large numbers of developers working in parallel we could make software projects go faster by adding more developers
- *We can* integrate the work of large numbers of developers using a domain-driven microservice architecture, auto-generated at project inception
- *Does it work?* 4 years of test projects involving 200 grad student developers say *Yes*. Transition from classroom to industry is underway.



PARALLEL AGILE

Agile (feedback-driven) management
Disciplined (plan-driven) design
Spiral (evidence-driven) lifecycle