KVL AgroTechnology

A PROPOSED SYSTEM ARCHITECTURE TO ENABLE BEHAVIOURAL CONTROL OF AN AUTONOMOUS TRACTOR

Professor Simon Blackmore
PhD candidate Spyros Fountas
Professor Henrik Have

AgroTechnology
The Royal Veterinary and Agricultural University
Agrovej 10
DK-2630 Taastrup
Denmark

Simon.Blackmore@kvl.dk

www.AgroTechnology.kvl.dk

Presented as a plenary paper
Automation Technology for Off-road Equipment
Hyatt Regency hotel, Chicago, USA
July 26-28, 2002
ASAE-CIGR

Blackmore, B. S., Have, H., and Fountas, S. (2002). *A proposed system architecture to enable behavioural control of an autonomous tractor*. Automation Technology for Off-Road Equipment. ed. Q. Zhang. 2950 Niles Road, St. Joseph, MI 49085-9659, USA, ASAE. pp.13-23.

A PROPOSED SYSTEM ARCHITECTURE TO ENABLE BEHAVIOURAL CONTROL OF AN AUTONOMOUS TRACTOR

SIMON BLACKMORE, SPYROS FOUNTAS, HENRIK HAVE

AgroTechnology, The Royal Veterinary and Agricultural University, Denmark
Simon.Blackmore@kvl.dk, spf@kvl.dk, hh@kvl.dk

ABSTRACT

To allow emergent behaviour from an autonomous tractor, a new sophisticated system architecture is proposed. It is fundamentally object oriented with only high-level message passing between the objects that allow the overall system to be developed without the complexity becoming too great to manage. Each of the agents (Coordinator, Supervisor, Mode Changer, Route Plan Generator, Detailed Route Plan Generator, Multiple Object Tracking, Object Classifier and Hardware Abstraction Layer) could be implemented as separate processors or job-computers on a common bus. This paper describes the overall layout and functions of the elements within the system and gives examples of identified instances.

INTRODUCTION

To be able to achieve the desired autonomous tractor behaviours as described in Blackmore 2001 [1] an inherently complex and sophisticated system architecture for control of the vehicle is needed. A subsumption model can be used as some of the more primitive behaviours and processes are subsumed within higher collective ones. Similarly, each of the main behavioural traits can be modelled by a process, which can be programmed on a processor. In this way we can create an object oriented systems architecture that allows us to model the complex behaviours we need but keep the overall system manageable. The agents with the architecture will follow the same definitions of object-oriented software in terms of encapsulation, inheritance and polymorphism. Encapsulation is of most use to us as it denotes a rigid sub-system boundary that we only need to define the interface and function at this time. Inheritance and polymorphism is a little more difficult to achieve in hardware but processing units that can fulfil the basic functions can be modified to suit particular applications. Communication between agents will only be at the highest information level, probably as text messages that can be understood by people as well, to allow easy analysis of the system.

As part of the design specification, safety and reliability must be designed in at the start. The system architecture must have redundant systems built into it to achieve fault tolerance and allow graceful degradation. We consider the definition of both the behaviour and the system architecture to be of paramount importance before constructing the tractor itself. In general terms, this is a description of how a system is constructed from basics and how the components fit together to form the whole system [2] In this paper, we are primarily concerned with the control system description and will not cover the other tractor systems so that this description can remain device independent at this stage.

BACKGROUND

Kortenkamp [3] identified that systems architecture is a set of inter-related components organized to achieve certain goals. Every component of the system has to be fully understood and to the interrelationships among the components have to be defined. Arkin [4] stated that behaviours are actually the answer to the question "What should the autonomous vehicle be able to do?" Rzevski [5] mentions that behaviour of a machine is a particular interaction of the machine with its environment over a period of time, defined by a particular set of inputs from and outputs into the environment over that period. In general terms, at every point in time, the vehicle is faced with a variety of feasible next states to which the machine could move, and it aims to find the transition that is likely to provide the maximum long-term benefit. The maximum benefit may be expressed in a variety of ways- like the minimum risk of failure, the shortest route to a destination and the maximum utilisation factor of a given machine.

This proposed systems architecture has been designed to accommodate the behaviours already defined. As this behaviour-based system consists of a number of different behaviours each of the processes must be configured in such a way as to allow the desired behaviour to emerge.

Table 1 lists the identified behaviours, their associated codes, whether the behaviour core is embedded in an agent or not, a short description of the proposed behaviour and what other behaviours it subsumes.

Table 1. Descriptions of behaviours

Behaviours	Code	Obj	Description	Subsumes
Explore	Expl		A behaviour that extracts information from the unknown local environment to populate the GIS.	DRP
Implement task	ImpT	*	A behaviour that is executed by the attached implement whilst carrying out the assigned task.	DRP (to target)
Refuelling	Ref		A specialised form of navigation back to a base station	Nav (to base)
Navigation	Nav		The process of moving safely to a required position at a given time	RtPl
Route Planning	RtPl	*	The static process (once only) that analyses all the a priori information to determine the waypoints of a route to the destination.	DRP
Detailed Route Planning	DRP	*	The dynamic process of identifying the best route to the next waypoint. (being modified by information from OTrk)	OTrk
Object tracking	OTrk	*	The dynamic process of tracking the closest object to the tractor	
Watching and waiting	WW		The tractor is doing nothing. The sensors and communications wait for input.	All
Self Check	SChk	*	A process that runs all the time in the background. It checks to see if all the parameters of the tractor are nominal. It keeps a log file and reports abnormalities	
Safety	Safe		Consists of different levels according to the existing situation.	All/none
Request To Start	RSrt		The behaviour from power up of the tractor and before it moves into any other mode. All systems are reset and checked before continuing.	
Request To Stop	RStp		This behaviour indicates that the system is ready for power off. It will be a terminal behaviour requiring that the power be shut off. During this process, the tractor may also put all the mechanical components into a safe neutral position.	

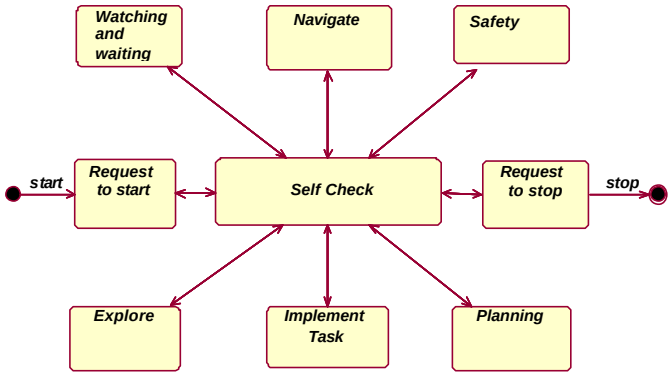

Figure 1. State diagram showing some of the behaviours and their transitions

Figure 1 shows a state diagram for nine behaviours of the tractor. When a request to start is received, initial checks are carried out to ensure that a safe transition to Self Check is viable. All transitions between behaviours will have to pass the self-check first. Self Check is the process where all parameters are checked against what is considered to be nominal. If everything is within limits then the transition can continue. If not, then alternative appropriate action will ensue. This process will be implemented by an Expert System. When a Request to Stop message is received, the tractor will go through a closing down procedure to make sure all the equipment is safe before stopping.

SYSTEMS ARCHITECTURE

In order to achieve these behaviours an object-oriented systems architecture design was developed. To design this architecture we mainly followed the Arkin (1998) assumption that robotic architecture designs refer to a software architecture, rather than hardware side of the system. In other words this architecture was a logical design to support all the behavioural modes. As an object oriented design approach was taken, the physical hardware can also match the logical design.

The benefit with the proposed system architecture is that we can divide the system into elements or agents and we can deal with them independently. Rzevski (1995) also supports this concept. He states that the architecture design usually divides the product into large modules that bring together a set of conceptually related concerns, with relatively narrow interfaces between them. A narrow interface minimizes the interaction between the modules it connects to. The advantage of well-defined interfaces is that if a change has to be made in one module during the design process, no change is needed in other modules unless the interfaces are modified. This interface definition also adheres to the object oriented design philosophy of encapsulation. Each agent will only communicate through high level or natural language messages. Each message will have the capability to pass specific parameters that are closely coupled to the particular message. This is the approach taken by a number of other researchers. (Konoglie etc) The schematic diagram of the proposed system architecture is shown in Figure 2.

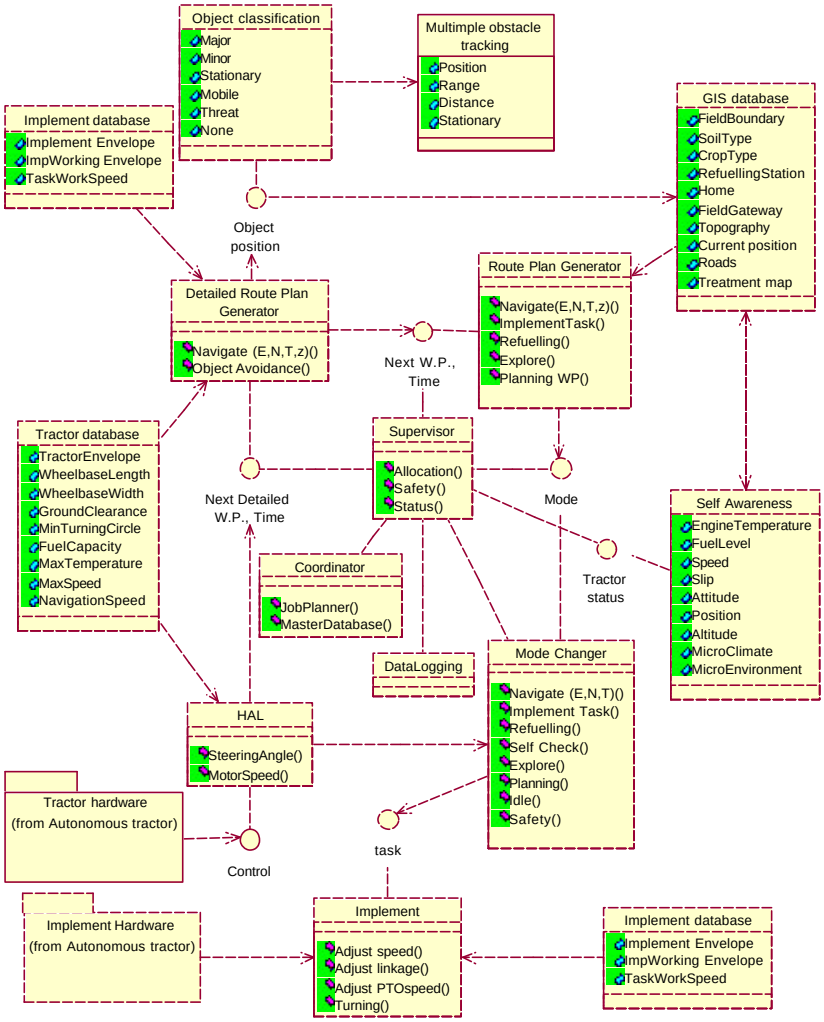

FIGURE 2. The proposed system architecture showing the main data flows

Agents

In the proposed systems architecture design there are two types of agents: processes and databases. Processes execute a coherent set of tasks to achieve an overall goal. This could be pure processing of data, (an example would be the Route Plan Generator) interfaced to sensors (e.g. Self Awareness) or closed loop control (e.g. Hardware Abstraction Layer). The databases store and retrieve data on demand. Each on-board process could be realized as a processor node, with only messages being passed between them, over a common bus. The current object oriented processes and databases are described here. The tractor itself is encapsulated within the Hardware Abstraction Layer (HAL). All closed loop control, which must be tightly coupled and cannot allow any time lag, is embedded within an agent, as there is no guarantee of timeliness of communications via the bus.

Messages

All communications between processes will take place through messages and message parameters. The messages will be at a high-level, or near natural language, for easy debugging and to retain an understanding of the system interactions when it adopts complex behaviour. There will be two types of messages, imperative and responsive. Imperative messages are sent out to instruct or request agents to do something, whereas responsive messages reply with information. We have identified a special class of responsive message, which is one where it will be issued at regular intervals or in certain conditions, such as the giving the current position every second or when an event happens. Table 2 shows some of the messages already identified.

Table 2. Examples of messages used within the proposed system architecture

Message	Type	Description	Parameters	Example
RequestPosition	Imp	What is the current position and bearing?	Easting, Northing, Bearing	RequestPosition(Easting, Northing, Bearing)
PostionNow	Res	Current position	Easting, Northing, Bearing	PostionNow(12300,12300,360)
RequestMode	Imp	Requests current mode of agent(s)	Agent(s)	RequestMode(All)
ChangeMode	Imp	Instruction to agents to change modes	Agent(s), mode	ChangeMode(All,Navigate)
ModeNow	Res	Gives current mode of agent	Agent and mode	ModeNow(HAL,Navigate)
PlanRoute	Imp	Request to plan route clear of obstructions between two points	Start point, Start time, End point, End time, Clearance	PlanRoute(12300,12300,Now, 12340,12340,11:30:15,2)
RoutePlanned	Res	Returns status and route plan	Status, Filename	RoutePlanned(OK,Route1.txt)
ClosestObject	Res	Gives position and information of closest object sensed	Easting,Northing, Status	ClosestObject(12305,12300, Stationary)
NextWayPoint	Res	Gives position, bearing and Estimated Time of Arrival	Easting,Northing, Bearing, ETA	NextWayPoint(12340,12340, 360,11:30:15)
NextDetailedWP	Res	Gives position, bearing and Estimated Time of Arrival	Easting,Northing, ETA	NextDetailedWP(12305,12305, 11:01:00)
Goto	Imp	Move to the indicated position	Easting,Northing, Tolerance, ETA	Goto(12305,12305,1,11:01:00)

This message passing system allows flexibility in the design as new functions or processes are needed, then new messages can be formulated. As they are all in text form, they can be stored in a log file for later analysis, the capability of replaying events and understanding how the system has reacted. Each message can be recorded with a time stamp to assess the temporal effects or conflicts.

Coordinator

The coordinating process will be carried out in the farm office on a PC. (All other processes can be realised on-board the tractor) It is likely to be a set of information screens and optimisation routines that can be used by the farm manager. High-level requests can be made, such as monitor the crop for nitrogen stress in field 10 for the next 2 days or carry out intra-row weeding in field 5 with implement number 2. The coordinating program can then allocate resources, (e.g. which tractors to use) prepare an initial route plan based on the GIS and develop a suggested instruction set for the tractor(s). The manager can then review the proposed itinerary and make adjustments to it before it is then downloaded to the tractor(s). The coordination process will also be in on-demand communication with the tractors and show the manager the current location and status of each

machine through a tractor mimic display. A prototype mimic display is shown in Figure 3. A real-time video link to steerable on-board cameras allows a better understanding of the tractor's environment.

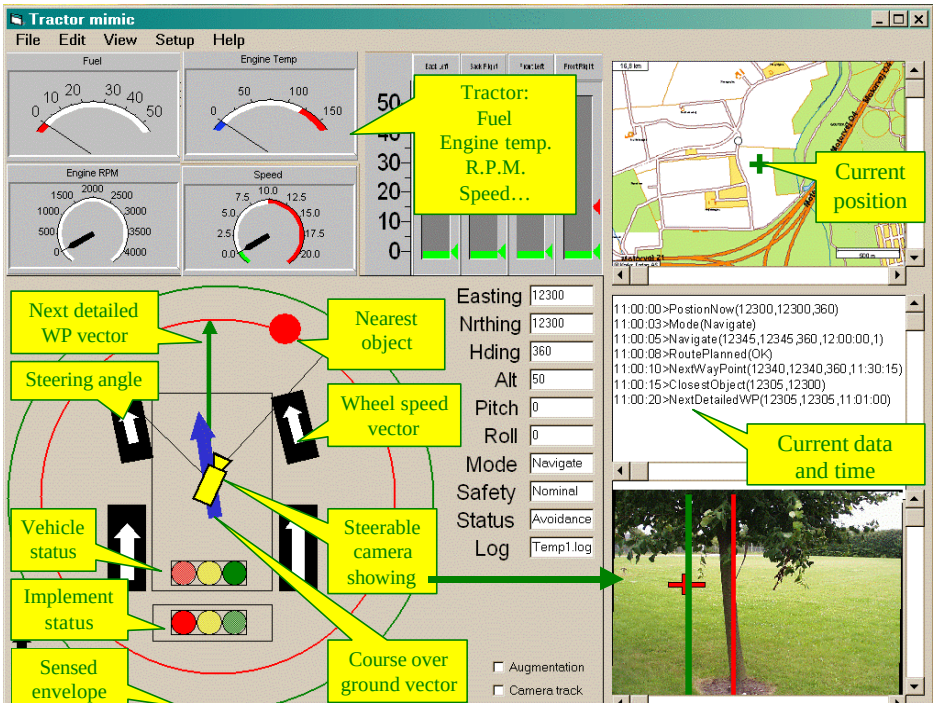

Figure 3. Prototype mimic display of tractor status

Supervisor

The supervising process will be hosted on a laptop computer on the tractor. Firstly, it will relay the appropriate tasks to be executed from the coordinator through a radio modem. Secondly, it will supervise different states of the tractor and keep a transaction log file of all the messages on the bus. Thirdly, it will present the current status of the tractor, as a mimic, on the computer's monitor to allow an operator to interact with it directly.

It is not directly used within the tractor control loop (except for communications with the coordinator as the laptop hosts the radio Ethernet client) but is one of the redundant systems that can supervise the other processes. It will have an expert system that can decide if any of the processes may be malfunctioning, in which case it can then intervene. The expert system can hold a set of sensible parameter values for comparison with what the processes are sending. It can also send simple requests to each process and check that it gets an expected reply. If it receives an unexpected reply then further checks can be made to identify the situation and take suitable actions like reset the process or shut down the whole tractor into one of the safety modes.

Mode Changer

The Mode Changer is a process that can assess the current situation from other processes and command each process to switch into a particular mode that can allow an appropriate overall behaviour. It may receive an imperative command from the coordinator, for the tractor to go to a particular position. It will check all processes to establish it is safe to switch modes and then issue the appropriate command to each process. An example would be where the manager instructed the tractor to navigate to a particular Easting and Northing near by. If each process responded that it was nominal, capable of accepting a new mode, and that the implement was secured ready for transport, then each process could be configured to allow the tractor to navigate to the required position.

Identified modes are: Navigate, Implement task, Refuelling, Self check, Explore, WatchAndWait and Safety.

Another function of the Mode Changer will be to assess the context of the tractor. It will identify pertinent messages on the bus and use an expert system to categorise the current context of the tractor, its task and situation. For example, if the tractor is carrying out a weeding task and fuel is getting low, it would be sensible to wait until the tractor reached the end of the row before leaving for to refuel. The Mode Changer constantly updates the context – in this case: currently weeding, in a row. The Self Awareness agent outputs a message that fuel is getting low and the Mode Changer will not change into Refuel mode until the row has been finished.

There appear to be two classes of finite states for the context: internal and external. Internal states can be assessed from analysing the actual messages on the bus. Internal contexts that have been identified are: Nominal, agent failure, unrecognised message, corrupt message, and parameters outside limits. External states are much more complex and are assessed by interpreting the messages. Current examples are shown in Table 3.

Table 3. Showing examples of external contextual situations

Name	Description
Nominal stationary	Tractor stationary, in one of the predefined behaviours (Route planning, Self Check etc)
Nominal Task	Tractor and implement carrying out predefined task
Navigating	Tractor moving freely, implement stowed
Avoiding	Following obstacle boundary
Threat	Shutting down while tracking approaching object
Assessing	Object sensed within threshold, tractor stopped and watching behaviour of object
Skid	Tractor moving faster than the wheels
Slip	Wheels moving faster than the tractor
Stuck	Wheels moving, stationary tractor
Sink	Reduced clearance under tractor
Tilt	Tractor beyond attitude limits
Weather	Tractor experiencing weather beyond set limits
Theft	Tractor shut down but moving after Threat (!)

Another special set of contextual situations is related to safety, which can arise from both internal and external situations. Safety is one of the most important considerations to take into account when constructing autonomous vehicles. There are many different safety requirements to be considered: Safety requirements for the tractor itself, safety requirements for the people and objects within the vicinity of the tractor, and finally safety requirements for the crops and plants it interacts with.

For the tractor, 6 different safety modes have been identified. These modes will be decided by the Mode Changer and reported continuously to the co-ordinator. They are:

1. Nominal safe operation
   The tractor is operating within normal parameters
2. Safe operation with warnings
   The tractor operating safely, but there are some warnings about abnormalities
3. Partial system shut down – mobile
   The tractor has problems that have partially shut down the system, although it remains mobile
4. Partial system shut down – immobile
   The tractor has problems that have partially shut down the system and made the tractor immobile.
5. Stopped – still communicating
   The system has fully shut down, but it still communicates with the co-ordinator
6. Dead
   The system has fully shut down and there is no communication with the co-ordinator, or the communication system has malfunctioned.

These safety modes can be instigated when failures are anticipated and graceful degradation occurs. Redundant hardware safety systems should also be employed ranging from bump switches being monitored by the Self Awareness agent, to inertia triggered wobble switches in line with the power relays for stop the tractor during physical intervention.

Route Plan Generator (RPG)

The Route Plan Generator can accept a completed route plan from the RPG in the Coordinator (where it knows about multiple tractors) or generate its own when asked to navigate to a certain position (where it knows only about itself). It will generate a series of waypoints that the tractor will follow to arrive at its final destination at the desired time. The waypoints will take into account all the prior knowledge in the GIS, such as gateways, preferred paths and tracks, as well as obstacles such as fences, ditches and public roads. At this stage of the planning it is assumed that the tractor will travel on the straight line between the waypoints, so careful positioning is crucial if the tractor is to stay on a narrow curving track. These series of points will be optimised for efficiency of the task in hand (minimum distance would be one criteria), such as to provide the best route to start at a gateway, go down every row in a field and come back to the gateway, avoiding the known obstacles. This route plan is predetermined before the tractor moves and includes the estimated time of arrival at each waypoint, knowing the start time, end time and distance to be travelled. If the desired arrival time at the target position computes to unrealistic speeds for the tractor then a warning will be given and a more realistic ETA suggested to the operator. The control processes will not allow the tractor to be operated outside predefined limits stored in the tractor database.

Detailed Route Plan Generator (DRPG)

The detailed route plan generator initially calculates a series of detailed waypoints linearly between the waypoints from the RPG at a set distance apart and sends steering and speed messages to the hardware abstraction layer to move towards the next target position. As the tractor gets to within a certain distance from a detailed waypoint, it switches to the next one in the series. This process will continue until it reaches the next waypoint, unless there is an obstruction in the way.

If an object is sensed within a predefined range and bearing of the tractor, then the DRPG will instruct the tractor to stop and wait. If the range and bearing of the obstacle changes, then the DRPG will consider it to be a mobile obstacle and wait for it to move outside the safety range and resume its navigation. If the obstacle moves towards the tractor then it will be seen as a threat by the Context Assessor and change the mode accordingly. If the object appears to be stationary, then its estimated boundary will be entered into the GIS and the tractor will try to circumnavigate the obstruction.

An obstacle will be deemed an obstruction if it is likely to interfere with the movement of the tractor or attached implement. If an obstacle is sensed by the obstacle tracking processes to be within the working safety envelope, (derived from the tractor envelope in the tractor database and the implement envelope from the implement database) a new detailed waypoint will be generated to take the tractor towards the next waypoint but at a safe distance from the obstacle. At this stage, the DRPG does not know the size or the extent of this unforeseen obstacle, so it must rely on the sensors to estimate the distance and plan the next detailed waypoint accordingly and update the GIS for future reference.

Multiple obstacle tracking

This process runs in parallel with the other processes all the time and uses intelligent sensing techniques to search the local environment around the tractor for any objects that may become obstacles. Intelligent sensing uses multiple, redundant, sensors to measure the same characteristic [6]. In this case we are interested in range and bearing to close objects. The sensors should use different physical methods to sense the same thing, which dramatically reduces the possibility that the overall system can be confused of corrupted, which could lead to catastrophic failure. It can estimate the range and bearing (relative to the tractor) of a number of obstacles within its sensing range and reports the closest one. An example is shown in Figure 4.

Figure 4. A small research vehicle with an ultrasonic and laser range finding systems

As the closest object is continually tracked, it is possible to halt the tractor and observe the object to see if it is moving. This will assist in the object avoidance behaviour. Similarly, when a stationary object needs to be navigated around, the range and bearing can be used to allow the tractor to follow the outline of the unknown object at a safe distance.

Object classification

The navigation process described so far has assumed a two-dimensional world, i.e. all obstacles are of infinite height, as is the tractor and implement. The object classification process may be needed to deal with three-dimensional objects interacting with the three dimensional vehicle. If there is a

small obstacle (e.g. a small stone) in the way, can the tractor go over it, or should it go round? This complex issue will need a separate 3 dimensional model of the tractor and implement in the future.

Hardware Abstraction Layer (HAL)

The hardware abstraction layer agent receives the position and time of arrival messages from the DRPG and then implements it on the physical platform. It will use an inverse kinematic model to determine the steering angles and wheel speeds before controlling the actuators. Each actuator will also have a transducer to measure the actuator to ensure accuracy. Closed-loop feedback will be used at a number of levels to ensure reliability, but all loops will be within the HAL agent (apart from the GPS messages) as the stability of closed loop feedback requires fast responses which cannot be guaranteed over the bus. As the tractor is encapsulated within the HAL agent, no process can get direct access the tractor control system unless it goes through the HAL or Self Awareness processors. Multiple nested feedback loops will be used to ensure reliable operation or sensible fault diagnosis. Some of the functions within the Hardware Abstraction Layer are shown in Figure 5.

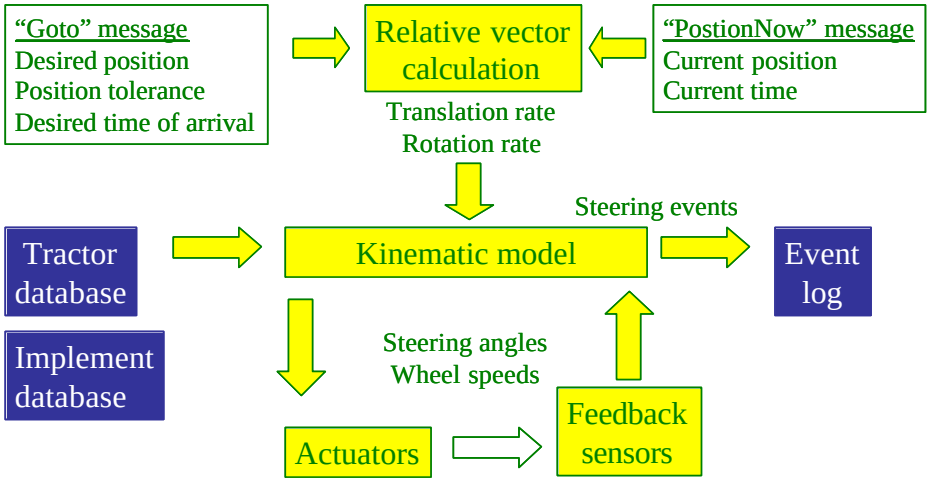

Figure 5. Some of the functions within the HAL

Self Awareness

A wide range of sensors on the tractor supply information about the tractor and implement hardware components such as position, attitude, engine temperature, fuel capacity and implement logistics. A number of other sensors provide information related to local environment such as altitude, air speed and air temperature from a mobile weather station. All of these data are used to update the GIS database and to send messages to the Context Assessment process to decide if there is a need to shut down in adverse weather conditions or unusual situations, like if the tractor turns over or gets stuck in mud.

Implement Task

The implement task process is highly specialised and tightly coupled to the particular activities of the implement. It gets parameters about the implement from the implement database and has control over the whole tractor whilst active. The Implement Database is embedded in the Implement Task agent and can be accessed through the agent interface. When a new implement is bought and connected to the bus, it can identify itself, what it does and what information it needs from the system. It will also allow the database and programs to be copied onto the Coordinator and HAL

agents for future reference and backup purposes. Examples of tasks are mechanical weeding, crop sensing etc.

Tractor Database

This is a database embedded inside the HAL that contains data about tractor dimensions, wheelbase length and width, ground clearance, minimum turning radius, fuel capacity, maximum temperature, maximum speed, navigation speed as well as any other parameters closely coupled to the tractor itself.

Implement Database

The Implement Database contains information about implement dimensions, implement-working dimensions, task working speed as well as any other parameters considering important for the implement task.

GIS database

The geographic information system (GIS) is the main database to store all the spatially related data. The GIS is essential for optimising the field operations for autonomous vehicles [7]. GIS database can contain information from asset surveys or exploration (see above). It provides all the permanent spatial attributes of the field, pertinent to field operations, transient data (where it sensed an object last time), provides attributes of a field that change during the growing season, e.g. crop structure and soil nutrient status and field operations maps. It is envisioned that a standard commercially available GIS will be used for this purpose that will be modified to allow on-line queries.

Expert systems

This system architecture employs many expert systems within it. They range from high level roles in the coordinator, down to hardware specific roles embedded within the agents. To show the subsumption or hierarchical relationships within the architecture, see Figure 6.

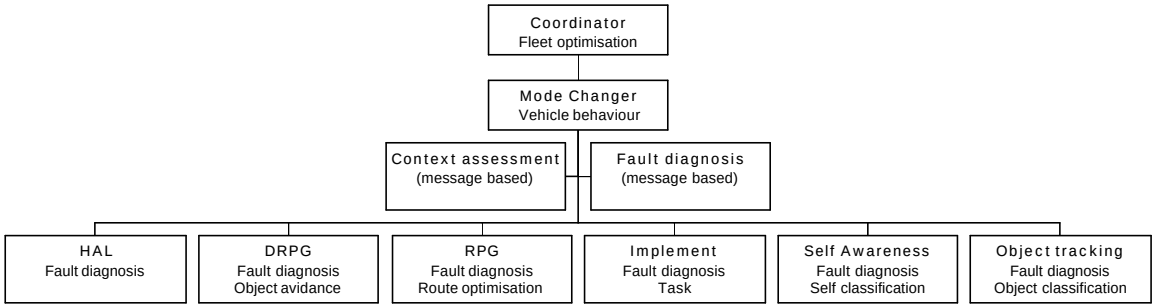

Figure 6. Hierarchical relationships between expert systems

Hierarchical redundancy can be seen in the fault diagnosis but they occur at different scales. The embedded fault diagnosis with each agent knows the specific function required by the agent and can use detailed knowledge to assess suitable functionality. This is in addition to the function itself, such as assessing the type of object in the Object Tracking process or route optimisation in the RPG.

Faults may occur from corruption of messages on the bus or failure of an agent. This should be recognised by the message based fault diagnosis process within the Supervisor. All of the messages are assessed by the Context Assessment process to classify the overall situation of the tractor. The Context Assessment process is embedded within the Mode Changer agent. The overall tractor behaviour mode will be changed by the Mode Changer based on information from all the lower processes. Similarly, the Coordinator will modify the individual tractor behaviours, based on instructions from the manager and the situation of other tractors.

DISCUSSION AND CONCLUSIONS

This architecture has been designed in an attempt to enable the emergence of behavioural traits in an autonomous tractor or other vehicle working in a semi-natural environment. The complexity of this system is huge in comparison to existing systems that attempt to carry out the same task but it should be seen as an attempt to embrace the sophistication needed by the architecture to deal with the complexity of the real world without making sweeping assumptions to simplify the problem that often leads to catastrophic failure of the vehicle and its interaction with its environment. The object oriented and message passing approach along with the tight coupling of agents to tasks allow information to move between agents, thus allowing this information to contribute to a particular behaviour. If new behaviours are needed resulting in the requirement for new processes or information, then they can be added as new agents with little difficulty. Even during development the object orientation allows modification or improvement within the agent without having to modify other system parameters. By taking the message passing up to the information level and keeping to absolute SI units, the interfaces should be relatively stable.

Although this work has been done with a tractor working in an agricultural context in mind, it is equally applicable in simpler more stable, non-natural environments. It can be equally applied to other sectors where behavioural control will be needed.

---

[1] B.S. Blackmore, H. Have and S. Fountas
Autonomous machinery in horticulture: A specification of requirements
Proceedings of the 6th International Symposium on Fruit, Nut and Vegetable Production Engineering conference, Potsdam, Germany, 11-14 September 2001

[2] J. Albus (1991): Outline for a theory of intelligence. IEEE Transactions on Systems, Man and Cypernetics, Vol. 21, No.3, May-june, pp. 473-509.

[3] D. Kortenkamp, R.P. Bonasso and R. Murphy,
Artificial intelligence and mobile robots, edited by
1998, AAAI Press

[4] R.C. Arkin
Behaviour based robotics
1998, MIT Press, USA

[5] **Mechatronics; Designing Intelligent Machines**
Vol. 1 Perception, Cognition and Execution; G Rzevski
Vol. 2 Concepts in Artificial Intelligence; John and Picton. 1995, The Open University

[6] Blackmore B.S. and Steinhauser T.
Intelligent Sensing and Self- Organising Fuzzy Logic Techniques Used in Agricultural Automation
ASAE Paper No. 93-1048 Spokane June 1993 4 pp.

[7] R.Earl, G.Thomas, B.S.Blackmore
The potential role of GIS for autonomous field operations
Computers and Electronics in Agriculture, Elsevier, Special issue; Navigating Agricultural Field Machinery. Ed Gerhard Jahns, Vol.25 Issue 1-2, 1 Jan 2000, pp. 107-120