



PARALLEL AGILE



CASE STUDY

HOW THE PARALLEL AGILE DEVELOPMENT
PROCESS AND CODEBOT WAS USED
SUCCESSFULLY TO CREATE A COMPLEX,
CROWDSOURCED TRAFFIC SAFETY SYSTEM

ABOUT CARMACAM

CarmaCam is a crowdsourced traffic incident reporting and traffic safety system. It utilises a crowdsourced approach to capture geo-tagged video clips of driving incidents and accidents. It includes a dashboard camera app, emergency reporting, as well as machine learning. The emergency reporting feature sends higher priority video clips directly to nearby first responders and gives them live-streamed access to the video of the incident.

The CarmaCam system has been developed using the Parallel Agile Process since 2017. Over 150 developers have participated in the development of CarmaCam.

CarmaCam includes a large Machine Learning component, much of which is Patent Pending.

In this case study, we'll show how Parallel Agile and CodeBot have enabled CarmaCam's development.

CarmaCam Family of Apps



CarmaCam® Safety

CarmaCam® Safety helps drivers improve road safety by easily reporting emergencies and unsafe traffic incidents. Don't get angry - Save a life!



CarmaCam® Mobility

CarmaCam® Mobility - for Parking Enforcement to quickly clear bus and restricted lanes by efficiently generating error-free parking tickets and improve parking enforcement.



CarmaCam® LE

CarmaCam® LE enables Law Enforcement to quickly respond to real-time reports of emergencies including DUI and hit and run.

FIG 1. CARMACAM HAS FOLLOWED THE PARALLEL AGILE PROCESS FROM PROOF OF CONCEPT THROUGH OPTIMIZATION TO PRODUCE A FAMILY OF MOBILE AND WEB APPS FOR TRAFFIC INCIDENT MANAGEMENT

THE PROCESS

CarmaCam is a crowdsourced system for identifying as well as locating traffic incidents and sharing them with law enforcement, insurance providers and emergency services.

CarmaCam was developed following the simplified spiral model that's used in the Parallel Agile Process. We apply the spiral model to each use case in the system and assign one developer to each use case.

Parallel Agile simplifies the spiral model into three phases:

1. Proof of Concept
2. Minimum Viable Product
3. Optimization

Parallel Agile introduces the concept of 'Executable Architecture' in the form of a database and Rest API that's automatically generated from a domain model using CodeBot.

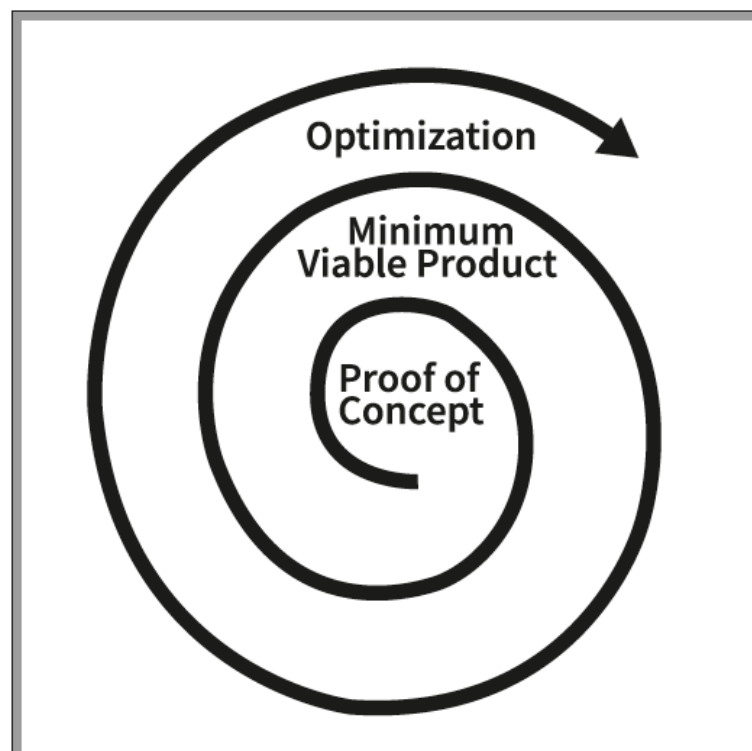


FIG 2. ESSENTIAL PARALLEL AGILE: EACH USE CASE CAN EVOLVE INDEPENDENTLY THROUGH THE THREE PHASES

PHASE 1: PROOF OF CONCEPT

The first phase was proof of concept, which was used to demonstrate that we were building the right system. This was designed by analysing use cases from the point of view of each of the end-users i.e.: driver, reviewer, and insurance provider, and building a UML domain model. In order to compile and coordinate these declarations into a phased schedule, we then used visual modelling.

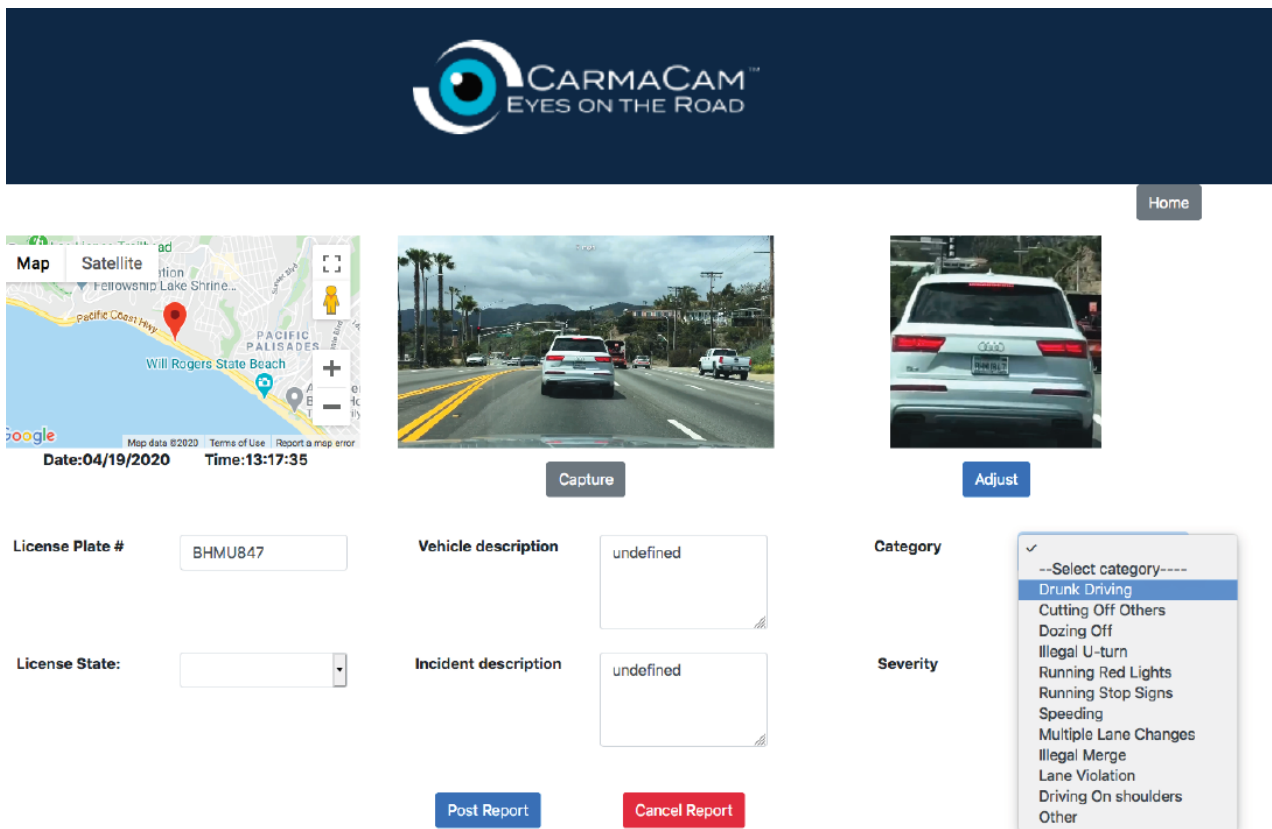
Having a well-defined use-case model to help organise all tasks was important to leverage parallelism with multiple developers. By the end of Phase 1, CarmaCam had its Proof of Concept. They had built an end-to-end prototype system with mobile apps (Both in Android and iOS) that could record and upload videos to the server. The server would send an email containing the link to the 'Incident Report' web form to the user. The video is viewable on the incident report form, and the user could provide more details on the traffic incident and post them back to the server.



FIG 3. FIG 3. IN 2017 , WE COULD CAPTURE
HD VIDEOS OF TRAFFIC INCIDENTS USING MOBILE PHONES

PHASE 2: MINIMUM VIABLE PRODUCT

By the end of the first phase, CarmaCam then identified a list of requirements and a set of working prototypes covering all the use cases of the system. The second phase involved taking the system through a more careful design pass to build a Minimum Viable Product (MVP) that met all the requirements. The initial challenge with the second phase was working with a new team of developers, with the exception of three developers who had carried over from the first phase. This meant nearly 100 per cent staff turnover. Hence, the use cases had to be reassigned to the new developers. The first couple of meeting sessions were spent getting new teammates up to speed on the progress made in the previous phase and help familiarize them with the prototype code that was built. Another goal for this phase was to have a scalable server application by migrating to AWS EC2. With the result set that we had collected from the server load tests, we determined the CPU and memory requirements for the AWS EC2 instance, along with how much of the memory was to be allocated to the database. We started the migration around halfway into the phase, when we had an improved, optimized, and more stable server application. The migration was a milestone in our project, as the application was now hosted in the cloud and we were one step closer to establishing a production server. By the end of phase 2, we had our MVP.



The screenshot displays the CarmaCam web application interface. At the top, the CarmaCam logo is visible. Below the logo, there is a map on the left showing the location of the incident, with a red pin indicating the location. The map includes a date and time stamp: Date: 04/19/2020, Time: 13:17:35. In the center, there is a live camera feed showing a white car on a road. To the right of the camera feed, there is a button labeled 'Capture'. Below the camera feed, there is a form for reporting an incident. The form includes fields for 'License Plate #' (BHMU847), 'License State' (a dropdown menu), 'Vehicle description' (undefined), 'Incident description' (undefined), 'Category' (a dropdown menu with options like Drunk Driving, Cutting Off Others, etc.), and 'Severity' (a dropdown menu with options like Running Red Lights, Running Stop Signs, etc.). There are also buttons for 'Adjust' and 'Post Report' (blue) and 'Cancel Report' (red).

FIG 4. THE CARMACAM WEB APP LETS USERS CAPTURE LICENSE PLATE NUMBERS AND INCIDENT DETAILS

OPTIMIZATION 1: EMERGENCY ALERT SYSTEM

The third phase was the optimization of the product. Again, there was nearly 100 per cent staff turnover, with a new batch of developers taking over the project. A new feature that we added to the dual-display of the mobile app was the DUI Alert or Emergency Reporting System. In addition to creating a traffic incident report, this feature sends higher-priority video clips to a new collection in the database. The Emergency Alert collection retains these high-priority alerts for only 30 minutes, and it's accessible via a geospatial (location-based) query. Our goal with this system is to “broadcast” video showing reckless driving behaviour or possible drunk driving to patrol cars within the desired radius, making law enforcement aware of these dangerous drivers in near-real-time. During this phase we focused on improving the system to make it industry ready, with a lot of user acceptance testing. Most of the resources were allocated for optimization, performance tuning, and adding new features to improve the product’s usability.

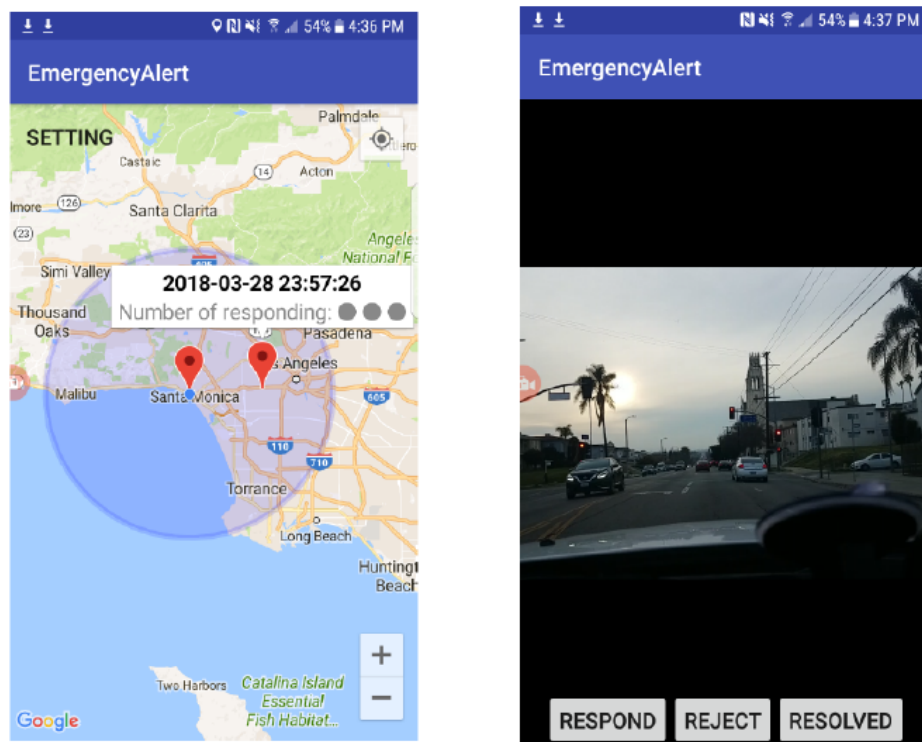


FIG 5. IF A DUI IS REPORTED NEAR A FIRST RESPONDER THEY ARE NOTIFIED INSTANTLY

OPTIMIZATION 2: ARTIFICIAL INTELLIGENCE REPORTING

We developed AI and Machine Learning capabilities because we needed to auto screen emergency videos for DUI, hit-and-run and we also need to sanity-check and prioritize incident reports for human review.

Emergency videos are screened for likely DUI by detecting "weaving" (i.e. multiple lane changes) at high speed. Videos that show erratic/impaired driving are then sent immediately to nearby first responders. For non-emergency traffic incidents like unsafe lane change or stop sign violation, our system performs sanity checks and prioritizes incident reports based on AI Scoring before videos go through an independent review.



FIG 6. IN 2019 WE DEVELOPED MACHINE LEARNING CAPABILITIES TO CHECK INCIDENT REPORTS AUTOMATICALLY.

FINAL PRODUCT

Thanks to the effective use of the Parallel Agile development system, structures and software CarmaCam was able to deliver Proof of Concept in less than a year, the MVP in the spring of 2017 and the Optimization process began toward the end of 2017. Overall, over 150 developers have worked across semesters from 2016 – 2020 to successfully take CarmaCam from an innovative idea into a marketable product. CarmaCam continues to evolve, with current research focusing on researching AI on the phone trying to optimise the Machine Learning capability so that it can be integrated into the dashboard camera app.

For more information on Parallel Agile visit www.parallelagile.com

For more information on CarmaCam visit www.carma-cam.com



FIG 7. THE CARMACAM MOBILE APP IS EASY TO USE

FIG 8. CARMACAM CONTINUES TO EVOLVE WITH MACHINE LEARNING CAPABILITIES IN THE MOBILE APP

