Agent Based Approach to Distributed Real-Time Systems Development

Paulo G. de A. Urbano

Fax: +49 711 685 7302
Phone: +49 711 685 7321
Email: urbano@ias.uni-stuttgart.de
Institute of Industrial Automation and Software Engineering
University of Stuttgart
Pfaffenwaldring 47, D-70550 Stuttgart – Germany

Abstract. The development of real-time systems is traditionally characterised by the usage of *a priori* design techniques which can guarantee that a set of tasks will meet their deadlines. This development methodology requires that the application's task set and the characteristics of each task – execution period, worst case execution time, etc. – are previously known. Although leading to proven systems, the adaptability to changes in the system's architecture is quite low, due to the static nature of the design. In the area of distributed real-time systems, complex applications are arising where the need for adaptation and flexibility can not be ignored. The concept of agent based software development is proposed as suitable to cope with the complexity of such systems, providing intrinsic characteristics of autonomy and self-regulation. As example, an automotive electronic control system is presented, as well as the vision of an agent based approach to the scenario.

1 Introduction

The development of real-time systems has always been a challenge due to the necessity of guaranteed response times. As definition, [1] real-time systems are the ones in which the correctness of the system depends on the logical results as well as on the time at which the results are produced. Traditionally, real-time applications are developed to control single purpose processes and are realised as isolated and self-contained units. In this context the number and characteristics of the tasks to be performed by an application are well known beforehand. Schedulability analysis can be made in order to guarantee that no deadline would be missed in run time by the specific configuration of hardware and software used in the application.

New applications on the area of distributed real-time computing have added a great deal of complexity to the systems, specially requirements on the dynamic adaptation to changing conditions. The development of real-time systems has been following the trend of distribution, mainly because a considerable amount of real-time applications is inherently distributed and dynamic. As example, we can name the network of electronic components in today's automobiles. Applications on this context have require-

ments on flexibility and self-regulation that are not possibly addressed by the traditional development process explained above. If a failure occurs in one electronic component engaged in an activity that requires co-operation with other components, the outcome of such activity is endangered. Due to their fixed relation and lack of self-regulation, the components could not autonomously modify their run time behaviour in order overcome the problem. One centralised monitoring unit could manage all the elements, but the approach of a decentralised decision architecture has the advantage of no single point of failure and reduced flow of data and commands, due to locally made decisions. The actual development level of micro-controller electronics, concerning processing power, allows the vision of self-regulating electronic components on the context of automotive applications.

The concept of autonomous and self-regulated processes have for long time been addressed in the computer science research area of agents. Arguably [3], agent based software engineering is the most promising conceptional tool available to develop complex systems. In the case of distributed real-time systems, the complexity arises from conflicting issues [4] to be addressed, like predictability and the necessity of flexibility to handle non-deterministic and non-trivial environments, faults and system evolution. This paper claims the principle suitability of the agent approach to real-time distributed systems and presents a vision of how this approach could be realised

The remainder of this paper is organised as follows: in Section 2 the basics concepts of distributed real-time development are presented, stressing the characteristics of the systems produced with the use of actual methodologies. In Section 3 application scenarios in the context of automotive industry are presented, in special the Autonomous Cruise Control (ACC) system. In Section 4 a possible agent approach to the application scenarios development is outlined. Conclusions and future work possibilities are addressed in Section 5.

2 On the basics

To support the argument that the agent based software engineering approach is suitable to the context of distributed real-time (DRT) systems, a description of these systems and their characteristics is made. Following, the traditional methodology of development applied to DRT systems along with the features of the resulting software are explained. Examples of applications that pose new requirements not fulfilled by the referred development methodology are then given.

2.1 Distributed real-time systems

A distributed real-time system is an integrated system composed of a set of dedicated hardware that monitors real-world processes, acts and reacts on events respecting time requirements. The elements of such system are inherently concurrent, and the need of synchronisation arises when co-operation is required. The co-ordination of action between the elements is achieved using a shared resource such as a communication channel in which the elements exchange data and messages. Applications like networked electronic components in an automobile, flexible manufacturing equipment

operation, intelligent highway systems are examples of distributed systems with real-time requirements. A graphical representation of such systems is given in Figure 1.

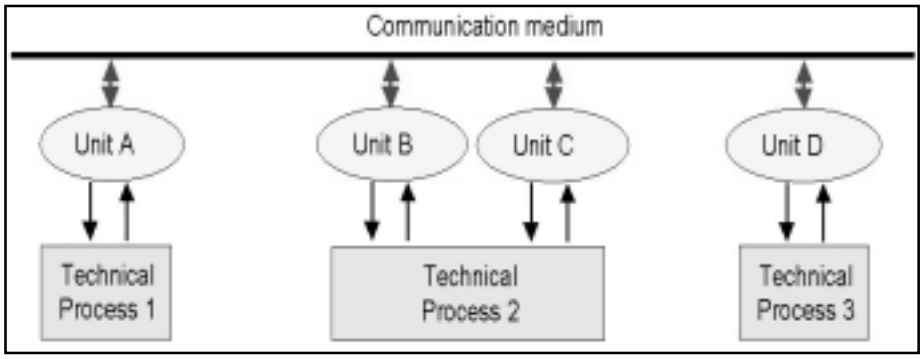


Fig. 1. General distributed real-time system

The nature of distribution of the system comes due to the fact that many applications in the context of real-time systems have to gather information from the environment and influence it in physically separated locations. As illustrated above, these systems are generally divided in controlling system and controlled system [5]. The controlled system can be viewed as the environment to which the controlling system is related. The power engine of actual cars, for example, is controlled by a micro-controller running a specific software. In this case, the micro-controller is the controlling system and the power engine the controlled one. The communication medium in the automotive context is represented by a bus system (e.g. CAN bus) over which the different micro-controllers inside the car can exchange information.

The real-time characteristics of low-level tasks - as the ones responsible for direct interaction with the environment – are tightly related to the timing characteristics of the controlled system. The process of sensing, for example, should be realised often enough to perceive the variations on the environment relevant to the controlling system, giving it an accurate picture of what is happening.

The periodicity is not always a must, as many events may occur in an asynchronous and unexpected manner. The perception of and reaction to asynchronous events is also regulated by timing requirements, in order that the reaction to these events comes also in real-time.

In a co-operative environment the timing requirements can also be interrelated, in a way that a series of events have to be performed before a deadline. These “end-to-end” timing requirements arise in applications where common goals should be reached through the combined work from the distributed system elements. A new level of complexity is reached is this situation, as the co-ordination of the individual elements have to respect a general deadline in order to produce a usable result. This is, nonetheless, unavoidable in dynamical applications with characteristics of high distribution.

2.2 Development processes

Traditionally real-time systems in general (and, in extension, DRT) have been developed using *a priori* knowledge of detailed characteristics of the tasks subjected to execution. The amount of tasks and their timing properties - as worst execution time and activation period - should be known beforehand in order that an static analysis and scheduling could be made. This procedure is successfully used in applications with a relatively isolated and centralised manner of execution. The number of considered external events is made small in order to simplify the analysis process, procedure that imply certain assumptions on the behaviour of the environment controlled by the real-time system. From this basis, the analysis should indicate if a schedule exist to the set of tasks under the made assumptions. This schedule should - theoretically - ensure that the tasks are executed in a way to fulfil the timing requirements. Even if the static analysis is successful and a schedule for the task set is found (and validated as correct), a deviation on the assumed behaviour of the environment could lead to undesired behaviour, like missed deadlines during the execution of tasks. In a context where the variation in the environment are relatively small, this risk could be neglected. But in the applications addressed in this paper, that could not be done.

2.3 New requirements and issues raised

New applications, as the one addressed in Section 3, featuring large number of interacting elements pose new requirements of flexibility and adaptation to DRT systems not present before.

The distributed system could now be formed by a unknown number of co-operating elements that engage in negotiation to achieve a common goal. The fixed nature of systems developed with the static analysis of tasks bring a clear idea of the system's runtime behaviour, but also makes them less flexible to, for example, run-time adaptation. The new requirements increase the complexity of the system, in a way that new issues have to be taken into account, as new elements being added or removed from the system or conflicting goals of different operating elements.

The actual development level of micro-controller technology – in what cost, footprint and processing power are concern - enables the formerly passive elements to have also software running inside them. That strengths the characteristic of distribution, as not only the functionality is distributed (reading input, processing, writing output) but also the control decisions could be divided between the elements.

3 Scenario presentation

3.1 Automotive systems

Automotive systems are addressed here in two application levels: inner systems, as micro-controller networks; and outer systems, regarding the automobile as the composing unit of the distributed system.

The network of micro-controllers is a reality in modern cars. The functionality of systems as different as torque, stability and motor control should exchange information and co-operate in order to achieve their individual and common goals. For example, when driving an icy road the stability system has to co-operate with the motor control, in order that it applies more power in the wheels and also with the torque system, ensuring that the extra power is divided among the driving wheels in the best proportion to avoid loss of control.

The second kind of applications, that will be taken in the next section as an example, models the automobile itself as a node in the distributed system. There are a number of types of applications, referenced in the literature as intelligent highway vehicle systems (IHVS). An IHVS system would perform tasks that are typically done by human operators and use advanced technologies from various fields such as image processing, computer vision, intelligent controls, and artificial intelligence (AI). The IHVS systems are classified into four categories [6]:

1. advanced traffic management system (ATMS);
2. advanced driver information system (ADIS);
3. freight and fleet control system (FFCS);
4. automated vehicle control system (AVCS).

The ATMS systems perform tasks such as surveillance, control, and management of freeway and arterial networks. Such applications include intersection traffic light control and congestion and incident management. The ADIS systems are responsible for such tasks as origin–destination calculation and motorist advisories. They provide information such as efficient alternatives for reaching destinations based on time and road conditions. The FFCS systems manage cargo and freight traffic. Since cargo and freight vehicles are massive in number and size, efficient systems are required to ease fuel consumption, congestion, and road wear and tear caused by such transports. The AVCS systems include vehicle platooning, obstacle avoidance, and autonomous vehicle guidance. The system concentrates on intelligent guidance systems for vehicles. They are capable of either driving the vehicle in a fully automatic manner or giving the human driver useful advice. Such systems would allow drivers to operate at high speeds and simultaneously reduce the probability of having accidents or collisions. The example given in the next section can be classified as an AVCS system.

3.2 Autonomous Cruise Control

An Autonomous Cruise Control (ACC) application [2] installed in a automobile receives information from the highway and from distance sensors, adapting the vehicle's speed to the permitted limits and also ensuring a safe distance to the other cars nearby. A schematic representation of the system is presented in Figure 2.

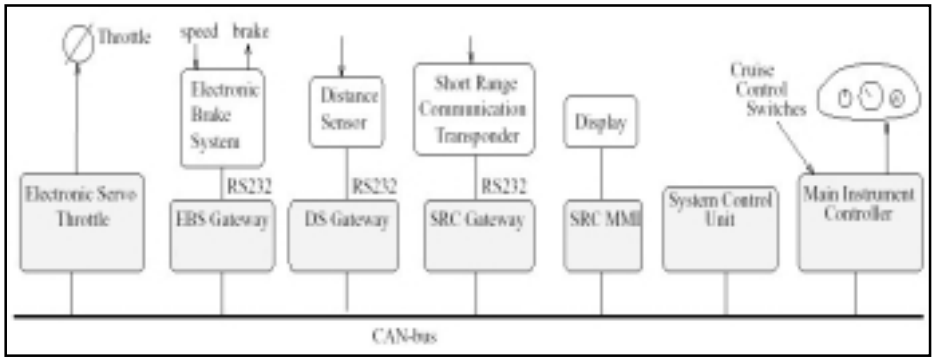


Fig. 2. Autonomous Cruise Control

The approach used in this implementation models each automobile as an isolated unit, with a low degree of interaction between the automobiles. Each one of them has individual goals (maintain speed and distance to the car ahead) but do not co-operate in possible common goals, as optimisation of flux in the highways, exchange of information about the conditions of the road, emergency alerts, etc.

4 Agent approach

An agent based approach to the ACC application models each automobile as an interacting element. Each car does not only receive information from the distance sensor, but also from its neighbours. In such way, the control of the speed and distance have better means to establish when and in which extend the automobile has to be controlled. The level of autonomy of the ACC application can vary from a advise to the driver to a direct intervention in the automobile control in emergency situations. A car with a sleeping driver can be brought to a lower speed automatically, for example. The big advantage of a co-operative is that the neighbours automobiles can be made aware of the problem and increase their distance to the potentially dangerous vehicle.

Innovative scenarios can also be imagined. While modelling each car as an agent with different goals, it is possible to give high priority of access to a free lane in a highway to police cars, ambulances, fire fighter's trucks, etc. The exchange of information between the agents would advise the driver of such situations far before the sirens could be heard or the flash lights seen. Traffic jams could be automatically identified by the automobiles stuck in it and their position forwarded to other vehicles or appropriate service systems located in the road side.

The agent based approach to the ACC application, in which information about the automobile itself and its surroundings is exchanged between vehicles and fixed stations located in the highway can be regarded as the basis of a large information system that has the potential of improving the traffic system, as well as providing new possibilities of services to drivers and highway maintenance companies.

5 Conclusions

DRT systems are developed nowadays using static analysis of tasks, a method that provides high degree of security but low level of system flexibility to runtime modifications.

New applications in the area of DRT systems - as the ACC system addressed in Section 3 - have requirements of flexibility, runtime self-regulation and scalability that add an extra level of complexity to the development process. Systems developed with prior knowledge of the tasks' runtime behaviour are intrinsically fixed, characteristic that is not specially necessary while developing applications that should work in a co-operative and highly dynamic environment.

This paper claims the suitability of the agent based software engineering to cope with the complexity levels present in the development of these new applications, and give a short vision of application scenarios in the context of automotive industry.

The ideas presented here are the basis for a ongoing research effort being made in the Institute of Industrial Automation and Software Engineering, University of Stuttgart. Among the next steps in this work there are planned the construction of a model in which a proof of concept implementation of an autonomous co-operative environment in which agent modelled elements communicate and negotiate to reach individual as well as common goals.

6 References

[1] J. A. Stankovic: Real-Time and Embedded Systems. ACM Computing Surveys 28(1): 205-208,1996.

[2] H. Hansson, H. Lawson, M. Stromberg, and S. Larsson. BASEMENT a distributed realtime architecture for vehicle applications. In Proc. IEEE Real-Time Technology and Applications Symposium (RTAS'95), pages 220--229, Chicago, May 1995. IEEE Computer Society Press.

[3] N. R. Jennings, "On Agent-Based Software Engineering", Artificial Intelligence 117 (2000), pp. 277-296, Elsevier Press, April, 2000.

[4] Stankovic, J. Reflective Real-Time Systems. Technical Report 93-56, University of Massachusetts, 1993.

[5] J. A. Stankovic. Real-Time Computing. Byte, pages 155--160, August 1992.

[6] M. Patel and N. Ranganathan. IDUTC: An Intelligent Decision-Making System for Urban Traffic-Control Applications, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, VOL. 50, NO. 3, MAY 2001