2016 23rd Asia-Pacific Software Engineering Conference

An Incremental V-Model Process for Automotive Development

Bohan Liu
Software Institute
Nanjing University, China
Email: hifibohan@gmail.com

He Zhang
Software Institute
Nanjing University, China
Email: hezhang@nju.edu.cn

Saichun Zhu
Research Institute of
Beijing Automotive Industry Corp., China
Email: chengderen@hotmail.com

***Abstract*—V-model and its variants have become the most common process models adopted in automotive industry guiding the development of systems on a variety of refinement levels. Along with the exponentially growing complexity of modern vehicle systems, however, the late verification and validation in the conventional V-model expand in uncontrollable ways that result in higher cost of development and higher risk of failure than ever. This paper describes an inc-V development process for automotive industry that improves the conventional V-model and variants by introducing and institutionalizing early and continuous *integrated verification* enabled by simulation-based development. We developed a continuous simulation model of the inc-V process, and the initial version is used to investigate the characteristics of the inc-V compared to V. The preliminary finding from the simulations of an example project is that the inc-V process is able to improve the traditional V process by saving effort, shortening duration, and increasing product quality. The finding also show how the advance of development technology impacts the systems engineering processes.**

***Index Terms*—process simulation, systems engineering process, inc-V process, V-model, simulation-based development**

I. Introduction

Today, the automotive industry is faced with some of the most difficult challenges in its over 100-year history. Environmental pressure in developed markets and rapid sales growth in emerging economies (such as Brazil, Russia, India, and China) are stretching the product portfolio of OEMs. The need to adhere to environmental regulation worldwide has pushed automakers into very complex technologies, such as low emission engines, hybrid vehicles, and electric vehicles. The resulting higher intensity of embedded (software) systems in cars and higher complexity of development processes put the automotive industry under stress.

In a modern car, almost all functions are electronically controlled and also interlinked. Vehicles had changed from machinery to systems of E/E (Electrical and Electronic) systems. Cars have up to 2500 software controlled functions, representing 10M (million) lines of code (LOCs) [1]. The code size is increasing rapidly to 100M LOCs in the recent years. The E/E systems determine over 40% of a vehicle development cost, most of which are directly related to software. This led to a leap in the complexity of the resulting overall E/E systems to a degree that the established vehicle development processes efficiently creating high quality mechanical systems were not able to cope with.

On the other hand, the recent global financial crisis has ushered in a monumental shake up in the automotive industry. The burden of legacy costs, broken product development processes, and poor quality have been documented at length [1]. Automakers are faced with a tremendous challenge meeting the rapidly changing needs in marketplace, as well as the increased diversity of customer segments. To tackle the challenge, a research performed 20 case studies of 10 significant product successes and 10 significant product failures in the American automotive industry over the past 50 years (1950s-2000s) provides statistical inferences regarding the factors driving the product to success and failure [2], and indicates '***development process***' is one of the two most critical factors, out of 14 key factors (e.g., '*technology*', '*fuel economy*', and '*safety*'), in success or failure of a new automotive product.

So far, however, the development processes in car industry are not well adapted to the needs of software intensive systems and software and systems engineering. For example, the traditional V-model development process often incurs very high cost during the late verification stage and linger response to the changes from customer and market, especially when system complexity is considerable. Although some V-model variants (e.g., W-model) were proposed, unfortunately, their '*promised*' improvements are difficult to be validated in varying global industrial contexts.

In this paper, we introduce and describe a new 'inc-V' process model for automotive development with emphasis on E/E systems. It is based on traditional V-model and enabled by the simulation-based system development techniques. We also report our initial attempt on investigating the impacts of this enabling technical advance plus the resulting process changes (early and continuous verification in the inc-V model) on the overall performance of the development project using process simulation, in particular project duration, effort, and quality.

This paper is structured as fellows. The next section describes the inc-V development process that is derived from the traditional V-model and enabled by simulation-based development technology. Section III elaborates a simulation model that is used to simulate and investigate the inc-V process. It is followed by an initial example of the inc-V process simulation and the preliminary findings in Section IV. Section V briefly reviews the related work. The paper ends up with the discussion and conclusion in Section VI.

1530-1362/16 $31.00 © 2016 IEEE
DOI 10.1109/APSEC.2016.56

II. INC-V SYSTEMS ENGINEERING PROCESS

Nowadays, to remain competitive in an ever more global business environment, automotive manufacturers increasingly work across geographical, socio-cultural and technical borders. They need to build ever more complex systems but release them faster. Being derived from the traditional V-model and enabled by simulation-based system development technology, an inc-V development process is devised to leverage modern vehicle development in practice.

A. Traditional V-Model and Variants

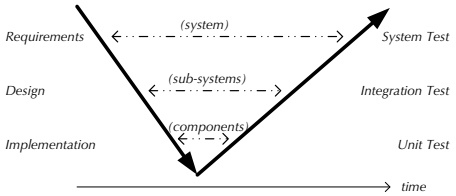
The **traditional V-model** is derived from the waterfall model but with emphasis on verification and validation activities in development processes [3]. Being established process model in systems engineering, the V-model (Figure 1-a) allows a deeper understanding of the interplay of creative and analytical processes over the course of a systems development. To be specific to vehicle development project, for example, starting with the specification of the desired complete vehicle characteristics, downward movement in the V-model (along the left leg of the V) denotes decomposition and specification - from complete system requirements to system design down to parts specification, design and evaluation of parts. From here upward (along the right leg of the V), the systems created out of the designed components are tested and validated against their specification in a hierarchical order - from components over sub-systems up to the complete vehicle. The V-model also represents the different levels of refinement of the systems, e.g., *requirements* and *system test* (validation of systems) happen at the same level.

In systems engineering, the deliverable and test object are not just executable code. The system is usually developed in a sequence of product-appearances that become more '*real*'. Hence, the **multiple V-model** (W**-model**) was introduced as a typical variant of the V-model (Figure 1-b). When developing an embedded E/E system in vehicle, for instance, first a *model* of the system is built upon a computer, which simulates the required system behavior. Once the model is found to be correct, code is generated from the model and embedded into a prototype. The experimental hardware of the prototype is gradually replaced by the real hardware until the system is built in its production form [4]. The reason behind such an approach is that it is cheaper and quicker to change the prototype (even the model) than to change the final product. The multiple V-model, based on the traditional V-model, takes the *sequence of product-appearances* into account.

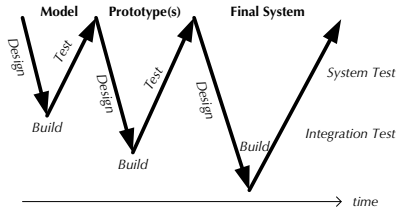
The V-model and its variants have become the most common process reference models adopted in vehicle development.

B. Model- and Simulation-Based Development

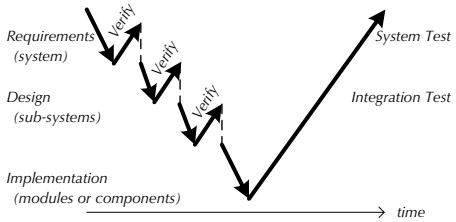
Model-Based Development as a widely employed development technique in automotive industry, addresses the heterogeneity of modern vehicle systems by focusing on computational models as the core design artifact [5]. The model enables a hierarchical design process where the entire system is first represented at an abstract level while model



(a) Traditional V-model



(b) Multiple V-model (W-model)



(c) inc-V model with early and continuous verification

Fig. 1. Traditional V, W and inc-V process models

elaboration iteratively refines this design and includes details as necessary to implement the required functionality. Thus, different models that may be playing different roles are required for the main stages of the development: specification, test and validation, and consecutive refinement. The ability to efficiently construct models combined with associated tools and systematic methodologies primes model-based techniques for success by providing a complete solution that enables concurrent engineering, performance analysis, automatic verification, building efficient specifications and execution models, code generation and optimization, and automatic refinement through different abstraction levels [6].

Although the model-based development enables simulation in some early development phases based on behavior modeling, it provides limited verification that only relates to the individual model. In other words, it is difficult to support verification across models on system or even subsystem levels. It is because in the model-based development most of these behavior models are platform independent, which implies the vehicle system (or subsystem) can not be verified until the integration of these models with the platform (i.e. integration test or system test). When the vehicle systems become larger or more complex, the benefits of model-based development turn to be slimmer.

Theoretically there may exist two approaches to solving the above problem. First, the barrier to early integrated verification might be avoided by applying a unified platform and modeling language. Unfortunately, it is unrealistic in most cases

in automotive industry. In real vehicle development, OEMs, suppliers and application vendors may employ various types of platforms implemented in different ways in a distributed environment, which is difficult to alleviate using the conventional model-based development. The other is to provide an interface that establishes effective communication across the platforms, and further enables the interplays among the models.

Simulation-Based Development is an advanced vehicle development technology that extends the conventional model-based development in various aspects. The models, which might be built upon diverse platforms, in their specific languages, and on different refinement levels, are able to communicate with each other within the framework [7]. As a result, it enables model-based virtual integration of vehicle systems (or sub-systems) across the platforms, and further makes the *early integrated verification* possible.

Some advanced systems development techniques related to simulation-based development appear recently. For example, System Architecture Virtual Integration (SAVI) [8], which is solely based on AADL modeling language, is a verification technique promoted by the Aerospace Vehicle Systems Institute (AVSI) with focus on the aviation industry. Another instance is Embedded Systems Technology's (ESTs) Specification and Simulation Environment (ESSE).

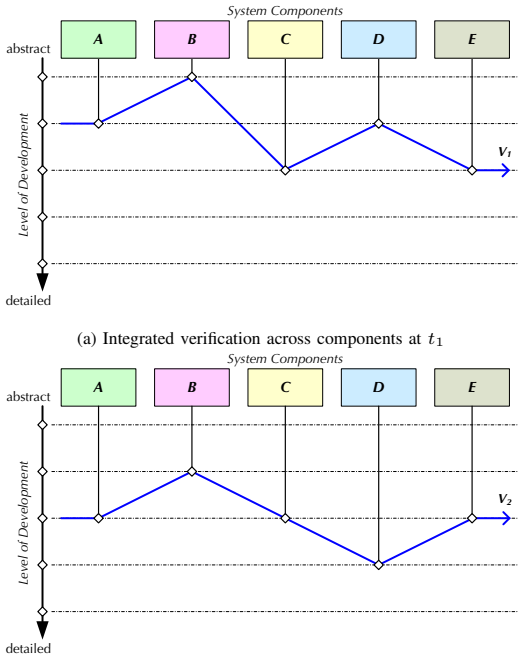
C. Analytical vs. Integrated Verification

Verification and validation (V&V) are two important activities to ensure the product quality in software and systems engineering. To be specific to a V-model process, verification is about evaluating the product against the specifications until the system test or acceptance test. Typical verification techniques may include *review-based verification*, such as inspection, requirements tracing, code review and so on; *static analysis*, such as formal or semi-formal static (non-executable) analysis (e.g., fault-tree analysis); and *simulation and prototyping*, which use dynamic techniques to evaluate a design (e.g., performance simulation). Software testing consists of the dynamic verification of the behavior of a program on a finite set of test cases, suitably selected from the usually infinite executions domain, against the expected behavior [9].

In order to clarify the different concepts of '*verification*' in this paper, we use **analytical verification** to represent the above *review-based verification* and *static analysis*. Note that sometimes the requirements specification can be translated into temporal logic, and then verified by a model checker. This is still classified into *analytical verification* in our study.

On the other hand, we use **integrated verification** to indicate that the product (virtually or physically integrated) is being verified by executing test cases. Both *analytical verification* and *integrated verification* can be used together or in isolation in systems development.

Figure 2 shows a simplified example that illustrates the simulation-based development and integrated verification. The system is supposed to have five components. They are being developed by different suppliers or vendors, based on distinct platforms, and progressed at various levels. At time t_1, the virtual system (with the five components), though on different refinement levels, can be integrated and verified by v_1. Due to a number of factors, e.g., complexity and size, the progress in developing these components might be different. Comparing t_1 and t_2 (Figure 2-a and -b), the advance of Component D's development is significant, while Component C and D almost remain their levels. The simulation-based development also allows another virtual integration and verification (v_2) at t_2.



(a) Integrated verification across components at t_1

(b) Integrated verification across components at t_2

Fig. 2. Integrated verification capability by simulation-based development

D. inc-V Process Model

An early '*integrated*' model-based verification is very challenging as different teams use their preferred modeling languages and work on different platforms. For example, the electronic control unit (ECU) models, the engine models, the whole car models and the car-in-traffic models are all built on separate system scales and refinement levels using different set of languages and tools. No unified application platform and modeling environment exist so that the integrated verification was impossible. All this makes the upward (right) leg of the V model expand in uncontrollable ways when the system becomes more and more complex or the requirements have to be changed later in the development life cycle. This situation is exacerbated when a complex vehicle product is developed together by multiple organizations in a distributed setting.

A number of new technologies are aimed to solve the problem. For instance, ESSE environment, supported by simulation-based development technology, focuses on letting the models for various platforms, at different levels of abstraction, in different modeling languages communicate to each other during early and integrated simulation. The technologies also use the new cheap multi-core CPUs to make

such approach scalable. This has significantly changed the conventional V-model development process into a new 'inc-V' process. Although people were able to verify early design artifact (models) in various analytical ways such as inspection, verifying the whole system by virtually integrating and simulating a set of models differing in a number of aspects (e.g., platform, abstraction, language and environment) is impossible without these new enabling technologies.

In the inc-V process, models and test cases at different levels of abstraction for different parts are constantly developed and refined, and more importantly, integrated. Simulations are constantly run to verify the *virtual* system (or sub-systems) rather than waiting the integration test and system test later. In Figure 1-c, the early '*verify*' upward leg are not just for verifying early design artifacts in isolation (like conventional *analytical verification*) and once but with other pre-existing models (such as models of reused components or from suppliers) and constantly throughout the life-cycle. Such inc-V processes may vary depending on the frequencies one wants to run the integrated simulation and the model exchange policies across multiple divisions and companies. For example, an OEM can develop a high level engine model (Component B in Figure 2-a) for its initial integrated simulation for verification. The OEM can pass this high level model with test cases and requirements/specifications to a Tier-1 supplier. The supplier would refine their models (for example Component D in Figure 2) during their development and virtual integration with the high level engine model (Component B), but supply their refined models back to the OEM to allow the concurrent development and enable more detailed *integrated verification* still early in the life cycle to make other adjustment.

Figure 3 shows the downward (left) leg of an inc-V process instance for vehicle development and elaborates the *architecture & optimization* phase as an example (in the dot-dashed block). The previous (*specification*) phase generates *executable specification* of and *system test suite* for the work product (component or sub-system) as outputs. The *architecture & optimization* phase uses them as inputs and generates refined *architecture design* and *architecture test suite* that are further fed into the subsequent (*optimization*) phase. During this step, the vehicle system (or sub-systems) is virtually integrated and verified against the *system test suite* using simulation. In terms of their refinement levels, the detected defects are returned to their origins, i.e. either the current phase, or the preceding phase(s) (e.g., *specification*) for rework.

III. INC-V PROCESS SIMULATION MODEL

In order to investigate the inc-V process described above, at the initial stage, System Dynamics (SD) is used to model and simulate an inc-V process instance. This section elaborates the construction of the inc-V simulation model.

A. Process Logic

Figure 4 illustrates the generic workflow in one development phase of inc-V process. The '*artifact*' here represents the hardware or software work product delivered through one

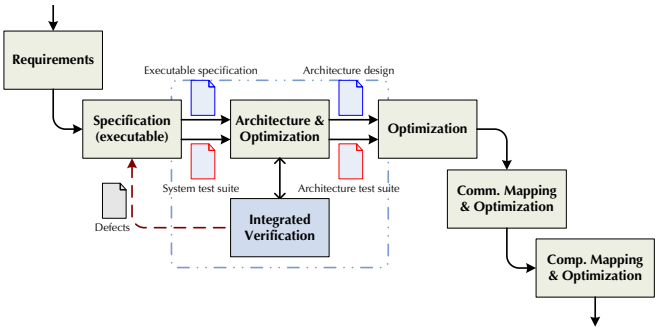


Fig. 3. An inc-V process instance for automotive development

development phase, e.g., specification, architecture design or code, and may be used as input to the consequent phase(s). The upper half of Figure 4 shows a similar workflow to one phase on the left arm of a traditional V model (Figure 1-a). In the beginning the artifact and test cases can be developed in parallel based on the work product and test suite generated by the previous phase(s). Once the development completes, the artifact can be verified, e.g., inspected or analyzed using *analytical verification*, according to the policy of the organization/project. When the number of defects found through verification is beyond a preset *threshold*, the artifact should be sent back for rework. A further verification is optional to ensure most of the faults are fixed in the current phase.

A development phase of a traditional V process normally ends up here once the rework has been done and no more verification is needed. While inc-V process will go one step ahead. As all artifacts delivered through the early phases become testable, the *integrated verification* may follow up. The verified artifact can be further tested using the test cases developed. In an inc-V process, the *integrated verification* (e.g., integration test) can be performed by the developers of the artifact. As a result, most defects captured in the test will be fixed on the spot.

Note that in an inc-V process, *analytical verification* and *integrated verification* are optional activities. For instance, when the test cases are ready, artifact(s) can be directly fed into *integrated verification* for testing after development, skipping *analytical verification*. Whereas, an inc-V process may turn to be a traditional V process when the *integrated verification* is ignored in each phase on the left leg of V.

B. Model Structure and Implementation

The inc-V process simulation model (version 1.0) was developed using Vensim, the most commonly used SD modeling and simulation tool in software process research [10]. Vensim provides a graphic workbench and a number of extra features on the top of SD, e.g., *views* and *subscripts*.

1) Views: Figure 5 shows the basic structure of the inc-V simulation model. The generic development phase of an inc-V process instance is modeled and organized by *views*, a mechanism offered by Vensim that facilitates development and understanding of large scale SD models. It also helps increase module reuse within a complex model. The current model is

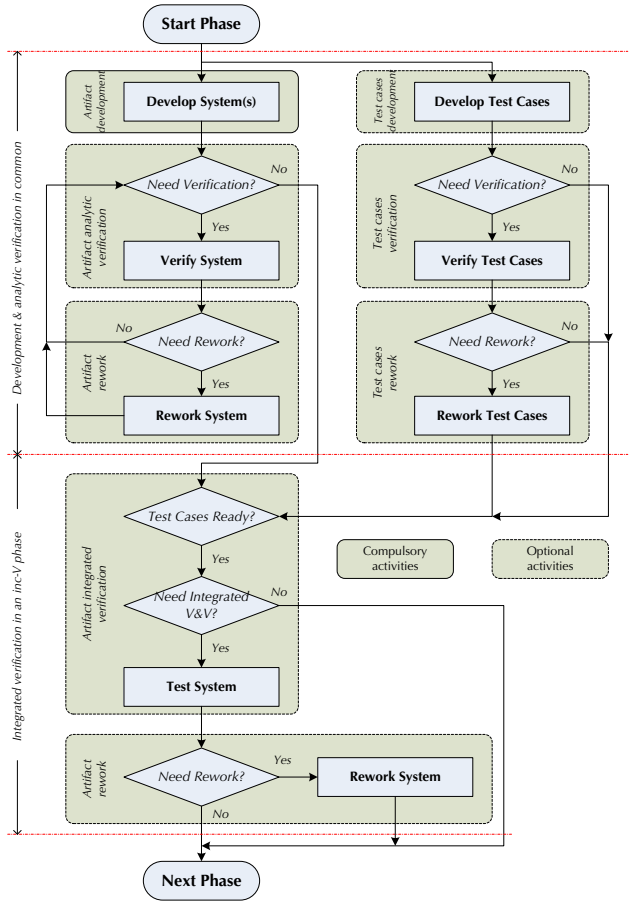


Fig. 4. Workflow in one phase of inc-V process

composed of nine *views*, seven of which correspond to the basic activities of the inc-V process abstracted in Figure 4, i.e. **artifact development** (AD), **test case development** (TD), **analytical verification** (AV), **test case verification** (TV), **artifact rework** (AR), **test case rework** (TR), and **integrated verification** (IV). In addition to the above activity-specific views, the inc-V model also includes another two views: **state control** (SC) and **resource management** (RM), which communicate with each activity during execution. The former oversees the state of process in real-time and controls the transitions between activities (views) and phases. The latter monitors the utility of resources (e.g., workforce) and allocates resources upon requests and availability.

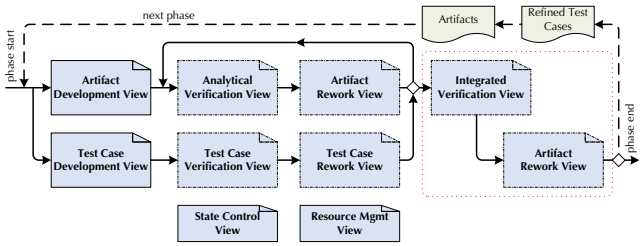


Fig. 5. View (activity) transition in one phase of inc-V process

2) Subscripts: The subscripting mechanism of Vensim enables the variables holding different values for multiple entities simultaneously. Each entity is assigned a distinct identifier called *subscript* (working as index of an array) that can be associated with different variables. The subscript value is used to access the corresponding entity. This feature leverages structure reuse within a simulation model. There are four important *subscripts* used in the current inc-V simulation model: '*Phase*' subscript identifies the development phases and realizes the conditional transitions between the phases during simulation; '*Activity*' subscript separates the values of the variables exist in multiple activities (views); '*Component*' subscript allows to trace individual work products as components (or subsystems) of a system to be delivered through the process; '*Developer*' subscript is used to identify and capture individual workforce (e.g., developer) available for the project.

3) Transitions: Two levels of transition happen during the simulation. At the lower level, when an activity finishes, simulation will trigger another activity based on the state of process and policy constrains. For example, after the *analytical verification* and *artifact rework* finish (cf. Figure 5), the simulation may progress to *integrated verification* if it is required in the project and the test cases are ready, or restart *analytical verification* if too many faults were found in the current round, or move directly to the end of the current phase. The SC view looks into the view's status stored in the *activity* subscript to control the activity transitions. At the higher level, a transition between phases is triggered when all *artifact rework* is done and no more verification is needed (the rightmost point in Figure 5). The simulation implements the phase transitions using the *phase* subscript.

C. Model Parameters

The inc-V process model has a large number of parameters that represent model's inputs and outputs, or are used for model calibration against empirical evidence specific to an organization, project, team, techniques and so on. Table I lists a subset of the model parameters as examples. Note that most of the parameters are subscripted. For example, [*developer's commitment*] may vary by individual, activity (e.g., *development* or *rework*), and phase.

TABLE I
A SUBSET OF INC-V MODEL PARAMETERS

Parameter	Value	Type	View
Need verification	Bool	Input	AV
Need integrated verification	Bool	Input	IV
Defect density threshold	Numeric	Input	AR
Developer's commitment	%	Input	AD
Normalized developer's productivity	Numeric	Calibr.	AD
Normalized test case effectiveness	%	Calibr.	IV
Normalized verification effectiveness	%	Calibr.	AV
Defect propagation multiplier	Numeric	Calibr.	AD
Defects detected by verification	Integer	Output	AV
Defects corrected	Integer	Output	AR
Residual defects	Integer	Output	All
Total effort	Numeric	Output	All

Input parameters specify project specific information, such as estimated product size(s), allocated team size and their

competency levels, as well as project-related policies, e.g., decisions on verifications and associated thresholds. Most of these information should be determined prior to the kickoff of a development project. Calibration parameters represent empirical characteristics of an organization or one type of development project, such as the normalized productivity on development, verification, rework and other activities. Output variables represent the purposes of process simulation, and are calculated during simulation when the input and calibration parameters are specified and the casual relationships are given.

At present as the inc-V model was only partially applied in some industrial trials without systematic data collection, we decided to apply literature-based calibration for generic systems development activities in combination with developers' experience for inc-V specific activities (i.e. early integrated verification) in modeling and calibrating the inc-V process. The literature sought for calibration are related to software engineering (e.g., [11], [12]), automotive engineering (e.g., [1]), and (embedded) systems engineering (e.g., [13], [4]).

IV. Simulation Study

In this section, the inc-V process is investigated using process simulation to answer the exploratory questions like "*Is an inc-V process able to reduce development cost, shorten project cycle and increase product quality?*"

A. Simulating An Example Project

An example system development project that employs the tradition V and inc-V development processes was simulated to investigate the impact of inc-V process on project performance in terms of duration, effort, and quality. The example project is to develop a sub-system of vehicle's E/E systems. In the initial investigation of inc-V process, in order to minimize the possible impact of system complexity, the sub-system developed in the example project is only composed of three components. The '*size*' of the components has been abstracted into a number of arbitrary-sized '*units*' of requirements, since it is a more informative reflection of system development and can be converted for specific size metric, e.g., *functions* of hardware or *lines of code* of software. The size of requirements of the sub-system is 205 units. The maximum team size may be up to 30, but the individual's availability and commitment to this project vary during the process.

The traditional V process of the example project was simulated by disabling the *integrated verification* activities in the phases of the inc-V model, but triggering the integration test at the end of the last phase. In other words, the developed system is verified against the test cases if and only if it is implemented (after the development phases on the left leg of V). Therefore, only *analytical verification* (i.e. review-based verification and model-checking) is allowed to be applied in the early development phases in simulating the V process.

When simulating the inc-V process, the *integrated verification* is enabled in the early development phases. Except this process change, there is no difference of inputs and configurations (i.e. *input* and *calibrated* parameters) between the simulations of V and inc-V processes, which makes a later comparison possible.

On the left leg of V and inc-V, only the first four phases of the inc-V process instance (Figure 3) were simulated because the last two computer-aided phases, i.e. *communication mapping & optimization* and *computation mapping & optimization*, are relatively automated steps, whose impacts on project duration, cost, and quality is little. On the right leg of V and inc-V, we did not simulate the *system test* because we are not yet aware of the inc-V's performance on this final step.

In addition, the *test case verification* and *test case rework* were disabled in both V and inc-V processes due to the assumption of organization's policy.

B. Comparing V and inc-V Processes

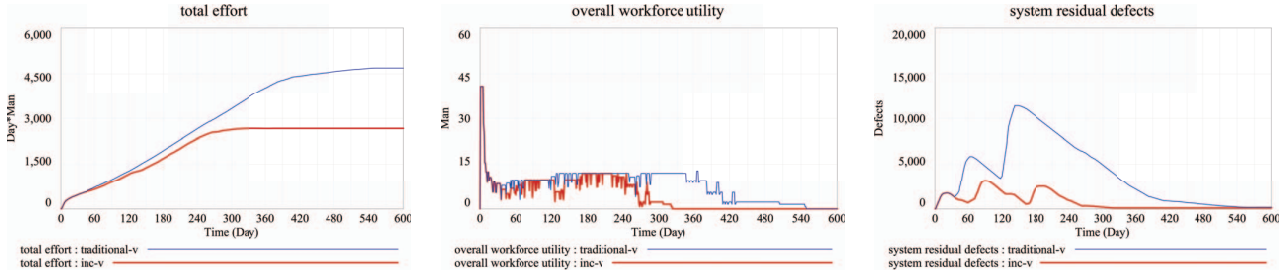
Figure 6 shows part of the simulation results. Note that in the V process simulation, due to the high defect level of the sub-system after the integration test, one more integration test is triggered to further detect and fix the residual defects.

The simulations of V and inc-V processes with the same settings except the activity of *integrated verification* enable the comparison (like in controlled experiments) and further the investigation of the impacts of the inc-V process. According to the simulation, V process and inc-V process of the example project finish in 549 days and 322 days respectively. Although the implementation of the inc-V process takes more time (on the *integrated verifications*) than the traditional V process in the early development phases, the simulation result (Figure 6-a) indicates that the example project may save 41% by applying the inc-V process as an replacement of the traditional V. Similarly, the example project is simulated to consume 2762 man-days for the inc-V process, while 4687 man-days are spent for the traditional V process.

Figure 6-b depicts the workforce distribution during the project life-cycle. The workforce level of the V process maintains a high profile until the last quarter of the project but followed with a long tail. The peak workforce utility of the inc-V process in contrast concentrates on the front end, and never goes beyond the workforce level of the V process.

We investigate the impact of the inc-V process on product quality in terms of the defect level, i.e. the number of defects residual in the released system (after integration tests in the simulation example). Figure 6-c shows the impact is significant. Compared to the V process, the defect level of the inc-V process stays at a lower level (never goes beyond 2500) and ends at 92 defects in the final sub-system. Whereas, the defect level of the V process may peak at 11300 and finally drops down to 172. The results indicate 46.5% improvement on product quality when adopting the inc-V process.

Table II shows some more performance indicators by phase for detailed comparison. As the *optimization* is not a typical phase in the traditional V-model process, the table does not compare the two processes for this phase except *entry time*. The '*entry time*' records the earliest date when the development moves into a new phase. Note that the components of the sub-system might be developed at different speed due



(a) Total effort of the V & inc-V processes (b) Workforce level during the V & inc-V processes (c) Defect level of the V & inc-V processes

Fig. 6. Simulation of the V and inc-V process examples

to a number of factors, e.g., component size, complexity, and resource availability. We only show the time when any component first kicks off a new phase. The results show that the inc-V process significantly lags behind the V in the first two phases. Although the V process starts *architecture* phase first, it ends later than the inc-V. The similar results are also observed on the phase effort. The inc-V process consumes more effort in the first two phases than the V, but saves a lot in the consequent *architecture* phase. The number of the corrected defects by phase may explain the above phenomenon. The inc-V process spends much more effort correcting the defects detected by *integrated verification* in the very early development phases (i.e. *requirements* and *specification*), which results in the temporary delay and higher effort. Compared to the V process, the extra effort on this activity in the inc-V looks a worthy investment.

TABLE II
COMPARISON BETWEEN V AND INC-V IN THE EARLY PHASES

	Requirs.	Specif.	Archit.	Optim.
V entry time	Day 1	Day 36	Day 117	Day 350
inc-V entry time	**Day 1**	**Day 60**	**Day 162**	**Day 248**
V effort (man-day)	391	781	3104	N/A
inc-V effort (man-day)	**538**	**891**	**1044**	N/A
V corrected defects	803	2602	10442	N/A
inc-V corrected defects	**1448**	**2865**	**2198**	N/A

The above comparison between the two processes shows the positive impacts of the inc-V process on project duration, effort, and product quality. The simulation results and findings to a large extent conform to the experience from a telecommunication system development project that employed the VaST Quantitative System Engineering Process [14], a precursor of the inc-V process.

Note that the current simulation does not include *system test* phase due to the lack of evidence. However, as the inc-V's defect level after *integration test* is much lower than the V, we believe that the final defect level of the inc-V after *system test* would not be higher than the traditional V.

There exist many researches (empirical and simulation studies) on the impacts of the *analytical verification*, especially *inspection* (cf. Section V). Although more *analytical verification* can be performed in the early phases, it is not able to replace the *integrated verification* at system level. For example, some faults on interoperability can only be detected when the system is being integrated (virtually or physically).

V. RELATED WORK

Process simulation has become a powerful tool in support of software and systems development process at various levels, including cognitive process understanding, operational process management, and strategic decision making [15]. Over two hundred process simulation models were built and reported [10], a few of which have interests in the V-model process and verification and validation (V&V) activities.

The V-model process and V&V activities have been investigated by process simulation with varying settings. Raffo et al. [16] modeled V&V as a portion of traditional V-model style development process (i.e. ISO 12207) adopted on NASA's software development projects, and created a discrete-event simulator that NASA can use to quantitatively assess the economic benefits of performing V&V activities on their development projects and to optimize that benefit across alternative V&V integration strategies for decision making. This work enabled NASA to effectively allocate scarce resources for V&V activities and provided NASA increased visibility into their alternative and optional V&V activities.

As another important example, GENSIM 2.0 [17] is a system dynamics based process simulator that models and simulates a generic V-model development process. It is constructed with three levels of refinement, i.e. requirements, design, code and their validation counterparts. GENSIM 2.0 is a pure continuous simulation model like the inc-V simulation model in this paper. Nevertheless, the differences between them are significant: *1*) GENSIM 2.0 simulates a generic V-model process, it neither represents real-world of V-model variants (e.g., W-model), nor an inc-V process described in this paper; *2*) GENSIM 2.0 only models the *review-based* verification in V-model process rather consider other *analytical* and *integrated verification*; *3*) though the both developed with Vensim, GENSIM 2.0 is composed of four *views* on each refinement phase, while the inc-V model is structured with seven more detailed *views* of activities within a phase; *4*) GENSIM 2.0 was calibrated with literature related to software engineering, while the current inc-V model was calibrated with literature in three domains, i.e. software engineering, (embedded) systems engineering, and automotive engineering.

Although several simulation studies investigated V-model process and V&V activities, none of them had been built with the realistic needs of application to embedded systems or automotive industry. However, these studies provide evidence

for the feasibility and effectiveness of process simulation on investigating and optimizing V-model development processes.

VI. Discussion and Conclusion

Large scale systems development is often a complex undertaking and fundamentally different from that of pure software or hardware systems. Complexity arises from the need to co-design and create software at low-level of abstraction that also interacts closely with hardware with strong emphasis on dependability and mission-critical constraints. Compared to the conventional software development, complexity also increases in the development processes of these systems.

The vehicle development process has changed dramatically in the past century [1]. The increasing use of E/E systems, particularly software embedded in these systems, has brought new challenges to automotive development process over the recent decade. A number of new technologies have been invented to tackle these challenges, which will eventually change the development process again.

Simulation-based development is an advanced technology that enables virtual integration of vehicle (sub-)systems across barriers (e.g., platforms) and simulation of the systems on varying refinement levels in the early development phases. This technical advance move the late integration test early, from the right leg to the left leg of the V-model. Correspondingly, an inc-V process is devised as a promising replacement of the traditional V-model process for vehicle development.

This paper introduces the conceptual inc-V process model and describes an inc-V process instance that is under industry trials. To investigate the impacts of the inc-V process on (automotive) systems development, an SD-based process simulation model is created based on our collaboration with the technology inventor and automotive industry.

Empirical research (e.g., controlled experiments) may provide premium quality evidence to evaluate an emerging technology. In evaluating a new system development process (inc-V model in this paper), however, such rigorous empirical evaluation is not only extremely expensive, but somehow impossible in the exploratory stage. Although the simulation-based development and the inc-V process discussed in this paper have been applied in some trials in a leading automotive OEM's settings, it is difficult to be adopted for the entire development life cycle of a complex system of systems like modern vehicle without any laboratorical evidence.

The simulations of the V and inc-V processes enable the comparison between them. Although the inc-V process consumes more resources in the early development phases, the results show the dominant benefits on *duration*, *cost* and *quality* on the overall project when applying the inc-V process in the same case. The positive impacts of inc-V from this research confirm the encouraging potential of this new process.

Some limitations exist in the research at the current stage. For example, the simulated system is relatively simple (with three components); the simulation model is built on System Dynamics which is not able to support the modeling of some complex features of simulation-based development (e.g., constant verification); a few model parameters are not calibrated against the empirical data as we have not found much empirical research on the new techniques and processes related to inc-V. We will continue to work on these limitations.

Our future research can be twofold: to refine the inc-V process model based on the feedbacks from industrial trials so that it can be successfully adopted and work well with organization's legacy processes; and to improve the inc-V simulation model with the support of more empirical evidence and more accurate modeling for in-depth impact analysis.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant No.61572251). We would also like to record our appreciation to the other contributors to the initial idea of the incremental V-model process.

References

[1] J. Weber, *Automotive Development Processes: Processes for Successful Customer Oriented Vehicle Development*. Springer, 2009.

[2] E. Hanawalt and W. Rouse, "Car wars: Factors underlying the success or failure of new car programs," *Systems Engineering*, vol. 13, no. 4, 2010.

[3] C. Haskins, *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities (ver. 3)*. International Council on Systems Engineering (INCOSE), 2006.

[4] B. Broekman and E. Notenboom, *Testing Embedded Software*. Addison-Wesley, 2003.

[5] A. Sangiovanni-Vincentelli and M. Di Natale, "Embedded system design for automotive applications," *Computer*, vol. 40, no. 10, pp. 42–51, Oct. 2007. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4343688

[6] G. Nicolescu and P. J. Mosterman, *Model-Based Design for Embedded Systems*. Boca Raton, F.L.: CRC Press, 2010.

[7] A. Abdallah, E. M. Feron, G. Hellestrand, and M. Wolf, "Hardware/software codesign of aerospace and automotive systems," *Proceedings of the IEEE*, vol. 98, no. 4, pp. 584–602, May 2010.

[8] P. H. Feiler, J. Hansson, D. de Niz, and L. Wrage, "System architecture virtual integration: An industrial case study," Software Engineering Institute (SEI), Carnegie Mellon University, Tech. Rep., 2009.

[9] *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. http://www.computer.org/portal/web/swebok/: IEEE Computer Society, 2010.

[10] H. Zhang, B. Kitchenham, and D. Pfahl, "Software process simulation modeling: An extended systematic review," in *International Conference on Software Process (ICSP'10)*, vol. LNCS 6195. Paderborn, Germany: Springer, July 2010, pp. 309–320.

[11] M. S. Fisher, *Software verification & validation: An engineering & scientific approach*. Springer, 2007.

[12] S. Wagner, "A literature survey of the quality economics of defect-detection techniques," in *International Symposium on Empirical Software Engineering (ISESE'06)*. Rio de Janeiro, Brazil: ACM, Sept. 2006, pp. 194–203.

[13] A. Engel, *Verification, Validation, and Testing of Engineered Systems*. Wiley, 2010.

[14] G. Hellestrand, "The quantitative design process in embedded system and strategic engineering," in *Embedded World Exhibition & Conference*, Germany, 2012.

[15] H. Zhang, B. Kitchenham, and D. Pfahl, "Reflections on 10 years of software process simulation modelling: A systematic review," in *International Conference on Software Process (ICSP'08)*, vol. LNCS 5007. Leipzig, Germany: Springer, May 2008, pp. 345–365.

[16] D. M. Raffo and W. Wakeland, "Assessing iv&v benefits using simulation," in *28th Annual NASA Goddard Software Engineering Workshop (SEW'03)*. Greenbelt, MD: IEEE, Dec. 2003, pp. 97–101.

[17] V. Garousi, K. Khosrovian, and D. Pfahl, "A customizable pattern-based software process simulation model: Design, calibration and application," *Software Process: Improvement and Practice*, vol. 14, no. 3, pp. 165–180, 2009.