SALI

Smart self-driving vehicle project – Final report

Edith Zavala[†], Christian Berger[∓], Xavier Franch[†], Jordi Marco[†]

[†] *Software and Service Engineering research group (GESSI), Universitat Politècnica de Catalunya (UPC), Barcelona, Catalonia, Spain*

[∓] *Department of Computer Science and Engineering, Chalmers and the University of Gothenburg, Gothenburg, Sweden*

Executive summary

The SALI project aimed at investigating how challenging runtime factors such as unpredictability, faults and limited resources affect and could be managed in self-driving vehicles (SDVs). In order to achieve that, a series of experiments have been conducted on the AstaZero test track where three main use cases were tested: a road accident, a sensor fault and critical battery level. Experiments were conducted in the city area as well as on the rural road. Three vehicles were utilized during the experiments; one of them running the SALI software solution. In the three use cases, the SALI vehicle had the objective of supporting the driver on his/her daily journey from work (i.e., an origin point) to home (i.e., a destination point). Machine learning techniques were applied to find patterns on driver's behavior (e.g., preferred routes). Different scenarios have been tested for each use case, for instance, with and without traffic. Thanks to the learning-based adaptive behavior enabled by our software solution, the SALI vehicle was able to timely react to the different runtime events, ensuring the self-driving functionality availability, adequacy and resilience. The results demonstrate the importance of correctly managing runtime factors that could affect SDVs, and the validity of our solution for correctly supporting this task.

I. Background

Smart (or intelligent) vehicles are systems capable of sensing contextual data (e.g., from the driver, the environment, the vehicle itself) and making decisions based on this data (e.g., turn on an alarm, activate self-driving functionality). These systems have become increasingly popular in the automotive industry. Smart vehicles have brought several societal benefits, for instance,

improving drivers' safety, experience and comfort (Zavala, Franch, Marco, Knauss, & Damian, 2018). At the same time, their control systems need to face challenging factors such as unpredictability, runtime faults and limited resources. Many researchers have focused on studying the self-driving functionality of modern vehicles in order to support driving challenging situations allowing the drivers to focus in other tasks, from working to relaxing (Berger, 2014). However, less effort has been done on studying how the challenging factors mentioned above affect to, and could be addressed in, SDVs.

In the SALI project, we propose to enable self-adaptation capabilities to SDVs, i.e., make them smart, for correctly managing challenging runtime factors affecting their performance. Self-adaptation capabilities such as self-healing or self-optimization could enable SDVs to recover from faults or lengthen the self-driving functionality operation when limited resources are available. In the self-adaptive systems field, these capabilities are typically implemented through feedback loops. Several models have been proposed in the literature for implementing such feedback loops. Among them, the most prominent is the MAPE-K reference model (IBM-Corporation, 2006; Kephart & Chess, 2003). In MAPE-K, self-adaptation is achieved applying four steps: Monitor, Analyze, Plan, and Execute. SALI adopts the MAPE-K loop for providing the self-adaptation capabilities to SDVs.

In short, the MAPE-K loop works as follows. The monitoring step collects data from the system to adapt (also known as target system), the environment and the users through sensors or other types of monitoring data sources such as cloud services. The collected data is then analyzed for understanding the current context. The analysis results are then passed to the Plan step for determining whether an adaptation is required or convenient. If an adaptation is planned, the Execute step is triggered and the adaptation is enacted on the target system. In order to operate properly, these four steps share a Knowledge base which stores design and runtime data such as metrics, logs, adaptation policies, etc. In the SALI project, adaptations consist of: activating alternative sources of monitoring data (*monitors* for short), when one fails or provides inaccurate data; performing a trade-off between available monitors, when the SDV is running out of battery; and, changing monitors' frequency, when data freshness is compromised. Our solution utilizes machine learning techniques which have been incorporated in the SALI's MAPE-K feedback loop for analyzing and determining whether an adaptation should be performed or not based on patterns learnt.

The SALI project proposal has been developed as an open software engineering solution consisting of a set of microservices available at the GitHub platform[1]. The solution can be reused

[1] https://github.com/edithzavala/opendlv
https://github.com/chalmers-revere/opendlv
https://github.com/edithzavala/OpenDaVINCI
https://github.com/edithzavala/loopa
https://github.com/edithzavala/ksam-loopa
https://github.com/edithzavala/ksam-opendlv
https://github.com/edithzavala/wekaservice
https://github.com/edithzavala/cityreporter

in other contexts as well as be extended for adding more intelligence to already smart vehicles. In the rest of this document, we describe how the SALI solution has been tested at AstaZero (Section II), what are the preliminary results (Section III) and the conclusions of our work (Section IV) as well as which are the lessons learned and how was our experience at the test track (Section V). Planned publications and disseminations activities are included at the end of the document (Section VI) as well as more details about the project partners (Section VII).

II. Purpose, research questions and method

The purpose of the SALI project was to investigating how challenging runtime factors such as unpredictability, faults and limited resources affect to and could be managed in SDVs. For conducting our work we have proposed the following research questions:

RQ1. How do runtime factors such as unpredictability, faults and limited resources affect the performance of current self-driving vehicles?
RQ2. How can the impact of those runtime factors be managed in current self-driving vehicles?
RQ3. Is the SALI project solution valid for supporting current self-driving vehicles affected by those runtime factors?
RQ4. What are the benefits and drawbacks of using the SALI project solution for supporting current self-driving vehicles affected by runtime factors?

In order to address the research questions we have conducted a series of experiments on the AstaZero test track. Three main use cases were tested: a road accident, a sensor fault and critical battery level. Experiments were conducted in the city area as well as on the rural. Three Volvo vehicles were utilized during the experiments: two XC90 (Snowfox and Greyfox hereafter) and one V40. The Snowfox XC90 was selected for testing the SALI project solution. This vehicle is equipped with a LIDAR sensor, a GPS and a camera. Moreover, a cloud service gathering traffic data and another one gathering weather data were incorporated. Before running the actual experiments, the Snowfox was trained for the sensor fault and the critical battery level use cases which were tested on the rural road. A test driver and a software component have simulated the self-driving functionality. The driving scenario exemplifies a driver that goes from work to home in a daily basis and utilizes the self-driving functionality in specific segments of that journey. From this training phase, patterns on self-driving functionality usage and typical route were inferred using different data mining algorithms. The resulting learnt behaviors are illustrated in Fig. 1.

Figure 1. SALI learning phase patterns

After the learning phase, the following use cases were tested:

- **UC1. Road accident.** The Snowfox and the V40 go on an urban road at an average speed of 20 km/h. In the AstaZero city area, the Snowfox departed from the point marked as *Work* and drove to the point marked as *Home* (see Fig. 2). Suddenly, the V40 witnesses a road accident and uses V2V communication for alerting the vehicles around (see Fig. 2). V40 is not able to timely recalculate its route and stops. On the other hand, the Snowfox successfully recalculates its route and reaches its destination. In order to recalculate the route, the Snowfox relies on traffic data gathered through a cloud service. Each execution of this use cases lasted around 1 minute. For this use case, we have designed two scenarios:

  - **Scenario 1. Parameter adaptation.** In order to correctly recalculate the route, the SALI vehicle adapts the traffic monitoring service frequency in order to gather the freshest data since the road accident just happened. As a result, the Snowfox changes its route.

  - **Scenario 2. Structure adaptation.** Since the road accident just happened, the SALI vehicle receives inaccurate traffic data from the cloud service. In order to improve the quality of the traffic data, the SALI vehicle substitutes the traffic monitoring service by V2V communication (i.e., it actively asks other vehicles around about the traffic conditions) (adaptation *a*). As a result, the Snowfox changes its route. In the new route, V2V communication is not useful anymore;

thus, the SALI vehicle adapts its monitors again, deactivating V2V communication and re-activating the traffic monitoring service (adaptation *b*).

Figure 2. Road accident use case

– **UC2. Sensor fault.** The Snowfox goes on a rural road at an average speed of 40 km/h. As in the training phase, the Snowfox departs from the point marked as *Work* and intends to arrive to the point marked as *Home* (see Fig. 3). Suddenly, one of its sensors fails (see Fig. 3). The Snowfox analyses whether an adaptation is required or not, and in case of yes, adapts itself activating alternative monitors (i.e., sensors, services or V2V communication) for gathering same or similar data. Each execution of this use case lasted around 4,5 minutes. For this use case, we have designed three scenarios:

  – **Scenario 1. Structure adaptation on a road with no other vehicles.** In this scenario alternative sensors are activated.

  – **Scenario 2. Structure adaptation on a road with other vehicles.** In this scenario, the Greyfox XC90 is placed in front of the Snowfox doing a similar journey. Given the increased driving risk, apart from the sensors activated in *Scenario 1*, V2V communication is activated and Greyfox position data is gathered by the Snowfox.

  – **Scenario 3. No adaptation on a road with no other vehicles.** This scenario is similar to *Scenario 1*. The main difference is the time at which the sensor fault is experienced. In this case, based on the patterns learnt, the self-driving functionality will not be used by the driver in the rest of the journey. As a consequence, the SALI vehicle determines that an adaptation is not worth it.

Figure 3. Sensor fault use case

- **UC3. Critical battery level.** Very similar to previous use case, in this use case, the Snowfox goes on a rural road at an average speed of 40 km/h. The Snowfox departs from the point marked as *Work* and intends to arrive to the point marked as *Home* (see Fig. 4). At some point, the Snowfox detects critical battery level (see Fig. 4). We have indicated a specific threshold for this purpose. The Snowfox analyses then whether an adaptation is required (and possible) or not. In case of yes, it performs a trade-off between available monitors and deactivates unnecessary ones, ensuring that the self-driving functionality can be supported till it is required. Each execution of this use case lasted around 4,5 minutes. For this use case we have designed three scenarios:

  - **Scenario 1. Structure adaptation on a road with no other vehicles.** In this scenario unnecessary sensors are deactivated.

  - **Scenario 2. No adaptation on a road with other vehicles.** In this scenario, the Greyfox vehicle is placed in front of the Snowfox doing a similar journey. Given the increased driving risk the SALI vehicle is not able to deactivate any monitor.

No adaptation is performed and a take-over request is sent to the driver (in this case, after the notification, we have manually stopped the module that was simulating the self-driving behavior).

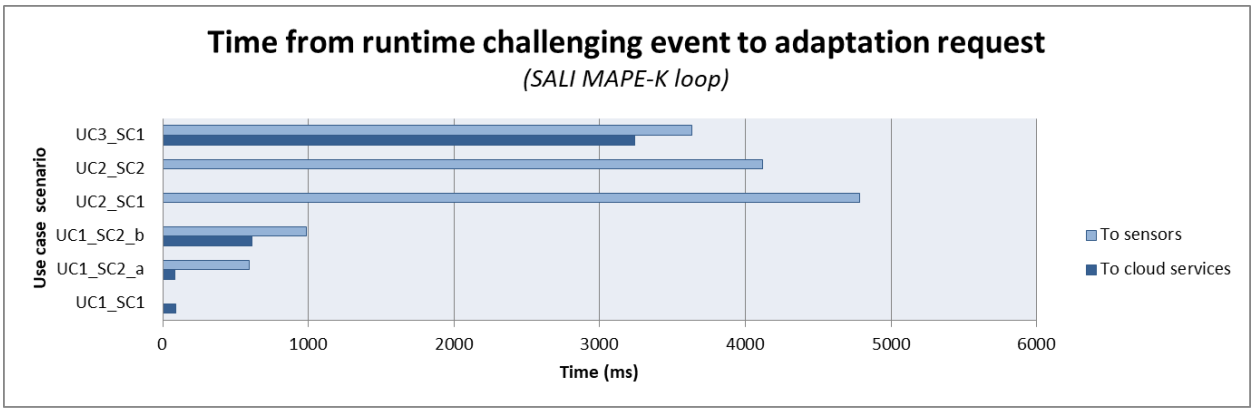
– **Scenario 3. No adaptation on a road with no other vehicles.** This scenario is similar to *Scenario 1*. The main difference, as in Scenario 3 of UC2, is the time at which the critical battery level is experienced. In this case, based on the patterns learnt, the self-driving functionality will be disabled by the driver shortly. As a consequence, the SALI vehicle determines that an adaptation is not worth it.

Figure 4. Critical battery level use case

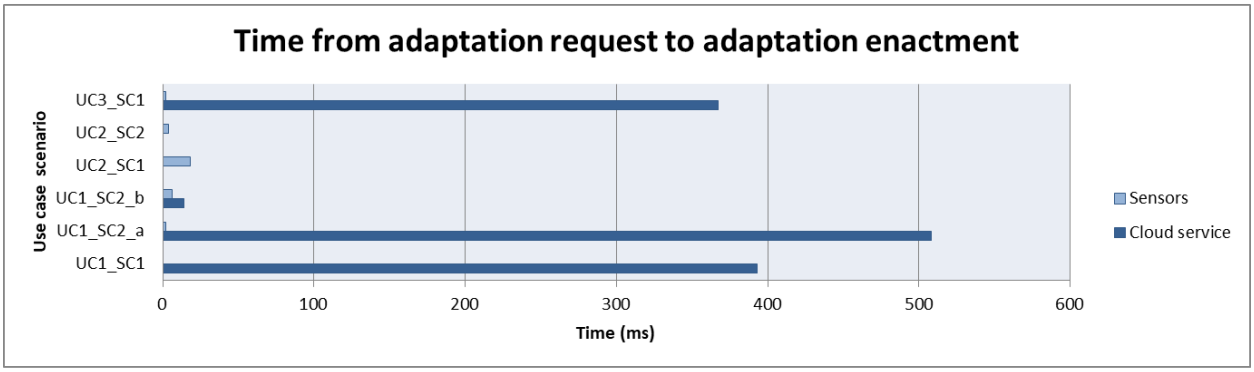
III. Results

Each use case scenario that we have described in previous section has been executed several times during our testing days at AstaZero (6 days in total, 1,5 at the city area, 3 at the rural road and 1,5 at the garage area). In this report, we provide preliminary results of those experiments. That is, we will present the results of one execution of each use case scenario. The complete

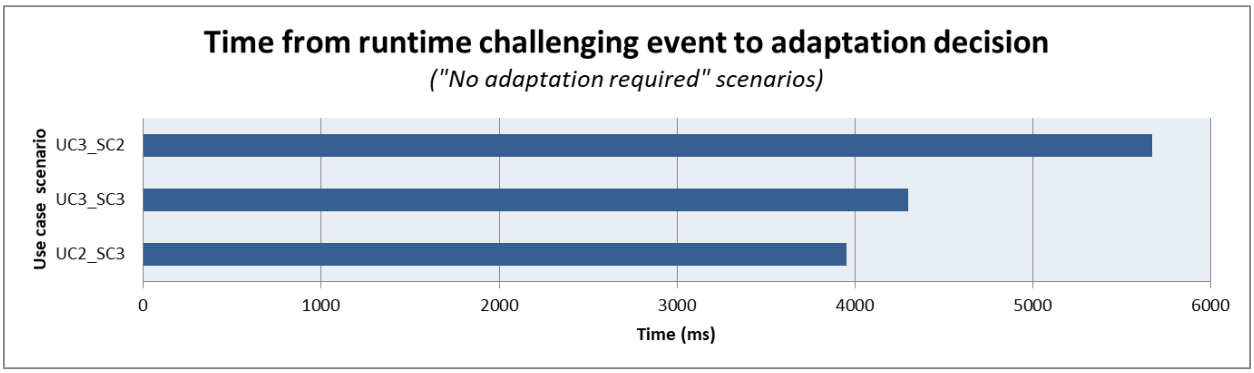
analysis of the results will be provided in the works listed in Section VI. All scenarios have been successfully executed, in terms of both functionality and timeliness. That is, when applied, required adaptations have been enacted on the SALI vehicle. Moreover, from the driver's point of view (person in charge of the experiment in this case), adaptations were executed at the right time, e.g., in UC1 the adaptation of the traffic data sources and the route recalculation was done before the SALI vehicle arrived to the intersection where route changes. Graphs shown in Fig. 5 summarize the results of the scenarios' execution in terms of response time.



(a)



(b)



(c)

Figure 5. SALI project preliminary adaptation results

Graph (a) of Fig. 5 shows the time spent by the SALI solution feedback loop for identifying and analyzing the runtime challenging event (i.e., road accident, sensor fault or critical battery level), determining vehicle's position in the near future and self-driving functionality usage in that position (for use cases UC2 and UC3, based on patterns), deciding the required adaptation and sending the adaptation request to the corresponding SALI vehicle components (i.e., sensors, could services monitors or both). When adaptations involved different components, requests were sequentially sent, first cloud services, second sensors. As it can be seen, for UC1 the feedback loop's response time is much smaller than for UC2 and UC3. This is due to the analysis complexity in the last two use cases. Predictions took around 3 seconds to be done. Optimizations in this step may reduce the whole feedback loop response time, e.g., decreasing the number of position points to be predicted. Even though, current response time worked well for the scenarios we have proposed.

Graph (b) in Fig. 5 shows the elapsed time since the adaptation request is sent from the SALI feedback loop to the vehicle components till the corresponding changes are actually executed on the components. In the case of sensors, the de/activation has been simulated by software components; thus, the enactment response time in real cases may differ from what it is presented in this report. On the other hand, the cloud services' response time does actually reflect real services changes. In this regard, it is worth to mention that we have noticed that most of the cloud services' enactment time was spent in communication. Particularly, for the first message exchanged between the feedback loop and the services' interface (we use a single entry point for both cloud services' adaptation). This can be clearly seen comparing first and second adaptations of UC1_SC2 (*a* and *b*), corresponding to traffic service deactivation and activation, respectively. However, we have to further look into the rest of experiments for providing a comprehensive conclusion. Finally, graph (c) in Fig. 5 shows the time spent by the SALI feedback loop for determining that an adaptation is not required. Similarly to the insights obtained from graph (a), in this case most of the time corresponds to the prediction task. Even though, the decision of do not enacting an adaptation was accurate for all the scenarios.

IV. Conclusions and outlook

In conclusion, the preliminary answers to our research questions are:

RQ1. How do runtime factors such as unpredictability, faults and limited resources affect the performance of current self-driving vehicles?

If they are not managed correctly, runtime factors such as the ones listed in RQ1 could critically affect SDVs at runtime. For instance, the self-driving functionality could become completely unavailable if runtime data is incomplete (e.g., due to a sensor fault). From the driver's perspective this situation does not only affect his/her safety but also his/her comfort and trust on SDVs.

RQ2. How can the impact of those runtime factors be managed in current self-driving vehicles?

In this work, we have briefly presented the SALI project solution. Our solution relies on runtime adaptation and machine learning techniques for reducing the impact of challenging runtime factors in self-driving vehicles.

RQ3. Is the SALI project solution valid for supporting current self-driving vehicles affected by those runtime factors?

From the preliminary results analyzed in this work, and our experience at AstaZero, we can say that the SALI project solution is a promising proposal for supporting SDVs in the presence of challenging runtime events such as road accidents, sensor faults and critical battery levels. However, more iterations of improvement and experimentation would be required for testing our solution in a greater variety of scenarios in which modern SDVs may be involved.

RQ4. What are the benefits and drawbacks of using the SALI project solution for supporting current self-driving vehicles affected by runtime factors?

Among the benefits of using our solution are: the ability of SDVs to be adapted at runtime and deal with challenging factors that otherwise would be not possible; and, the ease of adopting, reusing and extending the SALI software solution for supporting other types of adaptations and addressing other runtime factors that may affect SDVs. Among the drawbacks we can mention: the extra computation load that the SALI components may cause and the incompleteness of the solution since it has been developed specifically for addressing the runtime challenging factors described in our use cases.

V. Lessons learned, experience from testing at AstaZero

In the past, we have experimented with runtime adaptation and machine learning techniques in simulated self-driving vehicles (Zavala, Franch, Marco, Knauss, & Damian, 2015; Zavala et al., 2018). Although results were promising, we could not ensure that the approach may produce a similar behavior in a real vehicle. The opportunity of experimenting at AstaZero has allowed us to improve and demonstrate the validity of our approach. Moreover, it has allowed us to gather other kind of metrics, for instance, the adaptation timeliness from the driver's point of view. The support offered by the AstaZero staff was effective and efficient. We were allowed to test in different environments, use a variety of resources such as an extra vehicle and a soft car, and record our experiments in different formats, which is very useful for the future project's dissemination activities.

VI. Publication and dissemination

Zavala, E., Franch, X., Marco, J., Berger, C. HAFLoop: An Architecture for Supporting Adaptive Feedback Loops in Self-Adaptive Systems, *to be submitted*.

Zavala, E., Franch, X., Marco, J., Berger, C. Adaptive Monitoring for Self-Adaptive Systems, *to be submitted*.

Zavala, E., Franch, X., Marco, J., Berger, C. The smart self-driving vehicle: from autonomous to intelligent, *seminar*.

VII. Participating partners and contact persons

- Software and Service Engineering research group (GESSI), Universitat Politècnica de Catalunya (UPC), Barcelona, Catalonia, Spain.

  Edith Zavala (zavala@essi.up.edu), Xavier Franch (franch@essi.upc.edu), Jordi Marco (jmarco@cs.upc.edu)

- Department of Computer Science and Engineering, Chalmers and the University of Gothenburg, Gothenburg, Sweden.

  Christian Berger (christian.berger@gu.se)

- REVERE - Resource for Vehicle Research at Chalmers

  Fredrik von Corswant (fredrik.von.corswant@chalmers.se), Christian Berger (christian.berger@gu.se)

Acknowledgements

Thanks to CONACYT, for the PhD scholarship granted to Edith Zavala. Also, thanks to the vehicle laboratory REVERE for the resources and technical support provided during this project. This work has been partially supported by the AstaZero openresearch@astazero program (call 4 - 20180430) and the Spanish project GENESIS (TIN2016-79269-R).

References

Berger, C. (2014). From a Competition for Self-Driving Miniature Cars to a Standardized Experimental Platform: Concept, Models, Architecture, and Evaluation. *Journal of Software Engineering for Robotics*, *5*(1), 63–79.

IBM-Corporation. (2006). An architectural blueprint for autonomic computing. *IBM White Paper*, *36*(June), 34. https://doi.org/10.1021/am900608j

Kephart, J. O. J. O., & Chess, D. M. D. M. (2003). The vision of autonomic computing. *Computer*, *36*(1), 41–50. https://doi.org/10.1109/MC.2003.1160055

Zavala, E., Franch, X., Marco, J., Knauss, A., & Damian, D. (2015). SACRE: A tool for dealing with uncertainty in contextual requirements at runtime. In *23rd IEEE International Requirements Engineering Conference (RE)* (pp. 278–279). IEEE. https://doi.org/10.1109/RE.2015.7320437

Zavala, E., Franch, X., Marco, J., Knauss, A., & Damian, D. (2018). SACRE: Supporting contextual requirements' adaptation in modern self-adaptive systems in the presence of uncertainty at runtime. *Expert Systems with Applications*, *98*, 166–188. https://doi.org/10.1016/j.eswa.2018.01.009