24th Annual INCOSE International Symposium (IS2015)
Seattle, WA, July 10 – 16, 2015

Model-Based System Patterns for Automated Ground Vehicle Platforms

Troy Peterson
Booz Allen Hamilton
peterson_troy@bah.com

William D. Schindel
ICTT System Sciences
schindel@ictt.com

Abstract. Automated Ground Vehicle (AGV) platforms are proliferating across commercial, military, and consumer applications. Beyond diversity of form and application, AGVs can be manned or unmanned, and exhibit a broad range of automated control, from partial to fully autonomy, making these vehicles strikingly diverse.

This paper reports on application of Pattern-Based Systems Engineering (PBSE) to representation of automated ground vehicle platforms. PBSE is based upon reusable, configurable S*Models conforming to the S*Metamodel, expressed in any modeling language and toolset. The Patterns Challenge Team of the INCOSE/OMG MBSE Initiative has been practicing PBSE across applications, reported in this and other IS2015 papers.

A specialized class of Cyber-Physical Systems, AGVs are subject to intense interest, creating new opportunities, risks, and complexities. To address the diversity and complexity of these systems, the Embedded Intelligence (EI) Pattern, another S*Pattern, is being applied by the team to illustrate its applicability to an AGV Platform Pattern.

Automated Ground Vehicle Platforms

AGVs are rapidly and dramatically increasing in complexity which is changing the way we develop, manage and interact with these systems. A primary driver for this change is the increase in vehicular automation, which offers many benefits to include increased efficiency, safety and improved situational awareness among others. However, it also places significant demands on organizations to ensure rigor and trustworthiness by appropriately improving safety, security and reliability.

The rapid evolution of autonomy in ground vehicles is also driving competition and accelerating innovation in a reinforcing loop. The Boston Consulting Group's 2013 report on the most innovative companies noted that for the first time, there were more automakers than tech companies listed the top 20 (BCG, 2013). One surprise, not noted in the study however, was that every company listed in the top 10 is involved with ground vehicle automation.

The increased demand for automation and innovation brings with it risks which parallel the challenges outlined by the National Science Foundation (NSF) in regard to Cyber Physical Systems (CPS). Challenges which they state are both significant and far-reaching. The NSF defines CPS as "engineered systems that are built from, and depend upon, the seamless integration of computational algorithms and physical components" (NSF, 2014). They are systems which tightly intertwine computational elements with physical entities within aerospace, automotive, energy, healthcare, manufacturing and other sectors.

As an especially complex type of cyber physical system, Automated Ground Vehicle (AGV) platforms call for specialized methods to conceptualize and design for the deep interdependencies inherent within them. This paper will introduce Pattern-Based Systems Engineering (PBSE), the emergence and use of the Embedded Intelligence (EI) Pattern for systems with forms of embedded human or automated intelligence, and highlight some key implications and benefits as applied to AGV Platforms.

Model Based System Patterns

PBSE Overview

As a Model-Based Systems Engineering (MBSE) methodology, Pattern-Based Systems Engineering (PBSE) can address more complex systems, with reduction in modeling effort, using people from a larger community than the "systems expert" group, and producing more consistent and complete models sooner. These dramatic gains are possible because projects using PBSE get a "learning curve jumpstart" from an existing model-based pattern and its previous users, rapidly gaining the advantages of its content, and improving the pattern with what is learned, for future users. The major aspects of PBSE have been defined and practiced for many years across a number of enterprises and domains, including ground systems, but in a limited community. To increase awareness of the PBSE approach, in 2013 INCOSE started a Patterns Challenge Team within the OMG/INCOSE MBSE Initiative (INCOSE Patterns Team, 2014).

The term "pattern" appears repeatedly in the history of design, such as civil architecture (Alexander et al, 1977), software design (Gamma et al, 1993), and systems engineering (Cloutier, 2008). These are all loosely similar in the abstract, in that they refer to regularities that repeat, modulo some variable aspects, across different instances in space or time. However, the PBSE methodology referred to by this paper, based on S*Models and S*Patterns, is distinguished from those cases by the following important differences:

1. S*Patterns are Model-Based: We are referring here to patterns represented by formal system models, and specifically those which are re-usable, configurable models based on the underlying S*Metamodel. By contrast, not all the historical "patterns" noted above are described by MBSE models. It is for this reason that the current INCOSE PBSE Challenge Team effort is a part of the INCOSE/OMG MBSE Initiative.

2. Scope of S*Patterns: We are referring here to patterns which will usually cover entire systems, not just smaller-scale element design patterns within them. For this reason, the typical scope of an S*Pattern applications may be thought of as re-usable, configurable models of whole domains or platform systems—whether formal platform management is already recognized or not. By contrast, most of the historical "patterns" noted above describe smaller, reusable subsystem or component patterns. S*Patterns are similar to architectural frameworks, although they potentially contain more information.

Fundamental to Pattern-Based Systems Engineering is the use of the S*Metamodel (summarized by Figure 1), a relational / object information model. The S*Metamodel is intended to describe the "smallest possible model" necessary for the purposes of performing systems engineering or science. It provides the semantics to describe requirements, designs, and other information such as verification, failure analysis, etc. (Schindel & Peterson, 2013;

Schindel, 2013a; ICTT, 2013; Schindel, 2011c, Schindel, 2005a, 2005b, 2010). A metamodel is a model of other models—a framework or plan governing the models that it describes. These may be represented in SysML™, database tables, or other languages.

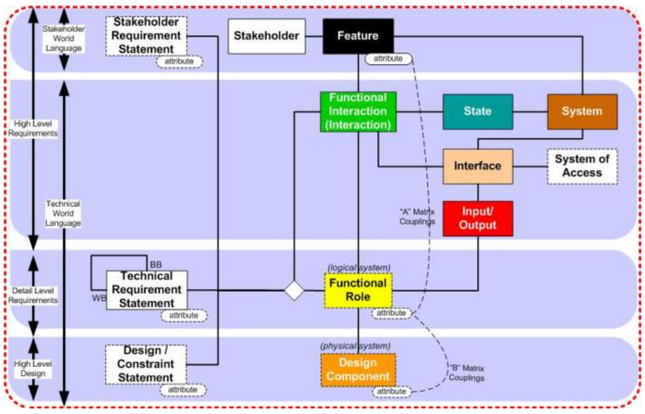
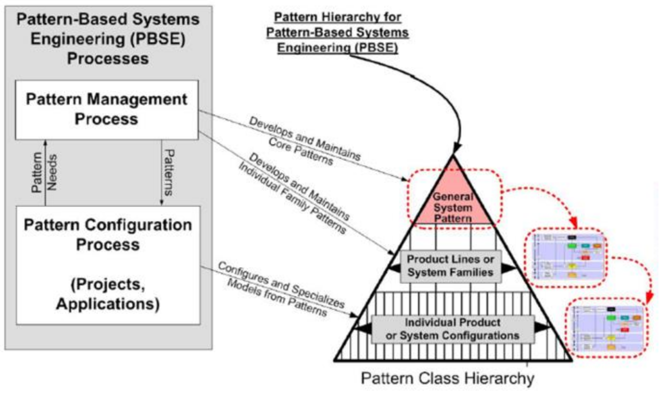


Figure 1: A summary view of the S* metamodel

Specifically, an S*Pattern is a re-usable, configurable S*Model of a family of systems (product line, set, ensemble etc.) as shown in Figure 2 below.



Pattern-Based Systems Engineering (PBSE) Processes
Pattern Management Process
Pattern Needs
Patterns
Pattern Configuration Process
(Projects, Applications)
Pattern Hierarchy for Pattern-Based Systems Engineering (PBSE)
Develops and Maintains Core Patterns
Develops and Maintains Individual Family Patterns
Configures and Specializes Models from Patterns
General System Pattern
Product Lines or System Families
Individual Product or System Configurations
Pattern Class Hierarchy

Figure 2: Pattern Hierarchy and Processes

Over several decades, this approach to Pattern-Based Systems Engineering has been practiced across a range of domains, including carrier grade telecommunications, engines and power systems, automotive and off road heavy equipment, telecommunications, military and aerospace, medical devices, pharmaceutical manufacturing, consumer products, and advanced manufacturing systems. (Schindel and Smith, 2002; Bradley, et al, 2010; Schindel, 2012b).

Engineers in these and many other domains spend resources developing or supporting systems that virtually always include major content from repeating system paradigms at the heart of their business (e.g., core ideas about airplanes, engines, switching systems, etc.). In spite of this, the main paradigm apparent in most enterprises to leverage "what we know" is to build and maintain a staff of experienced technologists, designers, application engineers, managers or other human repositories of knowledge.

The physical sciences are based upon the discovery of regularities (patterns), which we say express laws of nature. Although re-usable content has some history in systems engineering, there is less recognition of a set of "Maxwell's Equations" or "Newton's Laws" expressing the nature of the physical world, as the basis of those systems patterns. If Electrical Engineering and Mechanical Engineering disciplines have physical law at their foundation, can this also be so for Systems Engineering?

In support of that connection, the S*Metamodel is focused on the very physical Interactions that are the basis of all the observed laws of the physical sciences, and which we assert are at the heart of the definition of System (in this methodology) as a collection of interacting components. (Schindel, 2011b, 2013a) The S*Patterns that arise from the explicit representation of physical Interactions re-form the foundation of system representations to align more explicitly with the physical sciences.

A method for managing AGV complexity and types

Aspects of this PBSE approach are illustrated for Automated Ground Vehicles (AGVs) in the following section.

AGV Model-Based Pattern

AGV Diversity – Types/Product Lines

Autonomous ground vehicles are often broadly categorized as manned and unmanned and by their level or type of control (limited, partial or full autonomy) making the broad class automated ground vehicles especially diverse (Figure 3.)

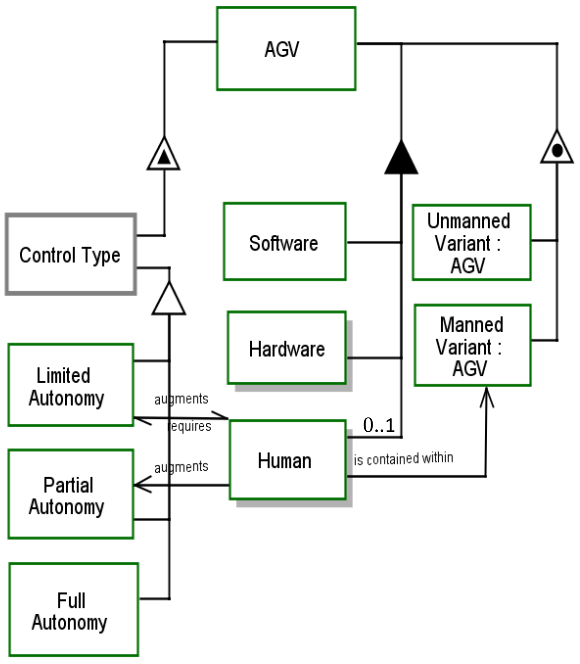


Figure 3: Broad Ground Vehicle Control Type and Manned/Unmanned Classifications

Figure 3 outlines these three types of ground vehicle control as well as manned and unmanned.. Furthermore, those that include partial autonomy can have algorithms and sensors embedded within the AGV or off-board at a central station, embedded within local infrastructure or otherwise (not shown in Figure 3). Regardless of how they are allocated the core logical elements and core function remain the same.

Vehicle Pattern

These broad categorizations of AGVs also span across industrial, military and performance/remote control and commercial applications. As a family of systems AGVs offer significant variation in size, mission, cost and many other attributes. Even as a very small sample set Figure 4 below aptly represents the diversity of Automated Ground Vehicle Platforms.

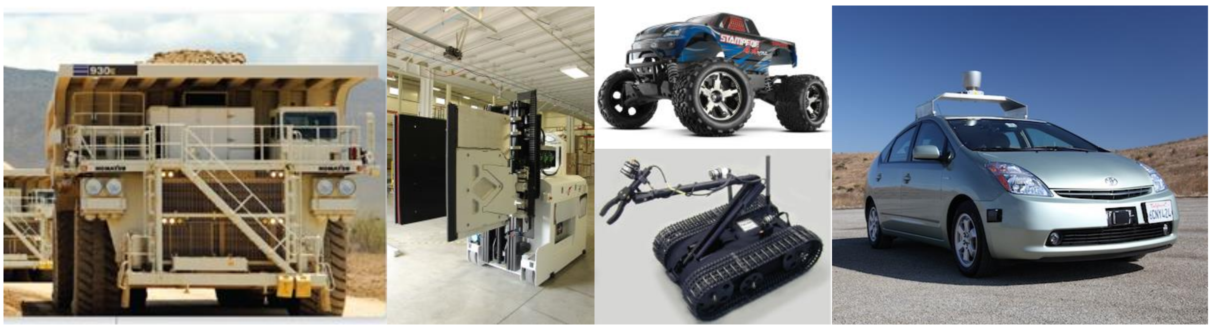


Figure 4: Automated Ground Vehicle Platform Diversity

Given the variety of applications the variability in the high level of design and implementation of such systems is not surprising. The general vehicle pattern represented in Figures 5-10however could represent any one of these diverse sets of platforms. The amount of information and configurability of this model through the application of PBSE permits rapid specialization/configuration through selection of required feature sets shown in Figure 5. Figures 5-10 on the subsequent pages show important features, interfaces and interactions required for autonomous operation. For instance the Remote Autonomous Operation Feature, Remote Access Management Feature, and from the Domain Model the Remote Management, GPS, and Environmental Interfaces.

Of course an elaborated model would be required to eventually arrive at any one of the platforms shown within Figure 3. To retain configurability the modeling effort would follow the approach outline in Figure 2 in a recursive fashion to accurately detail the system architecture of any particular platform family.

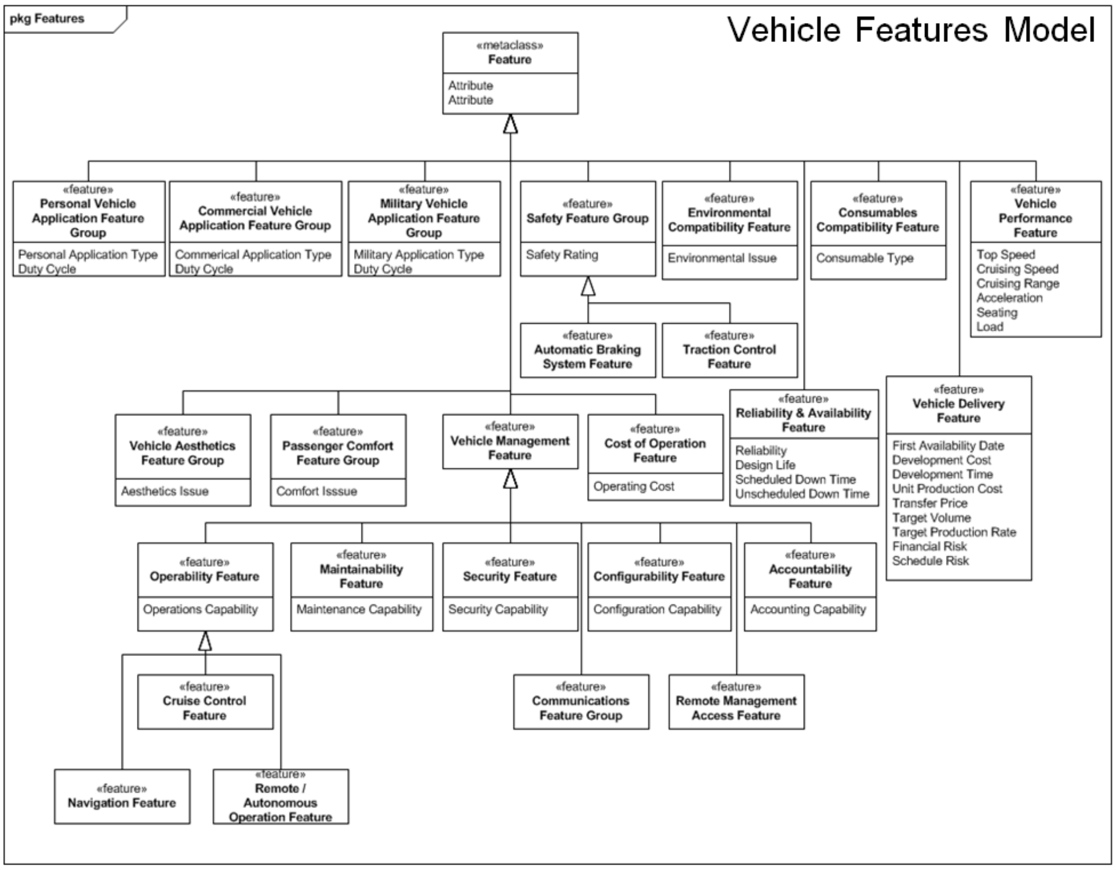

pkg Features
Vehicle Features Model
«metaclass» Feature
Attribute
Attribute
«feature» Personal Vehicle Application Feature Group
Personal Application Type
Duty Cycle
«feature» Commercial Vehicle Application Feature Group
Commerical Application Type
Duty Cycle
«feature» Military Vehicle Application Feature Group
Military Application Type
Duty Cycle
«feature» Safety Feature Group
Safety Rating
«feature» Environmental Compatibility Feature
Environmental Issue
«feature» Consumables Compatibility Feature
Consumable Type
«feature» Vehicle Performance Feature
Top Speed
Cruising Speed
Cruising Range
Acceleration
Seating
Load
«feature» Automatic Braking System Feature
«feature» Traction Control Feature
«feature» Vehicle Aesthetics Feature Group
Aesthetics Issue
«feature» Passenger Comfort Feature Group
Comfort Isssue
«feature» Vehicle Management Feature
«feature» Cost of Operation Feature
Operating Cost
«feature» Reliability &amp; Availability Feature
Reliability
Design Life
Scheduled Down Time
Unscheduled Down Time
«feature» Vehicle Delivery Feature
First Availability Date
Development Cost
Development Time
Unit Production Cost
Transfer Price
Target Volume
Target Production Rate
Financial Risk
Schedule Risk
«feature» Operability Feature
Operations Capability
«feature» Maintainability Feature
Maintenance Capability
«feature» Security Feature
Security Capability
«feature» Configurability Feature
Configuration Capability
«feature» Accountability Feature
Accounting Capability
«feature» Cruise Control Feature
«feature» Communications Feature Group
«feature» Remote Management Access Feature
«feature» Navigation Feature
«feature» Remote / Autonomous Operation Feature

Figure 5: Vehicle Features Package

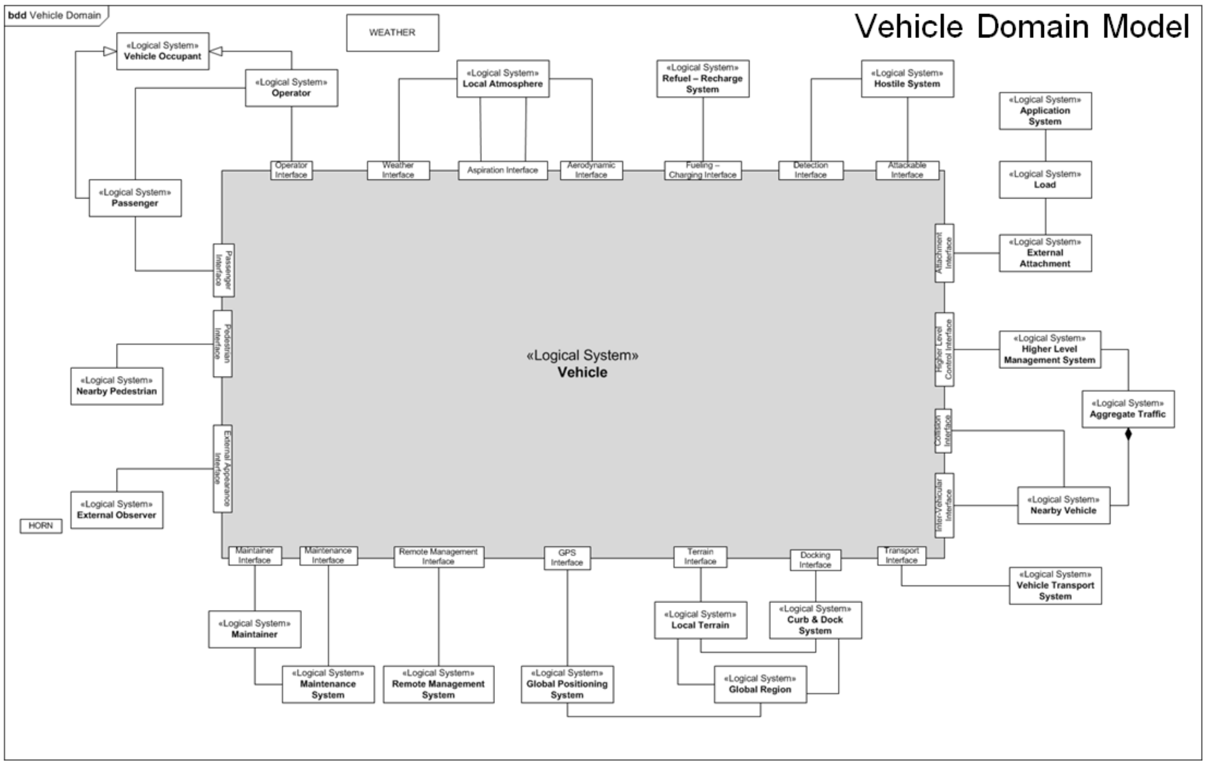

bdd Vehicle Domain
Vehicle Domain Model
WEATHER
HORN
«Logical System» Vehicle Occupant
«Logical System» Operator
«Logical System» Passenger
«Logical System» Local Atmosphere
«Logical System» Refuel – Recharge System
«Logical System» Hostile System
«Logical System» Application System
«Logical System» Load
«Logical System» External Attachment
«Logical System» Higher Level Management System
«Logical System» Aggregate Traffic
«Logical System» Nearby Vehicle
«Logical System» Vehicle Transport System
«Logical System» Nearby Pedestrian
«Logical System» External Observer
«Logical System» Maintainer
«Logical System» Maintenance System
«Logical System» Remote Management System
«Logical System» Global Positioning System
«Logical System» Local Terrain
«Logical System» Curb &amp; Dock System
«Logical System» Global Region
«Logical System» Vehicle
Operator Interface
Weather Interface
Aspiration Interface
Aerodynamic Interface
Fueling – Charging Interface
Detection Interface
Attackable Interface
Attachment Interface
Higher Level Control Interface
Collision Interface
Inter-Vehicular Interface
Passenger Interface
Pedestrian Interface
External Appearance Interface
Maintainer Interface
Maintenance Interface
Remote Management Interface
GPS Interface
Terrain Interface
Docking Interface
Transport Interface

Figure 6: Vehicle Domain Model

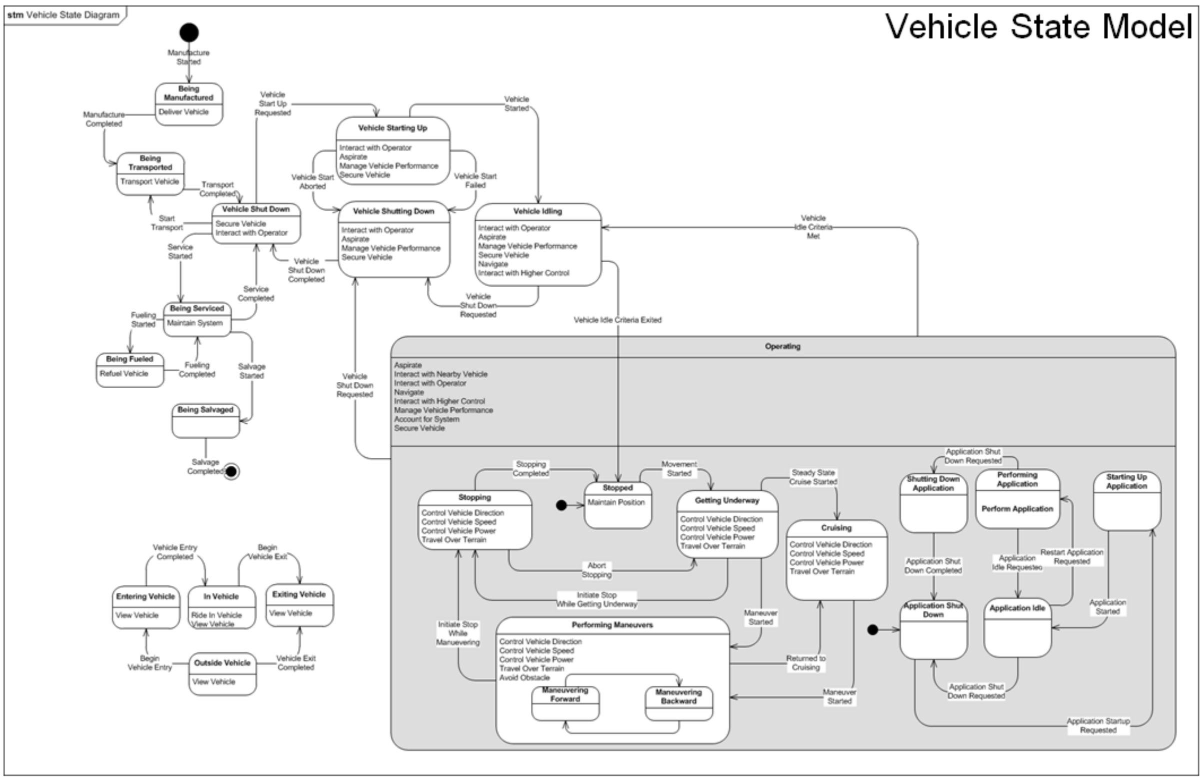


Figure 7: Vehicle State Model

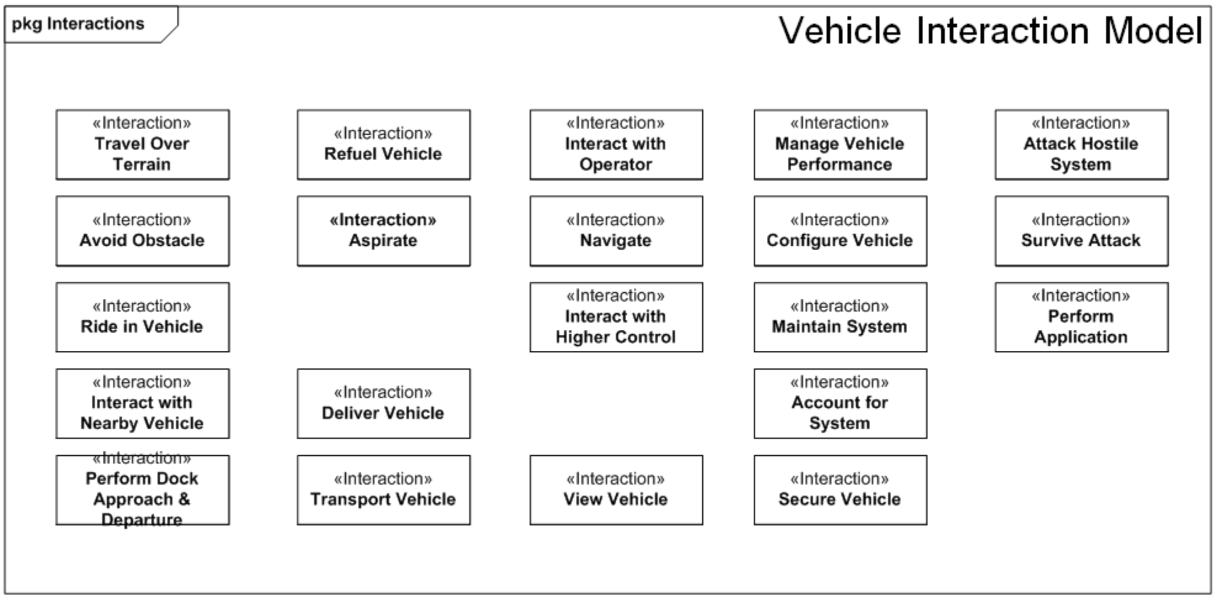

pkg Interactions
Vehicle Interaction Model
«Interaction» Travel Over Terrain
«Interaction» Avoid Obstacle
«Interaction» Ride in Vehicle
«Interaction» Interact with Nearby Vehicle
«Interaction» Perform Dock Approach & Departure
«Interaction» Refuel Vehicle
«Interaction» Aspirate
«Interaction» Deliver Vehicle
«Interaction» Transport Vehicle
«Interaction» Interact with Operator
«Interaction» Navigate
«Interaction» Interact with Higher Control
«Interaction» View Vehicle
«Interaction» Manage Vehicle Performance
«Interaction» Configure Vehicle
«Interaction» Maintain System
«Interaction» Account for System
«Interaction» Secure Vehicle
«Interaction» Attack Hostile System
«Interaction» Survive Attack
«Interaction» Perform Application

Figure 8: Vehicle Interactions Model

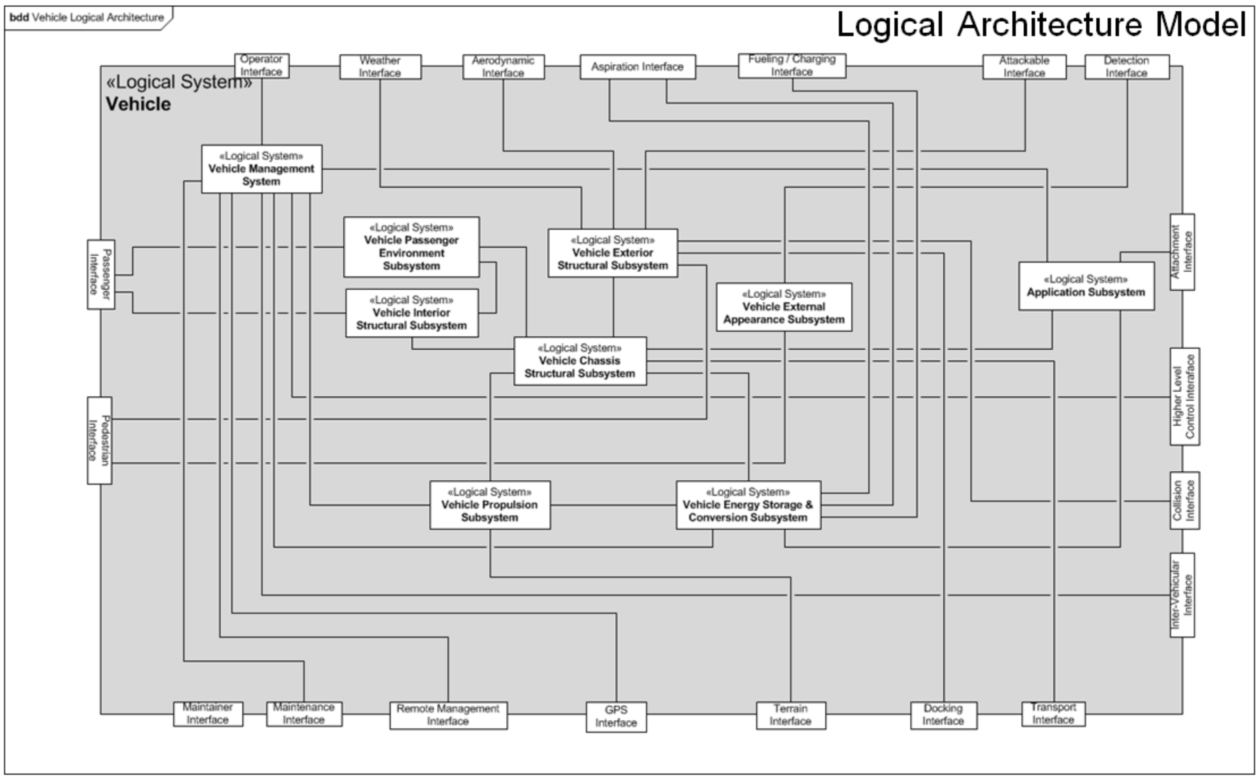

bdd Vehicle Logical Architecture
Logical Architecture Model
«Logical System» Vehicle
Operator Interface
Weather Interface
Aerodynamic Interface
Aspiration Interface
Fueling / Charging Interface
Attackable Interface
Detection Interface
«Logical System» Vehicle Management System
«Logical System» Vehicle Passenger Environment Subsystem
«Logical System» Vehicle Interior Structural Subsystem
«Logical System» Vehicle Exterior Structural Subsystem
«Logical System» Vehicle External Appearance Subsystem
«Logical System» Application Subsystem
«Logical System» Vehicle Chassis Structural Subsystem
«Logical System» Vehicle Propulsion Subsystem
«Logical System» Vehicle Energy Storage & Conversion Subsystem
Passenger Interface
Pedestrian Interface
Attachment Interface
Higher Level Control Interface
Collision Interface
Inter-Vehicular Interface
Maintainer Interface
Maintenance Interface
Remote Management Interface
GPS Interface
Terrain Interface
Docking Interface
Transport Interface

Figure 9: Vehicle Logical Architecture Model

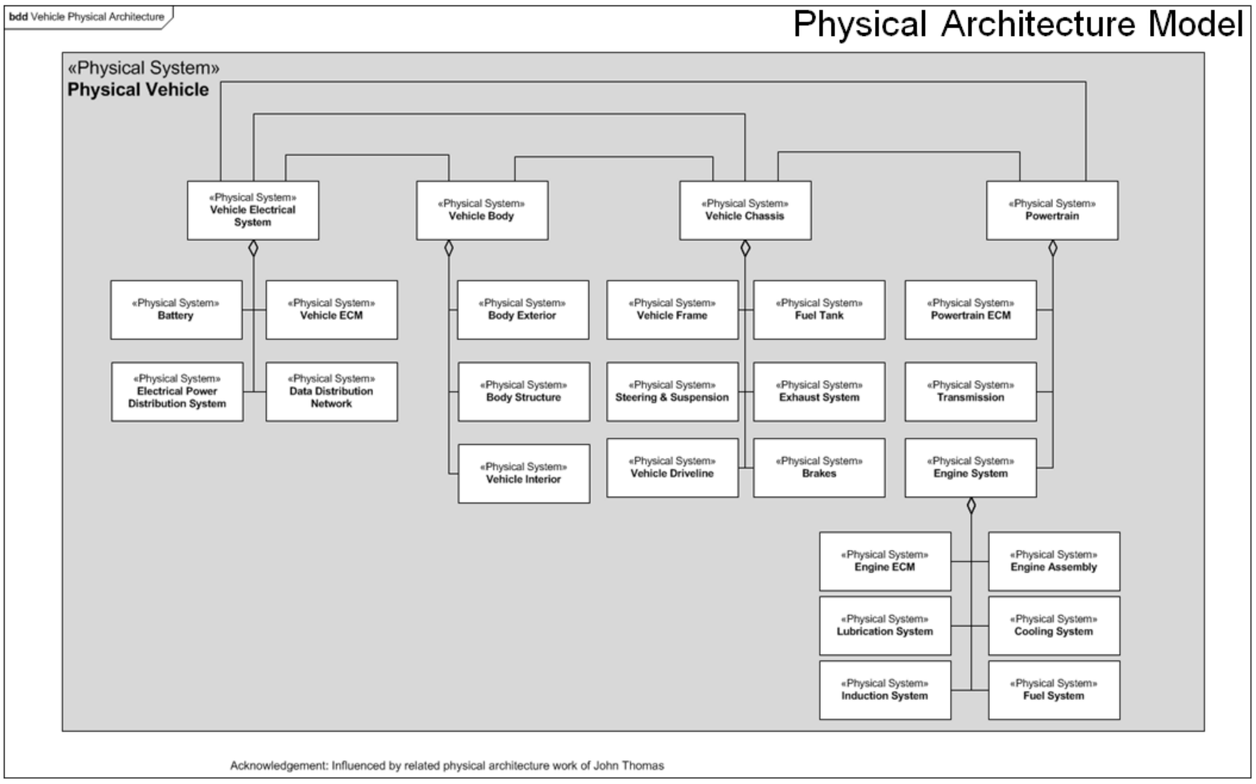

bdd Vehicle Physical Architecture
Physical Architecture Model
«Physical System» Physical Vehicle
«Physical System» Vehicle Electrical System
«Physical System» Vehicle Body
«Physical System» Vehicle Chassis
«Physical System» Powertrain
«Physical System» Battery
«Physical System» Vehicle ECM
«Physical System» Electrical Power Distribution System
«Physical System» Data Distribution Network
«Physical System» Body Exterior
«Physical System» Body Structure
«Physical System» Vehicle Interior
«Physical System» Vehicle Frame
«Physical System» Fuel Tank
«Physical System» Steering & Suspension
«Physical System» Exhaust System
«Physical System» Vehicle Driveline
«Physical System» Brakes
«Physical System» Powertrain ECM
«Physical System» Transmission
«Physical System» Engine System
«Physical System» Engine ECM
«Physical System» Engine Assembly
«Physical System» Lubrication System
«Physical System» Cooling System
«Physical System» Induction System
«Physical System» Fuel System
Acknowledgement: Influenced by related physical architecture work of John Thomas

Figure 10: Vehicle Physical Architecture Model

Unlike the diversity of physical implementation across these AGV platforms, whether manned or unmanned, wheeled or tracked, mine, city or wartime environments the means and allocation of control can have a very high level of commonality well represented by control patterns. More specifically the Embedded Intelligence (EI) Pattern applies well.

Embedded Intelligence (EI) Pattern

Many S*Patterns are discovered and expressed through PBSE, but the Embedded Intelligence (EI) Pattern is of particular importance to the subject of AGVs.

In the world of human-engineered systems, the term "embedded system" has come to be understood to mean a relatively low-level automated control of some sort, typically in electronic hardware/software form, to be inserted into a mechanical or other physical system. In its most common usage, this term is not used to describe larger scale automation, such as would be found for higher-level enterprise information systems or the largest cyber-physical systems. Since cyber-physical systems include both, this sort of divided perspective tends to suggest there are more differences between these levels than we believe are necessary.

Accordingly, the EI Pattern returns to the perspective of Norbert Weiner, who first coined the term "cybernetics" to refer to the study of control and communication in both living and human-engineered systems. (Weiner, 1965). This seems particularly appropriate as the term "cyber-physical system" is finding favor in a world in which we seek patterns to advise us at many different hierarchical levels, including systems in which human beings are "embedded".

The EI Pattern is an S*Pattern that emerges to describe intelligence in explicit models of evolving systems in the natural and man-made world—also referred to as the Management System Pattern. (Schindel and Smith, 2002). It describes the individual elements and overall systemic framework of embedded intelligence on a total system, whether the agents of that intelligence are information technology, human, hybrid, or other forms of management.

Norbert Wiener studied the mathematics of feedback control systems in nature generally, and engineered fire control systems in particular. Although classical feedback control quickly comes to mind in the management of system performance, more generally there are also other aspects of management. The four types of Embedded Intelligence Pattern functional roles that arise are shown graphically in Figure 11:

- Managed System (MDS): Any system behavior whose performance, configuration, faults, security, or accounting are to be managed--referred to as System Management Functional Areas (SMFAs) or in ISO terminology fault, configuration, accounting, performance, security (FCAPS). These are the roles played by the so-called "physical systems" in a cyber-physical system, providing physical services such as energy conversion, transport, transformation, or otherwise.

- Management System (MTS): The roles of performing management (active or passive) of any of the SMFAs of the managed system. These are so-called "cyber" roles in a cyber-physical system, and may be played by automation technology, human beings, or hybrids thereof, to accomplish regulatory or other management purposes.

- System of Users (SOU): The roles played by a system which consumes the services of an managed system and/or management system, including human system users or other service-consuming systems at higher levels.

- System of Access (SOA): The roles providing a means of interaction between the other EI roles. Engineered sensors, actuators, the Internet, and human-machine interfaces have contributed greatly to the emergence of the "Internet of Things".

In evolving systems, instances of these roles may arise individually, and over time lead to an emergent web of embedded intelligence. As further shown in Figure 3, these roles are organized into EI Hierarchies, in which localized EI functions contribute to subsystem intelligence and effectively higher level EI functions contribute to higher level intelligence. In engineered systems, such EI hierarchies may be found in automotive, manufacturing, or aerospace systems, for example.

In the case of the AGV Pattern, the Logical Architecture Diagram of Figure 9 shows the Vehicle Management System as a logical subsystem. This level of management is concerned with the vehicle as a whole. For consumer vehicles, it is frequently a hybrid of human operator (vehicle starting, steering, parking, stopping, etc.) and automation (automatic braking, vehicle stability control, etc.). Each of the other logical subsystems of the same Figure 9 are likewise eligible to have their own internal Management System—the next level down in the hierarchy. This management hierarchy continues downward, but also upward—there is management at higher levels, including traffic control, fleet management, site management, warehouse management, mission control, etc.

Like all S*Patterns, the EI Pattern also includes a pattern of Stakeholder Features, Functional Interactions, States, and other aspects which appear repeatedly in a pattern of relationships characteristic of embedded intelligence, whether through planned engineering or emergent evolution. These are not necessarily planned "top down" and the EI Pattern has been used to reverse engineer and describe complex hierarchies of embedded intelligent systems that emerged over time as either human-engineered sub-systems or as natural world systems that include living and other elements.

Within the EI Pattern, States (modes, situations) that arise include Situation Resolution Cycles. As shown in Figure 12, these reflect the idea that system stability over time requires a form of system regulation to "resolve" various "situations" that may occur from time to time, driving the managed system back to a "normal" or nominal state. Examples include:

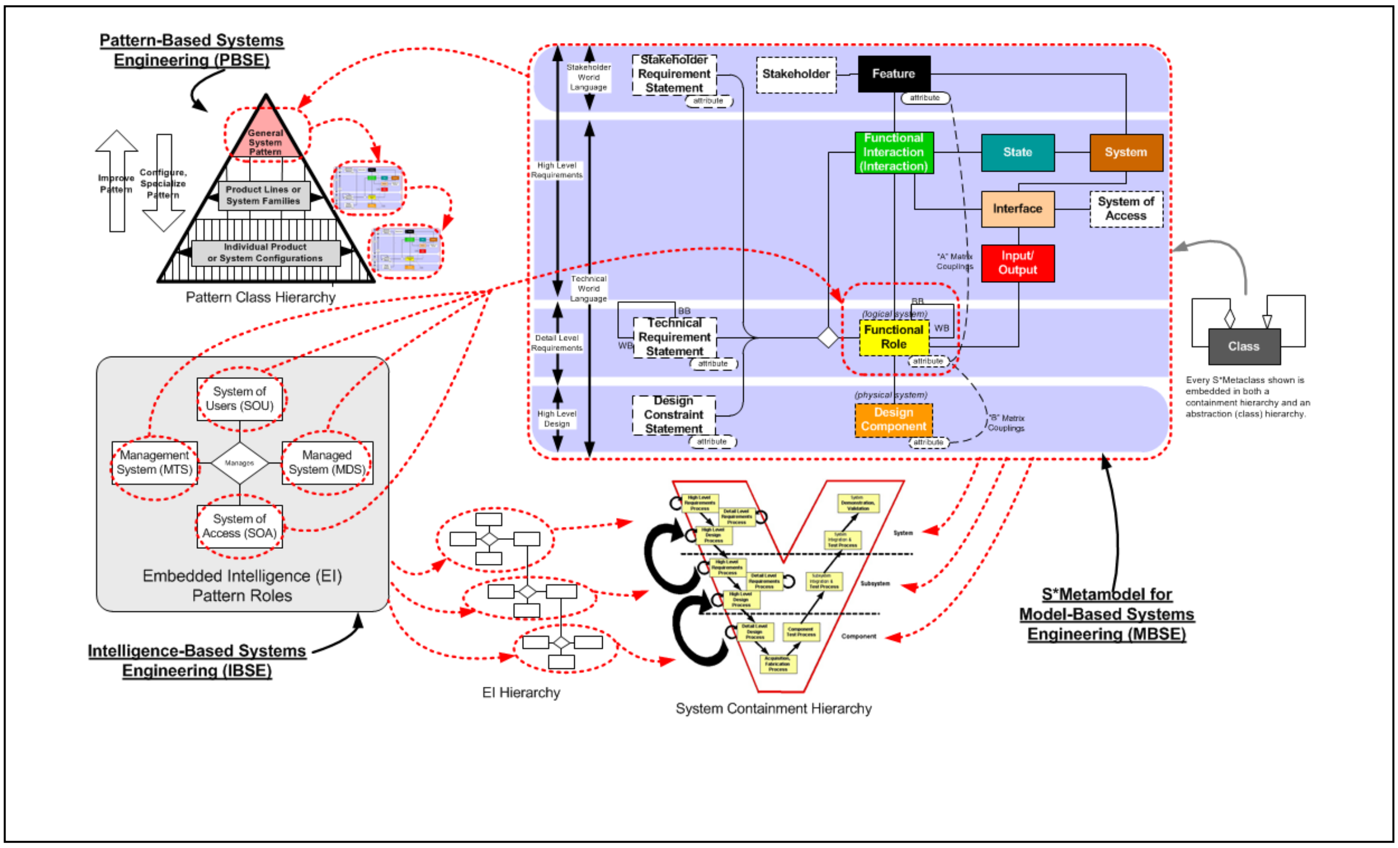
- Major Mission Resolution Cycles: These proceed through a series of mission states, from mission initiation to fulfilment, including planning.

- Minor Use Case Resolution Cycles: These similarly resolve various situational use cases.

- Resolution of Faults: These may include the recognition, diagnosis, repair, and recovery from system faults.

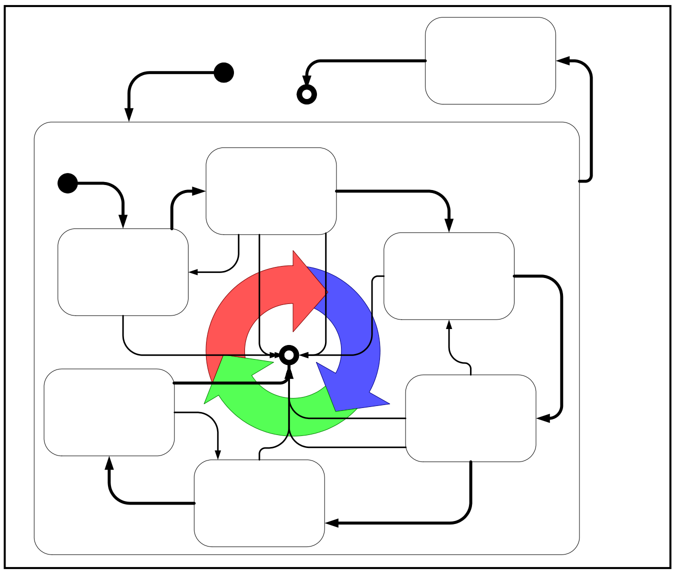
- Resolution of Service Requests: These may include resolution of requests for such services as re-configuration, security, or other situations.

If a system is capable of not only traveling a situation resolution cycle trajectory (as in Figure 4), but also recognizing that such a situation has arisen in the first place (as in Figure 13), we say that the system is “Situationally Aware”.



Pattern-Based Systems Engineering (PBSE)
Improve Pattern
Configure, Specialize Pattern
General System Pattern
Product Lines or System Families
Individual Product or System Configurations
Pattern Class Hierarchy
System of Users (SOU)
Management System (MTS)
Manages
Managed System (MDS)
System of Access (SOA)
Embedded Intelligence (EI) Pattern Roles
Intelligence-Based Systems Engineering (IBSE)
EI Hierarchy
Stakeholder World Language
High Level Requirements
Technical World Language
Detail Level Requirements
High Level Design
Stakeholder Requirement Statement
attribute
Stakeholder
Feature
Functional Interaction (Interaction)
State
System
Interface
System of Access
Input/ Output
"A" Matrix Couplings
BB
WB
Technical Requirement Statement
(logical system)
Functional Role
(physical system)
Design Constraint Statement
Design Component
"B" Matrix Couplings
Class
Every S*Metaclass shown is embedded in both a containment hierarchy and an abstraction (class) hierarchy.
System
Subsystem
Component
System Containment Hierarchy
S*Metamodel for Model-Based Systems Engineering (MBSE)

Figure 11: Emergence of Embedded Intelligence (EI) Pattern Functional Roles



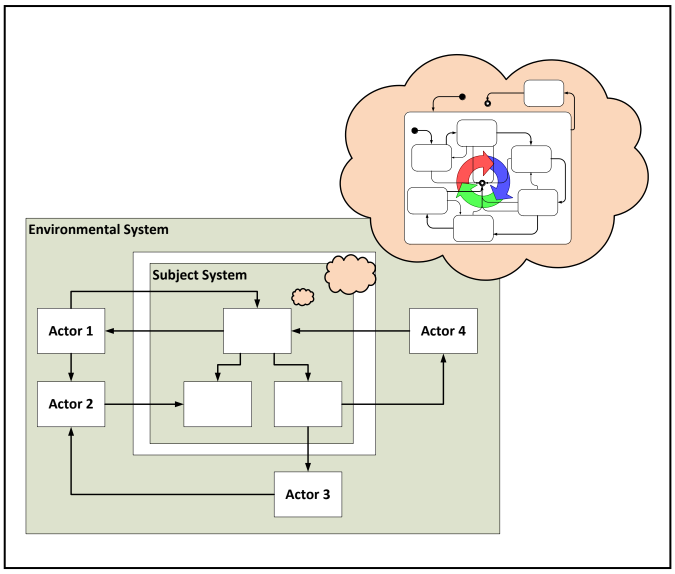


Figure 12: Situation Resolution State Cycles

Figure 13: Situational Awareness

If an operator interface panel has a “mode switch”, then the system relies on a human operator to recognize occurrence of a situation and place the system in the appropriate state cycle (mode). More advanced autonomous systems automate this (situation recognition and mode-switching) greater sophistication. In all these cases, Attention Models in the EI Pattern describe ability to recognize and resolve situations which arise within the finite limits of managed system and management system resources.

An example of this can be found in automated ground vehicles in three incremental levels of increasing autonomy – from limited to partial and then full autonomy. One can imagine the potential architectural differences between systems that have planned to evolve to full autonomy versus those that simply seek to integrate capabilities as they become available. In the second case the full EI pattern is not considered until after a system often experiences architectural lock in and it has limited integration opportunities.

An example of the benefits of applying the EI Pattern to CPS is the insight it provides about emergence of higher level controls and management effects in systems, which we have observed to be frequently overlooked or misunderstood—especially in high complexity systems such as autonomous ground vehicles and the larger domain systems they inhabit. The EI Pattern is based first and foremost on the logical roles of that pattern, not the physical technologies that perform those roles. A common thinking trap is to associate control system boundaries or scope with physical controllers or information system platforms. The EI Pattern teaches us that these boundaries are better understood using the scope of Logical Management System roles. It is commonplace in engineering of control systems or information systems to expend effort in “integrating” those information systems so that they can interact with each other (“talk” to each other). What is often overlooked, and what the EI Pattern enforces in our understanding, is that a higher level management system emerges logically in any situation in which two management system roles that have different managed system scopes interact with each other. This is a powerful way to understand how higher level management effects (whether positive or negative) emerge even if we did not initially recognize them.

Automated Ground Vehicle and EI Pattern Integration

The need to manage the inherent complexity within AGVs is very relevant today and the complexity is only accelerating. Google’s driverless cars are expected to hit the markets between 2017 and 2020. Automakers including GM, Ford, Nissan, Volkswagen, Mercedes Benz, Volvo, and Volvo Truck all have autonomous programs. As a specific case of AGVs modern automobiles have begun to incorporate driver assistance technologies that will, under certain circumstances, take momentary control away from the driver if they get in over their head including Collision Mitigation Braking, Panic Brake Assist, Electronic Stability Control, Blind Spot Warning, and Lane Keeping systems, among others. These systems by themselves can mitigate a significant number of causalities due to vehicle accidents. The rationale for automating civilian vehicles can be far reaching given: “A third of the American workforce spends more than an hour a day commuting. It’s boring and dangerous.”[insert reference] There are 30,000 deaths [insert reference] in a year because of car accidents, of which 90 percent were caused by human error.

To handle the diversity of AGVs which include application, manned and unmanned and level and types of control the integration of the Vehicle and EI patterns can provide a very powerful and compact way models these complex cyber physical systems. To accomplish this for an AGV would require the integration of the Vehicle and EI patterns. This is a straight forward exercise given both are compliant with the S* metamodel to form S* patterns. For AGVs this then permits the configuration and reuse of the general vehicle pattern and EI pattern to cover the diverse range of platforms within the AGV domain. Configuration of a specialized AGV can be accomplished specifically thorough the selection of features shown in Figure 5 to ultimately configure systems solutions represented in figure 10. The next thing the PBSE INCOSE/OMG Challenge Team will do is to incorporate feature selection following the precepts of ISO 26550.

AGVs to cover specific applications are most easily handled by application feature groups shown in Figure 5 which could be elaborated if necessary. More specifically, the Personal Vehicle Application Group, Commercial Vehicle Application Group and Military Application Group; selection of these features would turn on or off many of the external and internal interfaces and interactions to specialize the pattern for the particular application. Regarding manned and unmanned AGV variants this can also be configured from the general vehicle pattern through the selection (or not) of the Passenger Comfort Feature Group of Figure 5 which in turn would include (or not) the Passenger Interface on the Domain model of Figure 6, the Ride in Vehicle interaction from the interaction diagram of Figure 8 etc. Ultimately this would provide configuration options for implementation within the Physical Architecture Model element of Vehicle Interior shown in Figure 10.

To manage the type of control and ultimately the level of automation use of the EI Pattern and the chosen implementation of its functional roles (MDS, MTS, SOU and SOA) would determine the level of human control versus partial (hybrid) or full autonomous control. The Feature element which would determine much of this is the Vehicle Management Feature of the Vehicle Pattern shown in Figure 5. Here you can find features like navigation, cruise control, autonomous operation, remote management etc. Through the use of the EI Pattern allocations of sensing can be allocated to a driver, to multiple sensors on the platform or to a mix of these in conjunction with surrounding infrastructure and other vehicles all of which would fall into very specific roles within the EI pattern (depending upon the features selected). Selections of roles within the EI pattern ties directly back the vehicle pattern in many places; for instance: within the Domain Model (Figure 6) the Higher Level Management System, GPS, Terrian and Remote Management System. Within the Vehicle Interaction Model (Figure 8) interactions such as Avoid Obstacle, Interact with Nearby Vehicle, Navigate and Manage Vehicle Performance many be populated with different functional roles and inputs and outputs in fulfillment of the EI Pattern roles to appropriately meet the necessary features for a specific AGV.

The task of designing a system that synthesizes the interaction of sensors, software and electro-mechanical actuators to achieve autonomous mobility safely and reliably is both a challenge and an opportunity. Multiple layers of complex information must be processed, understood and communicated at many levels. The demands on component reliability will be extremely high. When safety or mission critical components do fail, diagnostic systems must detect failure and switch to backup or "limp home" systems. The complexity involved, while daunting, provides the opportunity to leverage commercial technical advances in autonomy, as well as advances in model based methods and more specifically the use PBSE methods to manage the inherent complexity within AGVs.

Follow on work

Elaboration of this work will be performed within a sub-team of the PBSE INCOSE/OMG Patterns Challenge Team whose focus is to model an Automated Ground Vehicle Platform. More specifically the sub-team will focus on the follow items:

- Expand the depth of both the vehicle and embedded intelligence pattern to build an Automated Ground Vehicle Platform Pattern. Initial work has focused on limited autonomy in an unmanned remotely controlled platform. In elaborating the model the sub-team will expand and detail feature selection and optioning using the precepts of Product Line Engineering (ISO 26550).

- Use the AGV model to further demonstrate the use of Design Structure Matrix (DSM) and Network Analysis as aids in architecting engineered systems. More specifically to aid with modularization, partitioning, visualizing allocations across Multi-Domain Matrices (MDMs) and investigating key network metrics and their ability to aid in architecting systems.

- Determine the applicability of ongoing standardization efforts affecting AGVs. In specific either SARTRE semi-automated truck platooning system or the AUTomotive Open Systems ARchitecture (AUTOSAR) Chassis Control Functional Architecture.

Conclusions

The systemic complexity of AGVs demands a systems engineering approach. It requires a systems paradigm which is interdisciplinary, leverages principals common to all complex systems and applies the requisite physics-based and mathematical models to represent them. It is a methodology which further differentiates the effectiveness of any MBSE approach and its ability to help manage the complex and interrelated functionality of today's Cyber Physical Systems. For the approach discussed in this paper, the "methodology" includes not only process, but more significantly the very concept of the underlying information those processes produce and consume, independent of modeling language and tools.

As a model-based systems engineering approach PBSE is particularly well suited to address cyber physical systems challenges and those of Automated Ground Vehicles. PBSE provides a data model and framework that is both holistic and compact. It addresses the core system science needed in designing automated ground vehicle platforms by making interactions, the heart of Cyber Physical Systems, more visible. PBSE and the EI pattern provides a rapid and holistic means to identify and manage system risk and failure identification, analysis, and planning essential to CPS. Both are also essential in establishing patterns of adaptive and hierarchical control which can be leveraged as a framework for engineering trusted systems. The Embedded Intelligence Pattern explicitly represents the logical roles which enable planned evolution and limits architectural lock in, effectively reducing switching costs and speeding technology integration.

References

1. (Alexander, 1977) Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., and Angel, S., *A Pattern Language*. Oxford University Press, New York, 1977.
2. (BCG, 2013) "The Most Innovative Companies 2013 – Lessons from Leaders" 2013 BCG Global Innovators Survey, BCG Analytics.

3. (Berg, 2014) Berg, E., “Affordable Systems Engineering: An Application of Model-Based System Patterns To Consumer Packaged Goods Products, Manufacturing, and Distribution”, at INCOSE IW2014 MBSE Workshop, 2014.
4. (Bradley, Hughes, Schindel, 2010) Bradley, J., Hughes, M. and Schindel, W., “Optimizing Delivery of Global Pharmaceutical Packaging Solutions, Using Systems Engineering Patterns” Proceedings of the INCOSE 2010 International Symposium (2010).
5. (Cloutier, 2008) Cloutier, R., Applicability of Patterns to Architecting Complex Systems: Making Implicit Knowledge Explicit. VDM Verlag Dr. Müller. 2008.
6. (Estafan, 2008) Estafan, J. 2008. Survey of model-based systems engineering (MBSE) methodologies. INCOSE MBSE Initiative.
7. (Gamma et al, 1995) Gamma, E., Helm, R., Johnson, R., and Vlissides, J., *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Publishing Company, Reading, MA, 1995.
8. (ICTT, 2013) Abbreviated Systematica Glossary, ICTT System Sciences, 2013.
9. (INCOSE Handbook, 2014) INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, Version 4, International Council on Systems Engineering (2014).
10. (INCOSE Patterns Team, 2014) INCOSE/OMG MBSE Initiative: Patterns Challenge Team 2013-14 Web Site: http://www.omgwiki.org/MBSE/doku.php?id=mbse:patterns:patterns
11. (ISO 15288, 2014) ISO/IEC 15288: Systems Engineering—System Life Cycle Processes. International Standards Organization (2014).
12. (ISO 26550, 2013) ISO/IEC 26550 “Software and Systems Engineering—Reference Model for Product Line Engineering and Management”, 2013.
13. (ISO 42010, 2011) ISO/IEC/IEEE 42010 “Systems and Software Engineering—Architecture Description”, 2011.
14. (NSF, 2014)“NSF-Funded Joint Efforts”, at www.nsf.gov/funding/pgm_summ.jsp?pims_id=503286
15. (Schindel, 2005a) Schindel, W., “Pattern-Based Systems Engineering: An Extension of Model-Based SE”, INCOSE IS2005 Tutorial TIES 4, (2005).
16. (Schindel, 2005b) Schindel, W. “Requirements statements are transfer functions: An insight from model-based systems engineering”, Proceedings of INCOSE 2005 International Symposium, (2005).
17. (Schindel, 2010) Schindel, W., “Failure Analysis: Insights from Model-Based Systems Engineering”, INCOSE International Symposium, Chicago, 2010.
18. (Schindel, 2011b) Schindel, W. “What Is the Smallest Model of a System?”, Proc. of the INCOSE 2011 International Symposium, International Council on Systems Engineering (2011).
19. (Schindel, 2011c) Schindel, W., “The Impact of ‘Dark Patterns’ On Uncertainty: Enhancing Adaptability In The Systems World”, in Proc. of INCOSE Great Lakes 2011 Regional Conference on Systems Engineering, Dearborn, MI, 2011

20. (Schindel, 2012a) Schindel, W. "Introduction to Pattern-Based Systems Engineering (PBSE)", INCOSE Finger Lakes Chapter Webinar, April 26, 2012.
21. (Schindel, 2012 b) Schindel, W., "Integrating Materials, Process, & Product Portfolios: Lessons from Pattern-Based Systems Engineering", in *Proc. of Society for Advancement of Materials and Process Engineering* (SAMPE), 2012
22. (Schindel, 2013a) Schindel, W. "Interactions: At the Heart of Systems", INCOSE Great Lakes Regional Conference on Systems Engineering, W. Lafayette, IN, October, 2013.
23. (Schindel, 2014) Schindel, W. "The Difference Between Whole-System Patterns and Component Patterns: Managing Platforms and Domain Systems Using PBSE", INCOSE Great Lakes Regional Conference on Systems Engineering, Schaumburg, IL, October, 2014
24. (Schindel, Peterson, 2013) Schindel, W., and Peterson, T. "Introduction to Pattern-Based Systems Engineering (PBSE): Leveraging MBSE Techniques", in Proc. of INCOSE 2013 International Symposium, Tutorial, June, 2013.
25. (Schindel, Peterson, 2014) "Pattern Based Systems Engineering – Leveraging Model Based Systems Engineering for Cyber-Physical Systems", Proc. of 2014 NDIA Ground Vehicle Systems Engineering and Technology Symposium, Systems Engineering Technical Session, August 12-14, 2014, Novi, MI.
26. (Schindel, Smith, 2002) Schindel, W., and Smith, V., "Results of applying a families-of-systems approach to systems engineering of product line families", SAE International, Technical Report 2002-01-3086 (2002).
27. (Weiner, 1965) Norbert Weiner, Cybernetics: Control and Communication in the Animal and the Machine, Cambridge, MA, MIT Press, 1965.

Biography

Troy Peterson is a Chief Engineer and Booz Allen Fellow operating as a firm-wide resource supporting top priority programs. Prior to Booz Allen, he worked at Ford Motor Company and operated his own engineering consulting business. He serves on the MSU Mechanical Engineering Department Advisory Board and INCOSE's Corporate Advisory Board. He's also a past president of INCOSE's Michigan Chapter and he co-leads the Patterns Challenge Team of the OMG/INCOSE MBSE Initiative.

William D. (Bill) Schindel is president of ICTT System Sciences. His engineering career began in mil/aero systems with IBM Federal Systems, included faculty service at Rose-Hulman Institute of Technology, and founding of three systems enterprises. Bill co-led a 2013 project on the science of Systems of Innovation in the INCOSE System Science Working Group. He co-leads the Patterns Challenge Team of the OMG/INCOSE MBSE Initiative