KTH Industriell teknik och management

Towards Autonomous Architectures
An Automotive Perspective

Sagar Behere (KTH)
Björn Liljeqvist (EIS by Semcon)

Technical report
Department of Machine Design
Royal Institute of Technology
SE-100 44 Stockholm

TRITA - MMK 2012:10
ISSN 1400-1179
ISRN/KTH/MMK/R-12/10-SE

Abstract

The use of embedded computers in modern automobiles is enabling increasingly autonomous features. Electronic power train management and applications in active safety, cooperative driving and navigation show an underlying trend of the transfer of responsibilities from the human driver to a vehicle's (semi-)autonomous subsystems. The logical culmination of this trend would be a completely autonomous vehicle. How should existing vehicle architectures be evolved to sustain the development and growth of autonomous functions? We explore the principal problems with existing architectures, caused due to ad hoc addition of (semi-) autonomous features and argue that it is time to rethink automotive architectures from an autonomous systems perspective. We introduce a pattern that can help architects and designers to think in terms of autonomy and suggest where the application of autonomous systems thinking should begin, in the context of architecture development.

1 Introduction

There is a rising trend of incorporating computers in the modern automobile. These computers interact with the mechanical subsystems in the automobile and are used for implementing features like anti-lock brakes, electronic powertrain management etc. Thus, the modern automobile can be viewed as a cyber-physical system with a growing number of functions.

The trend of this functional growth is towards increasing autonomy. This implies that decisions which hitherto needed to be taken by the human driver are being taken by the vehicle itself and the human driver exerts lower direct control over the machine's actions. The trend towards autonomy is not new. It started with the automatic transmission and continued with features like cruise control. Adaptive cruise control, active collision avoidance etc. are latest examples of this trend. The usage of computers accelerates this trend because computers can endow the machine with complex reasoning skills as well as a degree of control that surpasses human abilities. The logical culmination of this trend would be a fully autonomous vehicle which can safely drive between locations and care for its own maintenance needs and those of its passengers.

Can existing automobile electrical/electronic (E/E) architectures[1] handle the twin demands of functional growth and rising autonomy? The general consensus in the automotive industry is that it is extremely difficult for existing architectures to support sustained functional growth. This is because, as more and more functions are added, the system becomes too complex to design, test and certify in a cost-effective way. Furthermore, as more features are added, especially (semi-)autonomous features, there can be unexpected interactions among them leading to undesirable emergent behavior. Existing vehicle architectures do not provide elegant solutions to such problems.

[1]IEEE 1471 defines an architecture as, *"The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution."*

How should vehicle architectures be evolved to support functional growth and increasing autonomy? To answer the question, it is necessary to investigate the nature of the problem, before considering the nature of the solution. In this paper, we argue that it is time to rethink automobile architectures from an autonomous systems perspective. We put forth the hypothesis that thinking in terms of autonomy will lead to resolution of multiple architectural issues being faced today; and that in fact the issues arise in the first place because we do not recognize the automobile for what it is becoming: namely, an autonomous system. The hypothesis is examined by looking at some specific architectural issues, together with a discussion of how "autonomy thinking" can provide solution directions. We present a mental model that can help architects to think in terms of autonomy during the design phase. We also provide an idea of where such kind of thinking can be generally applied during the architecting process.

Autonomous systems and automotive architecture are studied in multiple research disciplines including software engineering research, artificial intelligence, robot autonomy and architecture [1, 2, 3, 4, 5, 6, 7]. Given the exploratory and problem formulation nature of this paper, we have chosen to refer to related state of the art in the various relevant sections of this paper.

Section 2 elaborates on the concept of autonomy and provides the context and scope for the rest of the paper. Section 3 describes a pattern that can be used as a mental model or frame while thinking about autonomy. Section 4 highlights some of the problems with existing automobile architectures, while section 5 provides an idea of how to go about applying autonomous thinking to the architecting process. Section 6 presents some conclusions and thoughts on future research goals.

2 Specifying autonomy

If a discussion of autonomy is to be of interest, it is necessary to discuss the scope and context of the autonomy in question. We begin by associating specific meaning to the terms 'simple', 'automatic' and 'autonomous'. Then we describe the property of having a Self, which we consider to be essential for the kind of intelligent autonomy that we address in this paper.

2.1 Simple, automatic and autonomous

All machines need energy and intelligence to perform useful work and may be divided into three types from an autonomy perspective: Simple, automatic and autonomous. For a simple machine, both energy and intelligence come from the user. For an automatic machine, only the intelligence comes from the user. For an autonomous machine, neither the intelligence nor power comes from the user, at least insofar as the autonomous function is concerned.

Simple machines or tools are mere amplifying devices that multiply or redirect a force used by the operator. Hammers, levers, wedges etc. belong to this category, but compound machines whose components are of this same category are also simple tools in this respect.

Automatic machines are those that transform energy to produce useful work, for example, a windmill or combustion engine. They are capable of performing a series of operations, ranging from very simple to highly complex sequences. Automatons have a cyclic nature, in the sense that they return to an initial state before repeating a sequence. This is true of any motor or engine, and also of microprocessors[2].

Autonomous machines are those that can gather information from their surroundings and produce a decision based on this information that leads to a successful, desired result. They may use automatic means to execute the decisions. This definition has been deliberately created to encompass many levels from the barely to the fully autonomous. As per this definition the simple mechanical centrifugal governor or thermostat as well as a sophisticated computer controlled cognitive robot are autonomous. All of them gather information from their surroundings and use it to generate relevant results. Whether the information gathering process is implicit or explicit, whether the decision making occurs due to deliberate programming or as a consequence of mechanical construction is irrelevant. The distinction lies in differing levels of autonomy. The level of autonomy of a machine could generically be defined as a function of various independent metrics and the relations between them. Thus,

$$A = f(m_1, m_2, m_3, \cdots, R)$$

where $m_1, m_2, m_3, \cdots$ are independent metrics, for example, number and complexity of internal models used to arrive at the decision, and R represents relations between them. One particular set of metrics is provided in section 3, but an elaboration of possible metrics and how to combine them to arrive at a value for autonomy is beyond the scope of this paper.

The ideas in this paper could be applied to intelligent, computer based autonomous systems and subsystems. In this context, we use the definition of intelligence given in [3], as *"...the ability of a system to act appropriately in an uncertain environment, where appropriate action is that which increases probability of success, and success is the achievement of behavioral subgoals that support the system's ultimate goal."* For intelligent autonomous systems, the ultimate goal and the criteria for its success would be defined externally, while goals and success criteria at levels internal to the system would be decided autonomously. For the fully autonomous vehicle of the future, the ultimate goals and success criteria would be defined by the vehicle designer and the human user of the vehicle.

2.2 The Self and the rest

The Greek word "autos" means "self". Automatic means self-moving. Autonomy means self-governing, capable of making its own decisions. But what is

[2]The microprocessor basically fetches the next instruction and executes it. The addition of software though, can make the computer more or less autonomous.

this "Self" in an engineering context?

The Self can be better understood when compared with a system that lacks a Self. A machine that has no Self is dependent on the decisions of another Self, that is outside that system. In the case of a regular car, the other Self is the driver. Introducing a Self in the system means that there is something (algorithms or rules, usually in the form of software) that can interpret a situation, follow instructions and control some sort of outcome.

Having a Self means having a point of view. From the perspective of the decision making Self, the surroundings appear to it in accordance with the perception processes and the understanding that is built into the system, i.e. internal models. What matters is not the surrounding reality as it is, but as it appears from this viewpoint. It is also in the Self, that the goals and criteria for their success end up.

The existence of multiple Selves in a system means that there could be natural delimitation between them. However, it is by no means certain that a system is designed taking this into consideration. A self could be a piece of software running on an ECU – but it could also be software that is distributed across several ECUs communicating over a databus. If the execution of a self is split up between components, it creates a need for possibly bandwidth-demanding communication, but more importantly, it makes it easy to overlook the fact that it actually is one single unit from the autonomy perspective, one self. Thinking in terms of Selves and non-autonomous non-Selves helps to understand the self-determining aspects of a system, and how they relate to each other.

Therefore, autonomous systems

- accept user commands – to the extent that a user exists – in the form of behavior requests, not as actuator setpoints. The Self controls the actuators at all times and does not expose them directly outside the system boundary.

- are invariant of how the "intelligence" of the Self is achieved. It can be rule based, algorithm based or whatever. What matters is not the nature of the intelligence, but the fact that the Self can accept behavior requests and use its actuators to fulfill them. In other words, "Autonomous is, as autonomous does!"

- ultimately need to have one single Self in charge of the system[3] and this is the Self via which the system is identified by its users and environment. The presence of multiple Selves directing the same system resource is a recipe for trouble. Some examples of problems that can arise out of multiple selves are given in section 4.

[3]although the system may contain subsystems with independent selves.

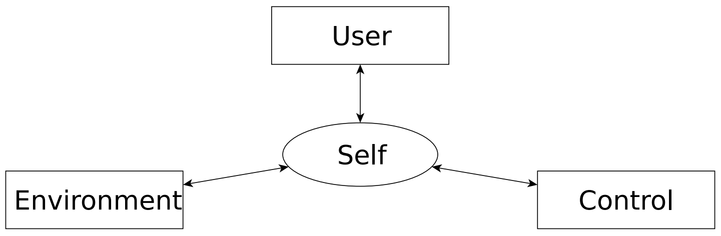

Figure 1: Components of the 3+1 pattern

3 3+1: An autonomous design pattern

In this paper, we use the term pattern for an abstract description of a class of architectures. The purpose of a pattern is to have a template on which to base the work of creating an architecture that could be implemented. It should express the fundamental principles of a solution and make it possible, or easier, to pose the right questions. When looking for a useful pattern for autonomous systems, the field of robotics serves as a useful source of inspiration, since many robots are designed from the start to be autonomous. Unlike cars, robots don't have much pre-autonomy baggage to account for.

In this section, it is postulated that all intelligent autonomous systems can be described by the "3+1" design pattern for context awareness. The level of autonomy is a function of the extent of presence and complexity of the pattern's components. The purpose of introducing the 3+1 pattern is that it enables the designer to see what is actually being constructed.

3.1 Components of the 3+1 pattern

The "3+1" pattern shown in Figure 1 consists of a decision making Self and internal models of the three domains that the system has to account for: the user, the environment, and the physical device to be controlled. In this pattern, they are referred to as Self (S), User (U), Environment (E) and Control (C). To make the pattern complete, a surrounding boundary, or shell, is included. Within the boundary, there is one system using internal protocols and logical interfaces; outside the boundary, there is the physical world that the autonomous system is operating in. When the things to model are physical objects, sensors are needed to build these models. When they are other autonomous systems connected using logical protocols and interfaces, there is no need for sensors, since the systems are in direct communication[4] – but the Selves only see each other through their internal models, they are not in direct contact.

U, E and C are known by S, but know nothing of each other. The self is the only part that makes decisions, the other three components are – from the

[4]A communication channel that delivers information from another system could be considered as a sensor.

perspective of the self – mere filters (U and E) or an actuator (C).

3.2 The User model

There must be something that tells the system "What to do?". Regardless of whether this is a car driver, that constantly provides intentions to the system, or just a set of rules or instructions hardwired into the system at the time of construction, as in the case of a remote space probe, an autonomous system always has a mission to fulfill. It does not have to be a human. Autonomous systems can work together where one system is the master of another (system-of-systems). From the perspective of the serving system, it does not matter if the user is a human or a machine: It will have its internal model of the user nonetheless. Why model the user? Why not simply let the user control the serving system directly? Because of the very definition of autonomy, the system is self-sufficient and supplies itself with the necessary control signals to maintain its operation. What it cannot do, is know what to do, what function to fulfill, without some other entity telling it. By modeling the user, the system providing the intentions, the Self is given a machine "awareness" and can distinguish between itself and the user. If the user is a human car driver, the model contains a way to translate what the driver commands, using ambiguous human signals, into a set of unambiguous control signals that the Self can consider, before making its decision.

3.3 The Environment model

When performing a task given by the user, the system has to take its surrounding into consideration. This is particularly true for robots and self-navigating vehicles, that need to use sensors of many types to build up an internal software representation of objects like roads, walls, other vehicles etc. It is in the environmental model that the result of the operations of the system will be visible.

Without an environment inflicting constraints, there is no "problem to solve", since the output of the system can be determined as a direct function of the input orders from the user. Also, without the feedback from the environment, the system cannot take its own actions into account when deciding what to do next.

3.4 The Control model

An output to control. This can be anything from a heating aggregate with one input signal, to a complete car, itself a union of hundreds of control parameters. Without an output to control, the system cannot do anything. A model of the system being controlled is necessary for the self to make the right decisions. The better the model, the more accurately it describes the physical reality that is being controlled, the better the system can predict what effects its actions will have. A distinction needs to be made between directly controllable outputs within the system (or "body") and indirectly controllable objects in the environment. The control model refers only to directly controllable outputs.

3.5 The Self

The Self is not a model, in the sense that it is not an internal representation of an external thing. It may be implicit or explicit, but it is an entity that constantly makes decisions, based on the above mentioned internal models of the external world. Without a Self, any action that delivers an output by taking user input and environment data into consideration, does nothing more than perform automatic and predictable responses to what the user is doing. With a Self, the system may choose to respond differently to the user requests based on, for example, its internal state. The Self could choose between "modes" of the system and modify behavior of the other three models.

In the case of an autonomous car, this actually means that the driver does not have the final say over what happens, albeit a very big influence. The driver provides intentions to be fulfilled, but the point of autonomous driving is to allow the car to make its own decisions when these are likely to be better, safer or more convenient than those of the driver. Making the driver still feel in control is an engineering challenge, but this does not preclude the existence of a Self.

3.6 Levels of autonomy

The 3+1 pattern provides a way to reason about the level of autonomy of a system. The latter factors in the presence and complexity of each component and it's relationship with the Self.

$$A = f(S, U, E, C, R)$$

In section 2, we asserted that a simple, mechanical centrifugal governor or thermostat is an autonomous system. It is a primitive system that does not have an explicit Self, no user model and rudimentary knowledge of the environment (rotational speed or temperature) and control (mechanical linkage, laws of physics). Further, it has no explicit environment model populated by the sensed parameter (rotational speed or temperature), nor an explicit model of the control system.

At the opposite end of the spectrum is the fully autonomous car. It would have a user model, that enables it to interpret the commands and intentions of the human user. It would have explicit models and representations of the environment it operates in, and these would be continuously updated by a stream of sensor information. It would have control models that can understand and predict the effects of actuators and use them to reach desired setpoints. Finally, there would be an explicit Self that is in overall charge and provides the identity of the system to the human user. Thus, it would not only have all the 3+1 components, but each component would have an appreciable amount of complexity. Therefore, the level of autonomy of this system would be very high.

4 Some problems with existing architecture

Existing automobile architectures have reached their present stage via a natural and obvious process of evolution. Initially, there was the completely mechanical vehicle. As the benefits of using the microprocessor as a machine component emerged, designers starting replacing purely mechanical functions with microprocessor controlled ones, with generally positive results. As the number of electronic control units (ECUs) increased, it was natural to connect them in some kind of a network, so that the units could share information for the purposes of collaboration, synchronization, error detection etc. Then, as the demand for network capacity increased, steps were taken to partition the network into sub-networks of functionally related ECUs. And that is the state of practice in vehicle architectures: a large number of ECUs executing a number of (semi-)independent functions. Significant new functions are usually added by connecting a new ECU (optionally, with new sensors and/or actuators) to the existing network. This pattern of "many intelligent/semi-autonomous units/nodes on a data bus" is now prevalent throughout the automotive industry. Built around it is a whole ecosystem of "how we do things", for example: grouping of functions, development of (black) boxes via consultants/vendors/contractors, safety and certification of the individual subsystems composed of one or several nodes on the bus.

Adding features in this *ad hoc* and bottom-up manner is not a sustainable approach. An immediate problem is the increasing number of dependencies between the features, both at a functional as well as technical level. Each feature ultimately exerts influence on some physical or behavioral aspect of the car (braking, acceleration etc.) When multiple features exert unarbitrated influence on the same aspect, the result can be chaotic. That this doesn't happen today is as result of careful arbitration strategies by the designers. But as the number of features and inter-dependencies increase, the problem of conflict resolution will become intractable. Indeed, it may no longer be possible for the designers to even think of all possible things that could go wrong with such a system. Therefore, this approach is not sustainable. What is needed is a systematic approach to the architecture that provides "correctness by construction" and enables composition of systems [8, 9] without unanticipated effects.

4.1 Competing Selves

A particular automated feature, such as adaptive cruise control (that maintains a fixed distance, measured in seconds, to the vehicle in front of it) can be implemented by adding a node to the vehicle network, with an ECU that monitors relevant sensor data and then controls the necessary actuators to achieve the desired distance. Imagine now adding another feature – emergency braking – implemented through another, separate ECU. This will be another node that monitors sensors and controls actuators based on sensor data. In fact, both of these ECUs can be seen as independent "co-drivers", operating on the *same actuators*. Both have a Self. To avoid situations where an actuator gets conflicting orders, something has to arbitrate and make a decision about which node should be the one controlling the actuator in question. This either means creating another Self, a master node, that handles requests to one actuator – or having the two nodes communicating and deciding between themselves which should take precedence and inhibit the other. In the latter case, the Self is not confined to one node. In the former, the question arises how to partition the responsibilities between nodes that handle these functions, and the master node that must do the selection.

Both approaches suffer from the same problem: Functionality that is similar in kind and uses the same actuators is implemented in separate units, as competing Selves. Thinking in terms of autonomy throws up the problem here. An autonomous system needs a single, cohesive Self that uses its components to generate the behavior expected of the overall system.

4.2 State space explosion

Many functions with many shared resources implies a large state space. In order to correctly design for this large state space, the designers carve up the state space into so-called modes that define behavior. Each function or subsystem can be in one of several tightly specified modes and the transitions among the modes are well-defined. But mode management is simply a different "level of zoom" in the design fractal. The initial problem is repeated when there exist a large number of modes in each subsystem and the overall system can be a combination of modes. There is a "mode space" explosion. Just as it is hard for a human being to keep track of multiple modes and their combinations, so also it is hard for an explicitly designed high-level controller to interact correctly with such a multi-mode system.

If we try to keep our system design tractable using principally similar approaches, we end up having to solve the same problem every few years. We merely keep applying the same solution to different levels of the same design fractal. But a principally different approach would break the design fractal and may solve the problem once and for all.

Naturally occurring autonomous systems seem to somehow solve this issue. In the human body, we have several low level functions, for example those regulating temperature, blood pressure etc. They are generally out of reach for the higher order Self (our mind) but still are mission critical in the sense that they just have to work and keep working. For short time reactions, we have reflexes where the Self has no time to react. Classical artificial intelligence (AI) and robot control architectures provide different system structures including deliberative (higher order planning) and reactive (sensory reaction) control systems. An example of a layered architecture for autonomous robotics is given in [10].

Such system structuring could be applied to automobile architectures too.

4.3 Partitioning and re-partitioning

The human brain can handle a limited level of complexity. It is difficult to design systems whose complexity is more that what our brains can handle. Therefore, ways are found to reduce the overall system complexity into manageable portions. Foremost of such ways, is to partition or divide the system into parts and decide their respective responsibilities and relationships. The partitioning is generally done by keeping in mind the functionalities and data flows. In current automotive practice, this is achieved by setting up a system with lots of nodes on a Controller Area Network (CAN) bus. As more and more functions/features get added to the system, existing partitions get redrawn, but each redrawing of the partitions follows, in principle, the same reasoning as the previous refactorings.

More recent automotive architectures are moving beyond the "flat" pattern of many nodes on a bus. Instead, they have a hierarchical structure, but it essentially follows the same methodology. Based on our experience with the automotive industry, the process evolves somewhat like this:

1. Initially, there are lots of nodes on a CAN bus, and bandwith problems arise. Therefore

2. Multiple CAN buses are added and existing nodes are redistributed among them. But as even more nodes are added, even this system starts becoming too complex

3. Then a tree like structure is created, with the main trunk being a high-bandwidth bus (like FlexRAY) and the branches being CAN or Local Interconnect Network (LIN) buses with individual nodes. The nodes are assigned to branches based on their functions and the dataflows needed between them i.e. via the traditional reasoning process

Thus, current automotive architecture structuring practices are not really solving the core problem of complexity management, but merely pushing it away towards the subsequent generation. This is an insufficient approach towards designing the autonomous car of the future. What is needed, is a pattern that is conceptually different from, "many intelligent/semi-autonomous units/nodes thrown together and made to cooperate on a (CAN) bus". Instead, patterns for structuring [11] and decentralization [12] should be explored that are inherently conducive to autonomy and which could serve as a basis for a different kind of partitioning.

5 Autonomy and automotive architecture: Where to begin?

Can principles of autonomous robot architectures [13, 4, 14, 6, 5] and the construction of intelligent machines be applied to automobiles? If we equate autonomy with a system following the 3+1 pattern, the conclusion would be that an architecture based on 3+1 will be more conducive to intelligent autonomy. What would that mean in practice? How can we maintain the peculiarities of automotive software, architecture and development processes [15, 16] and yet evolve them to autonomy? Where to begin this process?

Consider, for the sake of argument, a system that consists of various distributed sensors and actuators and a single central processing/reasoning unit that constitutes the Self. Let such a unit be denoted 'Big Fat Computer (BFC)'. The BFC reads all sensor inputs, has total control over all actuators and implements a single "thread of awareness". Thus, there is a single Self, which is One i.e. not split up into intermittently communicating parts, that understands both the behavior expected of the system, as well as the means available to achieve the behavior (sensors and actuators). Such a Self would result in a completely cohesive system, with no unresolved internal conflicts.

Creating such a system would still present problems. Automotive engineering has since long moved past a single monolithic computing entity and distributed systems have arrived and stayed. The real question then is, "How should a system be created that leverages the advantages of distributed system design and methods, while ensuring that all distributed parts effectively form a single, overall reasoning system?" In other words, how should a unique Self be created in a distributed way?

Could an autonomous system be composed out of autonomous constituent parts? Can the autonomous car have an autonomous engine, an autonomous transmission, an autonomous climate control and so on? From the perspective of the user of the autonomous car, this question is largely irrelevant. What is important is that there should be a single Self in the car that is accepting the user's commands, and which will be held accountable for the entire behavior of the car. Now, whether this Self should itself control every single actuator and function, or whether it should merely choreograph multiple autonomous systems to reach the end result is a design question. We need a process both to answer this design question as well as develop the answer.

Figure 2 shows a simplified view of how an architecture is developed. This is a top down process, that begins with the so called functional or conceptual architecture view [17, 18] that shows the function structuring. This is basically drawing up boxes and understanding their roles, behavior, hierarchies, dependencies and interactions. The function structuring decides how the system should be broken down into functional components and how the components will together fulfill the system's goals. It can also be thought of as the system behavior, on which a logical structure is imposed. The bottommost part is the so-called implementation architecture view. It shows the implementation

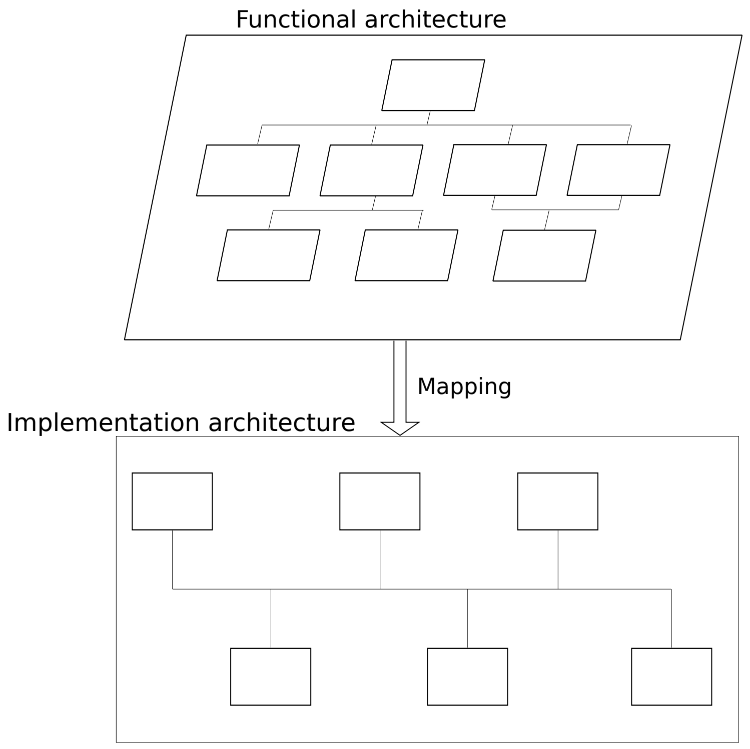

Figure 2: Architecture layers

specific details, the computing hardware and software, the communication and networking choices and so on. This is the architecture as it is actually implemented. The middle is the mapping from the conceptual to the implementation architectures. It shows which functionality is implemented where and how.

Systems thinking from an autonomous perspective can be applied to the conceptual and implementation architectures as well as their mapping. We suggest however, that the start should be made with the conceptual architecture because this is the foundation on which the rest of the development is made. How should the function hierarchies be designed in a way to support autonomy? What are the principles and rules that are conducive to autonomy, or detrimental to it? An automotive architect thinks primarily in terms of visible automobile features like the anti-lock brake system or the engine management system or the cruise control. An architect designing an autonomous mobile robot may think in terms of high level features that generate motion vectors and low level subsystems responsible for the execution of the motion vector. In any case, the functions, their partitioning and hierarchy is the first step. This can then be drilled down to the implementation.

6 Conclusion

Can anything at all be predicted about the future of cars as transportation devices? Yes, because no matter what the fuel will be, or what regulations the cars will have to conform to, it can safely be assumed that embedded intelligence will continue to be added to cars. Advances in other technological fields makes features like self-driving and navigation possible, eventually creating a demand. (There are big potential economic savings for society if self-driving cars could eliminate congestion, increase the usage of existing roads and of course minimize the number of accidents.) The functional growth in cars can increase the burden on the user, who has to keep track of them, or they will be of no use. Offloading this burden is, by definition, autonomy. The development of fully self-driving cars on a large commercial scale is no longer a question of possibility, but a matter of time.

It is time to stop thinking of the automobile as an increasingly mechanized (and computerized) horse-drawn carriage and instead think of it as an autonomous robot endowed with artificial intelligence. Such level of autonomy has a strong impact on the functional, software and electronic architecture of a vehicle. This impact must be studied and considerations for autonomy should be incorporated into the architecture. Research in automotive systems development must take into consideration and build on results from the areas of artificial intelligence and robotics.

This paper not only stresses on autonomy as an inevitable trend in the development of automobiles, it also argues that thinking from the perspective of autonomy has the potential to resolve some outstanding issues in current vehicle architectures. The mental model presented can be used to think in terms of autonomy. A starting point for the application of autonomous systems thinking during the architecting process has also been suggested.

The creation of an explicit autonomy based functional design for the modern automobile, which includes all current features and arranges them into autonomy friendly hierarchies would be a natural next step along this research direction.

References

[1] O. Larses, *Architecting and Modeling Automotive Embedded Systems.* PhD thesis, Royal Institute of Technology, Stockholm, Sweden, 2005.

[2] M. Broy, "Automotive software and systems engineering," *Proceedings Second ACM and IEEE International Conference on Formal Methods and Models for CoDesign 2005 MEMOCODE 05*, vol. 0, pp. 143–149, 2005.

[3] J. Albus, "Outline for a theory of intelligence," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 3, pp. 473–509, 1991.

[4] J. Albus, "A reference model architecture for intelligent systems design," *An introduction to intelligent and autonomous control*, pp. 27–56, 1993.

[5] B. Hayes-Roth, "A blackboard architecture for control," *Artificial Intelligence*, vol. 26, pp. 251–321, July 1985.

[6] A. J. Barbera, J. S. Albus, and L. S. Haynes, "RCS : The NBS Real -Time Control System," 1984.

[7] H. Tianfield, "Fundamentals and Architectures of Complex Distributed Systems," *Systems, Man and Cybernetics, 2008. SMC 2008.*, pp. 2471–2475, Oct. 2008.

[8] G. Brat, E. Denney, K. Farrell, D. Giannakopoulou, A. Jónsson, J. Frank, M. Boddy, T. Carpenter, T. Estlin, and M. Pivtoraiko, "A robust compositional architecture for autonomous systems," in *Aerospace Conference, 2006 IEEE*, pp. 8–pp, IEEE, 2010.

[9] S. Ilieva and M. Zagar, "GENESIS - A Framework for Global Engineering of Embedded Systems," *Genesis*, pp. 87–93, 2008.

[10] R. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal on Robotics and Automation*, vol. 2, no. 1, pp. 14–23, 1986.

[11] H. Tianfield, "Structuring of large-scale complex hybrid systems: From illustrative analysis toward modelization," *Journal of Intelligent &amp; Robotic Systems*, pp. 179–208, 2001.

[12] M. Torngren and J. Wikander, "A decentralization methodology for real-time control applications," *Control Engineering Practice*, vol. 4, no. 2, pp. 219–228, 1996.

[13] J. S. Albus, R. Lumia, J. Fiala, A. J. Wavering, and H. G. McCain, "NASREM - The NASA/NBS Standard Reference Model for Telerobot Control System Architecture," in *proceedings of the 20th International Symposium on Industrial Robots*, no. NIST 1235, NIST, 1989.

[14] J. Albus and F. Proctor, "A reference model architecture for intelligent hybrid control systems," in *Proceedings of the 1996 Triennial World Congress, International Federation of Automatic Control (IFAC)*, 1996.

[15] M. Broy, "Challenges in automotive software engineering," in *Proceedings of the 28th international conference on Software engineering*, ICSE '06, (New York, NY, USA), pp. 33–42, ACM, 2006.

[16] V. Schulte-Coerne, A. Thums, and J. Quante, "Challenges in Reengineering Automotive Software," *Software Maintenance and Reengineering, 2009. CSMR '09. 13th European Conference on*, pp. 315–316, Mar. 2009.

[17] D. Soni and R. Nord, "Software architecture in industrial applications," *Software Engineering, 1995.*, pp. 196–196, 1995.

[18] P. Drongowski, "Software architecture in realtime systems," in *Real-Time Applications, 1993., Proceedings of the IEEE Workshop on*, pp. 198–203, IEEE, 1993.