Aberdeen Group
A Harte-Hanks Company

Embedded Software

The Future of Innovation in Tomorrow's Products

October 2011

Michelle Boucher, Colin Kelly-Rand

Aberdeen Group
A Harte-Hanks Company

Executive Summary

Embedded software offers new and exciting opportunities to bring innovation to products. However, challenges such as changing requirements and increasing complexity make the development process difficult. This report offers guidance to help companies address these challenges so that they may improve their process for developing embedded software to bring products to market that meet market needs and ultimately yield greater profitability.

Best-in-Class Performance

Aberdeen used the following five key performance criteria to distinguish Best-in-Class companies with top performers achieving the following results:

- 88% of products launch on time
- 87% of products meet revenue goals
- 89% of feature requirements from project kick-off make it into the final product
- 89% of software is working to requirements at the original scheduled release date
- 89% of products meet cost targets

Competitive Maturity Assessment

When compared to competitors, firms enjoying Best-in-Class performance shared several common characteristics to support system design, including:

- 32% more likely to verify requirements have been met earlier in the process
- 20% more likely to adopt a modular architecture
- 25% more likely to test code earlier in the development lifecycle
- 44% more likely to improve the ability to manage software configurations

Required Actions

In addition to the specific recommendations in Chapter Three of this report, to achieve Best-in-Class performance, companies must:

- Use a change management process to govern changes to the requirements
- Implement requirements traceability across all stages of product development
- Conduct a gap analysis to ensure all requirements are traceable to a component

Research Benchmark

Aberdeen's Research Benchmarks provide an in-depth and comprehensive look into process, procedure, methodologies, and technologies with best practice identification and actionable recommendations

Aberdeen Group
A Harte-Hanks Company

Table of Contents

Figures

Tables

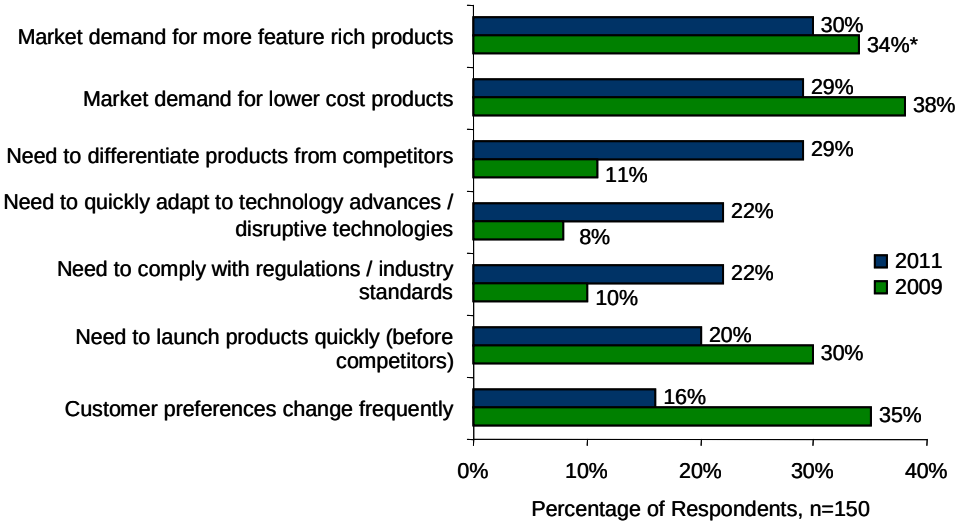
Chapter One: Benchmarking the Best-in-Class

Research from Aberdeen's November 2010 report, *Using Product Analytics to Keep Engineering on Schedule and on Budget*, found that the most successful discrete manufacturers are 30% more likely than their competitors to use electronics and embedded software to bring innovation to their products. While embedded software offers new opportunities for bringing innovation to products, the level of complexity it adds, as well as zero tolerance for bugs, brings many unique challenges to the development process. Where should companies focus to improve their development process? What are the best practices that should be considered? How have the economic changes over the last two years impacted the approaches to developing embedded code? To answer these questions, Aberdeen studied the experiences of 150 companies from June until September 2011 using an online survey and follow up interviews. Survey respondents with products that include embedded software report that over the last two years, the amount of embedded code in their products has increased by 18%, highlighting the fact that embedded software continues to become an increasingly critical part of product development.

The Business Impact of Embedded Software

To understand the external factors affecting the development of embedded software, respondents were asked to pick the top two pressures driving development improvements (Figure 1). The results were then compared to the results from Aberdeen's March 2009 *Embedded Systems Development: Three Proven Practices for Speed and Agility* report.

Figure 1: Top Business Pressures Driving Improvements in Developing Embedded Software



Source: Aberdeen Group, September 2011

It should be noted that in 2009, market demand for feature rich products was not included as part of the pressures question. However, making products more feature rich was rated as a top objective for improving embedded system development by 34% of survey respondents. This data has been included in Figure 1 as a reference, but it is not a straight comparison. Consequently, the data point has been noted with an asterisk to indicate the inconsistency.

Figure 1 represents some very interesting trends. In 2009, companies were slashing budgets just to survive, unemployment peaked, and costs were top of mind. Getting lower cost products out to market before the competition was a top business driver. Replacing physical components with embedded software was a way to take cost out of products. In addition software takes less time to produce. Further, the economic changes in 2009 resulted in sudden changes in customer preferences, particularly price sensitivity, that left companies struggling to quickly adapt their product strategies.

Now, two years later, things look much different. While cost is still important, a focus on more innovative products is now on top. The drop in the cost pressure is an indication that the economy may be improving, in addition to the fact that over the last two years companies have adapted to strategies that will result in lower cost products. Acknowledging that cost is still important, companies know they need to try harder to influence buying behavior. To do this, they are focused on adding more innovative features that improve the customer experience and competitively differentiate products. The time pressure is still important, but it is manifesting itself in the need to quickly take advantage of new technological advances that will help to further differentiate products from the competition. Finally, the pressure to comply with regulations and industry standards has grown quite a bit over the last couple of years. While users of desktop software programs have become tolerant of software bugs and software quality issues, users of devices with embedded software are not. People expect their phones, kitchen appliances, and cameras to work, bug free and simply applying a software patch is not an option. In the case of automobiles, airplanes, medical devices, or even factory controllers, a bug in the software could be far more catastrophic and can result in serious injury or even death. Recognizing this situation, industry standards for better coding are growing in importance.

"Bug tracking tools helped us to improve quality of code, trace errors and defects. Code reviews helped on detecting problems earlier, making the code clean and readable."

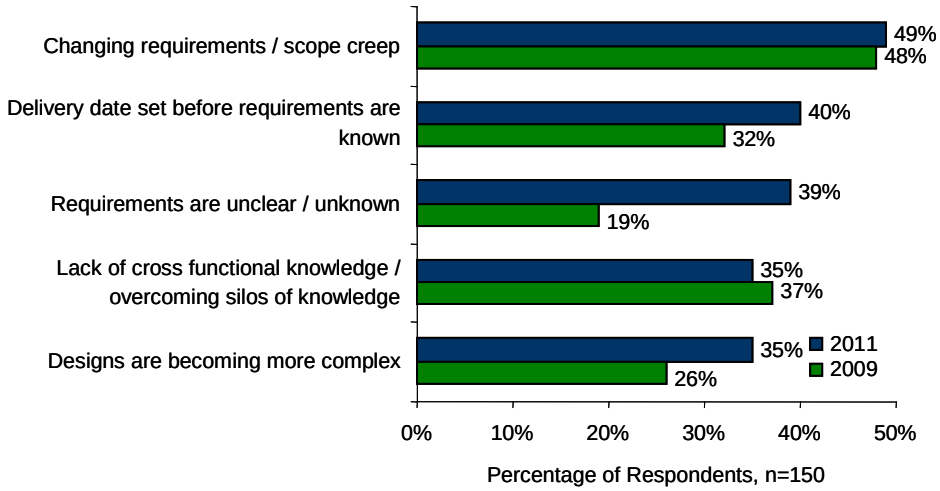
~Pablo Perinetti, Engineering Manager, Sitrack

Improving the process for developing embedded software is a smart way to address these pressures. Software costs less to produce than physical components so it continues to be a good way of taking cost out of products. In addition, its flexibility offers many opportunities to add innovation to products that will tailor the customer experience, driving customer demand and providing competitive differentiation. This is further complemented by a host of newer technologies such as GPS, wireless communications, smart grids, cloud technologies, and others which embedded software can take advantage of to bring even more innovation to products. All of these steps ultimately lead to bringing in more product revenue for the company.

Where to Focus when Improving Embedded Software Development?

To understand the biggest pain points that must be addressed in order to improve the process of developing embedded software, respondents were asked to pick the top three challenges of developing embedded software. These challenges were again compared to the March 2009 results (Figure 2).

Figure 2: Top Challenges of Developing Embedded Software



Source: Aberdeen Group, September 2011

As was the case in 2009, managing requirements and increasing product complexity continues to be the biggest source of pain when developing embedded software, and those challenges have become even greater in 2011.

As companies look to bring the greatest level of innovation to their products to optimize product revenues, and take advantage of the latest technologies, changing requirements and scope creep become big issues, as indicated by nearly half of survey responses. This situation means the developer needs to be able to identify how a change in requirements impacts the code or otherwise risks introducing bugs into the code. The next two challenges are further symptoms of this situation. While the requirements are in flux, the delivery date does not change. This makes it much harder for the developer to meet schedule deadlines and ultimately creates a risk of being late to market. In addition, this means that the schedule deadlines are set without considering the level of code complexity needed to meet the requirements. This creates time pressures on the developer than means either rushing the development process, sacrificing code quality, or being late. Changing requirements and setting deadlines prior to defining requirements also leads to a lot of uncertainty around what the requirements are, creating additional risks around code quality. This has become an even bigger issue in 2011 as respondents were 2.1 times more likely than they were in 2009 to indicate unclear requirements is a top challenge. This is further exasperated by increasing design complexity which was a top challenge in 2009, but is even bigger in 2011.

As embedded software becomes even more integral to product development, products are evolving into an integrated system of systems. This situation is contributing to increasing product complexity. This level of integration means that now with every decision engineers make, they must now consider its impact on other aspects of the product, not just within their own engineering discipline, but across engineering disciplines. This is challenging because engineers are experts in their respective engineering field, but are less knowledgeable about other engineering disciplines. This inherently creates natural silos of engineering knowledge. However, the interdependency of the different components created by different engineering disciplines means these silos of knowledge create a challenge. In addition, knowledge across the development lifecycle from linking what the customer wants, to what is developed, to what is tested must also be managed to ensure the final product will meet market needs and bring in expected revenues.

The Maturity Class Framework

To understand successful approaches for developing embedded software and the business impact it has on companies, Aberdeen benchmarked the performance of study participants and categorized them as either Best-in-Class (top 20% of performers), Industry Average (mid 50%), or Laggard (bottom 30%). The top business pressures driving improvements in embedded software are:

- Market demand for feature rich products
- Cost
- Managing requirements for competitive differentiation, regulations, and taking advantage of new technologies
- Time to market pressure

"Establish standards at the beginning of a project."

~Stuart McCallister, Researcher, Archangel Systems

Consequently, four key performance measures that indicate success with addressing these pressures were used to distinguish the Best-in-Class from Industry Average and Laggard organizations. The performance of each of these tiers is displayed in Table 1. These metrics show success with meeting market needs in order to achieve revenue targets, incorporating the intended features into the final product to drive customer demand and meet regulatory compliance, manage product cost, while still releasing products on time.

Table 1: Top Performers Earn Best-in-Class Status

Definition of Maturity Class	Mean Class Performance
Best-in-Class: Top 20% of aggregate performance scorers	▪ 89% of feature requirements from project kick-off that made it into final product ▪ 89% of software working to requirements at original scheduled release date ▪ 88% of product launch dates met ▪ 87% of product revenue targets met ▪ 89% of product cost targets met
Industry Average: Middle 50% of aggregate performance scorers	▪ 84% of feature requirements from project kick-off that made it into final product ▪ 75% of software working to requirements at original scheduled release date ▪ 71% of product launch dates met ▪ 74% of product revenue targets met ▪ 73% of product cost targets met
Laggard: Bottom 30% of aggregate performance scorers	▪ 73% of feature requirements from project kick-off that made it into final product ▪ 47% of software working to requirements at original scheduled release date ▪ 38% of product launch dates met ▪ 51% of product revenue targets met ▪ 39% of product cost targets met

Source: Aberdeen Group, September 2011

In order to meet market demand with the right features, and ultimately bring in more product revenue, those features need to make it into the final product. While the metrics for this show the Best-in-Class do this better than their peers, it is really the other metrics that tell the story of how much better. The Best-in-Class are 19% more likely than the Industry Average to have the software working to requirements at the original scheduled release date. When the software is not working, the options are to cut features or delay the release. Cutting features means potentially losing competitive differentiation or missing out on innovations that will drive customer demand. This has a direct impact on revenue. Given that the Industry Average are still able to get 84% of the original features into the product, they are more likely to push out the release date. Since the Best-in-Class have more of their software working, they are 24% more likely to meet their launch dates. Missing launch dates means being late to market, and consequently a shortened window of opportunity to bring in expected revenues. The Best-in-Class are 18% more likely than the Industry Average to bring in expected product revenues, clearly putting them at a competitive advantage. Further helping profitability, the Best-in-Class are 23% more likely than the Industry Average to meet their cost targets. This enables them to do a better job of addressing the number two market pressure, customer demand for lower cost products. By addressing this pressure, the

Best-in-Class overcome another barrier to influencing buyer behavior by addressing the needs of the price conscious.

As a result of their practices, the Best-in-Class enjoy other benefits too (Table 2). Because they are able to get more right the first time as well as improve the efficiency of their development process, they have been able to reduce the development cycle 2.2 times that of their peers over the last two years. This also helps their ability to meet launch dates. Most of their peers did see a small decrease in product cost over the last two years, averaging just 1%. One the other hand, the Best-in-Class have been able to reduce product cost by 7%. Finally, lower cost products, that have the features customers want, go to market on time and have the longest window of opportunity to bring in revenue, will be more profitable. The Best-in-Class are enjoying this competitive advantage with profit margins that increased 2.1 times that of their competitors over the last two years.

Table 2: Additional Best-in-Class Benefits

Performance Change over Last Two Years	Best-in-Class	All Others
Change in development cycle	9% Decrease	4% Decrease
Change in product cost	7% Decrease	1% Decrease
Change in profit margins of new products	15% Increase	7% Increase

Source: Aberdeen Group, September 2011

"I advise that you first model the system with as many parameters as the real system and simulate for all aspects before implementation."

~ML Shetty, Electronics Corporation of India Limited

The Best-in-Class PACE Model

Successful embedded software development requires a combination of strategic actions, organizational capabilities, and enabling technologies that can be summarized as shown in Table 3.

Table 3: The Best-in-Class PACE Framework for Embedded Software

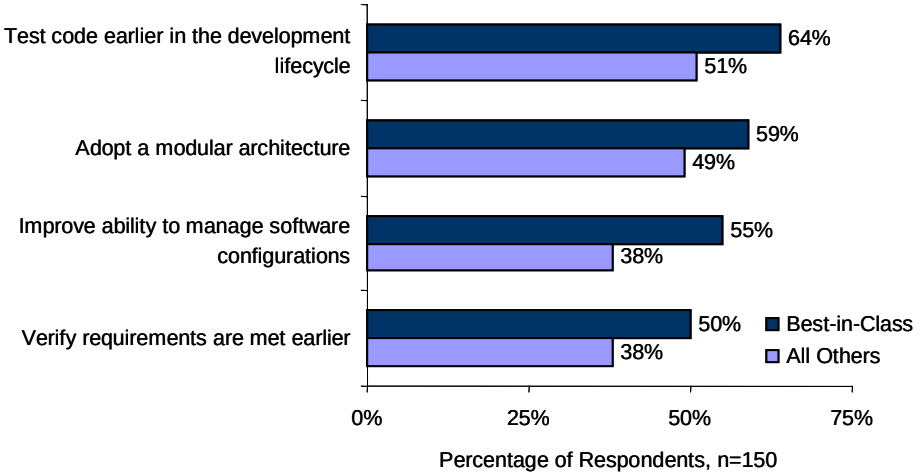
Pressures	Actions	Capabilities	Enablers
▪ Market demand for more feature rich products	▪ Test code earlier in the development lifecycle ▪ Adopt a modular architecture	▪ Customer needs defined prior to defining requirements ▪ Requirements prioritized for implementation ▪ Changes to requirements are governed by a defined change management process ▪ Test plan is linked to requirement ▪ Gap analysis conducted to ensure all requirements are traceable to a component	▪ Central repository for software ▪ Block diagrams ▪ IDE ▪ System engineering tools ▪ Integrated software modeling and coding ▪ Integrated embedded software and control design ▪ UML modeling ▪ Hardware/ Software dependency management

Source: Aberdeen Group, September 2011

Best-in-Class Strategies

Given the performance benefits enjoyed by the Best-in-Class, they are clearly doing a better job of addressing the challenges of developing embedded software. The top strategies implemented by the Best-in-Class are shown in Figure 3.

Figure 3: Top Strategies for Developing Embedded Software



Source: Aberdeen Group, September 2011

The Best-in-Class are focused on strategies that will help them manage increased product complexity as well as ensure they develop what they intended. By focusing on these things they improve code quality, make sure their code is in compliance, meet customer needs, and catch problems earlier when they are faster and less expensive to address.

The Best-in-Class are 32% more likely than their competitors to verify requirements are met earlier in the process and 25% more likely to validate code with tests earlier in the process. Both of these strategies help them ensure the original feature requirements make it into the final product as well as make sure the code is working as expected at the scheduled release date. In addition, this earlier validation helps to catch problems sooner in the development process, when they are easier to address, which leads to greater efficiency and more predictability within the development process, helping the Best-in-Class to meet launch dates.

"My advice is that if you don't have anything implemented then start at least using good and standard programming practices. Then add bug tracking and system design tools. It really does a great difference on your final product."

~Pablo Perinetti, Engineering Manager, Sitrack

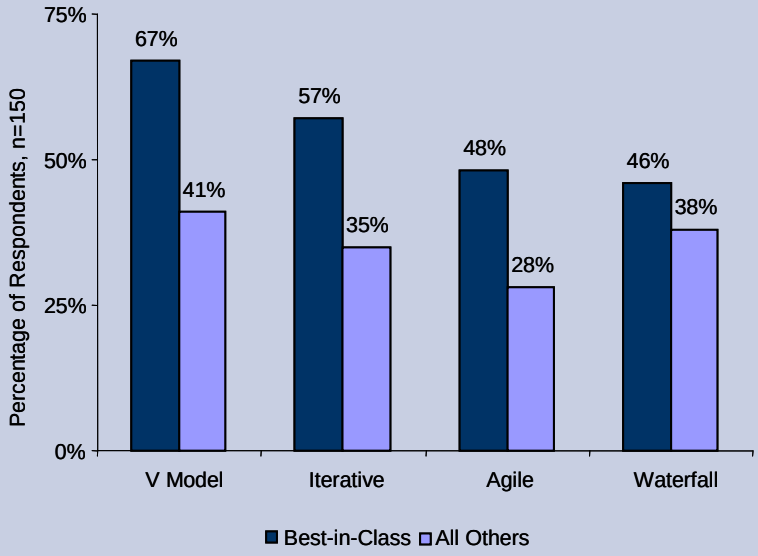
The Best-in-Class are 20% more likely than their competitors to adopt a modular architecture and 45% more likely to improve their process for managing software configurations. A modular architecture refers to a system made up of modules or components that can be swappedout, or new ones added, without impacting the rest of the system. This approach can be applied to both hardware and software. By managing code as modular elements, modules can be mapped to a particular function or requirement. Because each module can stand on its own, it is easier to test it and support the strategy to ensure requirements are met. If the test fails,

it is much easier to identify the bug because it is limited to one isolated module of code. By managing software configurations, code can be assembled based on the functions and requirements met by each module. This process promotes greater reuse which saves coding time in addition to improving quality because each module has already been validated to work. These time savings contribute to the Best-in-Class' ability to meet launch dates. The strategies also make it easier to ensure the needed requirements make it into the final product.

Aberdeen Insights — Strategy

Part of the strategy for developing embedded software involves the methodology used to develop the code. Over the years, many different methodologies have become available. Figure 4 shows adoption levels for some of the more popular methodologies.

Figure 4: Which Methodologies Govern Development



Source: Aberdeen Group, September 2011

The most popular method is the more structured V Model. However, both the Iterative and Agile methods also show high adoption and a large amount of differentiation between the Best-in-Class and their competitors, showing these methodologies contribute to Best-in-Class success. The V Model has been around much longer than the Agile approach so it makes sense that it would see greater levels of adoption.

continued

Methodology Definitions

Waterfall: A sequential top to bottom process where each development phase (Define Requirements, Design, Implementation, Verification. Maintenance) is completed before moving on to the next phase.

V Model: Can be considered an extension of the waterfall model, but the rather than top to bottom, it follows a "V" shaped framework to define the relationships between the phases with Requirements and Design on the left side of the V, Implementation on the bottom, and Verification, Test, and Maintenance on the right side sloping up.

Iterative: Rather than the more linear methodologies defined by the waterfall and V modes, the iterative methodology supports a more cyclical process where a small part of the code is defined, developed and tested. With each iteration, more functionality is added until the design is complete.

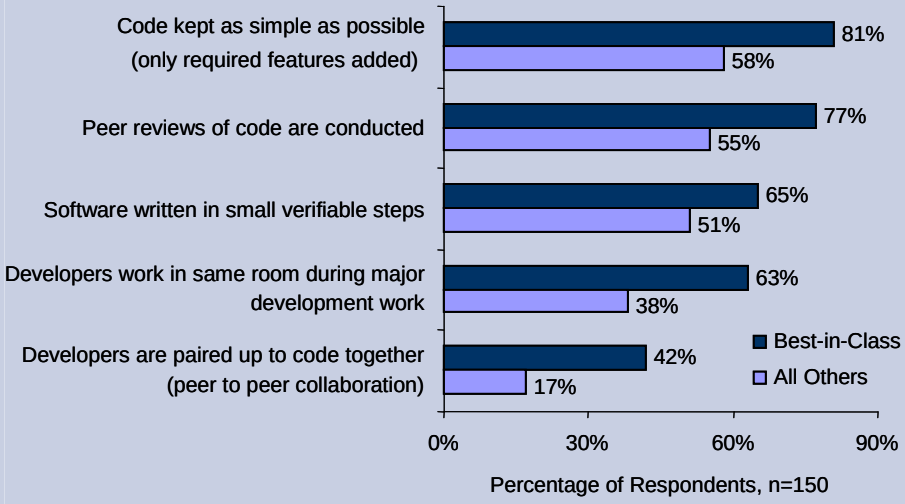
Agile: A newer methodology introduced in 2001. It builds upon the iterative methodology where requirements and design evolve in smaller incremental steps and code is kept as simple as possible. It encourages heavy collaboration among small teams and promotes greater flexibility to quickly adapt to changes.

Aberdeen Group
A Harte-Hanks Company

Aberdeen Insights — Strategy

However, many of the capabilities associated with the Agile Methodology are in place at Best-in-Class companies (Figure 5).

Figure 5: Agile Capabilities in Place



Source: Aberdeen Group, September 2011

This data suggests that one methodology is not used, but instead, development processes are evolving into a hybrid of the V Model with aspects of an Agile methodology incorporated into it. Given the unique pressures and challenges of developing embedded software, particularly lower tolerance for software bugs, development needs more focus on getting it right the first time as well as the ability to quickly respond to evolving requirements. Since the Agile methodology was developed with these challenges in mind, Best-in-Class companies are finding it is helping them to meet their objectives, but most have not yet committed to Agile 100%.

In the next chapter, we will see what the top performers are doing to achieve their success.

Aberdeen Group
A Harte-Hanks Company

Chapter Two: Benchmarking Requirements for Success

Chapter One demonstrated the pressures driving companies to improve how embedded software is developed and the challenges associated with it. It also described the efficiency improvements that can be realized with successful approaches to developing embedded software. Chapter Two explores the capabilities and enabling technologies the Best-in-Class use to execute those strategies, allowing them to enjoy a competitive advantage.

Case Study — Diamond Systems Corporation, Having Success with Reuse

Diamond Systems Corporation (DSC), a small form factor embedded computer company, produces components of a wide variety of embedded systems, alarm monitoring systems, and standard and wireless control panels. Its equipment has required embedded software since its first single board computer. Back then, there was not a great sense of code reuse as the hardware the code ran on had limited abilities and needed small and specific codes to operate. DSC operated in this manner for a long period of time.

That is, until now. The whole industry has changed and requires faster turn-around on a multitude of concurrent products. If DSC kept writing new code for each board the engineering overhead would go through the roof; new code per product was no longer a feasible operating method. James Moore, a software engineer with DSC can attest to that, "DSC had the tendency, as most places I have worked, to focus on getting the immediate deliverables out the door as fast as possible." Engineers wrote the software without much planning, which resulted in software that only did what was required for the project. Certainly if each of DSC's products were completely unique, that method would make sense as code reuse would not add much value, but that is not the case. Most of DSC's products are variations of the same product, with a minor buttress or precision and speed enhancement which are a perfect applications for promoting reuse.

Moore, as well as the rest of DSC, realized hard-coding was a "complete waste of time" and concentrated on moving towards a method that fit their product catalogue. To take advantage of the similarities within the products, DSC modularized their software code – factoring out the common chunks of code. The cost of the setup time for the engineers to basically review, cut, copy and paste chunks of code into identifiable and reusable chunks paled in comparison to the savings reusing code allows. James estimates that coding time for a software product decreased from a month to less than a week. This was another case of how superior upfront planning can set up huge gains down the road. There is an additional benefit as Moore now can improve the quality of the code by spending more time testing and not rushing code out the door.

Fast Facts

Over the last two years, Best-in-Class companies have seen a:

- √ 9% decrease in development time
- √ 7% decrease in product cost
- √ 15% increase in profit margins

Competitive Assessment

Aberdeen Group analyzed the aggregated metrics of surveyed companies to determine whether their performance ranked as Best-in-Class, Industry Average, or Laggard. In addition to having common performance levels, each class also shared characteristics in five key categories: (1) **process** (the approaches they take to manage embedded software development); (2) **organization** (how responsibilities are managed); (3) **knowledge management** (how embedded software development details are managed); (4) **performance management** (the ability of the organization to measure its results to improve its embedded software processes); and (5) **technology** (the appropriate tools used to support embedded software development). These characteristics (identified in Table 4) serve as a guideline for best practices, and correlate directly with Best-in-Class performance across the key metrics.

Table 4: The Competitive Framework

	Best-in-Class	Average	Laggards
Process	Customer needs defined prior to defining requirements		
	91%	76%	73%
	Requirements prioritized for implementation (must vs. nice to have)		
	81%	62%	54%
	Requirement changes governed by a change management process		
	77%	57%	39%
Organization	Requirements are mapped to the individuals assigned to code them		
	57%	44%	34%
Knowledge	Requirements traceability available across development stages (design, test, etc)		
	77%	64%	43%
	Test plan is linked to requirement		
	70%	57%	56%
	Tests on host machine are run on target without rewriting them		
	67%	51%	51%
Performance	Gap analysis conducted to ensure all requirements are traceable to a component		
	61%	35%	26%
	Software metrics captured to improve on-time delivery, quality, error rates, etc.		
	59%	33%	33%

"Increase visibility of the design process and artifacts across the organization."

~Bo Jönsson, Project Manager, Inor Process AB

	Best-in-Class	Average	Laggards
Technology	Technologies currently in use:		
	▪ 93% Central repository for software assets ▪ 89% Block diagrams ▪ 85% IDE ▪ 73% System engineering tools ▪ 65% Integrated software modeling and coding ▪ 65% Integrated embedded software and control design ▪ 64% UML modeling ▪ 48% Hardware/ Software dependency management	▪ 76% Central repository for software assets ▪ 66% Block diagrams ▪ 64% IDE ▪ 52% System engineering tools ▪ 36% Integrated software modeling and coding ▪ 40% Integrated embedded software and control design ▪ 40% UML modeling ▪ 33% Hardware/ Software dependency management	▪ 71% Central repository for software assets ▪ 58% Block diagrams ▪ 62% IDE ▪ 43% System engineering tools ▪ 30% Integrated software modeling and coding ▪ 35% Integrated embedded software and control design ▪ 35% UML modeling ▪ 17% Hardware/ Software dependency management

Source: Aberdeen Group, September 2011

"Moving to more standard emulator/hardware in loop processes, which shortens the test cycle, improved our development process."

~Director, Global Aerospace and Defense Corporation

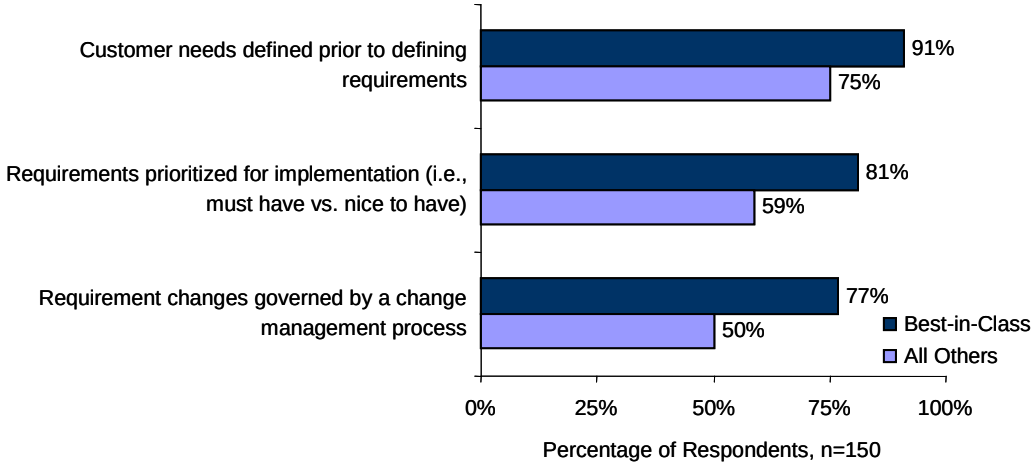
Capabilities and Enablers

Based on the strategies deployed to support embedded software development, the findings of the Competitive Framework and interviews with end users, Aberdeen's analysis of the Best-in-Class reveals where companies must focus to improve embedded software development to get the right product to the market. Processes, organizational responsibility, knowledge management, performance management, and technology all play a role in supporting this. Much of successful execution of the strategies depends on the right capabilities to capture and manage what is expected of the software. With the right capabilities in place, the technology tools will bring more value.

Process

The most differentiated processes implemented by the Best-in-Class are shown in Figure 6. The Best-in-Class focus on processes that enable them to capture and manage what is expected of the software. These processes provide the support needed to execute the strategies.

Figure 6: Best-in-Class Processes for Developing Embedded Software



Source: Aberdeen Group, September 2011

The top business pressure to improve embedded software development is the market need for more feature rich products. Getting this right will lead to greater customer demand for products, which ultimately leads to greater profitability. To accomplish this, the Best-in-Class start with understanding what the customer wants. They can then tie this to design requirements. In addition, with a better understanding of customer needs, they can then put a higher priority on requirements that will drive customer demand, and a lower priority on the things that are less important. Development can then focus on the most important requirements first. This prioritization also becomes important when deadlines are approaching and it is obvious that not everything will make it into the final product. When the prioritization has been done up front, with a link to market needs, it is much easier to make those difficult decisions around what should be cut.

"System requirements and understanding has become simplified with the use of simulation tools."

~ML Shetty, Electronics Corporation of India Limited

The top challenge is changing requirements and scope creep. To address this challenge, the Best-in-Class are 54% more likely than their competitors to ensure that changes to the requirements are under a formal change management process. A change management process limits changes to only those that are justified, and ultimately supports the goal of ensuring the final product meets market needs without additional complications.

Organization

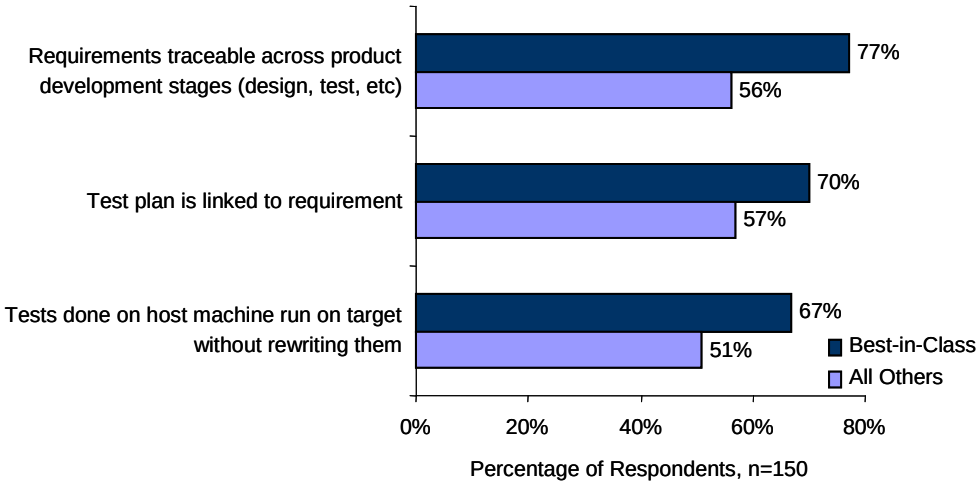
One of the top challenges is the lack of cross functional knowledge. While the inherent knowledge silos are difficult to address, supporting better collaboration will help to overcome some of the knowledge gaps. One of the first steps to supporting that collaboration is providing the team with visibility into who is working on what. That way, it is clearer to members of the team who they should reach out to with questions around interdependency between components in order to meet design

requirements. It also makes it easier to know who to notify when there are design changes that will impact them. With that in mind, the Best-in-Class are 30% more likely than the Industry Average to map requirements to the specific individuals who will code them.

Knowledge Management

After having the capabilities in place to capture market needs, map that to requirements, and assign that to the right person, the Best-in-Class have the right capabilities in place to make that knowledge available that will lead to successful execution of the strategies (Figure 7).

Figure 7: How the Best-in-Class Manage Knowledge to Support Embedded Software Development



Source: Aberdeen Group, September 2011

The top challenge of developing embedded software is changing requirements, in addition to increasing product complexity. Given the complexity of the design, when those requirements change, it is not easy to identify all the places impacted by the change. The Best-in-Class are 38% more likely than their competitors to address this by having the requirements traceable across the design lifecycle. This means that it is easy for them to understand the impact all the way from the market need, to the design, to the test plan. This traceability also helps them to execute on their strategy to verify that requirements are met easier in the design process. It also supports their ability to have the software working at release time.

"Make sure that the *entire* organization has knowledge and understanding of relevant subjects."

~Bo Jönsson, Project Manager, Inor Process AB

Furthering this, the Best-in-Class are 23% more likely than their competitors to link the test plan to the requirement. Thinking about the test plan when the requirement is defined ultimately leads to better defined requirements because thought must be given to how it can be validated up front. This helps to address the challenge of unclear requirements. This also helps the Best-in-Class execute on their strategy to test code earlier in the process because the test case is linked and it is clearer what needs to be validated.

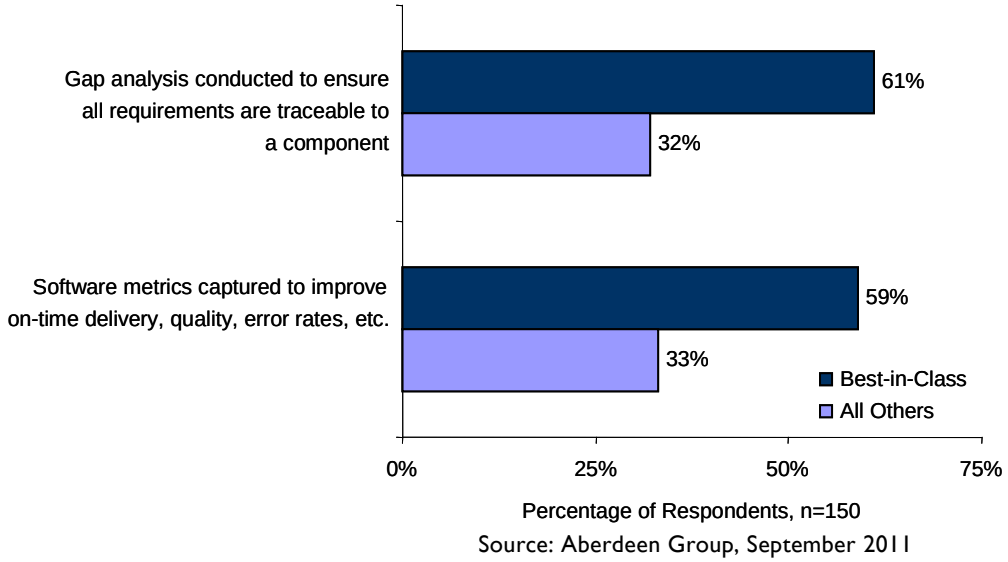
Ultimately this also contributes to the Best-in-Class' ability to have the software working as expected at release time.

Finally, the Best-in-Class are 31% more likely than their competitors to have the ability to run the same tests on the host and target machine. Not only does this make it easier to validate the code earlier in the process, but the end result is it leads to time savings. The ability to validate earlier means problems are caught sooner so they are easier to address. In addition, all the time that would have been needed to write the test for the target is saved. This contributes to the Best-in-Class' ability to meet 88% of their launch dates.

Performance Management

By managing performance, the Best-in-Class are able to continuously improve their process. The most differentiated capabilities they possess to manage their performance can be found in Figure 8.

Figure 8: How the Best-in-Class Measure Performance to Support Embedded Software Development



Source: Aberdeen Group, September 2011

"Use hardware / mechanical / software integrated virtualization layers to simulate the entire systems - no matter the cost. It *will* save you time in the hindsight of your development cycle. "

~James Moore, Software Engineer, DSC

A key piece of executing on the strategy to make sure requirements are met is to make sure there are components in place to meet each requirement. The Best-in-Class are 91% more likely than their competitors to perform a gap analysis to measure and manage this. Given the increasing level of product complexity, this can be challenging, but having the traceability in place and the ability to do a gap analysis, it is much easier to manage the process and have better visibility to the status on the implementation of each requirement.

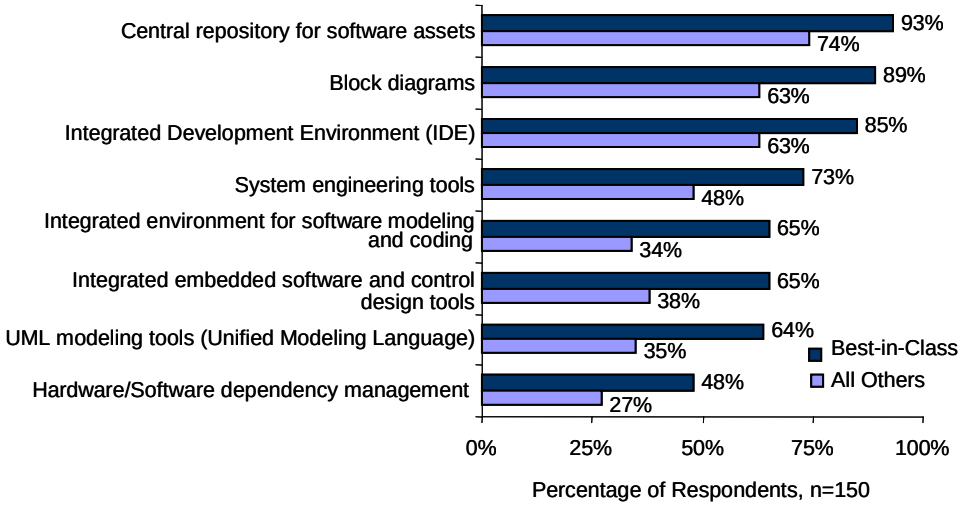
Furthering to support better visibility, the Best-in-Class are 79% more likely than their competitors to capture software performance and development metrics. Having visibility into these metrics makes it easier to catch potential

problems that could lead to quality issues or missed deadlines so that corrective action may be taken in time to keep the project on schedule.

Technology

A variety of tools are used by the Best-in-Class to support the development of embedded software (Figure 9).

Figure 9: Technology Tools Used by the Best-in-Class to Support Embedded Software Development



Source: Aberdeen Group, September 2011

A central repository for software assets as well as hardware and software dependency supports better collaboration within the development team, helping to overcome the lack of cross functional knowledge. With a central repository, it is easier to find the latest version of code, which is especially challenging when there are frequent changes. A better understanding of the dependencies between hardware and software also helps to provide the proper notifications about changes to one that impacts the other. Block diagrams, system engineering tools, and UML modeling help to manage design complexity and make it easier to execute on the strategy to develop a more modular design. Integrated development tools such as an IDE or tools that integrate modeling and code or embedded software and control design tools increase efficiency because work can be done in one environment. They also support better quality because work does not need to be recreated or copied over.

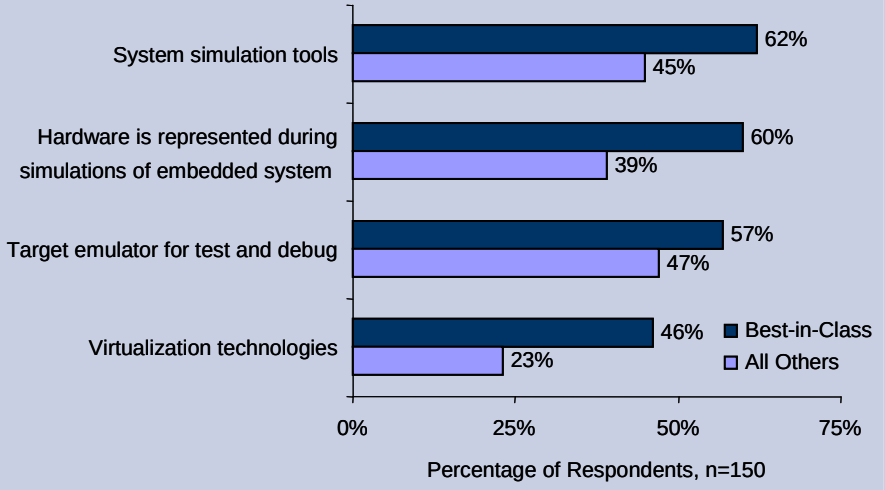
"I advise that you first model the system with as many parameters as the real system and simulate for all aspects before implementation."

~ML Shetty, Electronics Corporation of India Limited

Aberdeen Insights — Technology

Determining what will drive market demand, mapping that to requirements, and making sure those requirements make it in the final design are critical to ensuring the final product will bring in the expected revenues. Defining requirements, managing them, and ensuring traceability is the first step as everything trickles down from that. With a clear definition of what is needed, verifying and validating that was done is also critical. With the right capabilities in place to make it clear what must be verified and validated, technology can be used to support this test phase. The most differentiated technologies used by the Best-in-Class can be found in Figure 10.

Figure 10: Technologies to Support Test



Source: Aberdeen Group, September 2011

One of the Best-in-Class strategies is to verify requirements have been met earlier in the design. The Best-in-Class take advantage of simulation technologies to help them with this. This allows them to do quick verifications before physical prototypes even exist. The challenge is that embedded software is part of a large system and it is the interactions between hardware and software that are difficult to assess. The Best-in-Class look to use system simulation tools and are 54% more likely to make sure the hardware is represented during the simulation of embedded system functions. Further helping to identify problems early on, the Best-in-Class are twice as likely to use virtualization technologies to assess product behavior before a complete physical prototype exists. Finally the Best-in-Class are 21% more likely to use a target emulator, to both test and debug the software to provide another mechanism for testing earlier in the process. These technologies allow them to make sure the embedded software works correctly at release time so that they can meet their launch dates.

"Better design, better testing, more reuse, and guaranteed drop-in replacement compensation designs reduce the amount of RMA's you will receive."

~James Moore, Software Engineer, DSC

Aberdeen Group
A Harte-Hanks Company

Chapter Three: Required Actions

Whether a company is trying to move its embedded software development performance from Laggard to Industry Average, or Industry Average to Best-in-Class, the following actions will help spur the necessary performance improvements:

Laggard Steps to Success

- **Use a change management process to govern changes to the requirements**. Requirement changes is the top challenge of embedded software development. To address this, the Best-in-Class are 97% more likely than Laggards use this capability.
- **Implement requirements traceability across all stages of product development**. Requirements traceability helps to identify the impact of a change. The Best-in-Class are 79% more likely than Laggards to have this.
- **Support hardware and software dependency management.** Dependency management provides an understanding of how hardware and software components relate to each other making it easier to assess the impact of changes as well as enable better decisions for both hardware and software engineers. The Best-in-Class are 2.8 times more likely than Laggards to have implemented this.

Industry Average Steps to Success

- **Implement an integrated environment for software modeling and coding**. Modeling helps to address the challenge of increasing product complexity and an integrated environment for modeling and coding automates the process of generating code based on the model, saving time and reducing the chance of errors. The Best-in-Class are 81% more likely than the Industry Average to have this.
- **Capture software metrics for on time delivery, quality, error rates and other related metrics** to identify potential problems that could result in poor quality or missed deadlines. Best-in-Class companies are 79% more likely than the Industry Average to have this visibility which allows them to not only intervene and take corrective action before there is a problem, as well as continuously improve.
- **Conduct a gap analysis to ensure all requirements are traceable to a component.** The Best-in-Class are 74% more likely than the Industry Average to do this which helps them to verify that all requirements will be met.

Fast Facts

The Best-in-Class are:

√ **97%** more likely than Laggards to use a change management process to govern changes to the requirements

√ **79%** more likely than Laggards to implement requirements traceability across all stages of product development

√ **74%** more likely than the Industry Average to conduct a gap analysis to ensure all requirements are traceable to a component

Best-in-Class Steps to Success

- **Send automatic notifications to affected engineers when changes impact other subsystems.** Changing requirements as well as the lack of cross functional expertise are both top challenges. This capability will help to manage change better as well as support better collaboration across the team to simultaneously address both challenges. Forty-seven percent (47%) of the Best-in-Class plan to implement this capability.
- **Ensure that requirements definitions are measurable and metric driven**. This capability will make it easier to validate that requirements have been met and more clearly communicate the requirement. Forty-five percent (45%) of the Best-in-Class plan to implement this capability.
- **Create a software test prior to writing code.** By defining the requirement and then the test, it will be easier to confirm that the requirement is written in a way that clearly shows what is needed and that it can be validated. This will result in clearer requirement definitions, a top challenge, as well as help to support the strategies to validate requirements and test code earlier in the process. Forty-three percent (43%) of the Best-in-Class plan to implement this capability.

Aberdeen Insights — Summary

Embedded software offers opportunities to bring new innovation to products. However, challenges such as changing requirements and increasing product complexity make developing embedded software difficult.

The practices Best-in-Class companies are following to address these challenges are enabling them to release products to market that have the needed requirements to drive market demand and meet revenue targets. Ultimately, this has led to a 15% increase in product profit margins. Those looking to achieve similar success, should focus on:

- Managing requirements so that they are clearly communicated, yet maintain traceability to assess the impact of inevitable changes so that market needs are met
- Identify ways to manage complexity to simplify development and reduce risks of introducing software quality issues
- Support methods for reuse, and improve efficiency to help developers address the time pressures inherent to their development process
- Facilitate collaboration to capitalize on the collective expertise of the development team while avoiding delays resulting from miscommunication

Appendix A: Research Methodology

Between June and September 2011, Aberdeen examined the results, the experiences, and the intentions of 150 enterprises in a diverse set of industries.

Aberdeen supplemented this online survey effort with interviews with select survey respondents, gathering additional information strategies, experiences, and results.

Responding enterprises included the following:

- *Job title:* The research sample included respondents with the following job titles: Executive level manager (18%); VP/Director (15%); Manager (30%); Engineer (31%); and other (6%).
- *Industry:* The research sample included respondents from a wide cross section of industries. The sectors that saw the largest representation in the sample were aerospace and defense (31%); high tech (19%); industrial equipment manufacturing (14%); automotive (14%); medical devices (6%); and telecommunications (7%).
- *Geography:* The majority of respondents (49%) were from North America. Remaining respondents were from Europe (25%), the Asia / Pacific region (17%), and from the rest of the world (9%).
- *Company size:* Twenty-five percent (25%) of respondents were from large enterprises (annual revenues above US $1 billion); 30% were from midsize enterprises (annual revenues between $50 million and $1 billion); and 45% of respondents were from small businesses (annual revenues of $50 million or less).
- *Headcount:* Twenty-two percent (22%) of respondents were from small enterprises (headcount between 1 and 99 employees); 36% were from midsize enterprises (headcount between 100 and 999 employees); and 42% of respondents were from large enterprises

Study Focus

Respondents completed an online survey that included questions designed to determine the following:

√ What is driving organizations to improve how develop embedded software

√ The challenges they face developing embedded software

√ The actions these companies are taking to improve how they develop embedded software

√ The capabilities and technology enablers they have in place to support their development process

The study aimed to identify emerging best practices for the development of embedded software and to provide a framework by which readers could assess their own capabilities.

Aberdeen Group
A Harte-Hanks Company

Table 5: The PACE Framework Key

Overview
Aberdeen applies a methodology to benchmark research that evaluates the business pressures, actions, capabilities, and enablers (PACE) that indicate corporate behavior in specific business processes. These terms are defined as follows: **Pressures —** external forces that impact an organization's market position, competitiveness, or business operations (e.g., economic, political and regulatory, technology, changing customer preferences, competitive) **Actions —** the strategic approaches that an organization takes in response to industry pressures (e.g., align the corporate business model to leverage industry opportunities, such as product / service strategy, target markets, financial strategy, go-to-market, and sales strategy) **Capabilities —** the business process competencies required to execute corporate strategy (e.g., skilled people, brand, market positioning, viable products / services, ecosystem partners, financing) **Enablers —** the key functionality of technology solutions required to support the organization's enabling business practices (e.g., development platform, applications, network connectivity, user interface, training and support, partner interfaces, data cleansing, and management)

Source: Aberdeen Group, October 2011

Table 6: The Competitive Framework Key

Overview	
The Aberdeen Competitive Framework defines enterprises as falling into one of the following three levels of practices and performance: **Best-in-Class (20%) —** Practices that are the best currently being employed and are significantly superior to the Industry Average, and result in the top industry performance. **Industry Average (50%) —** Practices that represent the average or norm, and result in average industry performance. **Laggards (30%) —** Practices that are significantly behind the average of the industry, and result in below average performance.	In the following categories: **Process —** What is the scope of process standardization? What is the efficiency and effectiveness of this process? **Organization —** How is your company currently organized to manage and optimize this particular process? **Knowledge —** What visibility do you have into key data and intelligence required to manage this process? **Technology —** What level of automation have you used to support this process? How is this automation integrated and aligned? **Performance —** What do you measure? How frequently? What's your actual performance?

Source: Aberdeen Group, October 2011

Table 7: The Relationship Between PACE and the Competitive Framework

PACE and the Competitive Framework – How They Interact
Aberdeen research indicates that companies that identify the most influential pressures and take the most transformational and effective actions are most likely to achieve superior performance. The level of competitive performance that a company achieves is strongly determined by the PACE choices that they make and how well they execute those decisions.

Source: Aberdeen Group, October 2011

Aberdeen Group
A Harte-Hanks Company

Appendix B:
Related Aberdeen Research

Related Aberdeen research that forms a companion or reference to this report includes:

- *System Design: Get it Right the First Time;* August 2011
- *Using Product Analytics to Keep Engineering on Schedule and on Budget*; November 2010
- *System Engineering: Top Four Design Tips to Increase Profit Margins for Mechatronics and Smart Products*; October 2009
- *Embedded Systems Development: Three Proven Practices for Speed and Agility*; March 2009
- *Engineering Evolved: Getting Mechatronics Performance Right the First Time*; November 2008
- *System Design: New Product Development for Mechatronics*; January 2008

Information on these and any other Aberdeen publications can be found at www.aberdeen.com.

Authors: Michelle Boucher, Senior Research Analyst, Product Innovation & Engineering Practice, (michelle.boucher@aberdeen.com)
Colin Kelly-Rand, Research Associate, Product Innovation & Engineering Practice, (colin.kelly-rand@aberdeen.com)