

Software Engineering Aspects of Autonomous Vehicular Design

Abu Ashraf, John Fisher, Charles Hoffmeyer, and Jack Matheson
{amashraf,jnfisher,cchoffme,jkmathes} @ mtu.edu

Department of Computer Science
Michigan Technological University
Houghton, Michigan 49931

Abstract—Drive-by-Wire is the relatively new idea of automating the process of driving through the use of electronic communication and sensor detection. The creation of a strong decision-making system will promote safety as well as reduce the dependency our nation has on human based transportation. Other benefits include a significant reduction in manufacturing costs for producers, and fuel savings for consumers. This paper analyzes many aspects of a drive-by-wire system and provides a means for risk assessment to determine the feasibility of drive-by-wire in the near future. While revolutionary in theory, our immediate conclusion is that such a system is not tenable due to high safety concerns.

I. INTRODUCTION

The idea of a system of transportation that removes the most dangerous factor, human beings, has been around for quite some time. Although practical solution for a drive-by-wire system is still years off, a great amount of research has been dedicated to the problem. In any drive-by-wire system software is of great importance; if the software is flawed lives will be put at risk. Because of this inherent risk, it is important that the software involved be developed and tested extremely carefully- by applying fundamental concepts of software engineering. In this paper we will cover some of the key steps and considerations of the software design process that would be involved in a drive-by-wire system: requirements, design, testing, measurement, validation, risk analysis, reliability, maintenance and code reuse.

II. PROBLEM ANALYSIS

A. Environment

People in system: This project will need huge number of developers. The entire project group must consist of developers for coding, hardware specialists, knowledgeable mechanical engineers, electrical engineers, and civil

engineers who will work together for the success of the project. Interested entrepreneurs are needed to provide for the large required investment.

People affected: This new development will welcome everybody in the real world to be customer of the product. People from all over the world will be prospective users. Some developers will be necessary to maintain the software and implement new technology over time.

Machine in system: To actuate the project, it is required to have large server space. There should be a main database to store data for traffic, and road maps. The system will have high-speed computational capacity to generate output faster and react with vehicle sensory unit as needed.

Machine affected: Special controls in the highway system are needed. Every highway system where the automated vehicle will run must have magnetic sensor in each traffic lane to detect clear interaction with electronic steering wheel. For data communications, the highway system must have transmitter, which send and receive messages through electronic magnetic signal to keep track of all distributed system and share information with vehicles. This is expected that the highway system architecture may need revision. Traffic control system should be automated so that it can communicate with a central system to inform about traffic jams or signals and to help the vehicle control unit to take action.

Other items affected by software operations: Outside environment will be affected due to the drive-by-wire system. The highway system will have some change in operation, as interaction in driving will not be required. Older model vehicles will have less use and could be disposed of out of necessity. There should be an alternative way to undertake this kind of environmental issue.

B. Item produced

Items processed: The drive-by-wire system will process software with sufficient attributes that are needed for software operation. For example each vehicle will have a hardware, which will communicate with the central database that provides information for the vehicle. The vehicle itself will have sensory detectors that detect objects, control speed, brake, and transmission, and share data to the software for input. The output from the software actuates the sensor to react.

Items consumed: During software development process item consumed are person time, knowledge, developing cost, experimental cost and managing hardware/software attributes. For users the items consume to run the software will be electric power and necessary information regarding road and external environment. The advance technology can reduce the power consumption through solar power or from other resources. Information will be available from up-to-date database, sharing from other vehicle sources and or direct contact to the real life.

C. Functions

Functions performed by people: Drive-by-wire system make easy for people to drive vehicle without actually being present in the vehicle. For example user can choose a destination, as an input data then the system will take the vehicle to the right place in reasonable time. To actuate the process the input should be valid and the vehicle should have enough power.

Functions performed by machine: Drive-by-wire system software will control the hardware installed in the vehicle. It will use the database to collect information, find right direction to reach the destination and response to the traffic system to avoid accident. The time it will take depends on the traffic jam, speed limit of the road, and signals. Sometime the system could generate different direction to avoid frequent signals.

Functions needed to produce: Both hardware and software development are needed for this huge system. Special high tech mechanism in the vehicle is needed to react over the sensory detection and actuate the result. Software need to be very reliable and risk free because the system deals with human life, any malfunction might cause loss of life.

III. REQUIREMENTS

A. Functional Requirements

Drive-by-wire system should do the following:

- Each vehicle for this project will be electronically controlled.
- There should be vehicle autonomous driving system hardware in each vehicle, which control the vehicle and response to the human input.
- The vehicle will have automatic braking system
- Automatic route planning option in the system
- Automatic steering control and traffic lane control
- Automatic speed and transmission control
- The traffic control system should be compatible with drive-by-wire system so that the system gets informed early about the signal lights, traffic jam or any road construction.
- Vehicle should have intelligent sensor to detect obstacles, traffic lights and other objects during vehicle movements and react on it.
- The system would be able to communicate with main system or other vehicle in the system through wireless network and share necessary information.
- Redundant hardware to handle permanent faults.

Optional requirements:

- Message generating tool to inform the user about accident, inability to reach the destination or any other type of malfunction if the passenger is not present in the vehicle
- Memory control system to record past destinations and also expected rides in future.

B. Non-functional Requirements

Time: Drive-by-wire is a huge project and it will take plenty of time and effort to come up with an actual solution. Calculating time to complete the project is not easy but dependable time estimation should be available once the entire project plan is ready. The work will be split into different department of the development group and each group will have their own subprojects. The actual time will depending on LOC, number of people working on the project as well as their skills, and amount of testing required.

Cost: Development cost is very high. It is hard to determine actual cost before completing the entire project because the system is dependent on various environmental and functional issues. Cost estimated depends on human labor, hardware infrastructure, and costs of COTS software. It is also related to time needed. As an assumption most of the cost will go for remodelling highway system and developing the main operation control unit. The operation control unit needs multiple hardware related devices, which perform for each section of control unit.

It is also costly to make automated electrically controlled vehicle with special sensory mechanism. Distributed sensors and receivers may need to be placed all over the highway system to have easy access to the system. Finally the maintenance cost that includes recovering all kind of difficulty, failure as well as updating the traffic information that is needed forever. Users need to be trained or at least provide enough information with practical training to demonstrate how to operate, which will incur extra costs.

Security: The data transportation and user access should be highly secure so that nobody can control the vehicle other than the user. It is an important requirement, as to maintain customer expected standards from traditional vehicle models.

Portability: The system will communicate through wireless network; therefore it should be portable for every gateway system and suitable for all vehicle manufacturers.

Efficiency & Reliability: Although the project is expensive it should be reliable and efficient to use. Reliability and efficiency depends on the wide use of the system. Maintenance is also a big issue here because the more people use the software the more error or limitation comes out in focus, therefore continuous maintenance will make it more efficient and reliable.

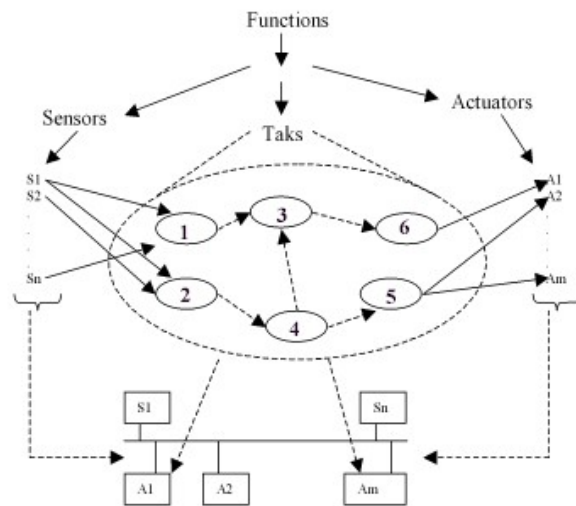
IV. DESIGN

The automated vehicle will have dual control, both autonomous and manual. The user will have opportunity to choose one of the two options at any time, either manual or automatic. The hardware redundancy is for increasing safety issue [1]. To minimize the risk it is necessary to have manual hardware control as an option, which will deal with any kind of critical software error.

The design process strictly follows the requirements of drive-by-wire system. Therefore, design might have some correction later on after requirement modification over time. The design process can be easily describe using data flow architecture where each process unit can be shown in a node and dividing farther into small units. The nodes will have input from outside environment then process the data, send the output, which would be input for another node. Finally the electronic mechanical system will receive the input to actuate the result. The operation control unit is subdivided into sensor, actuator and task [1].

In functional model, the task unit will control the over all process and control the sensor and actuator unit. The

sensor unit will detect and generate input. The input will go over other nodes to filter and process for actuator. The actuator takes action for specific situation. The data move through different nodes to actuate the result running through individual process unit and communicate with electric mechanical system to the right sensory unit of the vehicle. The entire system is controlled by electronic power system and communicates through bus system. Therefore, integrity methods, such as reliability, fault tree, hazard analysis, and risk classification for the design safeties are needed. [3].



[1]

Operation control unit: Operation control unit is the main station, which control and make available all information about drive-by-wire system. The control unit will be able to generate route planning, make available all information the system need.

Vehicle autonomous driving system: Vehicles of drive-by-wire system consists of some special electronic mechanism with electronic output, microcomputer to control the system, bus system to communicate through all sensory units, and electronic actuators [3]. The software will generate command to drive vehicles with the help of system receiver attached to the vehicle, control the communication links to each part of the vehicle such as brake pedal, steering wheel control and others through the receiver and sensory system [5].

Sensory process unit: The sensory process unit detects distance, obstacle, follow traffic lane, and other issues then filter the message and correlate sensory information and send to the task management. There are different types of partial sensory unit operate for the sensory process unit to work as a whole to keep track of running vehicle. Navigation sensor measures the distance and

collects location information from operation control unit [5]. Microwave sensor detects short distance obstacle and act faster with electronic actuator.

Actuator Module: This module estimates the state of external world gathering information from central control unit and make prediction from the sensory message. After evaluating based on estimation, the output is sent to the electronic actuator to promote the action. Actuator unit controls automated steering wheel, electronic brake pedal, throttle, transmission system, transfer case, parking brake using sensor detection, laser detection and data communication link [5].

Task Module: Task module will implement real-time planning for sensory message and control the electronic actuator. For safety check the task module will monitor all functions and perform error handling and fault treatment [1].

V. TESTING AND MEASUREMENT

It has been suggested that the safety requirement for such a system needs to be 100 times safer than conventional driving [2]. Due to the inherent need for safety in any drive-by-wire technology, extensive testing of the software systems involved is essential. In addition to testing, measurement of the software is also needed at each design phase. A safe, workable system will not be achieved without these two important concepts.

A primary source of software bugs, and the largest pitfall in a drive-by-wire system, is the operating environment. We can not replicate the users combination of hardware, peripherals, operating system, and applications in all of our tests [7]. Therefore, modelling the softwares environment should be our prime concern in the early stages of testing. In addition, a side-effect of the complexity of the system is that there are an infinite number of input domains (or test cases), so it is necessary to isolate these inputs into separate subdomains [6]. Subdomains in a drive-by-wire system can be broken up, for example, as follows: collision detection, lane alignment and position, current state relative to other vehicles, road conditions, etc. Dividing inputs allows us to implement unit testing very well, but it does little to show how the different parts of the system work together as a whole.

In the drive-by-wire system inputs should be seen as test scenarios, or events that encompass a number of physical input sequences [7], for example, a vehicle in front of us applying its breaks at a given speed. Test scenarios should be selected based on real world studies of normal traffic behavior as well as problematic

instances such as traffic jams, bad weather conditions, and accidents.

Because the environment in which a drive-by-wire system will exist is so complex, the verification system used with the software must be powerful and correct. In their paper, Fault-Tolerant and Secure Intelligent Vehicle Highway System Software - a Safety Prototype, McMillin, Sirois, Mahoney, and Budd outline a distributed system for drive-by-wire in which they used CCSP (a CSP-like syntax on top of C), called the Automated Highway System (AHS). In AHS vehicles exchange messages and program states. Assertions embedded within the code are evaluated upon execution depending on their truth values [4]. AHS is able to simulate different scenarios, or events, taken as input and allows for changes in the way the system determines its response to the events, such as the ability to enable or disable global state variables. The test system allows them to run simulations on a group of computer workstations and return results such as reaction time, resulting consequences (collisions, etc), and relevant states involved in the simulation. AHS is an excellent example of the type of extensive testing needed before any actual real-world test systems are built, it also gives the developers a way of manipulating the test environment that is fast and flexible, two attributes difficult to achieve on a test track.

After initial testing on simulators (computer workstations, scale-model tracks, etc) a large amount of testing is required in limited and large-scale real world tests. As changes are made during real world trials, regression testing will be required. Again, because of the time consuming and expensive nature of real world test scenarios, a system like AHS will be well suited to this application.

Measurability of any drive-by-wire system is not an easy task. Since the system is so complex, simple methods such as LOC and Function Points will not adequately give the designers a good measure of the software. A better system of measuring would be using an approach based on feature points (the number of algorithms involved). Since a drive-by-wire system is basically a grouping of algorithms that handle specific events based on input from sensors or surrounding vehicles in an ad-hoc network, determining complexity through feature points makes sense. Unfortunately, determining what is or is not an algorithm is somewhat arbitrary. Also, a good measurement system would also include a metric for determining coupling between methods (algorithms). Drive-by-wire only works if the many different classes

work well together (for instance, the methods associated with road condition checking, the collision system, and the automatic braking system), therefore it is imperative that as measurement deals with the hierarchy of classes and methods as a whole.

The key idea in measurability of a drive-by-wire system is to keep the method of measuring the software consistent from iteration to iteration. That way a developer or manager is able make judgments based on relative differences. Ultimately, testing any software involved in a drive-by-wire system is a long and arduous journey, and one that is vital to the success of any project. Measurement is required along with testing at each step in development to assess progress. A successful, working system will only be achieved after these two key concepts have been integrated.

VI. RISK ANALYSIS

Significant risk is involved when human lives are at the hands of machines, and their human software engineers. While humans are understood when fallible at driving, any mechanical substitute must be extremely precise and provide higher levels of safety at all times. Before any design or development is performed, detailed risk analysis must be done. This analysis is important to perform, as it provides validity to the project and airs downfalls to be considered in the initial design. For any company investigating autonomous driving vehicles, sources of risk include the cost of the project (research, design and development), safety (to the driver, others, and the vehicle), legal aspects and technical error (design, debugging, development, and fault errors).

Before work on any project begins, the leader should determine what the cost of the entire product will be, as well as how many units are anticipated to be sold and at what price. If a project will not make money for the corporation, or worse lose money, then the project should cease. The cost to the user for implementation of the final project must be considered as well. For the autonomous vehicle, great costs will be incurred in research and development for the simple reason that it has not been done before. In order to determine the most effective method of design, many failures will occur. Once a company completes the development of such vehicle, the marginal cost will be significantly lower to others through reverse engineering methodologies. Using the IEEE 1044.1-1995 Risk Classification Scheme, this could be classified as a medium risk. After the design has been fixed in place, production costs should be similar in comparison to standard vehicles. Certain care

must be taken not to require changes to the existing infrastructure and systems. For example, if a city must place electric tracks for the vehicles to run properly (and safely) there would be little opportunity for the projects success since costs will outweigh benefits. For no infrastructure changes, there will be zero risk.

Safety is the key to the deployment of drive-by-wire, but involves modest risk. This risk stems from the unpredictability of driver actions. In practice, Volkswagen has set a firm guideline for the safety of their autonomous projects: "Safety before Availability" which they define as "A vehicle, which is able to drive without human intervention (autonomous) by use of electric equipment, shall not entail human beings and/or property which is greater than the hazard represented by the conventional (human) driver!" [2]. Determining the level of necessary safety can be a daunting task. Should consumers be satisfied by systems that perform as well as humans, or will perfection be demanded. The United States Department of Transportation indicates that "more than 90% of crashes are the result of human error" [8]. By eliminating the driver, consumers are likely to expect 90% less accidents instead of merely meeting the human accident level. As such, safety is a high risk issue.

Legal risk is very important for automated driving systems. Traditionally, the driver is held responsible when accidents occur. In autonomous driving, there is no driver. The company may become legally responsible for all accidents. This is an unacceptable risk. The possibility of hardware and software failure is too great to assume responsibility for. Furthermore, the cost of recalling vehicles for deadly software programming could be tremendous.

Software errors are impossible to avoid or guarantee against for large, complex systems. Errors made during the design and development processes can be costly and/or injurious. Through software engineering and well-planned organization, many technical errors can be avoided. In [2], Volkswagen suggests two methods to avoiding software errors after extensive testing: functional testing and having a "electronic co-pilot". Traditionally, component redundancy has been used to protect safety in critical systems. Due to the high number of critical systems required, this could lead to be tedious, costly, and also dramatically add weight to the vehicle. Instead, they recommend creating redundant software in isolation. If one module fails due to software programming, the related module with different coding can take over. While it logically follows that software developed in isolation would produce stable results, in practice this

has proven not to be the case as the isolated teams have the same preconceptions in regards to the project (resulting in the same faults). The co-pilot alternative is more practical. An "electronic co-pilot" takes over if both software solutions fail and the system can no longer successfully self recover. The co-pilot is hardcoded to avoid accidents, and gracefully pull over to the side of the road to shutdown. Another solution is proposed [4] as an ad-hoc communication network for nearby vehicles. These equipped vehicles will communicate at short range to determine what the other is going to do next, removing uncertainty in future vehicle state. This network is also efficient in detecting software faults, by comparing communicated traffic with sensor/software detection results. In combination, these error tolerant systems dramatically lower risk of dangerous effects due to software error.

VII. RELIABILITY

Software reliability for drive-by-wire systems is incredibly difficult, because the fault conditions are not always known. This is due to uncertainty. In conventional software execution, reliability can be easily determined through path testing. A input value is given and results in an expected output. Those which have deviating output result in a fault condition. This can only work when outcome is certain.

In driving, we only have predictability. It is predictable that if a Michigan Tech student is driving in front of a road-raged professor, the student will switch lanes to avoid being rear-ended. It is also possible that in the same situation, the student switches lanes and the cruel professor (so full of rage) sideswipes the student off the road. We now have two situations which follow the same path, and have the same goal (to stay on the road and not be hit), yet one results in an unexpected fault.

While we cannot use scenario based testing to obtain a numeric value for reliability, we can create these results from testing the percentage of time in which the car performs as expected. Determining what is expected can also be difficult. To what standard do we hold the vehicle to? Is it a fault if the vehicle stays on the road when we expect an accident in a particular case? This must be answered by the software engineers before any form of reliability assurance can be made. It must also be standardized cross-industry if comparisons in reliability are to be made.

VIII. MAINTENANCE AND CODE REUSE

When engineering a large-scale safety critical system, such as an automated vehicle driving system, the level of projected maintainability must satisfy some core set of requirements. When not engineered with maintenance in mind, software bug fixing and firmware upgrades might be comprised of ad hoc solutions that will compound in intricacy upon further generations. Proper maintenance plans help to reduce the level of complexity for software systems, and also reduce the cost of developing and deploying software updates and safety revisions.

In automated driving systems, safety is a primary concern. Proper maintenance strategies in such environments yield higher safety ratings and allow for cost effective solutions to this cardinal charge. At the Volkswagen research group in Germany, an implied maintenance strategy is employed at all times. All team members develop and design project resolutions as a group. [2] As a result of this, each team leader is familiar with all aspects of development and can properly assimilate any problem or addition to the system with a maximum understanding of the system as a whole. The time to modify the system goes down while reliability goes up, allowing for better financial management and higher product yield, in this case, safety concerns.

Autonomous driving requires a layer of communication between every self-governed vehicle on the road. In addition, it must maintain strict interface quality and allow for backwards compatibility between models. This can all be achieved through exploitation of code reuse. Proper code reuse techniques can yield higher potential for all forms of analysis, especially in intelligent vehicles. McMillin et al [4] describe a simulation system for their proposed software architecture based testing schema. It was designed to uncover safety issues while allowing for fast prototyping and a large degree of scalability. The benefits of code reuse in this situation are quite clear. By deriving from a central code base upon new test suites, the research group was able to take advantage of a distributed system of virtual vehicles in their testing environment. Each distributed machine communicates via a network communications layer and runs identical elements of the safety critical software. There are many other opportunities for code reuse to resolve issues in engineering a car that drives itself. If all makes and models of vehicles were to draw from the same code base, bug fixes and security updates could be installed and distributed very easily. One example of this might be the communication protocol for which

automated cars communicate. Each car, in order to preserve safety requirements, should be able to notify and receive control signals to and from other automated vehicles. In order to allow for cost effective solutions, a car manufacturer would want to exploit the amount of code reuse necessary to allow for new models of the car to be released each year. This is an obviously ideal location for a code reuse policy. For a new car to be produced, old features cannot be implemented from scratch. That would drive up prices while delaying throughput. Instead, older and proven software is used. This is especially important in high technology vehicles, such as an automated car. The prices will no doubt already be quite high, and further cost does not need to be affected by poor design decisions, such as not taking advantage of code reuse.

IX. CONCLUSION

We have presented a series of software design model characteristics that could complicate the production of an automated vehicle system. Though each represents a fundamental area of software engineering, our results show that in highly experimental and safety critical systems, such as the one in our case study, proper engineering techniques directly affect the quality of the solution. Through previous work and theoretical future work, we have shown that automated vehicular design can yield high beneficial value, yet the cost is multipart.

REFERENCES

- [1] Ahstrom, Kristina. "Design Method for Conceptual Design of By-Wire Control: Two Case Studies". IEEE conference on Engineering of Complex Computer System, 2001.
- [2] Binfet-Kull and Heitmann. "System Safety for an Autonomous Driving Vehicle." IEEE International Conference on Intelligent Vehicles. Stuttgart, 28-30, Oct. 1998.
- [3] Isermann. "Fault-Tolerant Drive-by-Wire System". IEEE Control System Magazine, Oct 2002.
- [4] McMillin, Sirois, Mahoney and Budd. "Fault-Tolerant and Secure Intelligent Vehicle Highway System Software - a Safety Prototype". IEEE International Conference on Intelligent Vehicles. Stuttgart, 492-498, Oct. 1998.
- [5] Murphy, et al. "Ground Vehicle Control at NIST: From Teleoperation to Autonomy". Robot Systems Division, NIST. Gaithersburg, MD, 1993
- [6] Weyuker and Ostrand. "Theories of Program Testing and the Application of Revealing Subdomains". IEEE Transactions on Software Engineering, Volume 6, Number 3, May 1980, pp. 236-246.
- [7] Whittaker. "What Is Software Testing? And Why Is It So Hard?". IEEE Software, January/February 2000, pp 70-79.
- [8] US Department of Transportation, Intelligent Vehicle Initiative. "Driving Safely Into the Future With Applied Technology". Internet. WWW: <http://www.its.dot.gov/ivi/docs/IVIBrochure.pdf> [March 2003].