

Developing Smart Toys—From Idea to Product

Scott Boss, Smart Toy Lab (Connected Products Division), Intel Corp.
Henry Bruce, Smart Toy Lab (Connected Products Division), Intel Corp.
Charlie Case, Smart Toy Lab (Connected Products Division), Intel Corp.
Kendal Miller, Smart Toy Lab (Connected Products Division), Intel Corp.

Index words: Consumer Products, Extended PC, Smart Toys, Vendor Management, Vendor Qualification, Vendor Selection, Development Process, Product Life Cycle, Outsourcing, Product Integration

ABSTRACT

Creating an Intel® Play™ toy involves the collaboration of several different parties with a wide range of abilities. From toy designers, to engineers, to marketing operations and sales, the Intel Play team combines several disciplines to develop a creative play experience centered on the PC. This paper presents the life cycle of Intel Play toys by outlining and detailing the process used to create an end-user product from a budding idea. Starting with focus testing and concept approval, through product requirements and vendor selection, and finishing with architecture definition and product implementation, a lot of effort is expended before an Intel Play toy hits the shelf.

INTRODUCTION

Intel® Play™ toys are a combination of hardware and software that together form a unique play experience for children. The development of a Smart Toy involves parallel software and hardware efforts that must be continuously integrated to show progress and to validate architecture decisions. Software and hardware development at the Smart Toy Lab is a combination of product requirements, architecture definition, vendor selection, component development, and vendor management. Third-party vendors are engaged heavily for toy development, and this paper discusses the motivation, reasoning, and methods behind this approach. Integration brings the two worlds of hardware and software together

through validation of the complete product in order to guarantee a high-quality, end-user experience.

On the software side, third-party vendors and software packages are evaluated in order to provide solutions for software features. Several factors are weighed when choosing software vendors including technical background, user-interface design, cost, quality, and integration logistics. The final software solution combines external and internal resources to complete the software picture.

On the hardware side, a similar evaluation process takes place with a heavy emphasis on component cost due to the low-price expectations of the toy industry. A wide range of skills is required from the vendor including the ability to provide cost-effective components, the ability to design a proper enclosure for the toy, the ability to develop firmware and device drivers, and the ability to provide a quality manufacturing process. A successful hardware design is the correct balance between hardware architecture and vendor selection.

The integration effort is a daunting task: software and hardware must be seamlessly integrated into a complete product. All functions of the hardware and software must be validated in all possible usage scenarios. The integration effort must simulate and validate consumer usage in the virtually limitless space of available consumer platforms and configurations (many hardware combinations, many operating systems, and even many languages if the product is to be sold in the international market). To increase the integration burden even further, the toy industry revolves around the Christmas season, which imposes a hard stop deadline by which the product must be finalized. The final quality of Intel Play products is determined by the integration effort, which ultimately sets the bar for the PC play experience.

Intel Play is a registered trademark or trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Developing toys for the consumer marketplace presented Intel with a set of unique design, development, and integration challenges. These challenges are listed and described as well as the methods and approaches developed at the Smart Toy Lab for meeting these challenges.

The next time you walk into a toy store and see an Intel Play product on the shelf, think about the transformation that took place from a twinkle in a toy designer’s eye to the shrink-wrapped package in front of your eyes.

TOY DEVELOPMENT OVERVIEW

Toy development at the Smart Toy Lab (STL) has evolved and has been refined quite a bit in the few short years that the toy lab has been around. Toy components, hardware vendors, and software vendors vary from product to product, but the overall method and approach to building a smart toy remains constant. Each toy can be broken down or dissected into high-level areas and components for both the hardware and software portions of the toy. The details of a typical toy are outlined in the Toy Anatomy section. The quality of the toy and the ease of development are tremendously impacted by the decisions made during the product definition and the product architecture phases. Even the best development efforts cannot completely overcome or cover up poor definition or architecture decisions.

The development stage of a toy must be rigorous and well-managed since there are several parties involved both inside and outside of Intel. The toy continually takes different shapes and moves in new directions throughout the development timeline. In many cases, several vendors make changes to hardware and software components in parallel. These changes must be tracked and validated with pre-defined project milestones to prevent unmanageable defects and to guarantee on-time delivery of the product.

CHALLENGES

Intel® Play™ toys present a set of challenges that are typically not found in one area within Intel. These challenges span several areas ranging from design (brainstorming, concept selection, user interface definition), through engineering (architecture, vendor selection and management), to marketing (young consumer audience, toy-selling season). Each area can dramatically influence the toy feature set so it is important to keep requirements from each of the areas in mind when defining the toy.

Intel Play is a registered trademark or trademark of Intel Corporation or its subsidiaries in the United States and other countries.

With such a wide range of requirements, successful product implementation demands that all areas are well-represented and accounted for during product definition and development.

A challenge unique to toy design comes from the consumer expectation that toys should not cost a lot of money. Offering a low-cost toy is in itself an achievable goal; however, low cost quickly gets caught in a tug of war between high product quality and a rich product feature set. The Intel® brand name demands a high-quality product with a strong feature offering. While many toy manufacturers are able to compromise quality to save costs, Intel’s premium brand name brings with it the highest expectations of quality. Toy designers and engineers must constantly evaluate the tradeoffs between cost, quality, and feature set when defining a smart toy. Evaluation of these tradeoffs must be thorough as it ultimately determines the product’s success in the marketplace

TOY ANATOMY

Each Intel® Play™ toy can be summarized as a combination of physical hardware accompanied by a software suite (Figure 1). However, each toy varies tremendously at the lowest implementation level depending on its form and function. Basic design structures and typical components for hardware and software are outlined and described in this section.

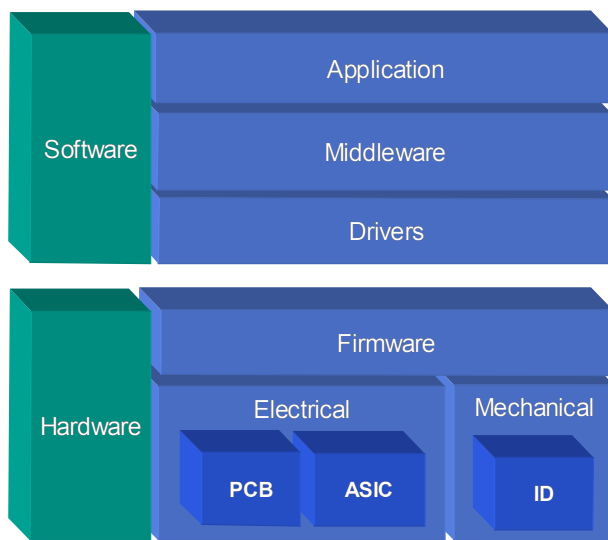


Figure 1: Toy anatomy

Intel and Intel Play are registered trademarks or trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Hardware

The hardware of a smart toy, by its very nature, typically has an embedded microprocessor, sensors of one type or another, a mechanical system to comprehend moving parts, and some firmware to control and tie the pieces together. The hardware elements listed above can be categorized into three main components: firmware, electrical, and mechanical. The electrical and firmware engineers define the hardware architecture. The electrical engineer is responsible for the Printed Circuit Board (PCB) that forms the “heart” of the toy. The mechanical engineer works closely with the toy designer on the toy’s enclosure and ultimately determines how the PCB fits within the toy.

Firmware

Firmware is the software that runs on the micro-controller or Application Specific Integrated Circuit (ASIC), and it is more complex if the toy supports an untethered mode of operation. As an example, customized firmware was required for both the Computer Sound Morpher (CSM) and the Digital Movie Creator (DMC) since they both support capture of media away from the PC. The user interface portion of the firmware (i.e., “look and feel”) is specified by the toy designer, and implemented by the firmware engineer.

Firmware programming techniques and complexity can vary greatly between products. The CSM handled button presses and the related capture, playback, and deletion of audio clips. Due to the simple nature of the microcontroller architecture and its task, the firmware was coded as assembly language. The DMC firmware was more complex than the CSM firmware. For a start, there were two microcontrollers: one handled the user interface, and the other sequenced the camera’s audio and video processing. The user interface firmware was of equivalent complexity to the CSM firmware and was implemented in a similar nature. The audio and video processing firmware were more complex; they were based on the Intel® 8051 processor. The firmware was coded in C and cross-compiled to the target. In this case, the microcontroller’s performance was pushed to the limit to enable recording of both video and audio. The video was recorded at a frame rate of 10 frames per second with a frame size of 320x240 pixels. The audio was recorded at a rate of 12kHz (samples per second).

Electrical

The electrical components form the core of the toy’s operation. Usually, an Application Specific Integrated Circuit (ASIC) and/or microcontroller do the bulk of the work as this enables low hardware cost and a small form factor. Selection of these two components is a key stage of the toy’s development. If the toy can be used when not

connected to the PC (often termed untethered operation), some form of status display (usually a customized liquid crystal display, or LCD) and memory are usually required. The electronics components are mounted on one or more PCBs to fit into the industrial design. Ideally just one PCB is used because inter-board connectors are expensive, and simple wiring can be unreliable.

Mechanical

Mechanical engineering (often termed M/E) is the “glue” that holds all the hardware together. Any moving parts required by the toy fall into the domain of the mechanical engineer and require thorough design up front. The mechanical engineer must also ensure that the toy is compliant with all toy safety regulations. This is a demanding task: tests must be conducted to ensure the product functions, with no sharp edges exposed, after being dropped and that there is no battery discharge if a small piece of wire is dropped into the toy’s enclosure.

A good mechanical design enables the industrial design to have the right external “look and feel,” the PCB to be neatly mounted inside the enclosure, and for the whole unit to be easily assembled.

Software

The software comprises three areas: drivers, middleware, and the application. Each area has its own purpose in the larger software whole and consists of a distinct functional set. The categorization of software into these areas provides for the following advantages:

- A large software project is broken into manageable pieces.
- There is a division of labor between in-house and out-of-house resources.
- A series of checks and balances is in place during the development cycle.

The details of the drivers, the middleware, and the application as well as the typical approach to their development are discussed next.

Divide and Conquer

Development of the software encompasses a wide range of activities, ranging from very low-level hardware communication programming to very high-level user-interface programming. Finding the necessary skill sets for the entire software stack in one place is often impossible. Therefore, we wrote the middleware in-house at the Smart Toy Lab and hired third parties for driver and application development. This approach is by no means the only solution, but it is now well-known at the Toy Lab and has proven to be successful for three separate products.

Drivers

Direct communication with the toy hardware is always a must for the software. Custom drivers are often necessary in order to implement toy-specific behavior. This driver work is always low level and is typically paired with the hardware provider. Interaction with the toy hardware ranges from simple input and output commands to full data streaming, depending on the toy. The toy feature set and toy-to-PC connections dictate heavily the amount and type of driver work that is needed. For example, the CSM required no driver work as it simply plugged into the already existing PC sound card and its driver. However, the QX3™ Computer Microscope and the DMC both required full streaming USB video driver solutions.

Middleware

Middleware serves several purposes including hardware abstraction, grouping of software features, and reduction of complexity. The main purpose of the middleware component is to abstract the hardware from the application by removing any direct communication with the driver from the application. The application uses the middleware components in order to interface with the hardware and to implement features beyond the scope of the application developer. Since the middleware is developed and validated at the Smart Toy Lab, it also serves the purpose of providing a vital porthole into both the application and driver development tasks. The communication between the application and driver can be monitored via the middleware and often application or driver defects can be short circuited and addressed in the middleware.

Application

The application provides all needed user-interface components as well as the majority of the functionality behind the user interface. It accesses the middleware components as necessary to implement complex features and to interface with the hardware. The entire multimedia application development process is encapsulated in this piece including media asset production and integration (art, sound, animations), component implementation and integration, and final application assembly. The look and feel of the software is carefully crafted here by experienced artists at a third-party development house with oversight and deep involvement by the toy designers at the Smart Toy Lab.

QX3 is a trademark of Intel Corporation or its subsidiaries in the United States and other countries.

TOY DEVELOPMENT

An Intel® Play™ toy starts as a new concept, moves through several approval stages during definition, matures during product development, and completes as a packaged end-user product ready for the store shelf.

Concepts and Prototypes

All toys start as a bright idea and then continue on through a myriad of refinements and approvals before there is any investment made to build a product. There are generally three types of prototypes used during the smart toy development life cycle.

During the early concept stage, product designers may make a variety of hard-foam or plastic models of potential products to help convey ideas, explore directions, and in some cases, test the concept with focus groups (Figure 2). While designers do some rapid foam prototyping in house, most of the models are made at outside shops by professional model makers, who employ Computer Numerically Controlled (CNC) machines and airbrush painting techniques to make photo-real, non-functional mockups.



Figure 2: Computer Sound Morpher plastic model

On the engineering side, often a concept breadboard is used to validate the fundamental technical risk in the project. For example, in the case of the Intel Play Computer Microscope toy, a prototype was built using an off-the-shelf optical microscope and a digital camera joined together (Figure 3). This combination was used to explore and specify magnifications, light levels, and project feasibility. The goal of the concept breadboard is to quickly understand the fundamental properties of the

Intel Play is a registered trademark or trademark of Intel Corporation or its subsidiaries in the United States and other countries.

toy, without necessarily worrying about size or cost at this stage.



Figure 3: QX3 microscope concept breadboard

The next stage of prototyping in the product life cycle is usually a form, fit, and function breadboard of the toy. This will generally consist of a Printed Circuit Board (PCB), machined or “rapid prototype” plastic parts, and software/firmware with very limited functionality (Figure 4). The purpose of this model is to validate the overall size of the product and provide an electronic breadboard for firmware development. This prototype presents a major challenge for the toy designer and hardware engineer, as they have to agree on an industrial design that satisfies both styling requirements and space envelope constraints. Often times, this prototype is also used at key marketing events to demonstrate the product.



Figure 4: QX3 final prototype

In addition to the several flavors of hardware prototypes, there are also software prototypes that are developed. Software prototypes are often simply push-button Windows* applications that demonstrate the ability to either implement a software feature from scratch or demonstrate the difficulty level of integrating a third-party software package. These kinds of prototypes are very important tools when determining the software feature list. A software prototype helps to crystallize the vision of the toy designer in some concrete form and helps the software engineer gauge the complexity of a software feature. By no means does the software prototype completely define a feature. It is simply a measuring stick that can be used as a piece of information when defining the software feature list, architecture, and user interface. It also serves as a resource load estimate tool when generating the project schedule.

Architecture Definition

In conjunction with the refinement and clarification of the toy concept, comes the definition and details on how to transform the concept into a product. Both hardware and software architectures are defined in parallel as it becomes clear exactly what features are required for the product. These two architectures are outlined in product architecture specifications, which contain component

*Other brands and names may be claimed as the property of others.

diagrams and sufficient implementation details on building the toy.

The software architecture must list implementation solutions for all required features for the software play pattern. The overall software architecture is fully described, and key technologies and potential vendor solutions are listed for each feature. This information becomes vital when constructing overall product schedules and cost estimates. It also serves to define functional development areas so that resources can be assigned accordingly. At this stage, the software architect must closely manage the delicate tradeoff between feature implementation and the software feature set. Decisions made during the architecture definition phase can have great impact during the product development phase and should not be taken lightly. Features must be analyzed and a clear development path must be identified for each feature before it can be included in the architecture.

The hardware architecture lists the major components to be used in the design; it tells how they must operate and how the toy interfaces with the user. The architecture also defines how the toy implementation is partitioned into an Application Specific Integrated Circuit (ASIC) and firmware and device driver components. The toy feature set typically demands leading-edge technology for implementation, so the architecture is developed in conjunction with the selection of a vendor that can deliver a leading-edge hardware solution.

In parallel with the hardware architecture, a mechanical design specification is drafted. The design specification covers all the dimensions of the product, which include how it fits together and how it looks.

Vendor Selection

External vendors play an important role in the development of Intel Play toys for both hardware and software development efforts. External vendors provide consulting expertise, development resources, and physical components for Intel Play toys. Vendor selection is an important step in the toy development process since the dependency on third-party vendors is a strong one.

Selecting a Software Vendor

The typical toy software is a full multimedia children's application with audio, animations, screen transitions, and lots of artwork. The workload for constructing the application can be divided into two areas, the user-interface development and the technology development that provides the functionality behind the user interface. Independent software vendors can provide solid solutions for toy software in both these areas. Identifying and selecting the best-qualified software vendor is a challenge in the toy development process. A discovery process is

used to search for potential candidates. A small set of vendors is chosen based on their reputation within the industry and their relevant product experience. The chosen vendors are subsequently evaluated for their technical background and abilities, tools and process, cost, quality assurance, track record, and relevant software experience. A written questionnaire is used to gauge these criteria. Depending on how much the vendor will be involved with the product, a more rigorous evaluation is sometimes required. For example, application vendors are involved from the ground up: they contribute to the software framework, art assets, user interface, and the play pattern. As a result, the questionnaire for application vendors is intensive, and they are interviewed several times before a decision is made. However, individual software components are usually much less complex and often include existing product components that can be used "as-is" or "off-the-shelf." The evaluation of these vendors is, therefore, less rigorous.

Selecting a Hardware Vendor

Hardware vendor selection is a complex task, as there are many components to the hardware as presented in the following list.

- industrial design
- acoustic and/or optical design
- mechanical engineering
- electronic engineering
- firmware development
- electronic module build and test
- toy assembly and test
- packaging

One or more hardware vendors may need to be engaged based on the complexity of the toy. Ideally, a single vendor is selected to develop and manufacture all hardware components (electrical, mechanical, and firmware). Selection of a single hardware vendor is generally not possible because of the complexity of a technology toy. The hardware vendor selection process is broken into several phases. Initially, the focus is on finding a vendor that can provide the ASIC or heart of the hardware. Here, chip-set vendors that meet the requirements of the architecture definition are evaluated. If the selected chip-set vendor does not also provide firmware development services, then an additional vendor may be needed to fill this hole. If the firmware coding is very simple, the work is sometimes done in-house, but this doesn't usually happen.

The next step is to find a vendor for the development and manufacture of the electronic module. This vendor is often described as the manufacturing vendor. Development of the electronic module requires the design and layout of the PCB+. The electronic module then needs to be fitted into the toy enclosure. This is a specialist task requiring mechanical engineering design skills. Typically, the manufacturing vendor can both design and provide the molded plastic parts that form the “skin” of the toy. Finally, the toy needs to be placed in a package for the toy shelf. Intel has demanding standards in this area that not all vendors can meet, so packaging is mostly handled in house.

Selection of the manufacturing vendors is made in close consultation with the Intel operations team. The operations team must evaluate the vendor for stability and capacity capabilities. As there are sometimes partnerships between technology and production vendors, the latter may influence selection of the former.

Vendor Management

Simply selecting and hiring a third party to build components does not automatically produce a finished product. Engaging third parties to build components requires close monitoring and tight integration points for success.

The approach taken in vendor management must be flexible, depending on the core competencies of the vendor and the complexity of the project. The overall goal is always to deliver a quality product on time, on schedule, and at the right cost.

Several key tools are used to facilitate the development process. The first such tool is the project “Map Day,” which occurs immediately after vendor selection. This is a mandatory on-site meeting wherein all stakeholders in the product development work out a full development schedule by negotiating key dependencies and integration points. The output of this meeting is the master project schedule that goes from project start to first customer shipment. It is some time after this period that the resulting integration timeline and its milestones (with relevant acceptance criteria) are baked into the vendor contracts.

With a schedule in place, the next step is to execute the agreed-upon plan. This is done via weekly team meetings between hardware and software vendors. These meetings are used to go over key issues and track progress. All parties will have team leads in their respective areas of expertise that will negotiate their way through development. Generally, these team leads serve as the sole contact points within their organizations for information exchange.

As Intel is the integration point for the hardware and software components, progress is also effectively tracked by validating interim deliveries of these components against the product specification and the Map Day schedule. Map Day is a meeting/process for planning new projects that require the integration of plans of multiple players or groups. As described below, integration points are designed to regularly validate the various components as they are integrated through the duration of the project.

Integration

Product integration attempts to provide the highest quality product for the customer, with minimal impact on the schedule, through constant evaluation of product health at defined checkpoints throughout the development cycle.

The Integration Engineer (IE) is the member of the core development team who is responsible for taking ownership of this particular challenge. Throughout the project, the IE remains focused on the final integrated product and its end user, while driving a variety of activities (called the integration process) that work to exhaustively validate the project schedule, product design, and product implementation.

Milestone Definition

If this integration effort is to be successful, it is critical to define a clear, rigorous, and methodical process. The starting point is a map of key milestones that mark the end of various phases: design and definition, pre-alpha, alpha, beta, release candidate, and finally a release to manufacturing. Using past products as a guideline, the durations between these milestones are scrutinized to ensure there is an appropriate amount of time to implement, test, and debug the features. Each milestone in this integration timeline contains exit criteria, each defined for that particular phase as an indicator for the status of the product. The IE will evaluate these criteria, and a resulting approval translates to a green light for various teams or external vendors to begin the next phase of the timeline. For those external parties that provide software or hardware deliverables, it also signals a contractual payment. However, if the product fails to meet the necessary criteria, those failures are evaluated and the team will do whatever is necessary (e.g., feature removal, additional engineering resources, etc.) to mitigate the risk to the next milestone and the overall project.

During the first phase, the IE is primarily architecting the appropriate validation. The engineer begins with an overall plan that describes the scope, the details of the timeline and milestone criteria, defect management, resources and tools, the various types of testing, a final approval checklist, and the methods and strategy required to successfully execute such an effort. Also during this

planning phase, the IE contributes to the product definition effort. The engineer will provide input to the feasibility of the feature set and, eventually, the lower level details of functionality and requirements of an Intel Play product. This is where the IE has the opportunity to set expectations of product quality and also improve upon new products by rolling in support issues or customer feedback from existing products.

As the product definition and product architecture specifications become more concrete, a feature matrix is defined for the hardware and software components of the product. This matrix articulates what gets implemented and when this occurs. Such a matrix provides two major benefits to the project: clear expectations for each milestone, and a method to align the implementation of the various features and functions for each layer of the product. The latter will be constantly evaluated as the project progresses. Such complex products require this kind of regular validation; otherwise, the project is prone to miss critical milestones, as a result of delays stemming from misunderstood dependencies.

Test Methodology and Tools

The actual testing efforts are broken into basic acceptance, functional, interoperability, compatibility, field, and localization acceptance testing. The IE has to carefully analyze the schedule, project requirements, product specifications, and resources to properly coordinate and manage all of these pieces into an efficient and effective validation effort that sets the bar high enough to protect the good name of Intel.

The general approach remains a black box (or end user) testing. Each type of testing varies greatly in scope and in the resources required. The basic acceptance testing serves to provide quick feedback on stability and core functionality, with a general target of execution time running four to six hours or less. Functional acceptance testing encompasses the big portion of the overall test plan, as it validates all features of the product, both in terms of correct implementation and in the broader sense of product performance, stability, and behavior in stress/boundary conditions. Interoperability testing focuses on how well the product behaves with similar hardware or software products, or how well it works with certain operating system features. Compatibility testing attempts to validate the product on a sampling of computer platforms that are indicative of the kinds of computers being used by the target customers. Given the inherently complex nature of this task, the Intel Play team utilizes a selection process and as much market data as possible to design a matrix of computer systems that will expose as many potential issues with those customer platforms as possible, while remaining cost-effective in the number of systems and test hours required. Field-

testing is a limited effort at acquiring early feedback from consumers with our product in their own homes or workplaces, using their own personal computers. This results in preliminary “real-world” data on potential usability and compatibility issues.

Certain tools are employed to track, utilize, or report all of these tests and their results. For the various test scripts, a proprietary test management system is used to create and maintain suites that contain the necessary procedures and their relevant scenarios. This is a database that contains a user interface that supports the authoring process, the assignment of procedures to a tester, and the generation of test-result reports. A third-party defect tracking system is used as a bug database (often one for software and one for hardware) for all internal and external team submissions and reports.

Not addressed in this document is component-level testing. Various software or firmware modules are always part of the overall integration effort, but the specific testing required for those pieces comes from a separate validation effort. The integration team coordinates with all of these teams, but the integration process always remains focused on the final product and its features from an end-user perspective.

Localization

Another factor to consider for an Intel Play toy is its readiness for other countries outside the United States. Intel Play currently delivers the English product version prior to any international versions, so the English product team does its due diligence to minimize the effort required on subsequent localization efforts. Therefore, some level of localization testing will occur on the English version. The goal of this effort is to ensure that the English version can run without any issues on international operating systems, the design is such that as few changes as possible will be required during the localization efforts, and that the localization kit included with the final gold release can be used by the localization team to easily modify assets and re-build the product for another language.

Product Implementation

As the product moves through the alpha and beta stages and becomes more stable and feature complete, the IE continues to increase validation efforts per the plan of record. The engineer works closely with other teams to communicate the results of testing and analysis against product requirements. In addition to overseeing a multitude of testing activities, the IE manages regular bug scrubs, drives ownership and prioritization of known issues, aids in the debugging effort through detailed characterization, and regularly reports product health to the various teams through indicators such as bug count, bug trends, and feature matrix status reports.

At the end of this integration process, the product enters the final two-week phase where it is evaluated against the golden checklist. Various procedures and checks are executed, and upon reaching the final seven days, all aspects of the release candidate are completely frozen. During the final seven days, the various tests continue until all validation procedures are verified to pass, known issues are resolved, all milestones are completed, and the product meets project requirements. The product is then declared golden for release to manufacturing.

Overall, the integration effort is a considerable task that, if orchestrated properly, will successfully serve to protect the brand name, maximize a positive customer experience, and minimize project risk.

SUMMARY

With a combination of hardware and software development and the integration savvy to merge the two, an Intel® Play™ toy evolves from a rough concept into a finished product. With considerable effort from all involved, the hardware transforms from breadboards and foam models to final circuit boards and highly polished plastics. The software follows a similar transformation from an artless engineering prototype to a full-blown graphical multimedia application. In the end, the final product on the shelf represents the full design, implementation, and integration effort of several vendors, all coordinated out of the Smart Toy Lab at Intel.

ACKNOWLEDGMENTS

Bob J. Hicke was the primary author of the original framework for the integration and vendor selection process described in this document. His vision has been expanded to create a strong foundation for the successful development of many Intel® Play™ toys.

AUTHORS' BIOGRAPHIES

Scott Boss is a software engineer who has been with Intel for eight years and has been working at the Smart Toy Lab for the past two years. Before the Toy Lab, Scott worked in the Intel® Architecture Labs on various multimedia projects ranging from audio infrastructures to 3D graphics. He received his B.S. degree in Computer Science from Valparaiso University in 1991 and his M.S.

degree in Computer Science from Purdue University in 1993. His e-mail is scott.d.boss@intel.com.

Henry Bruce is a hardware engineer who has been with Intel at the Smart Toy Lab for a year. Before Intel, Henry worked for VLSI Vision Ltd. (VVL) in Edinburgh, latterly acquired by STMicroelectronics. As the software development manager he developed expertise in video processing and device drivers. He received his B.S. degree in Electronic Engineering in 1985 and his Ph.D. in Image Processing in 1993, both from the University of Edinburgh in Scotland. His e-mail is henry.bruce@intel.com.

Charlie Case has been a hardware engineer at the Intel Smart Toy Lab for two years. Prior to working at Intel, Charlie was Director of Engineering for ThrustMaster, Inc., a leading PC game controller company. And for eight years prior to that, he was product-engineering manager at Gyration, Inc., a wireless PC peripheral manufacturer. He received his B.S. degree in Mechanical Engineering from The University of California, Berkeley in 1985, and his M.S.M.E. degree from Stanford University in 1990. His e-mail is charlie.w.case.jr@intel.com.

Kendal Miller is an integration engineer who has been with Intel for five years and has been working at the Smart Toy Lab since it was formed three years ago. Prior to this, Kendal worked in Intel's OEM Platform Solutions Division working with several external vendors on various consumer software titles. He received his B.S. degree in Computer Science from the University of Louisiana at Lafayette in 1996. His e-mail is kendal.miller@intel.com.

Copyright © Intel Corporation 2001. This publication was downloaded from <http://developer.intel.com/>.

Legal notices at <http://developer.intel.com/sites/developer/tradmarx.htm>.

Intel Play is a registered trademark or trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Intel and Intel Play are registered trademarks or trademarks of Intel Corporation or its subsidiaries in the United States and other countries.