RADICAL PRODUCTIVITY IMPROVEMENT WITH ONE PASS TO PRODUCTION (OPP)

Ravi Shankar
Florida Atlantic University
777 Glades Road
Boca Raton, FL 33431
(561) 297-3470
ravi@cse.fau.edu

Jaime Borras
Motorola
8000 W. Sunrise Blvd
Plantation, FL 33322
(954) 723-3797
Jaime.Borras@motorola.com

Abstract

One Pass to Production (OPP) has the goal of reducing the product development time of cell phones from 2 years (in year 2001) to 1 day. In a culture that is geared to continuous improvement, this initiative for radical improvement is both a great opportunity and a challenge. We are in the fifth year of research collaboration of an 8 year project. Allied technologies and methodologies have continued to mature during this period, giving us a rare opportunity to pace our development process almost in synch. Balancing the long-term OPP objective with short term technology transfer to keep the project alive has been illuminating and productive. This paper will provide a technical overview of the OPP flow and briefly discuss the equally important issues of technology transfer and academic training.

Acronyms Used

APD – Advanced Product Development Group
CA – Cycle Accurate software/system model
CPD – Current Product Development Group
DE – Discrete Event software/system model
EDA – Engineering Design Automation
FPGA – Field Programmable Gate Array
HW – Hardware modeling zone
IA – Instruction Accurate software/system model
MDA – Model Driven Architecture
MDD – Model Driven Design
MOC – Model of Computation
OPP – One Pass to Production
QOS – Quality of Service
SDL – Specification and Description Language
SW – Software modeling zone
SysML – System Profile of UML
TDD – Test Driven Development
TLM – Transaction Level Modeling zone
UML – Unified Modeling Language
UT – Untimed software/system model
xUML – Executable UML

Background

A shorter product development cycle allows a company to have predictable revenue since customer preferences do not have to be forecast substantially ahead of time and even a stumble can be corrected quickly and economically. The perception that this shifts costs around without economic benefit is easily disproved by studying the field of consumer electronics where the costs have come down, while the complexity has risen, thanks to miniaturization and increase in chip design productivity (and concomitant reduction in engineering cost). Complex embedded system design, while amenable to many design automation concepts of chip design, also has many unique challenges – interaction with non-engineering stakeholders, concurrent development of software and hardware, rapid technology change, increasing application complexity, and finally, integration challenges of a real-time embedded system. A typical waterfall-type development cycle has discrete (but overlapped in time) stages of Requirements (40% of product development cycle), Design (10%), Development (10%), and Prototyping/Testing (40%), fully reflective of these challenges. Professionals may be unable to communicate across disciplines leading to this behavior of discreteness. Success stories with notable gains within a stage may not lead to overall cycle improvement. Thus, we made attempts to understand the whole process and identify opportunities for improvement. This OPP initiative was started internal to Motorola [1] and became an university-industry alliance in 2003. While OPP is still an evolving process, there

1-4244-1041-X/07/$25.00 ©2007 IEEE

seems to be a clearer picture today of significant productivity improvement.

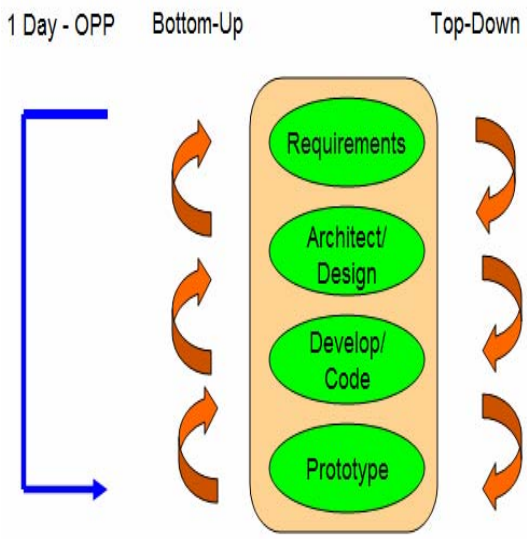
Introduction

Figure 1 depicts the One-day OPP in relation to top-down and bottom-up methodologies of product development. One needs a good mix of both top-down and bottom-up methodologies in developing a complex embedded system. In our approach, we use this to develop parameterizable (or customizable) subsystems that are developed a priori so that they can be integrated to develop our system in a day. The OPP system (subsystem) development involves a middle-out methodology to customize the subsystems (components) to meet the required QOS (quality of service) requirements. Figure 2 tries to provide our over-arching philosophy in developing the flow. The middle block represents the product development stages, with the up-pointed arrow indicative of abstraction and ambiguities. The right block represents availability of solutions, with today's methodologies maturing into languages and standards in the interim and eventually implemented in tools in a few years. This also implies that today's tools are not useful to address the needs a few years hence. The left block identifies the EDA (engineering design automation) methodologies that can be exploited in improving productivity. Abstraction is identified as the most mature, while acceleration is recognized as the most effective in demonstrating improved productivity. However, all the methodologies are needed in enhancing productivity.

While the challenges seem different in different stages, we did use certain common principles in developing our three layered (components -> subsystem -> system) flow:

Exploit Exponentials – Every domain where increase in size (or improvement in a quality) leads to exponential (as against a linear) increase in complexity is ripe for exploration. Complexity increases in software, hardware, and testing have been addressed by decomposition, multi-processing, and hierarchy respectively. Layered architecture allows one to hide and manage a continuum across different dimensions (hardware and software, for example). We have used this to manage developmental interactions across various engineering and other disciplines (via semantic web and multiple models of computation). A different type of exponential curve is noticed in the economics of fault identification. Figure 3 [adapted from 2] shows that an error introduced in the requirements phase will cost five times more to fix in the next downstream design stage if it is not fixed in the requirements phase. Further delay causes a rapid increase in cost. Figure 4 [adapted from 3] is another interesting exponential behavior that shows why software practitioners do not strive to reach a high system accuracy in their simulations: their productivity will fall rapidly. Both these (and other similar ones) represent challenges to be overcome if one desires to enhance productivity.

Figure 1: OPP Flow relative to other methods



- Adopt The Emergent: In any field, at a given moment, there are various similar and competing methodologies (as an example System Verilog and SystemC in 2001 or SDL and UML in 2003). It takes a few years for a winner to arise. We decided to objectively evaluate early on and pick the more suitable option, to avoid domino effect. This required more up front effort, but has proven useful in reducing complexity in decision making. Over the years, we have adopted the

following: SystemC for codesign [4], UML [5], Concurrency modeling [6], Component based design [7], multiple MOC - models of computation [8, 9], Semantic Web [10], MDA [11, 12], Eclipse [13], TDD - Test Driven Development, Agile MDD [14], and XUML [15].

Figure 2: OPP Automation Horizons

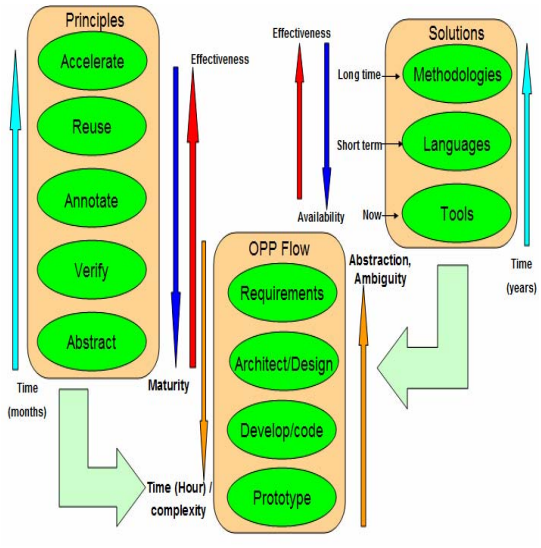
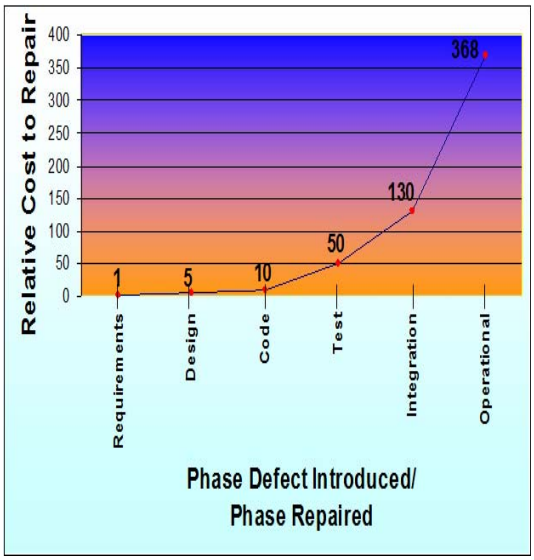


Figure 3 [2]: Cost of later error detection

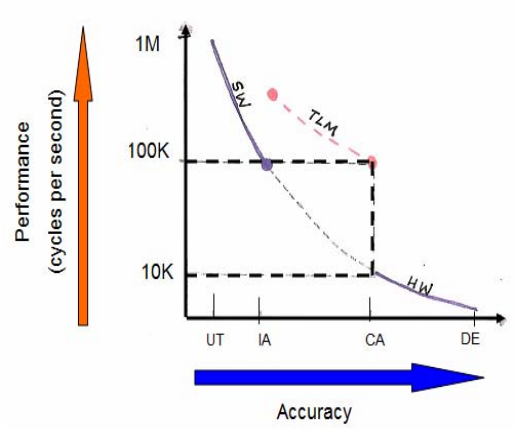


- Listen To The Experts: Dr. Dobb's Journal spotlighted in 2004 [16] the need for concurrent programming, as the Moore's law torch bearer, viz., the single processor, seemed to be losing its steam, thus paving the way for multiprocessor and NOC (network on chip) architectures to shine. Unfortunately, this changed the software paradigm from sequential to concurrent programming. And concurrent systems, though capable of higher performance relative to a single processor, are harder to program and also can fail mysteriously. Thus, given that the future systems would have tens to hundreds of processors, concurrency modeling at an abstract level seemed absolutely essential [6, 17, and 18]. Success of XML and Wiki, and passion of T. Berners-Lee, seemed to portend the success of Semantic Web. We use it for architectural component selection [19], access to a multi-disciplinary literature database to speed up search for related ideas in other disciplines, and ontology for the requirements and specification phases [20]. TDD (and Agile Verification) promises to increase productivity by reversing the order of the development (design) stage and the test (verification) stage [21, 22, and 23]. Finally, Eclipse, the open source environment, promises to not only give us access to both free and commercial plug ins, but allow us to develop our own to complement the capabilities that exist there. Innovations by others are helping us to fill in the gaps – a very good example is the evolution of MSC (message sequence charts) as a front end for the concurrency modeling tool we use [24].

- Empower The Engineer: The university group has developed many new courses, on SystemC, UML, Concurrency Modeling, Networks on Chip, Components, Performance Evaluation (with MLD that supports multiple MOC), Rapid Prototyping (with ImpulseC that supports software-hardware codevelopment), and Semantic Web. Motorola has hired many of the students trained in new methodologies and technologies. It is hoped that they would disseminate the new knowledge among the current pool

of engineers. Further, many Motorola engineers are enrolled in the university weekend MS program where they get exposed to these new concepts. This methodology transfer has become increasingly effective over the years. However, we do recognize the need for better knowledge transfer. Our semantic web initiative with Confluence (Enterprise Wiki Software) for an integrated literature database is focused on this need.

Figure 4 [3]: Simulation Performance Vs Accuracy.



Methodology

We initially articulate the organizational issues and challenges and how they shaped the process of development. The collaboration is between the university researchers and the APD (advanced product development) group at Motorola. APD defines the roadmap of technologies for future products. Any recommendation from the OPP group, after initial whetting by the APD group, would have to pass the acceptance test of current product development (CPD) teams. Thus, the proposed change would have to allow the CPD (1) to smoothly transition and (2) to realize a potential gain in the current process. To put another way, the long-term objective must have enough short-term wins for pro-active adoption. There were also heated discussions on the advisability of attempting this when continuous improvement could only yield a few percentage point improvement annually. While it is easy to describe the current methodology succinctly, our experiences in arriving at the current methodology, and the trials and tribulations that have occurred in the process, are important lessons for others.

Initially, we focused on SystemC, a software-hardware codesign language, as that seemed a natural way to extend the EDA concepts to higher levels. However, software programmers accustomed to significantly higher simulation speeds with ISS (instruction set simulators) did not see the benefits of this (see Figure 4) – that this would reduce the integration testing later on (see Figure 3). More abstract transactional level modeling (TLM) and acceleration were attempted to bridge the gap. An independent effort within Motorola has indeed shown the effectiveness of this approach [25]. Five of the university graduates trained in SystemC were hired by Motorola to promulgate the technology.

Later on, we shifted the focus to UML and autocode generation, only to realize that reality was different from claims. However, SDL appeared to be too engineering oriented that UML seemed a better choice for interaction with non-engineering stakeholders (such as Marketing and Service Providers) and software practitioners. The UML domain has progressed considerably since then, as others found it an useful methodology and pushed the envelope. Concurrent and independent development within Motorola has yielded significant autocode generation and productivity improvement [26]. Two more students trained in UML, concurrency modeling, and MLD were hired by Motorola to disseminate these technologies.

We are at present exploring xUML (executable UML) [15], a subset of UML, MDA (model driven architecture) [11], and MC (model compiler), in an agile methodology [14], to develop the product iteratively while exploring limited architectural options.

We also decided early on that in order to optimize our domain specific systems (mobile embedded systems with several applications, and with many QOS metrics, such as long battery life), multiple MOC had to be supported. We found it in Ptolemy [8] and its commercial form [9]. This made better sense to us as compared to transforming all requirements to a common MOC, as with Metropolis [27] – this

would have forced professionals to use an alien MOC, which may have led to non-optimal designs. The cell phone became a marvelous invention and a commercially successful product because it allowed professionals in various engineering disciplines to show their creativity in their domains of excellence. However, it has also become too complex to allow individual expression of creativity. So, the question was whether to ask all to map their expertise to a single MOC, or to allow them to express themselves in their favorite MOC, with an expanded role for a system architect to integrate these MOCs and do what-if scenarios so as to meet QOS requirements. We chose the latter option. However, the university's attempt to implement two designs (with 2 and 3 MOCs) using MLD failed – in fact, five graduate students failed in their attempts, despite their confidence that their model would work soon.

Search for answers led us to the same scenario as happens with integration: when different designs that worked in stand-alone mode were integrated, the integrated system failed. Subsequently, when we introduced abstract concurrency modeling [6] prior to MLD modeling, the MLD modeling not only worked, but the overall process showed a three fold improvement in productivity, with a functioning system at the end [17, 18]. To do QOS evaluations with MLD, we had to annotate the modules with performance characteristics, such as execution time, latency, noise, power and energy dissipation, and size metrics (area / code size), as well as their run-time requirements, such as memory and I/O needs. We developed our first few annotation methods and have since then identified a few more for this purpose [28]. All of these methods essentially estimate the characteristics and requirements of the module in isolation, in terms of computation and communication costs (cost is a generic term that describes various QOS metrics). Once they are modeled on MLD, an additional cost, viz., the concurrency cost, can be estimated. Performance measures subsequent to that give us the ability to decompose software onto different multiple processor architectures and determine a more optimum mapping that meets the QOS requirements [17]. Our current effort here is related to static optimization given the QOS metrics to be met [19]. A future goal is to evolve run-time optimization so as to increase user satisfaction and battery life.

Two other current initiatives are in the areas of Test and Verification, and Knowledge Management. The former involves TDD and methodology for integrated verification [21, 22, 23], while the latter is focused on the use of Semantic web at various levels of the hierarchy. An example is our paper on the use of SysML in the requirements phase [20]. Use of the semantic web and agile methodologies is expected to expand significantly over the next few years of the OPP project.

Results

In brief, our methodology uses both top-down and bottom-up methodologies with a significant component of middle-out modeling and analysis. Many of the current innovations/realities, such as UML, MDA, xUML, TDD, Concurrency, Semantic Web, SystemC, Design Patterns [29, 30], and reuse [31, 7] find a synergistic role in our overall methodology. Our cost model shows the feasibility of a one-month product development cycle with significantly reduced development cost. However, the limiting factor there might be the availability of low power FPGA-like technology, though much is changing there also [32].

Discussion

Meaningful benchmarks that Motorola CPD groups prefer are difficult to undertake in an university environment. However, concurrent and independent success stories within Motorola have kept interest alive in our methodology [25, 26]. Knowledge and technology transfers have occurred from the university to Motorola (and vice versa) successfully in several cases over the years, but it remains a challenge. A typical CPD engineer at Motorola has immediate deadlines to meet – and the OPP innovations must help him/her to succeed better in the job on hand. On the other hand, OPP focus is long-term, in developing a viable and radically different alternative to the current flow. Much patience and dialog between the two sides is warranted to achieve this goal. Significantly more effort on requirements productivity enhancement is planned for later [33].

Kennedy documents the magic of Toyota [34] in terms of its phenomenally high development productivity. The Lean development system documented there has five ingredients: entrepreneurial system designer leadership,

responsibility based planning and control, expert engineering workforce, set-based (as against infinite) concurrent engineering, and (recognition of) operational value stream to customer. The OPP project, though of a much smaller scale, seems to have followed this paradigm, due to conscious effort on both sides. It will be interesting to see its impact on a real product development cycle.

Conclusions

Motorola and FAU are involved in a long-term project to develop a radically different product development flow. This is envisioned to reduce the current product development period of 12 to 24 months to 24 hours. Inspiration from, and advances in, many other domains makes this a feasible goal. Judicious evaluation and use of these technologies, with the help of many test cases and benchmarks, has led us to a viable solution that may be useful for developing complex systems in other application areas.

Acknowledgment

This project is in its fifth year of funding from Motorola. Motorola engineers and FAU researchers constantly interact. This helps engineers voice their issues and challenges, and for the researchers to fashion better solutions that address the engineers' concerns. The researchers gain confidence by developing practical and useful solutions. We are grateful to the many participants on both the sides.

References

1. Borras, J., One Pass to Production, A Vision, Motorola Internal Document, 2001.

2. Bennett, T.L., and Wennberg, P.W., Eliminating Embedded Software Defects Prior to Integration Test, Cross Talk, December 2005, pp. 13-18

3. www.coware.com

4. Grotker, T., et al., System Design with SystemC, Kluwer Academic Publishers, Boston, MA, 2002

5. Ambler, S.W., The Elements of UML 2.0 Style, Cambridge University Press, Cambridge, UK, 2005

6. Magee, J., and Kramer, J., Concurrency State Models & Java Programming, 2nd edition, John Wiley & Sons, Ltd., Hoboken, NJ, 2006

7. Wolf, W., Computers as Components: Principles of Embedded Computing System Design, Morgan Kaufmann Publishers, San Francisco, CA, 2001

8. http://ptolemy.berkeley.edu/

9. http://www.mldesigner.com/

10. Passin, T.B., Explorer's Guide to the Semantic Web, Manning, Greenwich, 2004

11. Raistrick, C., Model Driven Architecture with Executable UML, Cambridge University Press, Cambridge, UK, 2004

12. Gasevic, D., et al., Model Driven Architecture and Ontology Development, Springer-Verlag, Berlin, Germany, 2006

13. Yang, Z., and Jiang, M., Using Eclipse as a Tool-Integration Platform for Software Development, IEEE Software, March/April 2007, pp. 87-89.

14. Ambler, S., The Object Primer, Agile Model-Driven Development with UML 2.0, 3rd edition, Cambridge University Press, Cambridge, UK, 2004

15. Mellor, S.J., and Balcer, M.J., Executable UML, Addison-Wesley, Boston, MA, 2002

16. Sutter, H., The Free Lunch is Over: A Fundamental Turn Toward Concurrency in Software, Dr. Dobb's Journal, 30(3), March 2005.

17. Agarwal, A, and Shankar, R., A Concurrency Model for NOC Design Methodology, IEEE Conference of High Performance Computing, Massachusetts Institute of Technology, September 2006

18. Jain, A., and Shankar, R., Software Decomposition for Multicore Architectures,

IEEE Conference of High Performance Computing, Massachusetts Institute of Technology, September 2006

19. Agarwal, A., et al., An Integrated Methodology For Qos Driven Reusable Component Design And Component Selection, IEEE Systems Conference, Hawaii, April 2007

20. Cardei, I., et al., Framework For Requirements-Driven System Design Automation, IEEE Systems Conference, Hawaii, April 2007

21. Mattu, B.S., et al., One Pass To Production (OPP) Test Driven Design Methodology For Component-Based System, IEEE Systems Conference, Hawaii, April 2007

22. Fraser, J., and Mattu, B.S., Test Driven Design Challenges, IEEE Systems Conference, Hawaii, April 2007

23. Choi, J.P., et al., Unified Test Environment-Integrated Platform For Bridging The Modeling, Testing And Code Development Flow, IEEE Systems Conference, Hawaii, April 2007

24. Uchitel, S., Incremental Elaboration of Scenario- Based Specifications and Behaviour Models Using Implied Scenarios, Ph.D Dissertation, Imperial College of Science, Technology and Medicine, London, UK, February 2003.

25 Ruff, A.V., System Level Simulation as a Platform for Software Development, IEEE Systems Conference, Hawaii, April 2007.

26 Foustak, M., Experiences in Large-Scale, Component Based, Model-Driven Software Development, IEEE Systems Conference, Hawaii, April 2007.

27 embedded.eecs.berkeley.edu/metropolis/

28. Shankar, R., et al., Annotation methods and application abstraction, IEEE Int'l Conference on Portable Information Devices, Orlando, March, 2007

29. Gamma, E., et al., Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, Upper Saddle River, NJ, 1999

30. Douglass, B.P., Real-Time Design Patterns, Object Technology Series, Addison-Wesley, Boston, MA, 2003

31. Jensen, R.W., An Economic Analysis of Software Reuse, Cross Talk, December 2004, pp. 5-8

32. Morris, K., Power – Suddenly, We Care, www.fpgajournal.com, April 26, 2005

33. Decker, B., et al., Wiki-Based Stakeholder Participation in Requirements Engineering, IEEE Software, March/April 2007, pp. 28-35

34. Kennedy, M.N., Product Development for the Lean Enterprise, The Oaklea Press, Richmond, VA, 2003