Time-bounded Adaptation for Automotive System Software

Serena Fritsch, Aline Senart
Distributed Systems Group
Trinity College Dublin
Dublin, Ireland
fritschs, senarta@cs.tcd.ie

Douglas C. Schmidt
Insitute for Software Integrated Systems (ISIS)
Vanderbilt University
Terrace Place, USA
d.schmidt@vanderbilt.edu

Siobhán Clarke
Distributed Systems Group
Trinity College Dublin
Dublin, Ireland
sclarke@cs.tcd.ie

ABSTRACT

Software is increasingly deployed in vehicles as demand for new functionality increases and cheaper and more powerful hardware becomes available. Likewise, emerging wireless communication protocols allow the integration of new software into vehicles, thereby enabling time-bounded adaptive response to changes that occur in mobile environments. Examples of time-bounded adaptation include adaptive cruise control and the dynamic integration of location-aware services within fixed time bounds.

This paper provides three contributions to the study of time-bounded adaptation for automotive system software. First, we categorise automotive systems with respect to requirements for dynamic software adaptation. Second, we define a taxonomy that captures various dimensions of dynamic adaptation in emerging automotive system software. Third, we use this taxonomy to characterise existing research projects in the automotive domain and identify promising areas of needed research in this field.

Categories and Subject Descriptors

D11 [**Software/Software Engineering**]: Software Architectures; A1 [**Introductory and Survey**]: Survey

Keywords

Dynamic Adaptation, Automotive Software Systems, Taxonomy

1. INTRODUCTION

The amount and complexity of automotive software has risen dramatically during the past several decades due to decreasing hardware costs, increasing computing/communication power, and growing demand for new vehicle functionality [15]. Modern vehicles contain more than 2,000 individual functions, such as airbag control software, X-by wire applications, and infotainment applications [35]. Software has thus become a dominant factor of the automotive industry [6].

ICSE 2008 Leipzig, Germany

Moreover, emerging wireless communication protocols, *e.g.*, dedicated short range communications (DSRC) [4], enable vehicles to interact with each other and with their surrounding environment. These protocols allow the integration of applications and data into vehicles from vehicular and/or infrastructure networks. Automotive services and systems that were traditionally vehicle-centric are thus becoming inter-vehicle and vehicle-to-infrastructure-centric [9].

Next-generation automotive systems (*e.g.,* computer-assisted vehicles) will run in highly dynamic environments (*e.g.,* inter-vehicle coordination) and will need to adapt their behaviour during runtime (*e.g.*, in response to frequent changes in their environment). Possible software-related adaptations include the dynamic allocation of resources, adaptation of (multimedia) content, and the adaptation of software itself. Resource adaptation is concerned with the proactive allocation of resources for better quality of service (QoS) [13], whereas content adaptation involves transforming content to adapt to device capabilities (*e.g.*, transcoding of content based on display resolution and processing capabilities [21]). Software adaptation, in contrast, involves the dynamic introduction of software modules and reconfiguration of software architectures, as well as changing parameters that affect automotive system software [23].

In general, the goal of software adaptation is to make automotive systems more evolvable, intelligent, and useful to drivers [11]. For example, a driver information system can dynamically integrate software modules, such as a location- and price-aware lodging discovery service based on driver preferences. Likewise, new software versions can be integrated into a system, *e.g.*, a new a digital signature algorithm can be integrated while a car is waiting at a tollgate. Other software adaptation use cases include reconfiguring a vehicle control system to operate through partial failures.

In automotive systems, software adaptations must often be time-bounded since stale information could trigger improper driver reactions. For example, a driver information system should update a display within a bounded time to ensure drivers are not informed about traffic jams that no longer exist. Likewise, adaptations should minimise software update time to ensure that software applications and data are fully integrated into vehicles before they are used.

Next-generation automotive systems differ in their need for dynamic adaptation support. This paper identifies the software adaptation requirements of the following four classes of automotive systems:

- **Vehicle-centric systems**, whose dynamic adaptation needs are limited, *e.g.*, handling error conditions

[37].

- **Driver information systems**, which require the dynamic adaptation of content, as well as periodic update of vehicle software to better reflect the current environmental conditions.
- **Cooperative driving systems**, which adapt their behaviour according to the surrounding vehicles and road conditions.
- **Vehicular sharing systems**, which use the processing power and data transmission of multiple vehicles to perform distribute computations.

After discussing the software adaptation requirements of these four classes of systems we define a taxonomy that characterises various dimensions of their dynamic adaptation, including binding time, constraints and type of adaptation, timeliness requirements, and adaptation trigger. We then use this taxonomy to classify existing research projects and reveal gaps in existing approaches with respect to time-bounded integration of software modules. We conclude the paper by identifying promising areas of needed research in this field to address these gaps.

The remainder of this paper is organised as follows: Section 2 motivates software adaptation in automotive systems via a managed highway scenario; Section 3 summarises related work on classifying adaptive automotive systems; Section 4 identifies four automotive system classes and their characteristics with regards to dynamic adaptation; Section 5 defines a taxonomy of the identified characteristics; Section 6 categorises existing research projects in terms of this taxonomy; and Section 7 presents concluding remarks.

2. MOTIVATING EXAMPLE FOR TIME--BOUNDED ADAPTATION

Managed highway scenario. To make our software adaptation discussions concrete, consider the following example from the domain of intelligent lane reservation system for managed highways. The goal of a managed highway is to reduce congestion, enable vehicles to maintain safe speeds, and allow emergency vehicles to arrive safely and faster at accidents [33]. One way to schedule and enforce vehicle QoS on a managed highway is to allow drivers to reserve lanes "slots". Moreover, lanes can be partitioned (*e.g.*, low vs. high priority QoS) and can be priced differently. For example, travellers could reserve slots on low priority lanes cheaper than travellers willing to pay extra for slots in a high priority lane that allow them to drive faster and reach their destination sooner.

In this managed highway scenario, vehicles indicate their destination and potential constraints or desires on their way points. To ensure proper admission control, vehicles wait in a queueing lane for their reserved slot to become available before entering the highway (*c.f.* Figure 1) A highway entrance assistance system (*e.g.*, a tollgate) uses short-range communication and relays between queued vehicles to ensure the vehicles have proper software versions and necessary hardware before allowing them to enter the highway. Example software could include warning applications, secure payment and communication algorithms, as well as infotainment applications, such as hotel and restaurant finder or car-to-car gaming; hardware could include road condition and vehicle motion sensors and sufficiently powerful on-board computers.

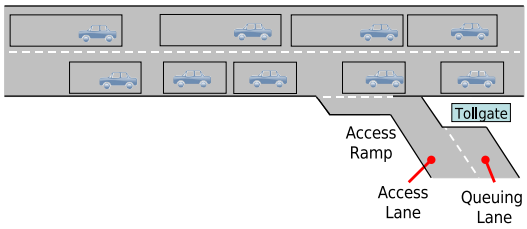

Figure 1: Managed Highway Scenario

Software adaptation mechanisms. The managed highway scenario described above motivates the need for various software adaptation mechanisms. For example, the assistance system could use an adaptation scheduling mechanism to determine which modules to download and integrate dynamically into vehicles based on adaptation policies. These policies specify the actions (*e.g.*, integration, upgrade, downgrade or deinstallation of software modules) to execute on the vehicle, as well as the properties with which (*e.g.,* version numbers or priorities) these modules must comply.

An example adaptation policy is the integration of all high-priority modules, *e.g.*, warning and security applications have higher priorities than infotainment applications and hence must be integrated before a vehicle enters the highway, whereas lower-priority modules are optional. Another adaptation policy is the dynamic update of an existing module to a newer version that is located at the assistance system, *e.g.*, updating a secure payment algorithm. Yet other adaptation policies include the downgrading and deinstallation of modules due to expirations of licences or the change of vehicle ownership [3].

The adaption process itself is time-bounded since the decision process of which modules to download and integrate and the actual download of the modules itself must be executed before the can vehicle enter the highway. These adaptation policies can be influenced by (1) the available memory on a vehicle platform, (2) module interdependencies and (3) versioning of the modules. For example, all necessary high-priority modules may not be downloadable due to timing or memory constraints. In this case, a new time slot may be needed for the vehicle. Likewise, the integration of a secure payment algorithm might trigger an additional integration of digital signature algorithms and secure communication modules that increase the overall adaptation time and might cause an overrun on the overall time bound.

3. RELATED WORK ON CLASSIFICATION OF ADAPTIVE AUTOMOTIVE SYSTEMS

Relatively little prior work has classified software for automotive systems and existing classifications focus mainly on embedded systems. For example, Karjalainen classified embedded control systems into six different classes, ranging from microcontrollers to special purpose control systems [18]. Each class within this taxonomy is identified by eight characteristic groups, *e.g.*, hardware architecture and processor capabilities.

Many automotive systems have stringent real-time requirements that can be classified according to their time con-

straints, *i.e.*, process execution behaviour, timing constraints, and degree of timeliness that they provide [5]. This classification focuses largely on developing integrated schedulers. Other classifications target coarse-grained distributed systems and their characteristics [34]. These characteristics are based on an explicit model of the time constraints as a *path* (the so-called path-based paradigm) and comprise granularity, triggering causes, and classes of data streams.

Dynamic adaptation of general (*i.e.*, non-automotive) software systems is discussed extensively in the literature. An overview is presented in [23], which focuses on compositional adaptation (*i.e.*, the reconfiguration of the software architecture itself) and classifies existing projects with regards to their support of compositional adaptation. This taxonomy comprises three dimensions: (1) where to compose, (2) the point in time of composition, and (3) the techniques used. The authors also highlight challenges, such as assurance and decision making, that must be taken into consideration by an adaptation technique.

Aksit and Choukair summarise approaches for deploying and dynamically adapting applications and software platforms [1]. They distinguish between approaches that deal with the dynamic reconfiguration of component architectures (*i.e.*, introduction and deletion of components) and dynamic adaptability (*i.e.*, the fast adaptation of application behaviour without reconfiguration). They list many technical challenges and solutions (such as maintaining application consistency) to address both approaches.

Although time-bounded adaptation is key to automotive system software this topic is largely absent from existing surveys and taxonomies. The remainder of this paper therefore presents a taxonomy of adaptive automotive systems with regards to time-bounded adaptation and compares this taxonomy with existing research projects.

4. CLASSIFICATION OF ADAPTATION REQUIREMENTS FOR AUTOMOTIVE SYSTEMS

This section presents the results of an extensive domain analysis of existing and emerging automotive systems aimed at classifying these systems in terms of their level of support for and characteristics of dynamic adaptation.

4.1 Vehicle-centric Systems

Vehicle-centric systems assist in the control of a vehicle's behaviour, such as braking or steering. This category contains all safety-critical systems, such as electronic braking systems (EBS) and X-By-Wire [10]. These systems have stringent safety and reliability requirements since they directly or indirectly affect vehicle behaviour.

In the case of a component failure in one of those systems, (safety) critical functionality must continue to operate. One way to ensure continued operation is to downgrade a system configuration to a fail-safe state while disabling non-critical functionality [37]. For example, critical functionality, such as a vehicle dynamics controller, can be degraded to a less sophisticated version, whereas climate control can be disabled. In this context, adaptation can be viewed as an error handling technique.

All component variants should exist at design time to ensure validity and safety of overall system behaviour. Adaptation should then choose a suitable configuration at runtime. Since components can have different importance levels the decision process may need to take their priorities into account, *i.e.*, the adaptation of the vehicle dynamics controller should be performed before disabling the climate control.

Adaptation timeliness is also important since adaptations should be executed quickly to ensure the safe operation of the application, *e.g.*, disabling climate control in a best-effort time bound. Safety-critical applications may even require adaptation to be performed in hard real-time bounds, *e.g.*, downgrading the vehicle dynamics controller within milli-seconds.

4.2 Driver Information Systems

Driver information systems provide information about the vehicle's surrounding environment and the vehicle itself, using internal and external sources. This category comprises a wide range of applications, including navigation guidance, warning about approaching emergency vehicles, and providing information about the vehicle's state. We categorise these systems into the three following sub-classes:

• **In-vehicle entertainment systems**, which deliver to passengers audio and video data obtained from other vehicles (*e.g.* via DSRC [4])) or the infrastructure (via 802.11p [22] or GSM [29]). Due to the instability of connections in vehicular *ad hoc* networks and the heterogeneity of in-vehicle entertainment systems, different data resolutions are needed to suit device and network capabilities [21]. In this context, adaptation is concerned with the changing video data resolution and modifying compression rates of audio and video data based on the current network conditions.

• **Warning applications**, which expand a driver's horizon by providing information about future hazardous road conditions, erratic drivers, and prioritised vehicles (such as emergency vehicles). These systems use information obtained from internal sensors, such as engine temperature sensors, as well as information obtained from other vehicles or the infrastructure, such as the number of neighbouring cars or current weather conditions. Since these systems execute in a mobile environment, information changes continuously, so applications must adapt the displayed information dynamically to better reflect the current conditions. In this context, adaptation involves executing actions depending on the current situation.

• **Travel information systems**, which perform navigation tasks, such as helping drivers locate optimal routes. Other value-added services, such as hotel, restaurant, and parking space locators, are in this category. These services are location-based and display information based on the current geographic location of a vehicle. Like warning applications, the software execution in these systems is directly affected by the external environment. In this context, adaptation involves the change of software parameters, by using either rule-based approaches or the strategy pattern [12].

Future driver information systems will leverage communication with the transportation infrastructure and other vehicles to allow the dynamic integration of new features into vehicles or the update of older features. For example, at a national border crossing vehicle manufacturers could install a specific type of a module in the driver assistance systems of all vehicles that displays additional information about the current country, *e.g.*, maximum allowed speed and specific road signs. In this context, adaptation involves structural and functional software changes. It can be triggered by the

user explicitly, *e.g.*, when connected to a 3G network, or triggered by short range communication between other vehicles and the transportation infrastructure.

Most of the software adaptations in the driver information systems described above should execute in a bounded amount of time to ensure the validity of the information. Adaptations may also depend on priorities between various systems. In the case of software integration, memory constraints and inter-dependencies between modules should be considered.

4.3 Cooperative Driving Systems

Cooperative driving systems require vehicles to coordinate their actual behaviour among themselves or with the infrastructure. Examples include adaptive cruise control, platooning, and adaptive steering. Whenever there is a change to a system parameter, such as the distance to the leading car, cooperative driver systems must react accordingly by changing their behaviour, *e.g.*, accelerating or braking. Adaptive cruise control involves maintaining a safe time-headway distance between vehicles to ensure emergency braking does not cause collisions between cars. The headway calculation system adapts a vehicle's headway by accounting for changed environmental conditions, vehicle dynamics, and safety considerations [17].

In this context, adaptation involves a change in behavioural parameters of a single car to ensure the coordinated behaviour of the group of vehicles to which it belongs. The behaviours of the collaborating vehicles should respond to constraints imposed from other vehicles and from the environment (such as the weather) [16]. This type of adaptation is considered *semi-dynamic* since the system adjusts its behaviour (*i.e.,* the actions to take) by changing system parameters (*e.g.*, the longitudinal control is determined by the distance and the time gap to the next vehicle).

4.4 Vehicular Sharing Systems

Vehicular sharing systems distribute data or computations on vehicles and are comprised of all inter-vehicle sharing systems. For example, an environmentally-conscious (*i.e.*, "green") vehicular sharing system could measure the aggregate carbon footprint of a road or region in real-time using distributed computing resources in the vehicles. If the footprint reached a critical threshold, vehicles could adapt their behaviour to reduce the pollution level, *e.g.*, by switching off their climatisation system, reducing their speed, or shutting down their engine if they are stuck in traffic congestion. In this sense, a vehicular network serves as a closed-loop control system, where disseminated messages trigger a corresponding response.

Data can also be distributed and shared amongst vehicles. In this context, the vehicular network can be viewed as a sensing and relaying network. For example, the delivery of audio or video data to passengers who want amusement along long journeys. These types of systems implement resource adaptation, *i.e.*, they provide facilities for monitoring and controlling dynamic resource usage of activities within a system [13].

5. A TAXONOMY OF AUTOMOTIVE SYSTEM ADAPTATIONS

Based on an extensive literature survey we defined a taxonomy that divides the dynamic adaptation requirements of automotive systems into five *dimensions*: (1) binding time, (2) adaptation timeliness requirements, (3) adaptation type, (4) adaptation constraints, and (5) adaptation trigger. Each dimension can take a finite number of values, which we refer to as *characteristics*. This section uses Kiviat diagrams [20] to visualise the different dimensions of our taxonomy and their possible characteristics and depict the adaptation requirements of the four classes of automotive systems presented in Section 4.

5.1 Taxonomy Dimensions

5.1.1 Binding Time

Binding time is defined as the point in time when the adaptive behaviour is composed with the business logic of an application [23]. *Static binding time* is used when all forms of adaptability are hardwired with the application, *i.e.*, all possible configurations and resource allocations are determined at design time of the application. A change in the adaptive behaviour triggers application reengineering and recompilation.

In *semi-dynamic adaptations*, all possible configurations of an application are determined at design-time. Depending on the current situation, however, a configuration can be dynamically chosen at runtime. In contrast, *dynamic binding* is the most flexible approach since it allows the introduction and alteration of software modules and the reconfiguration of the existing software architecture during runtime without stopping/restarting the application.

5.1.2 Type of Adaptation

The type of adaptation defines what is being adapted. *Resource adaptation* dynamically allocates resources based on the current conditions. *Content adaptation* determines the actions to take to adapt content to better suit device and network capabilities.

Software adaptation is a category that comprises *parameter adaptation*, *functional adaptation*, and *structural adaptation*. *Parameter adaptation* involves the modification of variable values that determine program behaviour. *Functional adaptation* allows application interfaces to remain constant and changes only the implementation parts, *e.g.*, updating of an existing software module to a newer version, whereas *structural adaptation* changes the actual architectural parts of an application, *e.g.*, by replicating objects or introducing new software modules.

5.1.3 Timeliness Requirements

The timeliness requirements of an adaptation characterise the time constraints under which the adaptation is executed. *Hard real-time constraints* require the execution of adaptation within a firm execution deadline. Adaptations executed under *soft real-time constraints* minimise the adaptation execution and blackout time. *Unbounded adaptations* are executed without any time bounds.

5.1.4 Constraints of Adaptation

This dimension comprises the various constraints that might affect an adaptation. *Memory constraints* impose a limit on the size of software modules that can be integrated and is especially a limiting factor in embedded systems. *Priorities* between modules implies an ordering of adaptations since

high-priority modules should be adapted before low-priority modules. Likewise, *dependencies* between software modules can affect the adaptation since the adaptation of one module can trigger an update of other modules.

5.1.5 Adaptation Trigger

These characteristics describe what triggers an adaptation. *Internal triggers* occur inside the system itself, *e.g.*, fault occurrence. External triggers, such as *vehicle-to-infrastructure* or *vehicle-to-vehicle*, are based on external information, either obtained by sensors or communication events with other vehicles or the infrastructure.

5.2 Automotive Systems Diagrams

To present the requirements of the four automotive system classes on adaptation in a visual way, we use Kiviat diagrams [20]. Dimensions of the taxonomy represent axes of the Kiviat diagrams and characteristics of the dimensions represent the set of differentiating characteristics. Since each class has different requirements, they all exhibit different Kiviat profiles and can therefore be compared easily. Kiviat diagrams also allow developers of automotive software to visually map the characteristics of their applications to one of the diagrams, which helps identify dynamic adaptation requirements.

Below, we present the representations in Kiviat diagrams of the four classes of automotive systems from Section 4 and describe their corresponding characteristics in terms of the taxonomy dimensions from Section 5.1. If a resulting diagram supports more than one characteristic in a dimension, the outermost characteristic defines the diagram.

5.2.1 Vehicle-Centric Systems

Figure 2 depicts the Kiviat diagram for vehicle-centric systems. These systems require functional adaptation, *e.g.*,

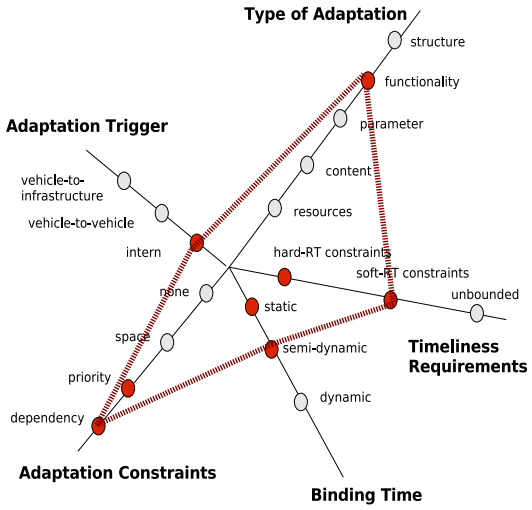

Figure 2: Kiviat Diagram of Vehicle-Centric Systems

downgrading and switching off functionality, when a fault occurs. Hence, the adaptation trigger is internal. Moreover, since these systems are safety-critical, a static or semi-dynamic approach for dynamic adaptation is appropriate and the adaptation actions should be executed with a bounded amount of time. Priorities and dependencies between modules affect the adaptation in these systems.

5.2.2 Driver Information Systems

Since driver information systems comprise a wide range of applications, we divided them with regards to their adaptation type. In-vehicle entertainment systems support content adaptation and the Kiviat diagram for these systems is shown in Figure 3. Since the resulting actions are determined at design-time of a system (*e.g.*, using event-condition-action rules) the adaptation is semi-dynamic. The execution time of the adaptation should be minimised to ensure the freshness of the displayed information. Adaptation is constrained by the available capabilities of the display device and current network conditions.

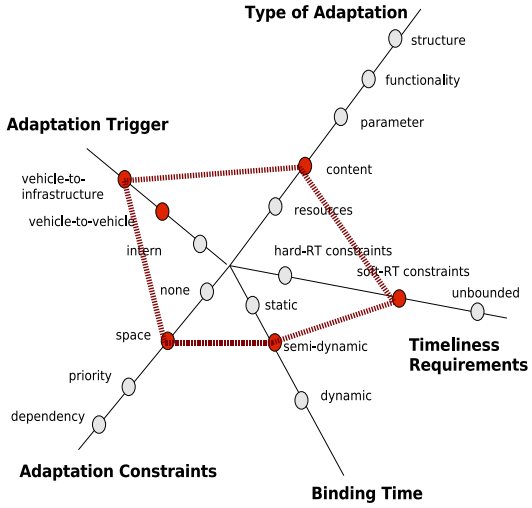

Figure 3: Kiviat Diagram of In-Vehicle Entertainment Systems

Warning applications and travel information systems support the whole range of software adaptation, *i.e.*, adaptation of parameters and also the integration of new services from the infrastructure and other vehicles. The profile of these systems is depicted in Figure 4. This adaptation can be triggered internally by drivers, as well as by inter-vehicle or vehicle-infrastructure communication as software modules are downloaded and integrated via established communication links. The binding time of this adaptation is dynamic since new configurations are determined at runtime of the system. The duration of the adaptation should be executed in soft real-time, *i.e.*, best effort adaptation within time bounds. Adaptation itself can be influenced by the available memory space on the vehicle's software platform. Dependencies between modules and priorities can also influence the scheduling and determination of which and how modules are integrated into the system.

5.2.3 Cooperative Driving Systems

The Kiviat diagram of cooperate driving systems is depicted in Figure 5. This adaptation concerns the change of parameters, such as speed, acceleration, and distance. It is triggered mainly by inter-vehicle and vehicle-to-infrastructure communication and the internal sensing of the current state of the vehicle.

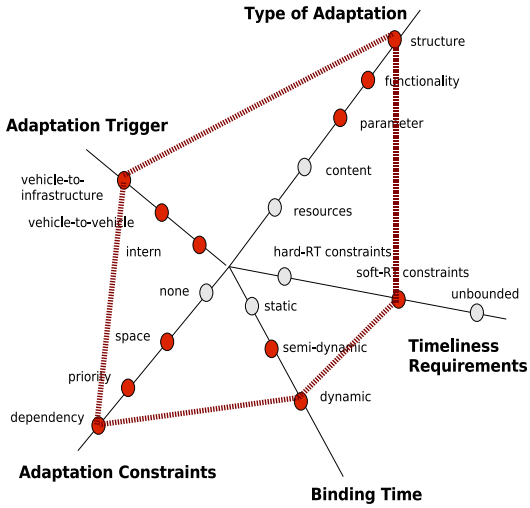

Figure 4: Kiviat Diagram of Warning Applications and Travel Information Systems

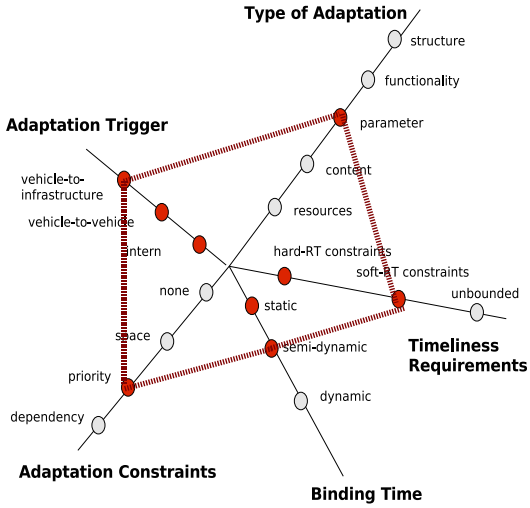

Figure 5: Kiviat Diagram of Cooperative Driving Systems

In cooperative driving systems, actions are determined *a priori* since they may have safety-critical effects on the adaptation. Hence, the binding time is semi-dynamic or even static. All adaptations should therefore execute within a bounded amount of time and are at least soft real-time constrained. In addition to coordination constraints, adaptation can be affected by priorities of triggered actions, *e.g.*, longitudinal control of a vehicle should be adapted before the change lane control to avoid rear-end collisions.

5.2.4 Vehicular Sharing Systems

Figure 6 presents the Kiviat diagram of vehicular sharing networks. These systems require dynamic allocation of

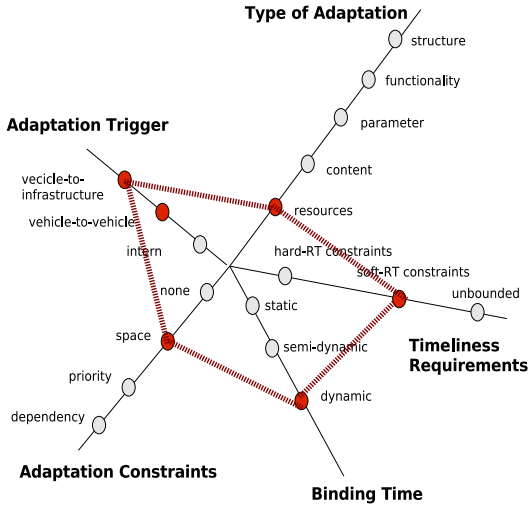

Figure 6: Kiviat Diagram of Vehicular Sharing Systems

resources (such as load balancing and membership mechanisms) since the underlying topology of the network changes continuously. Adaptations are triggered by messages received from other vehicles or from the infrastructure itself.

The binding time of adaptations for vehicular sharing systems is dynamic since new memberships and data or computational loads are dependent on the current status of the network. The nodes in these systems are mobile, so these adaptations should be performed within stringent time bounds. Resource constraints (such as available memory, computational power and bandwidth) impose constraints on the adaptation decision process.

6. APPLYING THE TAXONOMY TO CLASSIFY EXISTING PROJECTS

This section investigates existing research projects and maps them to our taxonomy from Section 5. For each automotive system class described in Section 4, we identified a research project that supports the most suited approach and use Kiviat diagrams to show their degree of adaptation support. We also discuss gaps in these projects with regards to their support of time-bounded dynamic software adaptation.

6.1 Vehicle-Centric Systems

The MARS [37] research project investigates dynamic reconfiguration in embedded automotive systems, *e.g.*, vehicle stability control systems. The approach supports dynamic adaptation at different levels, from coarse-grained reconfiguration of service providers at the system level to fine-grained configuration of behaviour variants. The Kiviat diagram for this system is shown in Figure 7. The adaptation itself

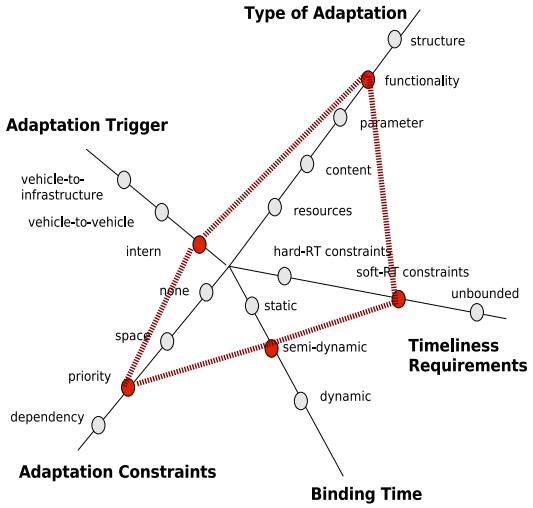

Figure 7: Kiviat Diagram of Mars

is triggered by faults occurring during runtime of the system. MARS considers inter-dependencies between software modules by explicitly modelling and analysing all possible configurations statically at design time of the system. This approach focuses on developing the analysis and modelling techniques for dynamic adaptation, however, rather than developing further adaptation techniques.

The Dynamically Self-Configuring Automotive Systems (DySCAS) [3] project aims at creating self-configurable embedded vehicle control systems that are based on existing middleware technologies and feedback control theories. In addition to structural and parameter adaptations, DySCAS supports closed reconfiguration, *i.e.*, graceful degradation in the presence of component failures. DySCAS is based on policies for achieving the self-management of the system. Policy reasoning includes the selection of the most appropriate reconfiguration in terms of priorities and urgency level. Space constraints can also be specified inside policies. The DySCAS project is still in an early stage, however, so no empirical results are available yet.

6.2 Driver Information Systems

There are a variety of driver assistance systems available, including navigation guidance and value-added locationbased services, as well as safety-enhancing applications, such as emergency vehicle and road warning applications [28].

TrafficView defines protocols and algorithms for the dissemination and gathering of information about vehicles on the road [24]. The system provides the driver with a dynamic view of the road traffic to help drivers in difficult conditions. The graphical user interface periodically displays all validated data sets, *i.e.*, data that is ensured to be neither conflicting nor outdated information. Adaptation here involves updating the display, *e.g.*, when a change in the validated data sets happen. It is triggered when messages from other vehicles are received. TrafficView only supports static binding time, however, and does not mention any consideration of constraints, such as dependencies.

The Safe speed and safe distance (Saspence) project is designed to provide suggestions of the proper velocity and headway based on the current driving conditions [2]. The project uses internal sensors, such as long- and short-range radar, as well as information obtained from the infrastructure and other vehicles, such as localisation and speed limits. Saspence dynamically adapts its warning suggestions based on the current deduced situation. The system follows a three-layered approach with (1) sensors providing input data, (2) specific algorithms processing the sensor information, and (3) human-machine-interfaces forwarding the warning messages to the driver. The algorithm processing should be executed within soft real-time bounds since otherwise vehicle safety cannot be ensured. This project, however, is still defining functional requirements.

The content adaptation of in-vehicle entertainment systems can be adopted from more general approaches, such as [21], which uses a decision engine to adapt content to fit the current capabilities of the device and network. This engine considers the user's preferences and devices' capabilities, provided in the format of the W3C Composite Capabilities/Preference Profile [19]. Based on the current network conditions and device capabilities, the content is transformed accordingly, *e.g.,* reduction of colour, scale of images, and use of various transcoding mechanisms.

Driver information systems not only handle adaptation of multimedia content, but also support software adaptation. Future use-cases for these systems require the dynamic integration of software into running systems. Application developers for these systems can leverage automotive middleware systems, such as AUTOSAR [27] and OSGI [31], that encapsulate the heterogeneity of computing platforms and communication protocols. These systems support the dynamic integration and reconfiguration of software.

For example, DynamicCon builds upon OSA+ [36], which is a scalable middleware for distributed real-time and embedded systems that support the dynamic deletion, addition, and replacement of services. It minimises the "blackout time" of an application by allowing state transfer while the old service is still running. Their approach assumes that the software being integrated or updated is locally available, however, and hence they do not consider fully dynamic adaptation as required by this class of systems. Its Kiviat diagram is illustrated in Figure 8.

The OSGI platform [31] provides a Java-based service platform that supports the remote installation, update, and lifecycle management of Java based applications. In this sense, the platform supports the fully dynamic adaptation of application components. The OSGI platform, however, only supports the adaptation of Java-based applications and hence does not support the integration or reconfiguration of heterogeneous software modules.

Fully dynamic adaptation is also supported by application and data synchronisation systems that support the remote integration and update of existing software and data. Platform-dependent approaches, such as Active Sync [25] and LDAP content synchronisation [30], support the synchronisation of information between devices. These appro-

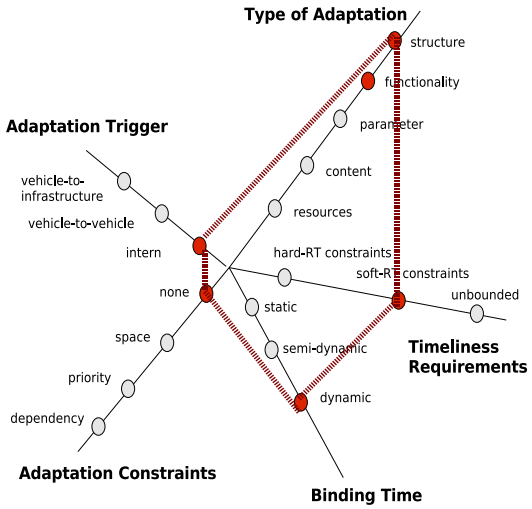

Figure 8: Kiviat Diagram of OSA+

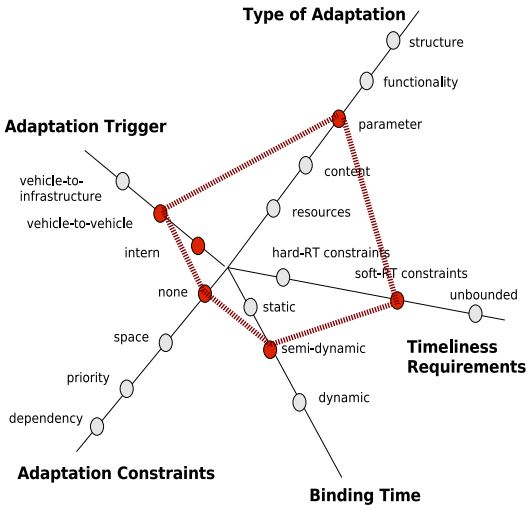

Figure 9: Kiviat Diagram of Auto21

aches, however, only support data synchronisation, *e.g.*, Active Sync synchronises personal information (PIM) and LDAP maintains a copy of a fragment of the LDAP Directory Information Tree.

SyncML [32] is a platform-independent information synchronisation standard that supports the integration and update of data, as well as software applications and firmware. It is based on XML protocols and provides multiple synchronisation modes, *e.g.*, client or server-side synchronisation and two-way synchronisation.

Current approaches for synchronising data and software binaries, however, lack support for time-bounded adaptation and do not address other adaptation constraints, such as memory, module inter-dependencies, and priorities.

6.3 Cooperative Driving Systems

Many approaches for cooperative driving are part of the Auto21 project that investigates fully autonomous vehicles [26]. The authors propose a three-layered architecture to realise decentralised coordination of (autonomous) vehicles [16]. The lowest layer senses the state of a vehicle and is responsible for a vehicle's behaviour. The management and traffic control layer determine the movement of each vehicle under the cooperative driving constraints.

Adaptation is executed on the longitudinal and change lane actuators, based on information obtained from the upper layers. The longitudinal actuator uses either distance or time-based information to change a vehicle's speed, whereas the change lane actuator follows a pre-defined lane function. The behaviour of both actuators is pre-determined and consists of static rules. The overall adaptation is executed within soft real-time constraints. Due to adaptation being related to parameter changes, it does not account for any constraints, such as memory or module interdependencies (*c.f.*, Figure 9).

Adaptive cruise control systems can be realised by distance policies [17]. In this approach, vehicles move within a safe distance to the vehicle in front of them to ensure that no collisions occur in case the vehicle in front brakes suddenly. This distance must be continuously adapted to reflect current environmental conditions, *e.g.*, icy roads, vehicle dynamics, and safety considerations.

Adaptation in an adaptive cruise control system can be executed by the vehicle control system, which is divided into two parts: the upper level calculates the desired control effort based on the current conditions, whereas the lower level computes the corresponding throttle commands. It is based on pre-defined commands and is executed with an effort of minimising the actual duration. The described approach is local and decentralised, *i.e.* no information from other vehicles or the environment is considered. Likewise, the adaptation itself is executed without considering any constraints.

6.4 Vehicular Sharing Systems

Vehicles equipped with storage capabilities can act as a store-and-forward mobile router for data dissemination, such as traffic information or audio and video data [7]. For example, next-generation vehicle entertainment systems assume video or audio files can be stored among several vehicles [14]. Depending on their current location, only a subset of these files are accessible for a vehicle and hence a policy framework is needed that predicts the availability of a file and the predicted time after which the file will be available.

The framework is dynamically adapted to the number of vehicles and the available files inside a cell. This information is continuously monitored and broadcasted to the decision making process that resides locally on each vehicle. To the best of our knowledge, however, the adaptation process does not take space/time constraints into account.

Within a geographic area, vehicles equipped with processing power can form an *ad hoc* grid computer that can autonomously solve distributed traffic flow control problems, such as lane merging. An example of such a system is VGrid [8], which is a grid computer and is realised by a four-layer protocol architecture in which the grid computing interface serves as a bridge between the applications and the network layers. This interface is responsible for the allocation of tasks based on the current network topology, *e.g.*, a critical task is replicated to a number of vehicles. A resource management layer controls the access to local resources on the vehicles. Task allocation takes computational time into account since nodes can move out of range before their com-

putations are finished.

VMesh is a vehicular wireless mesh network that serves as a dynamic sensor network, *e.g.*, for monitoring and collecting vehicular emissions [7]. The Kiviat diagram for this approach is shown in Figure 10. A key design challenge in

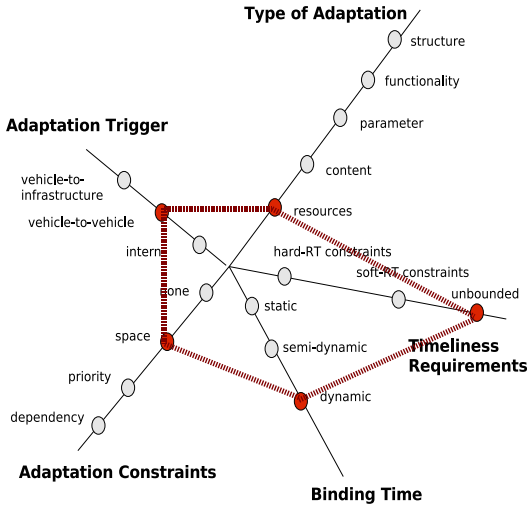

Figure 10: Kiviat Diagram of VMesh

VMesh is determining the penetration rate, *i.e.*, the number of vehicles that are part of the network. This rate should adapt dynamically to the number of vehicles available. The author's approach considers memory constraints at nodes to achieve a desired data throughput rate. It is unclear, however, whether the adaptation itself supports timeliness, *i.e.*, is executed within a specific time bound.

6.5 Discussion

Earlier in this section we classified existing research projects using the taxonomy from Section 5 and visualised their characteristics with Kiviat diagrams. By comparing these diagrams to the diagrams of the four classes of automotive systems depicted in Section 5.2, key gaps in existing approaches can be identified.

First, as we can see in Figure 2, dynamic adaptation in vehicle-centric systems is restricted to at most the dynamic choice of a suitable configuration at runtime. Since hard real-time systems must ensure the safety and consistency of their execution behaviour at all time, the dynamic insertion and reconfiguration of software modules is traditionally not allowed. Existing approaches, such as MARS and DySCAS (*c.f.*, Section 6.1), seem suitable to provide the required runtime adaptation as they replace software and include new devices and software into vehicle electronic systems at runtime.

Driver information systems (*c.f.*, Section 6.2) require not only adaptation of multimedia content, but they also need support for software adaptation, *e.g.*, dynamic integration of software into a running system. Developers of driver information systems can leverage TrafficView and OSA+, but these approaches assume that the software to integrate or to update is locally available. Saspence and OSGI offer full support for dynamic adaptation and SyncML provides data and software binaries integration, but they do not adapt or update software in time bounds. Moreover, these solutions do not account for time and space constraints as required by driver information systems.

Similarly, cooperative driving systems need time-bounded coordination between vehicles. Existing approaches, such as the Auto21 project [26], provide safety critical coordination. They do not, however, support stringent constraints that the executing platform could have on QoS, like the available memory space.

Finally, vehicular sharing systems monitor the space storage and computational power of the vehicles before distributing applications and data between vehicles (*c.f.*, Section 4.4). They do not, however, provide time-bounded dynamic allocation of resources and membership management.

7. CONCLUDING REMARKS

This paper analysed four automotive system classes and identified their software adaptation requirements, including degradation strategies in vehicle-centric systems and dynamically inserting components into software applications. We also defined a taxonomy to characterise various dimensions of dynamic adaptation, including binding time, adaptation type, timeliness requirements, adaptation trigger, and adaptation constraints. We used Kiviat diagrams [20] to represent this taxonomy and compared existing research projects of automotive system software using these diagrams and categorised existing research projects in the automotive domain in terms of this taxonomy. Our analysis showed that time-bounded synchronisation of applications and data is a key requirement for next-generation automotive systems that is not adequately covered by existing work.

To address the gaps in existing work, research is needed on techniques, tools, and platforms for composing automotive software under time constraints. For example, when deciding which modules to integrate or upgrade into vehicle, time and space constraints should be considered. Our future work is therefore defining a domain-specific modelling language and a QoS-enabled middleware execution platform that enables time-bounded composition of automotive system software.

Acknowledgements

The authors would like to thank Shane Brennan and Ashley Sterrit for their valuable input regarding this paper. This work is funded by Science Foundation Ireland under the Research Frontiers Program.

8. REFERENCES

[1] M. Aksit and Z. Choukair. Dynamic, adaptive and reconfigurable systems overview and prospective vision. In *ICDCSW '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*, 2003.

[2] M. Alonso, P. Garayo, and L. Herran. Functional requirements of the SASPENCE system. In *Road Safety on Four Continents*, 2006.

[3] R. Anthony and C. Ekeling. Policy-driven self-management for an automotive middleware. In *PBAC '07: First International Workshop on Policy-Based Autonomic Computing*, 2007.

[4] F. Bai and H. Krishnan. Reliability analysis of dsrc wireless communication for vehicle safety applications.

In *ITS '06: Proceedings of the Ninth IEEE Intelligent Transportation Systems Conference*, 2006.

[5] S. Banachowski and S. Brandt. Toward a taxonomy of time-constrained applications. In *RTSS '03: Proceedings of the 24th IEEE Real-Time Systems Symposium, Work in Progress.*

[6] M. Broy. Challenges in automotive software engineering. In *ICSE '06: Proceeding of the 28th international conference on Software engineering*, 2006.

[7] H. Chang, H. Du, J. Anda, C. Chuah, D. Ghosal, and H. Zhang. Enabling energy demand response with vehicular mesh networks. In *MWCN ' 04: International Conference on Mobile and Wireless Communication Networks*, 2004.

[8] A. Chen, B. Khorashadi, C. Chuah, D. Ghosal, and M. Zhang. Smoothing vehicular traffic flow using vehicular-based ad hoc networking and computing grid (vgrid). In *ITSC '06: Proceedings of the IEEE Intelligent Transportation Systems Conference*, 2006.

[9] G. de Boer and P. Vogel. Connecting the vehicle with the environment - trends and challenges. In *Dagstuhl Seminar Proceedings: Mobile Computing and Ambient Intelligence: The Challenge of Multimedia*, 2005.

[10] M. Krug et. al. Towards an architecture for safety related fault tolerant systems in vehicles. *ESREL '97: Safety and Reliablity Conference*, 1997.

[11] L. Fuentes and D. Jimenez. An ambient intelligent language for dynamic adaptation. In *OT4AmI '06; Proceedings of the Workshop on Object Technology for Ambient Intelligence and Pervasive Computing*, 2006.

[12] E. Gamma, R. Helm, R. Johnson, and R. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1995.

[13] S. George, D. Evans, and L. Davidson. A biologically inspired programming model for self-healing systems. In *WOSS '02: Proceedings of the first workshop on Self-healing systems*, 2002.

[14] S. Ghandeharizadeh and B. Krishnamachari. C2p2: A peer-to-peer network for on-demand automobile information services. In *15th International Workshop on Database and Expert Systems Applications*, 2004.

[15] K. Grimm. Software technology in an automotive company: major challenges. In *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*, 2003.

[16] S. Halle, J. Laumonier, and B. Chaib-Draa. A decentralised approach to collaborative driving coordination. In *ITS '04: Proceedings of the Seventh IEEE Conference on Intelligent Transportation Systems (ITS)*, 2004.

[17] X. Huppe, J. de Lafontaine, M. Beauregard, and F. Michaud. Guidance and control of a platoon of vehicles adapted to chaning environment conditions. In *SMC '03: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 2003.

[18] J. Karjalainen. A classification scheme for embedded control systems. In *IECON '88: 14th Annual Conference of Industrial Electronics Society*, 1998.

[19] G. Klyne, F. Reynolds, C. Woodrow, H. Ohto, J. Hjelm, M. Butler, and L. Tran. Composite capabilities/preference profiles: Structure and vocabularies. Technical report, W3C Ubiquitous Web Application Working Group, 2004.

[20] K. W. Kolence. The software empiricist. *ACM SIGMETRICS Performance Evaluation Review*, 2(2), June 1973.

[21] W. Lum and F. Lau. A context-aware decision engine for content adaptation. *IEEE Pervasive Computing*, 1(3):41–49, 2002.

[22] J. Luo and J. Hubaux. A survey of inter-vehicle communication. Technical Report IC/2004/24, School of computer and Communication Sciences, EPEL, 2004.

[23] P. McKinley, S. Sadjadi, E. Kasten, and B. Cheng. Composing adaptive software. *Computer*, 37(7):56–64, 2004.

[24] T. Nadeem, S. Dashtinezhad, C. Liao, and L. Iftode. Trafficview: traffic data dissemination using car-to-car communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 8(3):6–19, 2004.

[25] ActiveSync [Online]. http://www.microsoft.com/windowsmobile/activesync.

[26] Auto21 [Online]. http://www.auto21.ca.

[27] Autosar [Online]. http://www.autosar.org.

[28] Driver Assistance Applications [Online]. http://www.itsoverview.its.dot.gov.

[29] GSM Association [Online]. www.gsm.org.

[30] LDAP Content Synchronisation [Online]. http://www.openldap.org/doc/admin22/syncrepl.html.

[31] OSGI [Online]. http://www.osgi.org/.

[32] SyncML [Online]. http://www.openmobilealliance.org/tech/affiliates/syncml.

[33] N. Ravi, S. Smaldone, L. Iftode, and M. Gerla. Lane reservation for highways (position paper). In *ITSC '07: Proceedings of the 10th International IEEE Conference on Intelligent Transportation Systems*, 2007.

[34] B. Ravindran and L. Welch. A taxonomy of real-time systems. Technical report, The University of Texas at Arlington,, 1997.

[35] J. Schaeuffele and T. Zurawka. *Automotive Software Engineering.* SAE International, 2005.

[36] E. Schneider, F. Picioroagă, and U. Brinkschulte. Dynamic reconfiguration through OSA+, a real-time middleware. In *DSM '04: Proceedings of the 1st international doctoral symposium on Middleware*, 2004.

[37] M. Trapp, R. Adler, M. Foerster, and J. Junger. Runtime adaptation in safety-critical automotive systems. In *Software Engineering*, 2007.