An Analysis of a Layered System Architecture for Autonomous Construction Vehicles

Sara Dersten
System Architecture & Common Systems
Volvo Construction Equipment
Eskilstuna, Sweden
sara.dersten@mdh.se

Jakob Axelsson, Joakim Fröberg
School of Innovation, Design and Engineering
Mälardalen University
Västerås, Sweden
jakob.axelsson / joakim.froberg@mdh.se

Abstract—**It has been suggested in the literature to organize software in autonomous vehicles as hierarchical layers where each layer makes its own decisions based on its own world model. This paper presents two alternative designs for autonomous construction vehicles based on the layered framework 4D/RCS. As a first step, the typical use cases for these vehicles were defined. Then one use case for a hauler was traversed through the two alternatives to see how they supported safety, flexibility and the use of a product platform. We found that the coordination between bucket control and motion control must be done at a low level in the hierarchy and that the relationship between the vehicle actuators and the built-in autonomous system is important for how the software is organized.**

Keywords—system architecture; systems engineering; autonomous vehicles; automotive systems; systems-of-systems

I. INTRODUCTION

A strong trend within the automotive domain today is towards autonomous vehicles. Cars are more intelligent than ever and fully autonomous application are expected for many types of vehicles and machines, including cars, trucks, military vehicles, mining machines, agriculture vehicles, and construction equipment [1-3]. The main objectives are increased safety and efficiency, and less environmental impact. Studies have shown that around 90% of all car accidents are caused by human factors [4] and the hope is to reduce the number of accidents when the human driver is replaced by an autonomous system. The autonomous vehicles can communicate with other vehicles and surrounding traffic systems and adapt speed and acceleration to various traffic situations so that the fuel consumption is reduced [1]. Similarly, for working machines the expectancy is to be able to achieve a greater productivity and get rid of dangerous manual operations.

The focus of this research is on construction equipment; typically used in quarries, asphalt plants, mining and construction. Since these vehicles perform repetitive tasks and since the work areas are usually confined, they are very suitable for automation. Also, a requirement for an OEM is that an autonomous system will provide a platform and fit in several different types of vehicles on a site. Preferably, the electronic systems of the vehicles are members of the same product family since many requirements and architectural concerns are shared between them.

This research is partly supported by the Knowledge Foundation (KKS) through ITS-EASY, affiliated with Mälardalen University, Sweden, and partly by the VINNOVA FFI program.

On work sites, construction equipment vehicles cooperate to fulfill missions. A typical scenario for an articulated hauler is to get loaded by another vehicle, e.g. an excavator or a wheel loader. When the hauler is fully loaded the loader operator honks the horn as a signal to the hauler driver. The hauler drives to a pocket or a crusher for unloading [5]. Thus, the system is part of a system-of-systems and the system architecture will probably need to consider infrastructure and communication with other vehicles as well as with automation systems, e.g. support wireless V2X communication, efficient tasks distribution, or world model sharing [6].

The design of an autonomous construction vehicle differs from that of an automated car. The mobility of the car, or a truck, mostly concerns driving from a start point to a destination point. A construction vehicle, for instance an excavator, performs tasks while driving. The excavator must control a bucket to be able to dig, at the same time as it moves along a path. Therefore the mobility of these vehicles might be more complex than for a car.

Our research goal is to define a system architecture for the software and electronics that support a product platform for autonomous construction equipment vehicles. The 4D/RCS architecture [6] is a framework for how the software can be organized for collaborative military unmanned ground vehicles for accomplishing common goals.

The research question presented in this paper is if the 4D/RCS can be applied to construction equipment vehicles. Two hypotheses of how to organize the autonomous vehicle system are evaluated against safety, flexibility and support for a product platform.

The remainder of this paper is organized as follows. Section II gives an overview of related work. Section III explains the methodology for investigating the software architecture. Section IV presents the result. In Section V, the results are analyzed. Section VI presents a discussion of the analysis and the methodology and Section VII concludes the paper.

II. Related Work

The 4D/RCS architecture [6] deals with the complexity of planning missions, controlling actions, and perceiving the environment using a hierarchical layered structure. The responsibilities and priorities of the layers correspond to operational units in a military organization. The highest unit, or layer, is a *battalion* that plans missions for underlying companies. The plans span over a period of 24 hours on maps with a typical range of 100 km. The next layer corresponds to the *companies*, which in turn plan activities for their platoons, five hours ahead, on a 30km x 30km map. Each *platoon* plans vehicle movements based on the terrain features over an area of 10km x 10km. The platoon sends commands to its sections, which typically consists of tasks to be completed within 10 minutes. A *section* is a group of vehicles. The incoming commands are decomposed to scheduled cooperative activities and allocated to individual vehicles as a maneuverable corridor on the map. Each individual *vehicle* then assigns jobs for its *subsystems*, such as mobility, attention, communication and mission package. Maps are now generated from on-board sensors with a range of about 500 meters. The subsystem commands its sub-subsystems, controllers that plan and execute velocities and acceleration to optimize the vehicle dynamics. The layer is called *primitive* level and plans span over a period of 500 milliseconds on a sensor-generated map of range 5 meters. Lowest in the hierarchy is the *servo* layer. It has no map and only deals with actuator dynamics and reacts to sensory feedback from actuator sensors. All layers cyclically recalculate their plans about 10 times during the plan period. On each hierarchical level several instances can exist, such as several companies in a battalion or several individual vehicles in a section. Independent of layer, each instance, or RCS_NODE, is organized in the same way. The RCS_NODE has its own sensory processing, world modeling, value judgment, and behavior generation. Madhavan et. al [7] have collected research and implementations of the 4D/RCS. The reference can be used as an engineering guide for autonomous mobility systems.

Behere et al. [8] present a reference architecture for cooperative driving, where the relation between autonomous driving and cooperative driving is very close. The authors present services that the system must provide for cooperative driving. The central part is the world modeling which is the state of the vehicle and the vehicles around it, such as velocity, position etc. The other architectural elements are categorized after how they interact with the world model. The authors suggest that the information in the world model can be hierarchically structured in layers, as in 4D/RCS presented above. The minimum set of needed information in an instantiation of the reference architecture is dependent of the scenario requirements of the system use. Therefore the system use must be found for each instantiation before mapping the system to physical components. Furthermore Behere et al. give a useful list of both typical characteristics of a cooperative system and aspects to consider when choosing hardware and communication strategies.

III. Methodology

This section presents the methodology used to investigate how the software architecture should look like in our case of an autonomous hauler. The assumption is that the 4D/RCS framework is a plausible candidate to work on. The study was divided into two major steps, scope definition and architecture analysis, which are presented next.

A. Scope Definiton

Knowledge about system use, operational environment and system boundaries is essential to be able to analyze what needs the system architecture must support. Hence, we started to identify what kind of work site that was suitable for a first introduction of collaborative autonomous vehicles, what types of construction vehicles, and typical use cases on the site for the vehicles. We interviewed seven specialists experienced in either construction vehicles use at sites or within the development of autonomous systems. The identified use cases were modeled, according to Cockburn [9] in a UML tool, Enterprise Architect, for each type of vehicle. We identified which use cases would be appropriate to implement in a first introduction of the system by evaluating criteria extracted from the interviews against the use cases.

B. Architecture Analysis

The second step aimed to understand if and how the 4D/RCS framework could be adapted to fit the scope definition. Therefore, we selected a single use case, and manually stepped through it to see how it affected the different hierarchical layers of the 4D/RCS stack. In each step, we analyzed how two hypotheses of different ways of organizing the system responded to the three chosen quality attributes. These were safety, flexibility, and how well it would support a product line of several cooperating products.

IV. Result

In this section we present the results from the initial interviews and the execution of a selected use case through the hierarchical layered software architecture.

A. Scope Definiton

The most suitable work sites to introduce autonomous systems on are those where some kind of material is loaded, transported and unloaded, typically on quarries or on asphalt plants, due to the fact that the tasks are quite repetitive and that the site is a confined area. The construction vehicles for these tasks are excavators, wheel loaders, and haulers. We identified several use cases that all vehicles have in common, such as movement tasks, go to position, avoid obstacles, and supporting tasks, e.g. activate vehicle or perform system checkup. These are suitable to implement in a common software platform. When asking the respondents of the conventional usage of these vehicles, we found that there are many tasks performed today, that are not really done by vehicles, but which must be automated within our scope. One group of use-cases are related to communication between vehicles or between vehicles and the surroundings, such as honking to each other when a vehicle is fully loaded, or recognizing that a crusher is ready to receive material. Another group is related to site activities. Before the vehicles are installed at the site, many activities have to be completed, such as map creation, positioning of different material piles, definition of traffic rules, etc. During the operations there must be systems that monitor site performance, keep track on maintenance need, or counter environment conditions. So far, we found about 75 use cases divided into common use cases, use cases for each vehicle in scope and for the site. From the interviews we identified three criteria for analyzing necessary use cases for a successful first introduction of autonomous vehicles. First criterion was *A commercial and safe production solution*, i.e. a site production solution that is beneficial and safe for the customers. Second was *Completeness*, meaning some use cases might be needed to complete other use cases that fulfill some other criterion. The last criterion was *Easy implementable use cases*, i.e. use cases that are relatively easy to implement or already far along in the development, so called low hanging fruits, and consequently beneficial to introduce in a first phase of industrialization. We identified around 40 use cases that were necessary in a first introduction of autonomous construction vehicles.

B. Architecture Analysis

Suitable for automation, we selected a use case, *Build Pile*, to test against the 4D/RCS. It belongs to an articulated hauler, and involves tipping material into a neat pile. The use case is not as complex as the use cases of the other construction vehicles, e.g. the use case of digging with an excavator, but still the tipping concerns simultaneous control of vehicle driving and bucket movements. To get an overall picture we divided an asphalt plant site into the highest layers of the 4D/RCS architecture. The production manager on an asphalt plant normally has information several weeks, sometimes months, in advance of coming orders of asphalt. Still, the customers wait to place the definitive order until one day before the asphalt is needed, and the orders can be changed during the production day.

1) Division into off-vehicle 4D/RCS layers: The asphalt plant site was organized in a similar way as in 4D/RCS.

a) Batallion layer: The batallion corresponds to the full site, or several sites within the same asphalt supplier business. On this level, the production manager plans and replans the activities of the day to fullfill the incoming orders from customers.

b) Company layer: In the military, a company is normally a unit that is specialized for a certain task. Agents on a site (vehicles, humans etc.), could be an operational company that handles the activities related to feed the asphalt plant with material, a maintenance company that, for example, plows the roads on the site in the winter, or an office company that manages incoming orders and salary payments.

c) Platoon layer: The company is divided into platoons with less agents. On a small site probably only one platoon is needed, but on larger sites that span over an extensive area, probably more platoons are needed.

d) Section layer: A section is a group of vehicles that cooperate to fullfill a common task. The participating vehicles must know the location of other vehicles and cannot complete the task without the others. Still, a section might consist of only one vehicle.

2) Scenario description: Our scenario is an asphalt plant site, see Fig. 1. The Batallion layer plans how material should be tranported and loaded into stockpiles. In that way the asphalt plant can be supplied with the right material at the right time more easily. The layers below refines the plan. We will follow an articulated hauler working in section 1 of the plant. The section consists of an excavator that loads material on two haulers that transport the material and build piles nearby the plant. In Fig. 1, another section, consisting of one loader, feeds the plant with piled material. The Section level, the lowest layer off-board the vehicle will command hauler A to execute the task. The hauler A will follow a given path from the excavator to the area where the pile is located and receive the command BuildPile.

We will now walk through the command BuildPile. The hauler is in a position next to the pile area and has just received a clearance to enter. The hauler must detect if there is already a pile within the area and from that position calculate a dump position. Thereafter it must enter the area and back up into dump position, and then dump the bucket material onto the pile, possibly while moving. Normally the driver uses a simultanous combination of the bucket controls, the accelerator pedal, and the gearshift, during the building. When the vehicle is in dump position, and after judging there is no risk of tilting, the tip brake is enabled and the bucket is lifted. When the bucket is lifting, the driver can increase the engine speed, and thereby the bucket lifting speed, by pressing the accelerator pedal. When the bucket is lifted enough, the driver shifts gear to position D (drive) but without pressing the accelerator pedal. This disables the tip brake and in this way the vehicle will be pushed forward by the material that leaves the bucket.

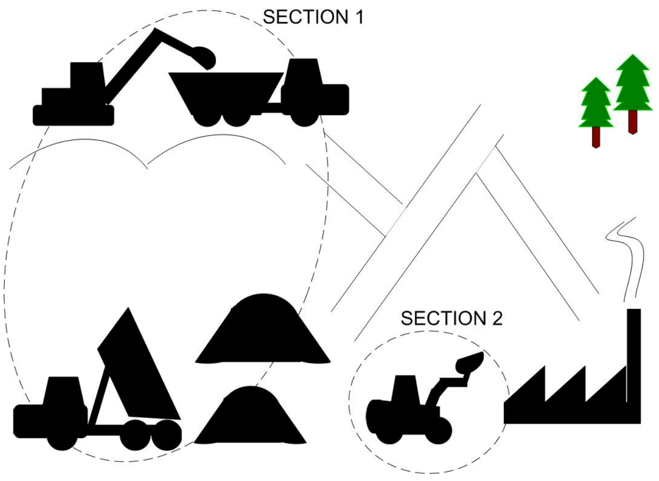

Fig. 1. Illustration of the scenario. Section 1 consists of two haulers and one excavator that are collaborating to move material to piles nearer the plant. Section 2 consists of one loader that feeds the plant with the material.

3) Division into in-vehicle 4D/RCS layers: We were interested to find out how the layers in the vehicle should be organized into software blocks. We stated two hypotheses of how the software could be organized, which are presented below. The 4D/RCS framework suggests a subsystem called a mission package. A mission package contains all the necessary hardware, like mechanics, actuators and sensors, and software layers that are needed to execute the specific mission of the vehicle. A typical mission package could be a package mounted on the vehicle to perform reconnaissance, surveillance, and tracking missions. In this way, the rest of the subsystems and their respective subordinate layers will be organized in the same way for all vehicle types, and in this way we have a platform.

a) Hypothesis H1: Our first hypothesis is that this approach is suitable for a platform for site vehicles. The platform should include the subsystems that are common for all vehicles, such as vehicle mobility and attention. The mission package executes the missions that are specific to the vehicle, which for a hauler includes bucket mobility. The organization of the system into nodes according to the first hypothesis is illustrated in Fig. 2.

b) Hypothesis H2: The second hypothesis is that the bucket control system cannot be isolated from the vehicle propulsion system on the sublayer level, see Fig. 3. The control of the hauler bucket must be coordinated with the control of the vehicle propulsion and both the two systems need to communicate with the same nodes on the servo layer to control engine or hydraulics. In our second hypothesis the full vehicle mobility, including both vehicle propulsion and bucket movements are organized in the same node at the subsystem level. In [8] a bottom-up solution of the organization is suggested to avoid that several nodes control the same nodes on the servo level.

The vehicle level plans the commands that it will send to each subsystem node within the nearest minute. Each command spans over a period of five seconds. For both H1 and H2, the attention subsystem is commanded to detect a

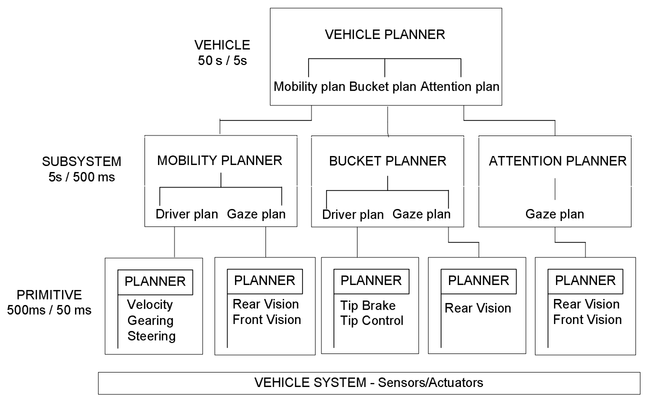

Fig. 2. The organization into nodes according to the first hypothesis, H1. The coordination between vehicle propulsion and bucket motion is managed on the vehicle level. On each node there is a planner that makes and holds plans for its subordinated nodes. The times stated for each layer stands for how long the calculated time is and how often the plan is recalculated.

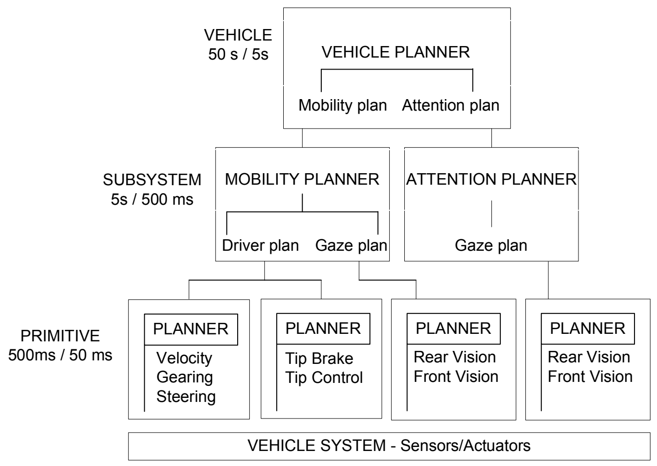

Fig. 3. The organization into nodes according to the second hypothesis, H2. The coordination between vehicle propulsion and bucket motion is managed on the subsystem level. The times stated for each layer stands for how long the calculated time is and how often the plan is recalculated.

pile position and from the return value the vehicle level node plans how the vehicle will enter a position for dumping the bucket material. Fig. 4 illustrates how the command *DetectPile* may traverse the hierarchical levels. The main difference on this level is when the vehicle level node plans the dumping of the bucket material on the pile. For H1 the vehicle calculates and plans all coordination of the motion and bucket subsystems to fulfill the command. For H2 it only sends the command *TipOnPile* to the common vehicle motion node.

On the subsystem level, H1 has three nodes, the mobility subsystem node, the bucket node and the attention node. Each of these subsystems has a planner that for each incoming command makes plans in a period of five seconds and refines them every 500 milliseconds for their respective sub-subsystems on the primitive level. 4D/RCS suggests a gaze plan to be used on all mobility nodes on the primitive level. The gaze plan is used for navigation and for detecting obstacles, such as pedestrians on the path, so that accidents can be avoided. Therefore, all subsystems have to plan for a gaze node on the subordinate level and there will exist three gaze nodes on the primitive level, one for attention, one for the bucket and one for the mobility of the vehicle. H2 has only two nodes on the subsystem level, one for the total vehicle mobility and one for attention. Therefore the main difference between H1 and H2, is that the planning of the coordination of the bucket motion and the driving of the vehicle is made on the subsystem level for H2 instead of on the vehicle level for H1.

The nodes on the primitive level receives commands every 500 milliseconds, and from that sends commands to the nodes on the Servo level each 50 milliseconds. We created time frames of 500 milliseconds from the incoming commands to the primitive level to see which nodes on the servo level that were affected within each time frame. The reason was to see if any actuator was affected by more than one primitive node at the same time. This did not happen for either H1 or H2, but we saw a risk that it could happen for

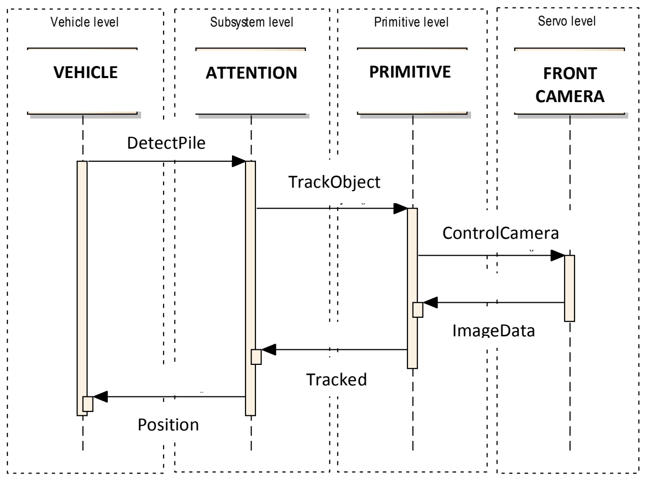

Fig. 4. Execution of command DetectPile.

H1 if the front vision systems are rotatable. H1 had two gaze nodes on the primitive level that affected the same actuators on the servo.

V. Analysis

This section presents our analysis of how the two different approaches, corresponding to hypothesis one, H1, and hypothesis two, H2, meet the needs on flexibility, safety, and support of several different vehicles in a product line.

A. Flexibility

As mentioned earlier, all planners regularly recalculate their plans. The organization according to H1 where the bucket mobility and the vehicle propulsion were organized into separate subsystems needed time frames of 5 seconds for every re-planning of the coordination of the total vehicle motions. The coordination was smoother in the H2 organization. Our example command *BuildPile* was simple and nothing went wrong so no re-planning had to be done, but in a real scenario of a hauler building a pile, there could be disturbances to the plan. There could be obstacles in the way, or the bucket is not lifted as fast as was planned by the vehicle layer. Fig. 5 illustrates how the subsystems of H1 and H2 receive commands from the vehicle level and plans for the nodes on the primitive level. When the tip brake is enabled, the lift of the bucket has to wait longer for H1 than for H2. The reason is that the planning of coordination between bucket control and vehicle mobility in H2 is made on a lower layer and the commands are given in smaller time frames. In this way the command for lifting the bucket, *BucketUp*, which is given after the command *EnableTipBrake*, can be given an earlier time for start.

B. Safety

We identified the risk of subsystems that share the same actuators. For H1, one example was if rotatable vision systems, used for obstacle detection that is used both by the bucket control and by the vehicle motion control. Maybe there is a greater risk that also the actuation of vision system movements will be done simultaneously by several nodes on the primitive levels if the coordination is planned with too long time frames. In H2, the same gaze node manages the obstacle detection for both bucket control and vehicle motion control so the risk is lower. For both H1 and H2, the attention subsystem is isolated and has its own gaze node on the primitive level. However, the attention tasks are always isolated in time from the bucket control and vehicle motion, so we do not see a safety risk here. They are performed when the vehicle is not moving. Also, the assumption of using rotatable vision systems might be questioned. It might be better to use several fixed cameras that are cheap and without troublesome mechanics.

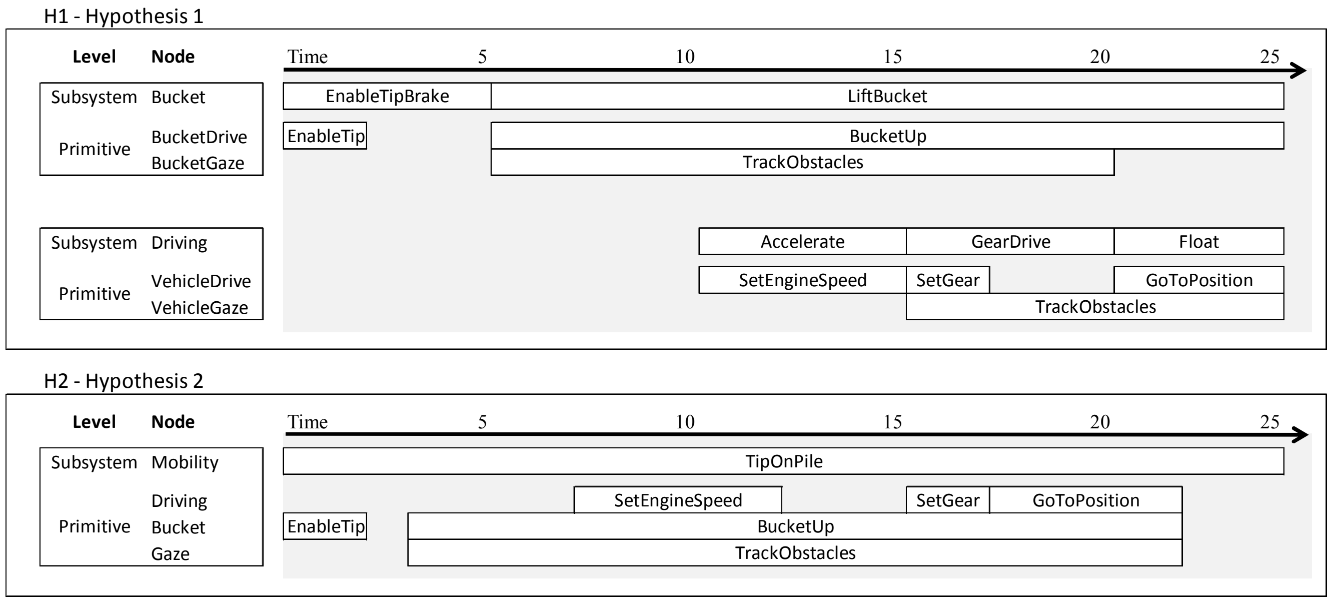

Fig. 5. Thee subsystems of H1 and H2 receives commands from the vehicle level and plans for the nodes on the primitive level. After the tip brake is enabled, the bucket can receive commands much faster in H2 organization than in H1 organization.

C. *Support several products*

We found that the organization according to H1 was more appropriate for structuring the system into a platform. The vehicle specific components, like bucket control, and the common vehicle components, like vehicle propulsion, were isolated in separate nodes. In this way the bucket control for a hauler could easily be changed to bucket control for a wheel loader. However, in H1 the coordination of motion of the vehicle was planned on the vehicle level. This means that the node must be aware of what tasks the vehicle is able to execute and generate plans for the best possible behavior of the nodes on the lower levels. The nodes for vehicle mobility on all lower levels can be shared between all vehicle products to some extent.

VI. Discussion

There are many reasons for having a common platform for autonomous vehicles. It is common practice in automotive industry and it is less expensive to maintain when components and software can be shared in several products. The developers get more effective by keeping the knowledge of the platform when going from one project to the other. All collaborating vehicles on a site will automatically share the same communication protocol and can respond to commands from the same site management system. However, we realize that it is not easy to isolate the functionality that is common to several vehicles from the functionality that is specific for the individual vehicle. The reason is that the mission package for construction vehicles, like the bucket control systems, need to set the same actuators as the vehicle propulsion mobility systems. For example, both need to control engine or rotating visions systems.

We investigated two hypotheses of how to organize the system for managing coordinated motion control. In the first hypothesis, H1, the coordination planning was managed on a high hierarchical level. It gave better possibilities to isolate specific vehicle functionality from general construction vehicle functionality than the second hypothesis, H2, but the problems occurred at lower levels in the hierarchy. Vehicle modes that define which system is currently allowed to control servos could be one solution for H1. Another solution might be some sort of prioritizing between the systems. That would need extensive investigation to ensure that the right system is given the highest priorities. The third solution is to have interconnections between the nodes on the same hierarchical level. The communication would not only be to share sensory information but also plans for accessing the same actuators. However, this would cause more nestling and increasing complexity, which is the problem that we want to avoid.

The second hypothesis, H2, of the organization of nodes gave a smoother coordination of the bucket motions and the vehicle propulsion due to that the planning was managed in shorter time intervals on lower hierarchical levels. Another benefit was that, in this case, the systems did not to try to access the same actuators. However, in this organization the bucket motion system and the vehicle propulsion system are nestled together in the same nodes. That might be a problem for defining a platform. Behere et al. [8] suggests that the functionality of the world model and its surrounding elements is implemented as plug-ins that can use different mechanisms for the different roles. That could be a plausible solution if H2 is used. In [10] a plug-in methodology for AUTOSAR based systems is suggested.

We have only executed and investigated one command given by the section level to the vehicle. To make any conclusions about what the final organization of the system will look like, we have to investigate the rest of the identified use cases for both the hauler, and the rest of the concerned vehicles. Further we used this methodology to identify what data is needed to execute all commands in each node on all levels of the hierarchical organization. When we know the final organization, we will use this methodology to identify the needed data on each node, and from that develop strategies for the allocation of functionality to hardware and topologies for the network and sensors. We also must define data that pass upwards in the hierarchical stack. This methodology can also be used to investigate how faults, from e.g. sensors, propagate between the layers in the stack.

VII. Conclsuion

The way of decomposing missions and distributing them on a group of collaborating autonomous vehicles, described in the military framework 4D/RCS, could be useful for construction equipment vehicles. However, the framework is mostly developed for assigning missions that concern position shifts, where autonomous construction vehicles need to simultaneously control tool movements and vehicle progress. We have investigated two ways of organizing the vehicle systems into nodes in hierarchical levels to deal with this problem and analyzed them against flexibility, safety implications and how well they support a product platform. We realized that the choice is a trade-off between the flexibility of the vehicle and support of a platform.

We suggest that functionality for vehicle propulsion and functionality for other motion systems that are vehicle specific, e.g. bucket control, is closely connected on lower hierarchical levels. This can be achieved, either by adding interconnections between the subordinates to the systems or to implement them as a dedicated subsystem.

In the future, more use cases will be executed through the hierarchical levels to define a final organization of the system. The use cases must cover the full life cycle of the system, for example diagnostics and maintenance. We will use the presented methodology to identify needed data for nodes to execute their received commands. We will use this information to identify sensor requirements and hardware topologies.

References

[1] R. Bishop, “A survey of intelligent vehicle applications worldwide,” IEEE Intelligent Vehicles Symposium, 2000, pp. 25–30.

[2] J. Albus, R. Bostelman, H. Tsai, T. Chang, W. Shackleford, and M. Shneier, “Integrating learning into a hierarchical vehicle control system,” Integr. Comput. Aided. Eng., vol. 14, 2007, pp. 121–139.

[3] T. Bak and H. Jakobsen, “Agricultural robotic platform with four wheel steering for weed detection,” Biosyst. Eng., vol. 87, 2004.

[4] National Motor Vehicle Crash Causation Survey, DOTHS811059, NHTSA, 2008.

[5] D. Rylander and J. Axelsson, “Lean method to identify improvements for operation control at quarry sites,” In Proc. 30th Int Symp on Automation and Robotics in Construction and Mining. Montreal, Canada, Aug. 11-15, 2013.

[6] J. Albus, H.M. Huang, E. Messina, K. Murphy, J. Maris, A. Lacaze, Alberto, et. al. 4D/RCS: “A reference model architecture for unmanned vehicle systems version 2.0,” National Institute of Standards and Technology, Gaithersburg, Maryland.

[7] R. Madhavan, E.R. Messina, and J.S. Albus, Intelligent Vehicle Systems: A 4D/RCS Approach. New York, Nova Science Publishers, 2006.

[8] S. Behere, M. Törngren, and D.J. Chen. ” A reference architecture for cooperative driving,” in Journal of Systems Architecture, vol. 59, pp. 1095–1112, 2013.

[9] A. Cockburn, Writing effective use cases. Addison-Wesley, 2001.

[10] J. Axelsson and A. Kobetski, “On the conceptual design of a dynamic component model for reconfigurable AUTOSAR systems,” In Proc. 5th Workshop on Adaptive and Reconfigurable Embedded Systems and in ACM SIGBED Review, April, 2013.