Quo Vadis, Virtual Platforms?

A Posse ad Esse, hic Platforma Virtutis Est!

Graham R. Hellestrand
Embedded Systems Technology, Inc.
San Carlos, California, USA
g.hellestrand@essetek.com

***Abstract*—System level Virtual Platforms are alive and well and in use in complex real-time distributed cyber-physical systems. These systems are both much more complex and more advanced than the QVVP'12 call for papers described. This short paper indicates the state of the art in system-level virtual platforms and outlines the process by which such platforms are generated, in part, automatically.**

Keywords - virtual platform; system architecture space; specification notations; differential equations; differential algebraic equations; decomposing architectures; translation to software; mapping software to execution units; mapping communications to networks; cyber-physcal systems

I. INTRODUCTION

Platformae virtutum are alive and well. Demonstrating themselves to be indispensable, well beyond the simple platforms of the 1st decade of this millennium. They provide high fidelity models supported by high performance simulation of entire cyber-physical systems and systems of systems involving thousands of vehicles. In general, cyber-physical systems (CPS) represent the set of architectures of real-time physical systems that consist of distributed plant – physical elements constructed from mechanic, thermodynamic, fluidic, electric, etc. technologies - and their networked electronic control units (ECU) that execute software to effect control. Systems and systems of systems do not admit formal proofs of equivalence to a specification - but that capability, in the verification arsenal, is not irrelevant for small parts of systems [1].

This leaves high fidelity modelling and simulation as the available verification technology. Even then, there is large, but finite, cost in time and money in both the high fidelity modelling and verifying of modern processors with their multiple, out-of-order, register renaming pipelines, and complex memory hierarchies. The cost and quality of the process is due in large part to the 100,000 or more tests in the processor silicon verification suites, which are required to be run with the model matching in function and timing the silicon artifact. The advent of multi-core computers is of huge benefit in the machines required to simulate large, networked, real-time controllers managing multiple coordinated plant models and running, potentially, tens of millions of verification tests. The usual parallelization of separable tasks effectively addresses the time taken to simulate systems using regression suites. However, simple-minded separation of system components, when executed on separate cores, may readily produce simulation run times significantly higher than the combined system simulating on a single core.

The usability of multi-core, single chip (MCSC) computers is not pervasive. The redundancy in MCSCs does not deliver fault tolerance and its advocacy in this use is ill-founded.

In the real-time CPS world of automotive systems design, there is a well-defined orthodoxy in which systems of differential equations (DE) (or differential algebraic equations (DAE)) are used as specifications of physical systems in which, initially, control is not separate from the model of physical plant that it controls. This seemingly arcane methodology presages the transition of real-time (control) system design into a largely automated process. So perhaps it is time for the virtual platform community to move along from the relatively simple world of computing platforms that just execute software. It is time to address the real-world platforms of software and hardware generated from systems of DEs/DAEs (often with a sprinkling of finite state machine (FSM) models) executing on modelled, networked controllers. [2] The controllers are ECUs that execute software and drive modelled actuators and sample modelled sensors in order to control the distributed plant. And cooperate so that the whole system meets both its functional and timing requirements. This is a somewhat different starting point for the discussion Quo Vadis, platformae virtutum?

II. 2012 STATUM ARTIS CUM PLAFORMAE VIRTUTUM

Platformae virtutum have the capability to support the high fidelity models of large-scale, real-time, networked control systems together with continuous domain plant that effect systems and sense their state. Such systems include terrestrial, aeronautical and nautical vehicles operating singly or in traffic or in transport systems involving all three types of models. This is quite a leap from the state of the art at the end of 2009. At that time, VaST Systems Technology Corp's technology led in high fidelity processor, bus and peripheral device modelling and simulation having enough performance from a single core of a computer to efficiently support real-time software development and undertake limited architectural work for 2 or so ECUs managing about the same number of plant models [3]. Such technology – even though it still maintains leadership in the ECU and software simulation arena - is insufficient to address the entire system, let alone the system of systems, modelling and simulation challenge.

III. 2012 STATUM ARTIS CUM SYSTEM SIMULATION

The state of the art in system simulation has taken a step function increase in capability beyond that available generally in 2009. The incremental advance in simulation performance from 1995 to about 2008 was halted by the current leakage problem in the silicon technology that consequently limited clock speeds. At the same time, the advent of increasingly capable multi-core computers from about mid 2007 to the present – forecast to continue throughout this decade - has been nothing less than stunning and, for distributed simulation technology at least, broke the barrier limiting single core performance. The continuing advance is evidenced by Intel's announcement of the availability of 10-core, hyper-threaded x86, single chip computing systems in 2012. This will seal the transition of super-computers to being the work-horse computers of the modestly funded laboratory – perhaps a 1,000 core machine for about US$100k by 2013/2014. With this spectacular feat of silicon technology – not computer architecture, today there is at least one provider of high performing, scalable simulation [EST 2011]. Using such systems, models of entire vehicles, powered by high fidelity models of internal combustion engines (ICE) and hybrid electric vehicle (HEV) drive-trains, can be simulated using various traffic scenarios to compute fuel consumption and gaseous emissions for the former, and state of charge for the latter. EST itself has demonstrated 2,000 low fidelity vehicle models, each equipped with EST's high fidelity IEEE 802.11p radio + IEEE 1609.4 DSRC MAC models - having both multicast & unicast communication modes - operating in traffic to demonstrate the unsafeness of the current global DSRC standard .

IV. ON ARCHITECTURE AND MAPPINGS – CREATING OPERATIONAL VIRTUAL PLATFORMS

A process for building virtual platforms is described below. It is distinguished by the use of a mathematical executable specification of the system that is used to drive the virtual platform building process and for verification of the developing modelled operational platform. The operational platform consists of continuous time plant, controllers consisting or discrete time electronics, and software and/or hardware directly derived from the mathematical control equations of the plant, and the networks connecting communicating controllers. The process is described briefly below.

A. Defining Terms

The use of these terms varies widely. Architecture – unless qualified by terms, such as, software – deals with the structural, functional, and timing attributes of parameterized CPSs - where the parameterization is of structure, function and timing. Behaviour we regard as a consequence of those 3 factors.

B. Notations that Simultaneously Capture Function and Time

In the world of real-time physical systems, the specification notation that has stood the test of mathematical enquiry is systems of differential equations that typically have time as an independent variable. These equations are parameterizable – with bound parameters - and give rise to an enumerable set forming an architectural space that can be searched for architectures satisfying a set of objective functions. Optimization at this level does not involve physical technologies – including software. Differential algebraic equations are frequently used in place of differential equations in this step [4].

C. Decomposing Monolithic System of Equations

Decomposing the system of monolithic equations into sets of equations representing physical subsystems (plant + control), together with their inter-communications is the next step. The function and timing of the set of decomposed equations must be verifiably the same as the monolithic set which behaves as an executable specification. Each of the equation sets might be further decomposable to some desired level of correspondence with some physical unit of plant + control. The equation set forms the specification of the physical unit. The level of decomposition may be set by some structural parameter.

For each modelled unit, the control and plant equations are separated from the executable specification, which also identifies the communication (feedback) paths between the plant and its control. The control is another set of DEs (or differential algebraic equations) that effect change in the plant via actuators, sample change in the plant via sensors, and coordinated connected units by controlling communications between them..

D. Translation to Software or Hardware – Almost No Hands!

For a number of specification notations - Simulink, Modelica, etc.- the equations defining control can be semi-automatically translated to either software or hardware. Within the foreseeable future, and much like the assembly code vs higher level language efficiency wars, the quality and size of the software generated in the translation process will be regarded as acceptably good. The outcome moves verification to an abstraction level above software. Providing the translation tools are demonstrably (ideally, provably) correct, this reduces the number of verification tests required and hence time to perform verification. In the software-dominated systems of modern controller design, this is a significant outcome.

E. Mapping Software to Execution Units

Primarily in this process, control has been associated closely with the plant of its specification. However, choosing an ECU with appropriate assisting hardware that is capable of executing the control software and guarantees the same timing and function characteristics as its defining equation set is the objective. This provides some freedom in determining whether ECUs can be shared between different threads of software controlling different plant. The scheduling of competing threads to access common resources must guarantee the satisfaction of the specification timing constraints imposed by their respective equation sets.

The real-time control system of a vehicle – drivetrain, braking, steering and suspension – may be pre-empted by active safety control in order to initiate autonomous crash avoidance actions involving any or all of the real-time control subsystems. In addition, fault tolerance will be required for safe autonomous

control. Both factors confound the ability to share readily single ECU resources between threads controlling disparate plant.

F. *Communicating between Controllers Linked by their Specification Equations*

A final detail is the mapping of communication between controllers via networks. Consideration of both latency and bandwidth are necessary and, as with all mappings to underlying physical resources, scheduling of access and residency must observe the constraints imposed at the abstract architecture level.

V. The Modelling and Simulation Test of Efficacy

The test of necessity is what can modelled systems, such as those described here, do that a corresponding physical system cannot.

1st Destructive Testing: In the same manner that modelling and simulation is used in the aerospace industry to test systems to destruction in order to assess likely hazards, the same should be true for all CPS modelling and simulation. It is poor engineering practice to kill test drivers, test pilots, passengers and by-standers. Increasingly, the killing of the same entities via emissions - both gaseous and particulate – is seen in the same light. In modelling, one can kill as many modelled human-objects as needed to investigate hazards and measure the degree of safety of systems. The integrity of such testing depends on the validation (demonstration that the models match the function and timing of their physical counterparts) of components and sub-systems that make up the system.

2nd Verification: The exhaustive verification of complex physical systems – particularly CPSs - is not possible within realistic times and for realistic cost. Models can be used to perform orders of magnitude more testing than can be done using physical artifacts. This approach to testing is enabled by the modern computer and the technology that enables thousands of them to be deployed concurrently in executing partitioned tests. In CPSs, this level of testing is usually a necessity.

3rd Optimization: The systematic optimization of complex systems depends on a set of objective functions and the ability to traverse a parameterized architectural space of CPSs. Such architectural spaces involving vehicles and traffic, for instance, may be extremely large and irregular [5]. A general approach to optimizing CPSs, is to use an empirical process, such as Design of Experiments. Whether empirical or more conventional mathematical search processes are used, the requirement is to measure the responses of the objective functions for each point of interest in the architectural space. This may necessitate measuring the responses of thousands, or even hundreds of thousands, of models of even relatively mundane CPSs. For a modelled and parameterized CPS, it is possible to simulate concurrently thousands of its various architectures – each characterized as a vector of its parameter values. Where these parameters are structural, the set of physical counterparts may represent quite different physical builds of a vehicle. It is not feasible to construct enough physical vehicles to enable a sufficient traversal of the architecture space to be confident that an optimum vehicle has been produced. Using physical vehicles does provide a practical approach to optimization, which is the reason that it is not done as part of the current automotive engineering process – even though particular control systems might indeed be optimized - typically, using model-based design.

The adequacy of the non-modelling approaches to automotive and other CPS engineering is demonstrated by their failure to meet even necessary conditions of safe and cost effective engineering. This therefore establishes the efficacy of the model-based CPS engineering process.

VI. Summary & Conclusions

This short paper discusses a number of critical issues in systems engineering that highlight the fundamental need for model-based, cyber-physical systems engineering processes based on specification and operational virtual platforms.

In doing so, it underscored the role of platformae virtutum in enabling safe, efficient and effective CPS engineering processes. And in the process indicated that the industry standard automotive CPS engineering process is not safe or efficient in either a technical or economic sense.in regard to the optimization of products.

Happily, I report, the advance of platformae virtutum well beyond the scope that triggered this workshop - thereby establishing a base for an improved CPS process that will deliver safe, optimized products.

Acknowledgment

The author wishes to acknowledge the contribution that many engineers and engineering managers in automotive and communications companies (OEMs, Tier 1s and Tier 2s) have made to the ideas presented above. Despite the odds, they persisted over the last 15 years in introducing the real world to a repentant academic.

References

[1] Hellestrand, G.R. (2005): Systems Architecture: The Empirical Way — Abstract Architectures to 'Optimal' Systems. Democritus and Plato Re-engage Digitally 2,400 years on! 5th ACM Intl. Conference on Embedded Software, Jersey City, USA, Sept 2005, pp 146-157.

[2] Abdallah, A., Feron, E., Hellestrand, G., Koopman, P. & Wolf, M., "Hardware/Software Co-Design of Aerospace and Automotive Systems," Proc. IEEE, April 2010. (IEEE)

[3] Winters, F.J., Mielenze, C. and Hellestrand, G.R. (2004): Design Process Changes Enabling Rapid Development. Proc. Convergence 2004 P-387, Oct 2004, 613-624, Society of Automotive Engineers, Warrendale, PA.

[4] Hellestrand, Graham R. (2006): Quantitative Embedded System Architecture and Performance Analysis. Ch 7 in Platform Based Design at the Electronic System Level. Ed. Burton, Mark and Morawiec, A, Springer.

[5] Ahmed Abdallah, Wayne Wolf, Graham R. Hellestrand: Using Empirical Science to Engineer Systems: Optimizing Cache for Power and Performance. DSD 2008: 325-333.