[24] R. Toledo, C. Sotomayor, and A. Gomez-Skarmeta, "Quadrant: An architecture design for intelligent vehicle services in road scenarios," in *Proc. Monograph Adv. Transp. Syst. Telematica*, 2006, pp. 451–460.
[25] N. Kaempchen, K. Weisst, M. Schaefer, and K. C. J. Dietmayer, "IMM object tracking for high dynamic driving maneuvers," in *Proc. IEEE IV Symp.*, Parma, Italy, 2004, pp. 825–830.

Supporting Agile Change Management by Scenario-Based Regression Simulation

Christian Berger and Bernhard Rumpe

***Abstract*—Many system-development projects today are mainly driven by the complexity of their software interacting with sensors or actuators in an embedded context. Autonomous vehicle development is a domain where it seems inevitably necessary to apply modern development techniques to cope with complexity, increase development efficiency, and ensure appropriate quality. Furthermore, changes that are triggered by customers or inventions of competitors, as well as bugs, enforce a comprehensible, if necessary, yet agile development process with stringent quality management. In this paper, we describe the agile efficiency- and quality-focused change management mainly based on scenario-driven regression simulation used in the CarOLO project for the development of an autonomously driving vehicle to compete in the 2007 Defense Advanced Research Projects Agency (DARPA) Urban Challenge program. The main contribution is the demonstration of the modern software engineering techniques' applicability to develop distributed embedded systems.**

***Index Terms*—Agile software engineering, autonomous driving, Caroline, CarOLO, change management, Defense Advanced Research Projects Agency (DARPA) urban challenge, intelligent algorithms, regression testing.**

I. Introduction and Motivation

More than one third of major project risks for software and system-development projects arise from late changes—either in a customer's needs or due to incorrect or incomplete requirements [1]. Further analysis of the Standish Group report yields that only 16.2% of the examined projects were in time and budget mainly because of the aforementioned reason. Five years later, that ratio has not improved significantly [2]. Therefore, the Standish Group suggests an iterative development process, including direct contact with the customer itself. Modern development processes for small-size to midsize projects like Extreme Programming (XP) [3] or the Rational Unified Process (RUP) [4] are directly based on an iterative principle considering that suggestion.

However, besides the iterative development process, further interpretation of the report yields that a consistent change-management process addressing the main risks of projects is necessary to complement an iterative development process itself. The main goal of the change-management process is to record and track change requests like rearrangements in a customer's needs and to schedule their processing in a regular iteration cycle.

Manuscript received March 31, 2008; revised July 20, 2009; accepted February 21, 2010. Date of publication March 29, 2010; date of current version May 25, 2010. The Associate Editor for this paper was L. Li.

The authors are with the Department of Software Engineering, RWTH Aachen University, 52074 Aachen, Germany (e-mail: berger@se.rwth-aachen.de; rumpe@se-rwth.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2010.2044571

Although agile development is mainly applied in business system development nowadays [5], it is still waiting for an adaption for embedded software development. One reason is the inherent need for documentation of quality-ensuring activities in embedded systems such as vehicles that are inherently incompatible with today's agile methods. Another maybe more important reason is the pretended inapplicability of agile development to traditional engineering techniques as known by mechanical and electrical engineers. Although very successful in their domains, these processes do not match very well, and thus, an alignment of these two different styles of development needs to be carefully realized. In particular, research projects, like the development of autonomous vehicles, are particularly suitable in demonstrating how agile development of complex embedded systems is possible when relying on an appropriate tool support.

We use the 2007 Defense Advanced Research Projects Agency (DARPA) Urban Challenge as an example to illustrate the agile change management process in the CarOLO project. Besides being able to add and adapt technology as well as architecture in an agile way, we were able to keep track of problems, ideas, and concepts while working with only a limited set of personal resources available. Thus, for keeping the tight schedule in the competition, we applied an agile change management for processing change requests.

To embrace changes, we were forced to preserve the system's quality without involving manual testing procedures all the time. We, thus, realized an automatic regression testing system that uses CxxTest [6] together with unattended and automatic system simulations allowing the software to virtually unattendedly drive complex scenes for the so-called *regression simulation*.

Having decided to use an agile development process, which uses and adapts substantial elements from XP and Scrum [7], we also had to keep in mind that although developers could meet regularly, we had to tackle the split-up of the development sites in Germany and the U.S. We, thus, used our change management process and tooling to handle globally distributed software development for ensuring the quality of software contributed from different sites—sometimes without being tested on the vehicle before being committed to the versioning system because of the limited availability of the vehicle itself, due to either a tight schedule, or the physical presence on the other continent.

In the following, we describe the competition and our contribution, followed by an overview of the change management that is used in the project. Furthermore, we show the integration of system simulation in our development and the change management process by outlining the software architecture and briefly describing our regression simulation framework. Finally, we show the applicability of our change-management process during the regular development and during the semifinal of the 2007 DARPA Urban Challenge.

II. CarOLO Project

The CarOLO project run by the Technische Universität Braunschweig developed an autonomously driving vehicle called "Caroline" for the 2007 DARPA Urban Challenge. In the following, the competition and our vehicle Caroline are described.

A. 2007 DARPA Urban Challenge

The 2007 DARPA Urban Challenge was the continuation of the well-known Grand Challenge series from 2004 and 2005. The goal in the DARPA Grand Challenges was to autonomously drive through an *a priori* unknown terrain by mainly following a route of predefined Global Positioning System waypoints. The challenge was to safely navigate through the unknown and rough terrain with only stationary

Fig. 1. Caroline: Autonomous vehicle for participating in the 2007 DARPA Urban Challenge.

obstacles. The earlier Grand Challenges took place in an area between Barstow, CA, and Primm, NV.

The 2007 DARPA Urban Challenge, however, considerably increased the requirements: Stationary obstacles, as well as dynamic obstacles such as other vehicles and autonomously driving vehicles, were simultaneously on the track. Autonomous vehicles had to obey traffic rules like staying inside a lane, yielding right-of-way at intersections, or keeping minimum distances while following other vehicles. The challenge took place in Victorville, CA, at the former George Air Force Base; therefore, the military area was rebuilt to simulate an urban environment containing intersections, lane markings, and parking lots, for example.

Our team participated in track B of the 2007 DARPA Urban Challenge [8]. To get invited to the semifinal, we had to demonstrate our vehicle's capabilities by conducting a so-called site visit that we carried out in San Antonio, TX, at the Southwest Research Institute in June 2007. Track B participants also had to qualify for the site visit by submitting a video in April 2007 showing the vehicle's capability to overtake a stationary vehicle in its own lane.

The site visit consisted of a demonstration of the vehicle's emergency control system, the capability of staying in lane, overtaking a stationary vehicle as already demonstrated in the video, and the correct behavior at an intersection.

After evaluating the results of the site visits, only 35 teams out of initially 89 teams were invited to the semifinal, taking place between October 24 and November 3, 2007 [9]. The semifinal, which is called the National Qualification Event (NQE), consisted of three test areas. DARPA inspected the emergency control system and tested correct merging behavior at intersections in test area A, long and unmanned runs in partly rough environments with stationary obstacles in test area B, and correct behavior at intersections with dynamic replanning of the initially planned route to be driven in test area C.

The performance of every autonomous vehicle was evaluated, and only 11 teams, including team CarOLO, participated in the 2007 DARPA Urban Challenge Final Event [10], which took place on November 3.

B. Team CarOLO's Vehicle: "Caroline"

Caroline, as depicted in Fig. 1, is a 2006 Volkswagen Passat station wagon that offers great technical possibilities to realize our intended system architecture, e.g., by reusing the available drive-by-wire interfaces. Due to the scope of this paper, we will not explain the hardware modifications that were made to the vehicle but outline only some key concepts of our software architecture, allowing us to quickly embrace new change requests. Further details of the vehicle can be found in [11].

As shown in Fig. 2, our software architecture mainly consists of several loosely coupled modules each working on a certain task, like fusing the sensor's data from the sensors that are mounted on the vehicle or finding lane markings in video sequences realizing a data-processing chain. Thus, we split the entire task of driving in unknown urban environments into several submodules following "separations-of-concerns" [12] by designing a special application programming interface (API) providing necessary basic features like communication or threading for each module. Besides the API, the layered pipes-and-filters architecture enabled us to reuse core modules in the regression simulation, like the "Actorics" module for steering and accelerating the car by encapsulating the real vehicle interfaces and using a virtual car model instead [13]. This modular software architecture allowed us to perform system simulations as explained in Section IV-A.

III. Change Management

We describe some general considerations of change management before we discuss the process that we used in the CarOLO project. First of all, change management is the task of managing and enabling changes rather than preventing them. However, in projects with fixed schedules and limited budget, every change request or new requirement could be regarded as a risk to the product quality or schedule, as mentioned earlier. This is one important difficulty in handling changes.

Nearly every software and system development project is subject to changing circumstances like rearrangements in a customer's needs or modifications in a system's context due to the system's nature. Simply changing the system according to the requested new behavior could possibly break parts or the entire developed system. Therefore, changes obviously need to be planned, tracked, and released with respect to the quality of the system.

Change management not only consists of a process to change the concrete system or subsystem but rather also integrates tools and should include stakeholders like customers, project leaders, and developers that are affected by the change request. Furthermore, change management is a vital part of the overall development and engineering process.

A. Agile Development Processes

Agile development processes like XP or Scrum are suitable for small-size to midsize project teams [14] targeting short-term iterations with deliverables according to a customer's needs who itself is explicitly a part of the development team as the contact person to reduce communication's barriers. These processes aim at *continuous integration* of features and delivering artifacts early and, thus, iteratively.

A typical development cycle in these processes is called iteration and lasts about one to three weeks starting with an iteration "planning." During the planning, all features to be integrated in the release built at the end of the iteration are discussed *together* with the customer who defines the set of tasks to be realized, including prioritization. During the iteration, short daily meetings summarize the state of every single task and escalate arising problems early to the management or the customer. These problems or misunderstandings can be directly discussed with the customer and, therefore, can be solved quickly. For finishing an iteration, an artifact in the RUP or a deliverable in XP has to be built and handed over to the customer after performing additional acceptance tests.

B. Agile Change Management

Agile software engineering is not only meant for applying suitable processes and organizational structures for the development of high-quality software in iterative development cycles based on a customer's

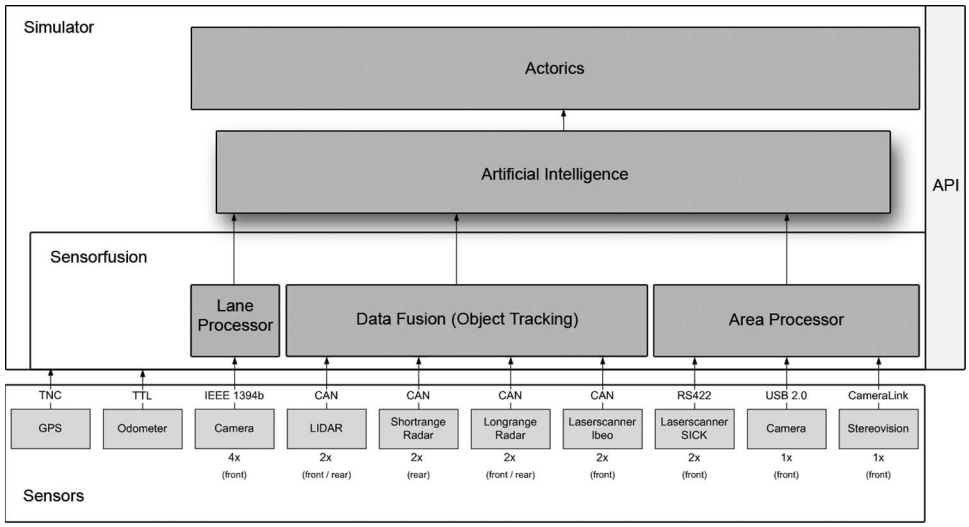

Fig. 2. Caroline's software architecture.

demands. It also needs a reliable change-management system that registers change requests as requested by rearrangements in the customer's needs or by software bugs, tracks the process of a change request, and defines a suitable benchmark for determining whether a change request is fulfilled. It also needs a release process to build a new software release containing the changes while ensuring the overall system's quality.

Therefore, change management is a vital part of agile development processes and is even recognized in the agile manifesto [15]: "Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage." Hereby, changes are not delayed or rejected, but rather encouraged since they are driven by the customer itself or by implications due to technical limitations [16].

IV. Agile Software Development Using Scenario-Based Regression Simulation

Here, we outline the agile change management that we established within the CarOLO project. Since it mainly bases on our virtual design and testing approach, the system simulation is described first. Next, scenario-based regression simulation using our system simulation is explained. Finally, the integration of this concept in our agile change management is described.

A. System Simulation

As already mentioned, the software architecture that is designed for Caroline allowed the compositional reuse of every module in other modules. Therefore, it is possible to embed an algorithm that is implemented to generate a driving trajectory into a simple vehicle simulation, as shown in Fig. 3.

Due to the modular design of the entire system, our visualization environment could be transparently used to visualize online data either from the vehicle or from our simulation. For further details of our simulation environment, see [11], [13], and [17].

B. Scenario-Based Testing

For using our simulation, a set of input data must be provided. These data consist of a digital map of the area to be driven and some information about the surroundings like simple polygonal shapes. To realize the digital map, the original route definition network format

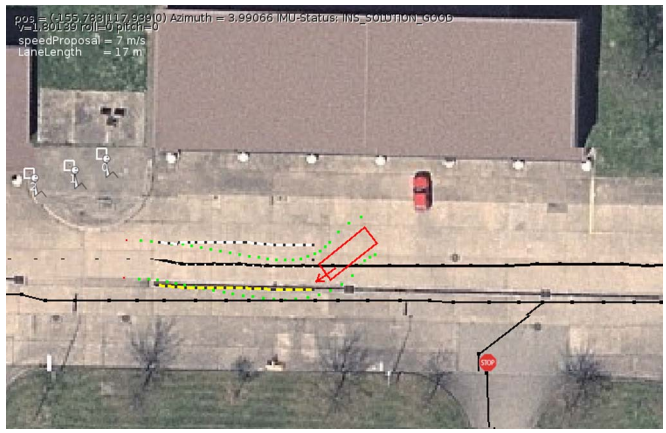

Fig. 3. Simulation of Caroline.

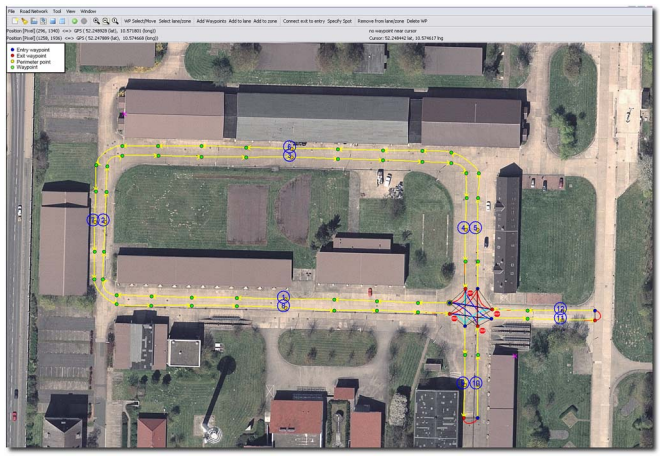

Fig. 4. RNDF editor.

(RNDF) consisting of waypoints and provided by DARPA for all competitors was used.

In Fig. 4, our 2-D RNDF editor is shown. Using this editor, an easy way of producing scenarios consisting of an RNDF-compliant digital map as well as simple surroundings using polygonal data could be established. Using these files, our system simulation was used to compute input data based on the current position on the digital map

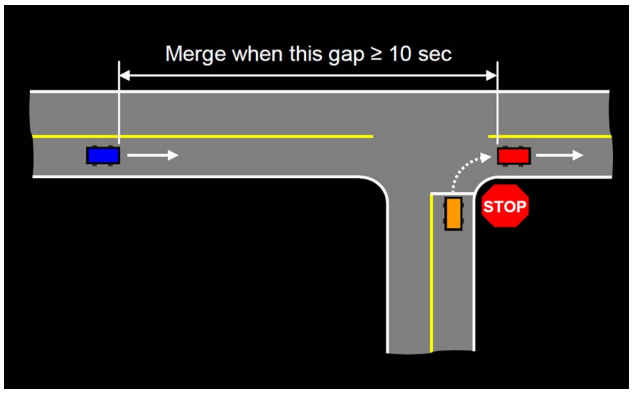

Fig. 5. Schematic drawing of a requirement provided by DARPA.

and the surroundings to feed our planning algorithms for generating driving trajectories. These trajectories were used to compute updated positions of the vehicle on one hand. On the other hand, the resulting position and orientation of the vehicle based on the generated trajectory were used to validate different criteria, from which some are listed in the following.

- *Position*. The vehicle's position on the digital map must be on the route to reach a predefined waypoint. Thus, no detours should be chosen by the planning algorithm.
- *Velocity*. The computed vehicle's speed must not exceed the predefined limits.
- *Acceleration*. The same applies for the acceleration.
- *Timing*. The vehicle must complete its mission within the intended time frame on one hand and must not exceed certain time limits at intersections, for example, on the other hand.
- *Traffic rules*. The vehicle must obey traffic rules at four-way stops, for example.
- *Safety distances*. Due to the main goal of the entire competition, the distance to all stationary and dynamic vehicles was computed at any time.

These aforementioned criteria could be validated online between two steps in the simulation. Thus, a final report containing information about Caroline's performance in the simulation could be generated automatically.

C. Regression Simulation

Besides the interactive use of the simulation together with the visualization for every developer, we combined the simulation with our continuous integration system. Therefore, it was possible to unattendedly and automatically validate changes on our continuous integration system using all available scenarios after committing the changes to the versioning server.

V. Application in the CarOLO Project

In the following, the applications of the aforementioned concepts in the CarOLO project are outlined. First, the application during the regular development of Caroline until October 24 is described. After that, the use during the semifinal from October 24 until November 3 is presented.

A. Application Until the NQE

As mentioned before, agile processes actively integrate the customer and its requirements into the development process. On one hand, DARPA was the so-called external customer in the CarOLO project.

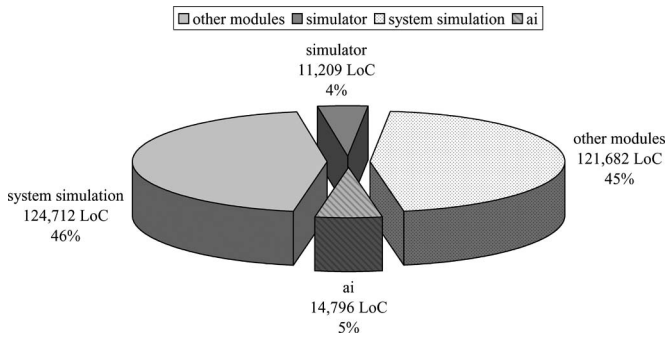

Fig. 6. Lines of the code for the modules "ai," "simulator," "system simulation," and other modules.

Fig. 7. Merging situation in track A.

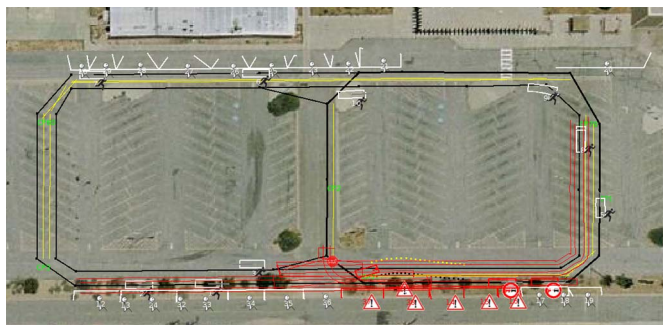

Fig. 8. Merging situation of track A remodeled for simulation.

On the other hand, due to the pipes-and-filters-based data-processing chain of our software architecture, every developer of a module was the customer of the preceding module's produced output data. Thus, we could simply integrate the internal customers due to the overall design of the software and discuss inconsistent aspects of the specification for the challenge directly with DARPA.

Due to the concept of scenarios using our editor, requirements provided by DARPA as shown in Fig. 5 could be easily modeled to be used in our simulation. Furthermore, due to the concept of regression simulation as mentioned before, we could continuously determine the software's performance with respect to the competition's rules that are set by DARPA. Therefore, leaving hardware problems unregarded, it was possible to estimate the software's quality at any time *before* testing it on the real car.

Furthermore, an automatically generated report containing the results of the vehicle's performance in the simulation was provided using the Web-based planning tool Trac [18]. Therefore, the results from regression simulation supported the weekly iterative planning for our

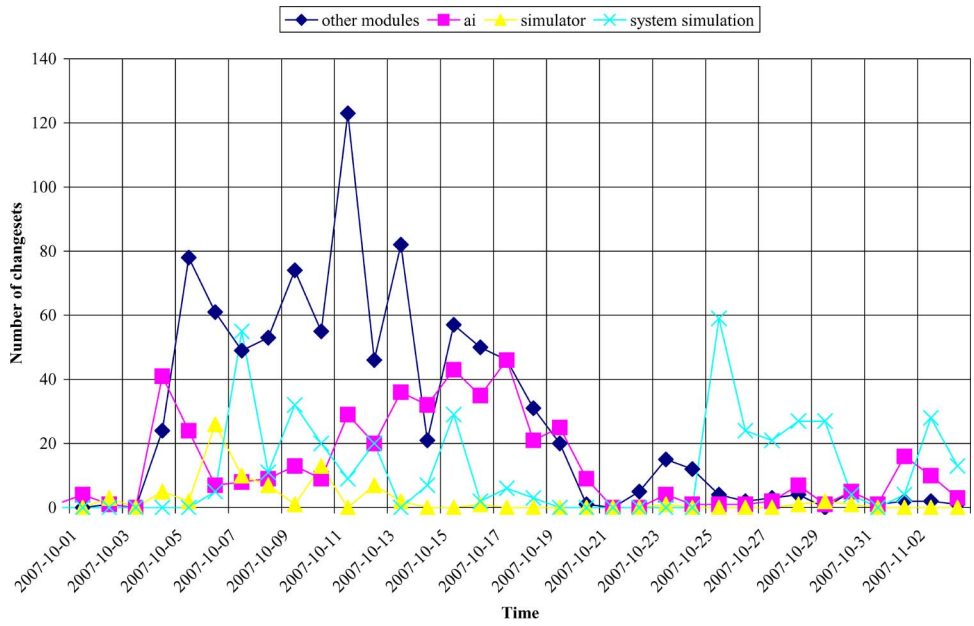

Fig. 9. Change sets over time for the last five weeks in the 2007 DARPA Urban Challenge.

team. The importance of this concept is shown in Fig. 6. Nearly 50% of the code is produced for regression simulation to ensure the software quality of Caroline.

B. Application During the NQE

Since the 2007 DARPA Urban Challenge was a competition, the tasks to be passed by all participants in the semifinal were not published until October 24. Therefore, one of the misinterpreted requirements of which we became aware dealt with time gaps and vehicle separation during merging. The DARPA requirement document stated that if the vehicle merges between two vehicles into a lane of moving traffic, it must maintain at least a two-vehicle safety distance from vehicles at the front and must accelerate such that the vehicle behind is able to maintain its safety distance and a constant speed. Furthermore, one of the fundamental rules demanded safety first, as already mentioned before.

First, we set up an overcautious behavior that yielded long breaks at intersections and produced an unconfident impression from DARPA's point of view. The situation is depicted in Fig. 7, where the intersection with oncoming traffic is shown.

It seemed that this was just a simple adaption to our intersection behavior, but decreasing safety time gaps could have unforeseen side effects to other behaviors as well. For example, vehicle separation during left turns is affected in almost the same manner. Therefore, a large series of testing of all kinds of intersection situations was implied by such a change. To find optimal parameters for our planning algorithms, the situation for track A was simply remodeled using our editor, as shown in Fig. 8. Therefore, it was possible to virtually practice this task several times before being judged by DARPA again. Furthermore, having modeled the other requirements specified by DARPA for the regular development as mentioned before, we were confident that changes that are applied to the software would not break any other functionality.

Fig. 9 contains an analysis of the change sets made to the revision system during the last five weeks of the competition. It is strongly remarkable that during the test phase in San Antonio, TX, that we carried out from October 1 until October 21 right before the NQE took place, on average, approximately 21 changed lines to the artificial intelligence were made per day to fix the remaining bugs and find the optimal parameters. In Victorville, CA, this value decreased to only four lines per day.

However, the change sets that are applied to the system simulation and the simulator got a peak in system simulations right at the beginning of the NQE due to remodeling of the test tracks A and C to interactively and automatically virtually practice our vehicle's software. Track B could not be seen publicly and, therefore, could not be remodeled adequately.

As shown in Fig. 10, several change requests occurred during the NQE either due to identified bugs in the software or due to misunderstandings in the implemented requirements that are set by DARPA. Testing every single change committed to the versioning server to all regular scenarios that we modeled before the NQE, as well as all scenarios that we became aware of during the NQE, we could ensure the software's quality at any time and qualify for the 2007 DARPA Urban Challenge Final Event.

VI. Conclusion

A tight schedule was a characteristic of the 2007 DARPA Urban Challenge. Despite trying to avoid late changes before important milestones, even in the semifinal, some changes to the system became necessary. Following the agile manifesto and the competition as well, these change requests had to be processed to fulfill the customer's needs, where the customer in our case was DARPA. Therefore, even late change requests were handled and processed preserving the system's quality while improving the entire system. The scenario-based regression simulation system was the core tool to enable agile software development and change management as a competitive advantage in the challenge.

The CarOLO project, therefore, has shown that agile software development with additional tooling greatly helped in achieving a high-quality goal with the limited resources that were available. In particular, we have adapted agile development to embedded systems and to partially distributed development using modern software-development techniques such as daily builds, continuous integration, and scenario-based regression simulation.

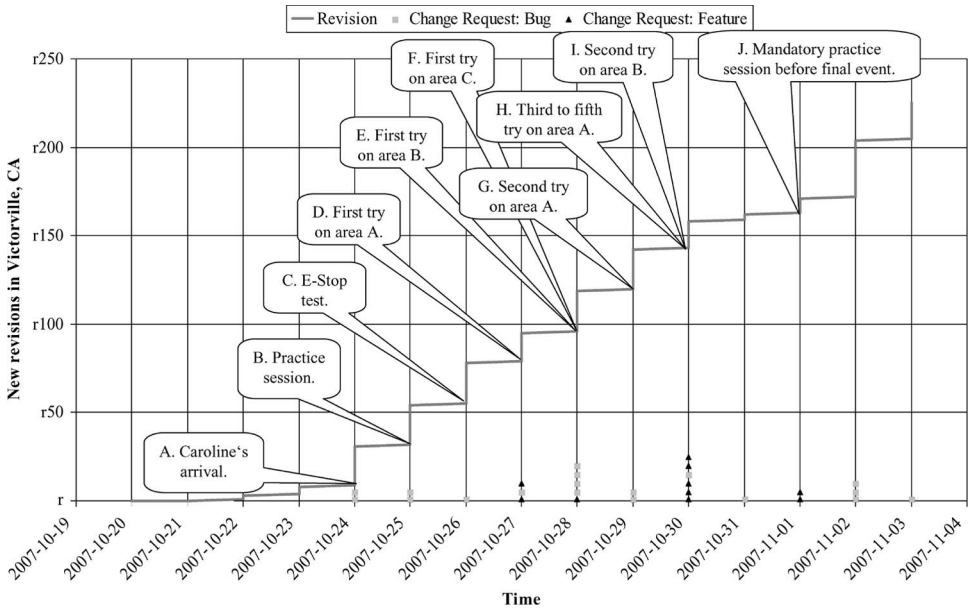

Fig. 10. New revisions and change requests per day during the 2007 DARPA Urban Challenge NQE.

We are confident that agile software development will consolidate and make its way into the development of complex systems with larger embedded software parts and, thus, can be aligned to traditional engineering processes as, for example, practiced in the automotive industry. Moreover, these alignments will be necessary, as software is a competitive factor in intelligent driver-assistance systems, and both quality and timely development are crucial for any original equipment manufacturer.

ACKNOWLEDGMENT

The authors would like to thank their colleagues and students from five institutes of the Technische Universität Braunschweig for the corporate development of Caroline. Furthermore, the CarOLO project would not have been realizable and successful without the extensive support from many people from the university and local industry that had sponsored materials, manpower, and financial support. The authors would also like to explicitly thank Volkswagen AG, Ingenieurgesellschaft Auto und Verkehr GmbH, Niedersächsisches Forschungszentrum für Fahrzeugtechnik, and the Ministry of Science and Culture of Lower Saxony. Furthermore, they would also like to greatly thank Dr. Bartels, Dr. Hoffmann, the President of the Technische Universität Braunschweig Prof. Hesselbach, Mr. Horch, Mr. Lange, Prof. Leohold, Prof. Lienkamp, Mr. Kuser, Mr. Rauskolb, Prof. Seiffert, Mr. Spichalsky, Prof. Varchmin, Prof. Wand, and Mr. Wehner for the various ways in which they assisted us.

REFERENCES

[1] Standish Group, *CHAOS*, 1995. [Online]. Available: http://www.projectsmart.co.uk/docs/chaos-report.pdf

[2] Standish Group, *EXTREME CHAOS*, 2001. [Online]. Available: http://www.smallfootprint.com/Portals/0/Standish%20Group%20-%20Extreme%20Chaos%202001.pdf

[3] K. Beck, *Extreme Programming Explained: Embrace Change*. Reading, MA: Addison-Wesley, 2005.

[4] P. Kruchten, *What is the Rational Unified Process*, 2001.

[5] D. J. Anderson, *Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results*. Englewood Cliffs, NJ: Prentice-Hall, 2003.

[6] E. Volk, *Cxxtest/xunit-Like Framework for C++*, 2004. [Online]. Available: http://cxxtest.sourceforge.net/

[7] M. Beedle and K. Schwaber, *Agile Software Development With Scrum*. Englewood Cliffs, NJ: Prentice-Hall, 2002.

[8] DARPA, *Urban Challenge Rules*, 2007. [Online]. Available: http://www.darpa.mil/grandchallenge/docs/Urban_Challenge_Rules_102707.pdf

[9] DARPA, *Urban Challenge Semifinalist Announcement*, 2007. [Online]. Available: http://www.darpa.mil/grandchallenge/docs/UC_Semifinalist_Location_Release080807.pdf

[10] DARPA, *DARPA Announces Urban Challenge Finalists*, 2007. [Online]. Available: http://www.darpa.mil/grandchallenge/docs/FinalistsUpdated.pdf

[11] F. W. Rauskolb, K. Berger, C. Lipski, M. Magnor, K. Cornelsen, J. Effertz, T. Form, F. Graefe, S. Ohl, W. Schumacher, J.-M. Wille, P. Hecker, T. Nothdurft, M. Doering, K. Homeier, J. Morgenroth, L. Wolf, C. Basarke, C. Berger, T. Gülke, F. Klose, and B. Rumpe, "Caroline: An autonomously driving vehicle for urban environments," *J. Field Robot.*, vol. 25, no. 9, pp. 674–724, Sep. 2008.

[12] E. Dijkstra, *Selected Writings on Computing: A Personal Perspective*. New York: Springer-Verlag, 1982.

[13] C. Basarke, C. Berger, and B. Rumpe, "Software & systems engineering process and tools for the development of autonomous driving intelligence," *J. Aerosp. Comput., Inf., Commun.*, vol. 4, no. 12, pp. 1158–1174, Oct. 2007.

[14] A. Cockburn, *Crystal Clear: A Human-Powered Methodology for Small Teams*. Reading, MA: Addison-Wesley, 2004.

[15] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas, *Manifesto for the Agile Software Development*, 2001. [Online]. Available: http://agilemanifesto.org/principles.html

[16] B. Appleton, S. Konieczka, and S. Berczuk, "*Agile*" *Change Management—From First Principles to Best Practices*, 2003. [Online]. Available: http://www.cmcrossroads.com/articles/agile-cm-environments/

[17] C. Lipski, K. Berger, and M. Magnor, "vIsage—A visualization and debugging framework for distributed system applications," in *Proc. WSCG*, Feb. 2009, pp. 1–7.

[18] Edgewall Software, *Trac*, 2007. [Online]. Available: http://trac.edgewall.org