

Vehicle System Development: A Challenge of Ultra-Large-Scale Systems

Robert Baillargeon
General Motors Corporation
robert.c.baillargeon@gm.com

Abstract

Scale and complexity have been the topical elements of the vehicle system development for years. Whereas traditional systems engineering methods have been believed to be the solution to vehicle design challenges, recent shifts in the vehicle domain question their adequacy. Specifically, the character of vehicle systems, and their development environment, has shifted away from singular-use independent systems to multipurpose collaborating systems. The characteristics of this change are contained within the themes of scale, diversity, and dynamics. Each of these dimensions is contributing to the “wickedness” of vehicle systems in understanding, development, deployment, and maintenance. Such complexity and dynamics of interactions, by both users and developers, make vehicle systems exhibit the characteristics of Ultra-Large Scale (ULS) systems. The scale and concerns of vehicle system development challenge traditional approaches, and demand new fundamental research to reason the design, manage the scale, and create the capability for developing future vehicle systems.

1. Introduction

Vehicle systems of today pose engineering challenges of large scale to system development. A vehicle system may include up to 60 microprocessors, multiple internal communication networks, and increasingly external telematic connections. The applications include software for conveniences such as climate control, for dynamic real-time controls such as vehicle stability control, and for entertainment systems such as high-end rear seat audio-visual systems. The deployment of so-called “vehicle-to-vehicle” and “vehicle-to-infrastructure” applications based on ad-hoc external communications further increases the pervasiveness and diversity of vehicle control functions. These together, make vehicle systems have the aspects of Ultra-Large-Scale (ULS) systems.

Several technologies and methods have shown potential to address the challenges of the vehicle development domain; however further growth and matur-

ity of these technologies must be achieved and are essential to apply to ULS systems including their inherent cyber-physical characteristics. A key improvement required is the capability of handling complexity, diversity, and conflict as primary concepts with mechanisms to identify and control these concepts in various dimensions. In the rest of this paper, we will present the ULS challenges exhibited in vehicle systems, technology gaps experienced in the development of future vehicle systems, and promising research directions that may address these gaps.

2. Design Challenges of Vehicle Systems

At the core, the critical aspect of ULS systems is their size. The size adversely influences the development and deployment of systems because the traditional assumptions on “systems” used in existing design methods become invalidated, such as stability and consistency. It is observed that the complexities of systems grow at non-linear rates relative to their size. In this section, we present the challenges experienced in vehicle system development, which have caused multiplicative effects in complexity in the same manner as ULS systems.

2.1 Large Development Scope

Two dimensions of the system scope make the developing vehicle systems challenging. The first is the volume of artifacts that are generated to support development. This challenges the ability to reason about systems and interactions based primarily on numerical challenges which often lead to non-linearity challenges to methods. Many tools fail when non-linear growth of dependency paths exists in the development, which creates barriers to the design and analysis of unconstrained, large-scale systems. Activities such as software builds for large-scale systems demonstrate this challenge quite well, yielding a multitude of solutions (library components, build farms, incremental builds, complex build process, and release processes, etc.) that work with varying degrees of effectiveness.

The second is the diversity inherently associated with vehicle systems at such scale. The vehicle system is required to work in an environment with the integra-

tion of motion controls (in the style of flight controls), convenience functions, entertainment devices, and heterogeneous communication connectivity. The collaboration of a broad set of domain principles, such as co-existence of continuous and discrete control, closed and open loop, and synchronous and asynchronous, is essential to implement such integration. The diversity introduces contention and conflict among objectives and applications that are not normally observed in small-scale systems. The scale further makes traditional methods incapable of resolving all conflicts in the system because of the required wide scope of comprehension.

The challenge of scope in both dimensions is mainly caused by the inadequacy of scaling tree structures that represent a singular, centralized source of authority for common arbitration in vehicle systems. As a solution, decentralized decision and control, which has never been attempted in this domain, is necessary to manage domain and problem spaces. The development of collaborating systems with decentralized control is a paradigm without sound scientific foundation to deliver predictability of design practices without further research.

2.2 Diverse Development Teams

The vehicle systems of today are designed and manufactured by large global teams. The situation will continue and grow. This is to manage the global economy and the scale of system development, however by doing so faces a series of challenges of integrating the diversity of expertise, region, and responsibility. It is natural that when systems become large, and ultra-large, their development teams also become large. System development still remains a social activity where critical information is communicated through social interaction (e.g. meetings, hall conversations, telephone calls, e-mails, etc.). The exponentially-increasing interaction amongst a large number of development team members becomes a critical bottleneck, since they are most conducive in a one-on-one environment. Although creating smaller sub-teams may control the growth of interactions, the deep hierarchy of the multi-tiered teams seriously questions the scalability of this method when a vast number of developers work on ultra-large scale systems.

Besides the size, site and regional diversity further introduces the new challenges as the locality disrupts the normative, face-to-face human interaction. However this is often balanced with the value of regional diversity aiding in understanding different usage scenarios in different localities. The challenge results in a set of social challenges for ULS systems on natural interactions, cultural differences, and disjoint time schedules. All of these introduce roadblocks to devel-

opment productivity based on the social nature of system development and evolution.

With the diversity of functionality of ultra-large scale systems there is also a diversity of expertise and domain knowledge. Such diversity of knowledge, experience, and technical training leads to diverse observations as to the role and function of the developing system. This results in unbalanced or potential conflicting views of objectives and priorities of the system even within the staffs of the same development teams. Such conflicting views create a natural disruptive force in the stability of development, such as requirements, providing an undercurrent of persist change as the perception and knowledge of developer or even the roles of team members' change.

Site diversity also introduces the technical challenge of asset management. Asset management, as in change and configuration management, becomes a discipline unto itself in managing multiple streams, baselines, and merges to insure stability and integration of data sets within and across regions. The development environment provided by the traditional approaches using file-orientated configuration tools and file based editing tools (modeling, IDEs, documents, etc.) is inefficient and complex. This results in the inability to arbitrarily apply tooling to manage the complexity independent of asset organization, relationships, and ownership resulting in a critical process binding.

The collective result of these challenges is a latent reactive process that slows all aspects of design. The primary cause for such challenges appears to be the lack of the capability of traditional development processes in managing scale and diversity characteristic of ULS systems.

2.3 Risk-Related High Cost

The investment costs of vehicle systems, as also seen in many ULS systems, impose a unique economic situation of becoming monopolies unto themselves. While odd, it is often true for such a system that while an original justification can be created to initiate the development of a system, it is very unlikely that the development can be initiated again with the same investment level anytime in the near future. So simply by initiating the development of a ULS system bars the creation of another, competing or replacing, ULS system with the same intended functionality. Such a self-contained monopoly demonstrates several unique, cost-related characteristics, including:

- Cost for high risk organizational behavior resulted from the viability of organization tied to single projects, and significantly higher thresholds for re-designs.
- High cost for long-term commitments, including the large resource commitments for extended pe-

riod of times and singular owners for long term maintenance and upgrades.

- Cost of physical systems that dictate the lifespan and investment patterns on the corresponding cyber system.

These financial implications resulted from the risk management strategies are unique for large-scale development and drive different decision structure. With high failure costs experienced in large-scale development, it is obvious that new methods of controlling development risks and life-time costs of systems must be developed to support more scalable risk management strategies, eliminate or minimize the risk-related cost, and improve development success rates.

2.4 Deployment with Unplanned Changes

The deployment and maintenance of vehicle systems are unique based on their size and investment costs. As ULS systems, the lifespan of a vehicle system is measured in years of duration, with durations over a decade common. The size and investment costs require the systems to have a long lifetime and to adapt to the surrounding environment and society.

The long lifetime requires ULS systems to be able to adapt in the field. Given very few changes can be planned; it is difficult to adapt when the unplanned changes occur. These unplanned changes include both changes in the deployed system and changes of interactions between the user and/or environment and the system. These may cause changes of user's objectives and expectations which only exacerbates the "wickedness" of the problem. Often, the latter changes of interactions, are the unknown, and traditionally are learned through valuable prototypes. But as vehicle systems become ULS systems, the capability to execute prototypes in development is greatly limited due to scale and cost. Additionally, ULS systems are rarely completely overhauled in the field, so only partial changes will occur to the entire system. The rest of the system must continue to function while preventing adverse side effects from being introduced by the partial changes. It is these dynamic and unplanned changes lead to exponential combinations of configurations and increase the difficulty to reason about systems that are anticipated to be faced in the future of vehicle systems.

As not all configurations could be reasoned about during development, new design paradigms and run-time architecture are required to build robustness into systems even when instability exists in deployed applications. To this end, research of adaptive methods and architectures is of valuable in the automotive domain as manufactures strive to develop products more rapidly, deal with uncertainty, and provide opportunities to de-

ploy enhancements to vehicles throughout their lifecycle.

3. Technology Gaps

The technology gaps relative to ultra-large systems, of which vehicle systems will become, are critical to research plan of the future. While it remains to be seen whether our first assessment as to the fundamental ULS challenges are accurate, they are important to research as a class. It is likely that as this classification, future vehicle systems will be "wicked". Developing methods to resolve such "wicked" problems may simultaneously change our understanding of them. In this vein, we do not intend to identify specific technologies for solutions. Rather we present a list of technology in capabilities where it is anticipated the technologies are likely to adapt over time to meet the needs.

A. Support of Run-Time Upgradeability.

The upgradeability of future ULS systems is unique with regard to the duration of which the system must be upgradeable and the breadth of possible upgrades. The upgrades may happen to hardware and/or software and at all points from subtle functional improvement to large-scale full functional change. All of which must be handled gracefully. Further, the delivery mechanisms for these changes are likely to take place in the field rather than in static controlled environments. None of today's development methods and runtime architectures can manage such uncertainty in future upgrades, thus new research is required to develop methods and technologies to support runtime upgradeability.

B. Support of Design and Run-Time Regulation with Decentralized Control.

Vehicle systems, as with ULS systems, need to have a regulated environment as opposed to a fixed interaction structure for evolution. This can be analogous to the evolution of ecosystems and cities. Since the systems don't finish evolution at the conclusion of the initial design cycle, the self-regulation of system components must exist at runtime. The run-time concepts of self-regulation and decentralized control are required to allow for evolution of a system's design. However, methods and architectures to support evolving systems with regulated behaviors do not exist. New methods and architectures must be developed these new concepts.

C. Methods for System Compositions.

One of the largest roadblocks in developing ULS systems is the development timing and financial commitments. Developing systems from proverbial "clean sheet" does not exist and yet the ability to ad-hoc reuse and compose does not exist at the demanded level. Where there has been significant ef-

fort in developing integration and component methods, the concepts of compositional methods have not grown to the level that meets demand. Methods are needed to better capture knowledge about the character, properties, of systems when composed of a set of components, a form of component property mathematics. Methods are also needed to understand the impact to the overall system when parts of the composed systems are replaced.

D. Methods for Platform Isolation.

As many ULS systems, the initially-designed platform of vehicle systems is often neither persistent nor permanent. Platform-dependent designs are fragile and may pose significant rework and re-validation if retargeting during development. Side effects of these characteristics include the stunting of desirable system evolution due to limited applicability to a platform, and reduction in feature growth rate due to resources consumed with platform portation issues. To deal with platform uncertainty, new research is needed to provide methods for platform agnostic design, definition, characterization, and redeployment.

E. Support of Multi-Site Development.

A common burden to large scale software development is the dimension of multi-site development. These are the technical and social aspects of developing systems in diverse environments. This will require growth in design methodologies to support these challenges to productivity and importance of diverse organizations. Tooling and design environments must also grow to more directly support these constraints and desires.

F. Techniques for Conflict Resolution.

Vehicle systems face the same ULS issue of managing system-wide conflicts at both design and run-time. The invalidation of assumptions of coherency and non-conflicting objectives is driven by scale and diversity of usage. New techniques are required to develop and deploy systems in a way that allows the graceful management of conflicts and regulated system behaviors can be achieved while still enabling vehicle system evolution.

4. Promising Technologies

While there is no singular answer to managing the complexities of scale and dynamics of ULS systems, there are several technologies that have been growing and are applicable in the large vehicle system development. The question remains whether their growth can scale to ULS systems and/or can they be combined in novel ways to address the specific ULS system concerns. Following is a list of technologies that are currently being explored to manage complexity for the

future vehicle systems domain and may be applicable to ULS system issues.

4.1 Product Line Development

Product Lines are concepts to deal with large-scale product development, which handle product variants of similar capabilities. It focuses on methods for planned reuse to reduce time-to-market and production cost of single products. Such concept of planned reuse and customization methods are critical for vehicle systems.

Product Lines are not a complete answer. Their success depends on the development of a large asset base to support a product line. It also depends on the inherent desire within product lines to drive to stabilities in the system definitions. Product Lines also have points of fragility in their target architectures and design languages. The fragile points often dictate the lifespan and applicability of the product lines' asset base.

While product lines typically target large volume products, or product families, much of the knowledge developed from them will be applicable to ULS. The desirable results from product line development include methods for large scale system development, assessment of contract based decoupling, utilization of component architecture, evolution management in parallel development, and techniques for composition. It is conceivable that a timely creation of an ULS system will be achieved through the composition of multitude of product lines in different domains.

4.2 Domain Specific Languages

For decades modeling has been advocated as a benefit of supporting abstraction and developing higher quality designs for large-scale systems. However, success has been limited, and modeling methods have faced significant challenges in navigating the cost-benefit curves to justify existence.

The unwieldy application of generic modeling languages to specific domains has not been productive. The tangling of dimensions of complexity by using the same notation and semantics has greatly reduced the value and applicability of generic modeling languages, and results in disruptive design that devalue the usefulness of models. The same notation and semantics in generic modeling languages further forces acceptance of a single lifecycle formulation whereas each dimension has a unique lifecycle of applicability.

While generic modeling languages lack the concise expressivity of dimensions that are natural within the domain space, domain specific languages (DSLs) enable focused language expressions. However, because of the natural isolation of DSLs from a generic modeling language community, research on the foundations of DSLs should be targeted to facilitate the development of core competences in the languages. The objective is then to utilize the knowledge in the DSL com-

munity to strike the balance of productivity and integration between and amongst other domains and languages to yield the benefits of decoupling which will allow for stronger design foundations for the developing and evolving ULS systems.

4.3 Aspect Oriented Methods

Aspect-Oriented (AO) methods have been developed to deal with the non-hierarchical nature of systems. Given not all perceptions of a system are decomposed in the same manner, the AO concept could be very successful in describing the multitude of perspectives and complexity dynamics in large-scale systems. Since ULS systems have such diverse perspectives, utilizing AO methods to individually characterize the perspectives and weave them to an implementation is beneficial and highly desirable.

While most research in AO methods has focused on those that support a single developer applying multiple perspectives to a design, using AO method to support large-scale teams becomes a very important growth area. It is conceivable that allowing one to express expertise based on his/her role through strongly-typed advice paths could be valuable. It can also facilitate the cooperative development of large-scale organizations. This can leverage the weaving technologies and better align design technologies with organizational structures and expertise.

AO methodologies need to continue growing to create more robust weaving technologies, including the methods of creating weavers. The weaver technologies will facilitate rapid development of new weavers as well as the ability to verify the generative results of weavers with respect to design properties. Additionally, the growth of weaving methods must support methods to manage the complexities and conflicts between aspects. Coupled with the development of weaving technologies, there must be more work to determine the right path of the AO methods, either as a generic AO language or more tightly-focused DSLs. Under either scenario, the implementations must leverage common underlying AO methods and frameworks.

4.4 Generative Methods

Generative methods are a collection of technologies and capabilities that allow the generation of derivative products from a source description. They fall under the implementation technologies such as meta-programming, model-driven architecture (MDA), and model transformations. While their implementations might vary, their objective is to improve developers' productivity and provide higher quality products.

Generative methods allow more expressive form to transformation that can extend the transformation, often beyond a standard one, to include more complex context-sensitive optimizations and elaborations using ap-

plication of design rules from multiple dimensions. The research must then leverage the intersection of models, aspects, and generative technologies to develop greater fitness to purpose. However, many of current generative methods demand a high investment cost prior to achieving the desired return. Growth is needed to reduce the entry cost of these methods for using in large-scale development.

The high quality and increased productivity are characteristics demanded by vehicle system development and ULS systems alike. What is unique is that ULS systems are some of the few applications which can absorb the cost of generative methods initial investment costs. It is likely that generative methods do not stand on their own but need support by other development concepts such as modeling, DSLs, and AO methods.

5. Conclusions

Vehicle systems show significant character of ULS systems, if they are not strictly ULS systems themselves. Although the development cycles are smaller than typical ULS systems, the rapid growth of vehicle development complexity introduce similar challenges. In ULS systems, it is the lack of scalability of the problem description and understanding that often restricts the development of desirable systems. The development challenge is in part due to the "wickedness" of such systems, i.e., one can obtain more knowledge only after the start of design, while such knowledge changes the one's perception of the problem, which then changes the solution. This appears to be a reflection of scale, recursive dependency, and the resultant uncertainty and the lack of methods to deal with these dimensions.

The growth in methods and technologies in developing ULS systems is critical to the stability of executing software systems and success of vehicle system development. The documented failure rate of current large system development indicates that we are not prepared to manage the size of future systems in a predictable manner. The pervasiveness and communication of computing resources is rapidly creating an environment of ad-hoc large scale, and potentially ultra-large scale, systems which impact users' daily lives. The methods to introduce stability in the developed and deployed planned ultra-large scale systems are required breakthroughs to aid in the development of robust systems which evolve into participating in ad-hoc large scale systems. The resolution of the challenges of ULS systems are critical for the development the vehicle systems of the future.

6. References

S. Mellor, J. Balcer, “Executable UML”, Addison-Wesley Professional; 1st edition (May 15, 2002).

K. Czarnecki, U. Eisenecker, “Generative Programming: Methods, Tools, and Applications”, Addison-Wesley Professional; 1st edition (June 6, 2000).

Kiczales, Gregor; John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Lopes, Jean-Marc Loingtier, and John Irwin (1997). “Aspect-Oriented Programming”, *Proceedings of the European Conference on Object-Oriented Programming, vol.1241*, pp.220–242.

P. Clements, L. Northrop, “Software Product Lines: Practices and Patterns”, Addison-Wesley Professional; 1st edition (August 20, 2001).

Software Engineering Institute, “Ultra-Large-Scale Systems: The Software Challenge of the Future”, June 2006.

D. Schmidt, “Model-Driven Engineering”, *IEEE Computing*, vol. 39, pp 25-32, February 2006.

Rittel, H., and M. Webber; "Dilemmas in a General Theory of Planning" pp 155-169, *Policy Sciences*, Vol. 4, Elsevier Scientific Publishing Company, Inc., Amsterdam, 1973

Y. Bar-Yam, “When Systems Engineering Fails --- Toward Complex Systems Engineering”, *International Conference on Systems, Man & Cybernetics*, 2003, Vol. 2, 2021- 2028, IEEE Press 2003.

R. Baillargeon, G. Rushton, “Model-Driven Product Line Software Development Process”, SAE 2005 World Congress and Exposition.

R. Baillargeon, J. R. Flores, “From Algorithms to Software – A Practical Approach to Model-Driven Design”, SAE 2007 World Congress and Exposition (to be published).