Managing a Student Project with Enterprise Architect – Part 1

Doug Rosenberg
ICONIX and USC

www.iconixsw.com

Introduction

For the past several years I've enjoyed a mostly informal association with the University of Southern California Center for Systems and Software Engineering (**USC CSSE**). I was on-staff at USC a few years ago teaching SysML and Model Based Systems Engineering, but for the last few years I've been mentoring Computer Science grad students in two Masters courses: CS577 Software Engineering and CS590 Directed Research. The Directed Research (DR) course is basically a mechanism for students who are about to graduate from the Masters program but are one or two units short of the required number to pick them up by participating in a project with a mentor from industry (that would be me). Students are expected to work 5 hours per week per unit.

Teaching at USC is fun (I graduated from SC back in ancient times), gives me an opportunity to work with a lot of bright young software engineers, to stay current on new technology (in particular cloud-connected mobile app development) and also gives me an excuse to work with **Prof. Boehm** (author of **Balancing Agility and Discipline** among numerous other titles), who has happily taken an interest in some of my ideas related to improving productivity by innovating better software processes and allowing me to test my ideas out with USC grad students.

This process work has included the development of the **Resilient Agile** process, an attempt to develop a better agile methodology that started out as an experiment called **Massively Parallel Use Case Modeling** that we did with the CS577 class a few years ago where we developed a complete location-based advertising system by handing one use case to each of 47 grad students and having each student develop their use case independently.

This semester I'm working with a group of 15 Masters students, mostly taking a single unit of DR. One student is taking two units, so my team has an effective time budget of 80 student hours per week. Although the semester at USC is 16 weeks long, by the time the student teams get formed, and with midterms and finals, we've got about 12 usable weeks of student time. So it works out to a time budget of roughly 1000 student hours (that's about half-a-person-year at 40 hours a week) over a 3 month schedule.

Because I like challenges, we're attempting a "crowdsouced bad driver reporting system" this semester, and because we need to be really productive, we're using Enterprise Architect to coordinate all of the student homework. This is the first article in a series that will describe how we do.

Crowdsourced Bad Driver Reporting System – what and why?

I got this idea for this project while I was working on a consulting assignment that required me to commute from Santa Monica to the Los Angeles Airport (LAX) area, putting me on some of the busiest freeways you'll find anywhere. In particular the transition from the Santa Monica freeway (Interstate 10) onto the San Diego freeway (Interstate 405) is a spectacular hotbed of bad driving practices.

During the time I was working this consulting assignment I was also mentoring a team of USC students one evening a week on building a photo-sharing mobile app that uploaded images to the cloud. One day after a particularly harrowing commute, I came up with the idea of building a crowdsourced video database for insurance companies using a voice-activated "dashboard cam" app to capture and upload video. So here's the mission statement I gave this semester's students:

> ***Mission is to eliminate bad drivers from the road system by providing a risk database to insurance companies so they can charge more to bad drivers. Business model is "software as a service".***

When I decided to write up a proposal for a student project, I did a little research and it turns out that bad drivers are an enormous problem in society. In 2012, 92 people were killed on average each day in the United States, there were 30,800 fatal crashes in the US that year. And the current numbers are even worse, given how many people are texting and driving. These numbers only reflect fatalaties, not injuries or property loss.

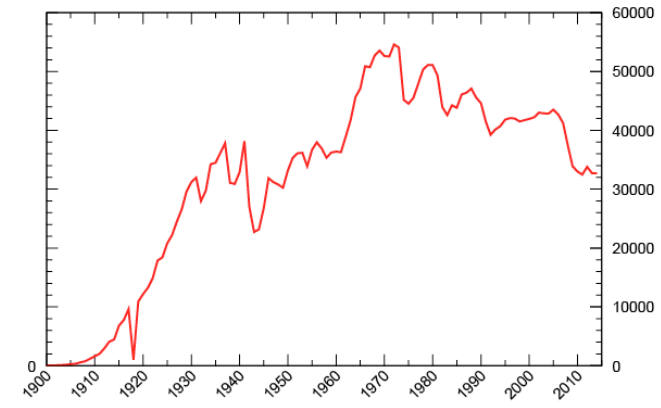

Figure 1 - There are too many bad drivers on the road

The basic operational concept of our system is as follows:

> *Dashboard camera continuously records looping video. Voice command to mobile app (e.g. "Hey, Siri, report a bad driver") triggers the autosaving and uploading of 15 second video clip to a cloud-database. License plate numbers are extracted from imagery (optional), and video/report metadata is filed by license plate number. Insurance companies can query against license plate numbers to see if there are any bad driver reports logged against a vehicle while issuing policies.*

The major differences between our app and existing dashboard camera apps will be

1) voice activation for initiating a bad driver report
2) cloud database for filing bad driver reports
3) queryable cloud database for insurance companies,
4) send drunk driver immediate alert to police department
6) quality monitoring of bad driver reports by insurance companies and/or others
7) auto-extraction of license plate number from video

In short, we've got the following use cases:

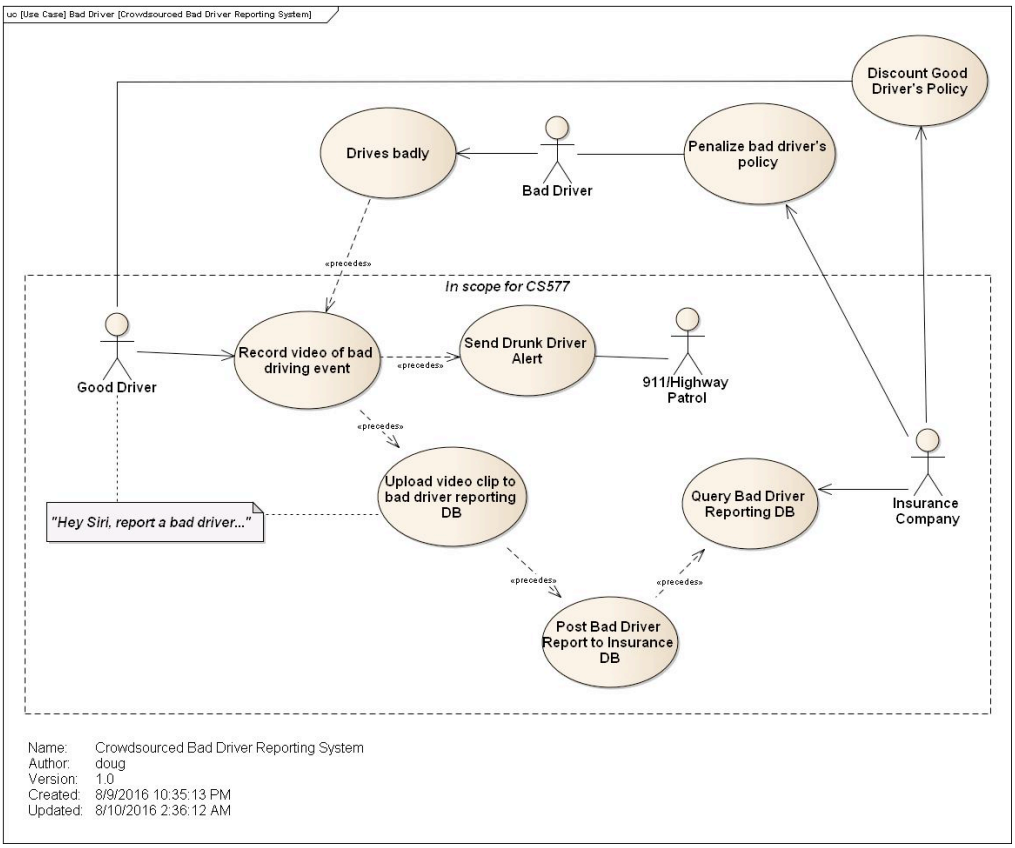

Figure 2 - Bad Driver Reporting System Use Cases

The architecture of the system is shown here: mobile apps for iOS and Android, connected to a Mongo DB repository via a Node JS RESTful API, combined with Angular JS pages to handle web-forms for posting and reviewing reports.

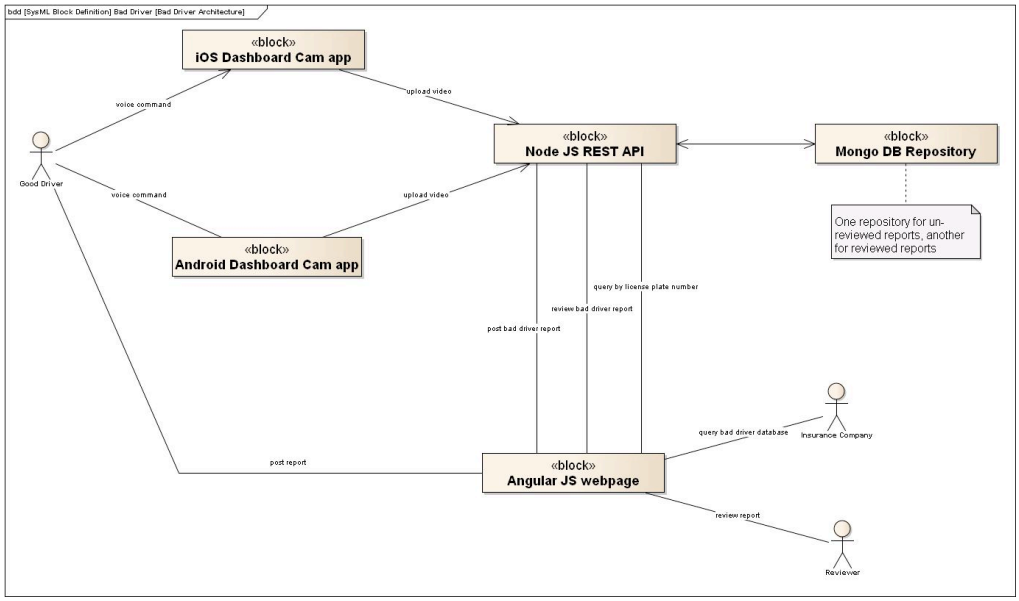

Figure 3 - Bad Driver Reporting System Architecture

Helping me teach the class is our VP Training from ICONIX, Hunter Deane.

Hunter and I deployed our student army as follows:

- 4 students on the Android team
- 3 students on the iOS team
- 5 students building web-app pages
- 1 remote student working on extracting license plate numbers from video
- 2 students on the test team (using the ICONIX/DDT add-in for Enterprise Architect)

Students are following the approach of 1) "Build the Right System" (using EA to model use cases, requirements, storyboards, analysis diagrams, etc.) followed by 2) "Build the System Right" (detailed class diagrams, sequence diagrams, etc.)

We're also deploying an experimental "secret-weapon" for the first time on this project. A PhD student-developed code generator that generates NoSQL databases and REST APIs from EA class diagrams.

Are we crazy to think that we can get this system built in 3 months with a total of half-a-person-year of developer time? Stay tuned for our next article to see how we're doing.