

Separating Core from Context Can Bring Rapid Returns in Application Transformation

Transcript of the second in a series of sponsored BriefingsDirect podcasts on the rationale and strategies for application transformation.

[Listen](#) to the [podcast](#). Find it on [iTunes/iPod](#) and [Podcast.com](#). Download the transcript. [Learn more](#). Sponsor: [Hewlett-Packard](#).

Gain more insights into "Application Transformation: Getting to the Bottom Line" via a series of HP virtual conferences Nov. 3-5. For more on Application Transformation, and to get real time answers to your questions, register to the virtual conferences for your region:

[Register here](#) to attend the Asia Pacific event on Nov. 3.

[Register here](#) to attend the EMEA event on Nov. 4.

[Register here](#) to attend the Americas event on Nov. 5.

Dana Gardner: Hi, this is [Dana Gardner](#), principal analyst at [Interarbor Solutions](#), and you're listening to [BriefingsDirect](#).

Today, we present a sponsored podcast discussion on separating core from context, when it comes to [legacy enterprise applications](#) and their [modernization](#) processes. As enterprises seek to [cut their total IT costs](#), they need to identify [what legacy assets](#) are working for them and carrying their own weight, and which ones are merely hitching a [high cost](#) -- but largely unnecessary -- [ride](#).



The widening cost-in-productivity division exists between older, hand-coded software assets, supported by aging systems, and replacement technologies on newer, more efficient standards-based systems. Somewhere in the mix, there are core legacy assets distinct from so-called contextual assets. There are peripheral legacy processes and tools that are costly vestiges of bygone architectures. There is [legacy wheat and legacy chaff](#).

Today we need to identify productivity-enhancing resources and learn how to preserve and modernize them -- while also identifying and replacing the baggage or chaff. The goal is to find the most efficient and low-cost means to support them both, through up-to-date data-center architecture and off-the-shelf components and services.

This podcast is the [second in a series](#) of three to examine Application Transformation: Getting to the Bottom Line. We will discuss [the rationale](#) and likely returns from assessing the true role and character of legacy applications and their actual costs. The podcast, incidentally, runs in conjunction with some [Hewlett-Packard \(HP\)](#) webinars and [virtual conferences](#) on the same subject.

[Register here](#) to attend the Asia Pacific event on Nov. 3. [Register here](#) to attend the EMEA event on Nov. 4. [Register here](#) to attend the Americas event on Nov. 5.

With us to delve deeper into the low cost, high reward transformation of legacy enterprise applications is [Steve Woods](#), distinguished software engineer at HP. Hello, Steve.

Steve Woods: Hello. How are you doing?

Gardner: Good. We are also joined by [Paul Evans](#), worldwide marketing lead on Applications Transformation at HP. Hello, Paul.

Paul Evans: Hello, Dana. Thank you.

Gardner: We talked in the [earlier podcast](#) in our series, a case study, about transformation and why that's important through the example of [a very large education organization in Italy](#) and what they found. We looked at how this can work very strategically and with great economic benefit, but I think now we are trying to get into a bit more of the how.

Tell us a little bit, Paul, about what the stakes are. Why is it so important to do this now?

Evans: In a way, this podcast is about two types of IT assets. You talked before about core and context. That whole approach to classifying business processes and their associated applications was invented by [Geoffrey Moore](#), who wrote [Crossing the Chasm](#), [Inside the Tornado](#), etc.



He came up with this notion of core and context applications. Core being those that provide the true innovation and differentiation for an organization. Those are the ones that keep your customers. Those are the ones that improve the service levels.

Those are the ones that generate your money. They are really important, which is why they're called "core."

Lower cost

The "context" applications were not less important, but they are more for productivity. You should be looking to understand how that could be done in terms of lower cost provisioning. When these applications were invented to provide the core capabilities, it was 5, 10, 15, or 20 years ago. What we have to understand is that what was core 10 years ago may not be core anymore. There are ways of effectively doing it at a much different price point.

As [Moore points out](#), organizations should be looking to build "core," because that is the unique intellectual property of the organization, and to then buy "context." They need to understand, how do I get the lowest-cost provision of something that doesn't make a huge difference to my product or service, but I need it anyway.

An [human resources system](#) may not be something that you are going to build your business model on, but [you need one](#). You need to be able to service your employees and all the things they need. But, you need to do that at the lowest-cost provision. As time has gone on, this demarcation between core and context has gotten really confused.

As you said, we're [putting together a series of events](#), and Moore will be the keynote speaker on these events. So, we will elucidate more around core and context.

The other speaker at the event is also an inventor, this time from inside HP, [Steve Woods](#). Steve has taken this notion of core and context and has teamed it with some extremely exciting technology and very innovative thinking to develop some unique tools that we use inside the services from HP, which allow us then really to dive into this. That's going to be one of the sessions that we're also going to be delivering on this series of events.

Gardner: Okay, Steve Woods, we can use a lot of different terms here, "core and context," "wheat and chaff." I thought another metaphor would be "baby and bathwater." What happens is that it's difficult to separate the good from the potentially wasteful in the legacy inventory.

I think this has caused people to resist modernizing. They have resisted tinkering with legacy installations in the past. Why are they willing to do it now? Why the heightened interest at this time?

Woods: A good deal of it has to do with [the pain that they're going through](#). We have had customers who had assessments with us before, as much as a year ago, and now they're coming back and saying they want to get started and actually do something. So, a good deal of the interest is caused by the need to drive down costs.



Also, there's the realization that a lot of these tools -- [extract, transform, and load \(ETL\)](#) tools, [enterprise application integration \(EAI\)](#) tools, reporting, and [business process management \(BPM\)](#) -- are proving themselves now. We can't say that there is a risk in going to these tools. They realize that the strength of these tools is that they bring a lot of agility, solve skill sets issues, and make you much more responsive to the business needs of the organization.

Gardner: This definition of core, as Paul said, is changing over time and also varies greatly from organization to organization. Is there no one size fits all approach to this?

Context not code

Woods: I don't think there really is a one size fits all, but as we use our tools to analyze code, we find sometimes as much as 65 percent or more of an application could really not be core. It could just be context.

As we make these discoveries, we find that in the organization there are political battles to be fought. When you identify these elements that are not core and that could be moved out of handwritten code, you're transferring power from the developers -- say, of [COBOL](#) -- to the users of the more modern tools, like the BPM tools.



So there is always an issue. What we try to do, when we present our findings, is to be very objective. You can't argue that we found that 65 percent of the application is not doing core. You can then focus the conversation on something more productive. What do we do with this? The worst thing you could possibly do is take a million lines of COBOL that's generating reports and rewrite that in

[Java](#) or [C#](#) hard-written code.

We take the concept of core versus context not just to a possible off-the-shelf application, but at architectural component level. In many cases, we find that this is helpful for them to identify legacy code that could be moved very incrementally to these new architectures.

Gardner: What's been the holdup? What's difficult? You did mention politics, and we will get into that later, but what's been the roadblock from perspective of these tools? Why has that been decreasing in terms of the ability to automate and manage these large projects?

Woods: A typical COBOL application -- this is true of all legacy code, but particularly [mainframe](#) legacy code -- can be as much as 5, 10, or 15 million lines of code. I think the sheer idea of the size of the application is an impediment. There is some sort of inertia there. An object at rest tends to stay at rest, and it's been at rest for years, sometimes 30 years.

So, the biggest impediment is the belief that it's just too big and complex to move and it's even too big and complex to understand. Our approach is a very lightweight process, where we go in and answer to a lot of questions, remove a lot of uncertainty, and give them some [very powerful visualizations and understanding of the source code](#) and what their options are.

Gardner: So, as we've progressed in terms of the tools, the automation, and the ability to handle large sets of code, the inertia also involves the nontechnical aspects. What do we mean by politics? Are there fiefdoms? Are there territories? Is this strictly a traditional kind of human nature thing? Perhaps you could help us understanding that a bit better.

Doing things efficiently

Woods: Organizations that we go in have not been living in a vacuum, so many of have been doing greenfield development; when they start out to say they need a system that does primarily reporting, or a system that does primarily data integration. In most organizations those fiefdoms, if you will, have grown pretty robust, and they will continue to grow. The realization is that they actually can do those things quite efficiently.

When you go to the legacy side of the house, you start finding that 65 percent of this application is just doing ETL. It's just parsing files and putting them into databases. Why don't you replace

that with a tool? The big resistance there is that, if we replace it with a tool, then the people who are maintaining the application right now are either going to have to learn that tool or they're not going to have a job.

So, there's a lot of resistance in the sense that we don't want to lose anymore ground to the target architecture fiefdom, so we are going to not identify this application as having so many elements of context functionality. Our process, in a very objective way, just says that these are the percentages that we're finding. We'll show you the code, you can agree or disagree that's what it is doing, and then let's make decisions based upon those facts.

If we get the facts on the table, particularly visually, then we find that we get a lot of consensus. It may be partial consensus, but it's consensus nonetheless, and we open up the possibilities and different options, rather than just continuing to move through with hand-written code.

Gardner: Paul, you've mentioned in the past that we've moved from the [nice-to-have to the must-have](#), when it comes to legacy applications transformation and modernization. The economy has changed things in many respects, of course, but it seems as if the lean IT goal is no longer something that's a vision. It's really moved up the pecking order or the hierarchy of priorities.

Is this perhaps something that's going to impact this political logjam? Are other organizations and business and financial outcome folks, who are just going to steamroll these political issues?

Evans: Well, I totally think so, and it's happening already. If you look at this whole core-context thing, at the moment, organizations are still in survival mode. Money is still tight in terms of consumer spending. Money is still tight in terms of company spending. Therefore, you're in this position where keeping your customers or trying to get new customers is absolutely fundamental for staying alive. And, you do that by improving service levels, improving your services, and improving your product.

If you stay still and say, "Well, we'll just glide for the next 6 to 12 months and keep our fingers crossed," you're going to be in deep trouble. A lot of people are trying to understand how to use the newer technologies, whether it's things like [Web 2.0](#) or [social networking](#) tools, to maintain that customer outreach.

Those of us who went to the business school, marketing school remember -- it takes \$10 to get a customer into your store, but it only takes \$1 to keep them coming back. People are now worrying about those dollars. How much do we have to spend to keep our customer base?

Therefore, the line-of-business people are now pushing on technology and saying, "You can't back off. You can't not give us what we want. We have to have this ability to innovate and differentiate, because that way we will keep our customers and we will keep this organization alive."

Public and private sectors

That applies equally to the public and private sectors. The public sector organizations have this mandate of improving service, whether it's in healthcare, insurance, tax, or whatever. So all of these commitments are being made and people have to deliver on them, albeit that the money, the IT budget behind it, is shrinking or has shrunk.

So, the challenge here is, "Last year I ran my IT department on my theoretical \$100. I spent \$80 on keeping things going, and \$20 on improving things." That was never enough for the line-of-business manager. They will say, "I want to make a change. I want it now, or I want it next week. I don't want it in six months time. So explain to me how you are going to do that."

That was tough a year ago, but the problem now is that your \$100 IT budget is now \$80. Now, it's a bit of a challenge, because now all the money you have got you are going to spend on keeping the old stuff alive. I don't think the line-of-business managers, or whoever they are, are going to sit back and say, "That's okay. That's okay. We don't mind." They're going to come and say that they expect you to innovate more.

This goes back to what Steve was talking about, what we talked about, and what Moore will raise in the event, which is to understand what drives your company. Understand the values, the differentiation, and the innovations that you want and put your money on those and then find a way of dramatically reducing the amount of money you spend on the contextual stuff, which is pure productivity.

Steve's tools are probably the best thing out there today that highlight to an organization, "You don't need this in handwritten code. You could put this to a low cost package, running on a low cost environment, as opposed to running it in COBOL on a mainframe." That's how people save money and that's how we've seen people get, as we have talked earlier, a [return on investment \(ROI\)](#) of 18 months or less.

So it is possible, it can be done, and it's definitely not as difficult as people think. The point of the tools is that they allow us to see the code. They allow us to understand what's good and bad and to make very clear, rational, and logical decision.

Gardner: Steve Woods, we spoke earlier about how the core assets are going to be variable from organization to organization, but are there some common themes with the contextual services? We certainly see a lot of very low-cost alternatives now creeping up through [software as a service \(SaaS\)](#), cloud-based, outsourced, mix-sourced, co-located, and lots of different options. Is there some common theme now among what is not core that organizations need to consider?

Woods: Absolutely. One of the things that we do find, when we're brought in to look at legacy applications, is that by virtue of the fact that they are still around, the applications have resisted all the waves of innovation that have preceded. Sometimes, they tend to be of a very definite nature.

A number of them tend to be big data hubs. One of the first things we ask is for the architectural topology diagram, if they have it, or we just draw it on a whiteboard,, they tend to be big spiders. There tends to be a central hub database and you see that they start drawing all these different lines to other different systems within the organization.

The things that have been left behind -- this is the good news -- tend to be the very things that are very amenable for moving to modern architecture in a very incremental way. It's not unusual to find 50-65 percent of an application is just doing ETL functionality.

A good thing

The real benefit to that -- and this is particularly true in a tough economy -- is that if I can identify 65 percent of the application that's just doing data integration, and I create or I have already established the data integration center of excellence within the organization, already have those technologies, or implement those technologies, then I can incrementally start moving that functionality over to the new architecture. When I say incrementally, that's a good thing, because that's beneficial in two ways.

It reduces my risk, because I am doing it a step at a time. It also produces a much better ROI, because the return on the incremental improvement is going to be trickling over time, rather than waiting for 18 months or two years for some big bang type of improvement. Identifying this context code can give you a lot of incremental ROI opportunities, and make you a much more solid IT investment decision picture.

Gardner: So, one of these innovations that's taken place for the past several years is the move towards more distributed data, hosting that data on lower-cost storage architectures, and virtualizing behind the database or the storage itself. That can reduce cost dramatically.

Woods: Absolutely. One of the things that we feel is that decentralizing the architecture improves your efficiency and your redundancy. There is much more opportunity for building a solid, maintainable architecture than there would be if you kept a sort of monolithic approach that's typical on the mainframe.

Gardner: Once we've done this exercise, variable as it may be from organization to organization, separating the core from the non-core, what comes next? What's the next step that typically happens as this transformation and modernization of legacy assets unfolds?

Woods: That's a very good question. It's really important to understand this leap in logic here. If I accept the notion that a majority of the code in a legacy application can be moved to these model driven architectures, such as BPM and ETL tools, the next premise is, "If I go out and buy these tools, a lot of functionality is provided with these tools right out of the box. It's going to give me my monitoring code, my management code, and in many cases, even some of the testing capabilities are sort of baked into the product."

If that's true, then the next leap of logic is that in my 1.5 million lines of COBOL or my five million lines of COBOL there is a lot of code that's irrelevant, because it's performing management, monitoring, logging, tracing, and testing. If that's true, I need to know where it's at.

The way you find where it's at is identifying the duplicate source code, what we call clone code. Because when you find the clone code, in most cases, it's a superset of that code that's no longer relevant, if you are making this transformation from handwritten code to a model-driven architecture.

What I created at HP is a tool, an algorithm, that can go into any language legacy code and find the duplicate code, and not only find it, but visualize it in very compelling ways. That helps us drill down to identify what I call the unintended design. When we find these unintended designs, they lead us to ask very critical questions that are paramount to understanding how to design the transformation strategy.

So, if you accept the premise of moving context code to componentized architecture, then the next thing you should be looking for is where is the clone code and how is it arranged?

Gardner: Do we have any examples of how this has worked in practice? Are there use cases or an actual organization that you are familiar with? What have been some of the results of going through this process? How long did it take? What did they save? What were the business outcomes?

Viewing the application

Woods: We've often worked with financial services companies and insurance companies, and we have just recently worked with one that gave us an application that was around 1.2 or 1.5 million lines of code. They said, "Here is our application," and they gave us the source code. When we looked into the source code, we found that there were actually four applications, if you looked at just the way the code was structured, which was good news, because it gives us a way of breaking down the functionality.

In this one organization, we found that a high percentage of that code was really just taking files, as I said before, unbundling those files, parsing them, and putting them into databases. So they have kind of let that be the tip of the spear. They said, "That's our start point," because they're often asking themselves where to start.

When you take handwritten code and move it to an ETL tool, there's ample industry evidence that a typical ROI over the course of four years can be between 150 percent and 450 percent improvement in efficiencies. That's just the magic of taking all this difficult-to-maintain spaghetti code and moving it to a very visually oriented tool that gives you much more agility and allows you to respond to changes in the business and the business' needs much more quickly and with skill sets that are readily available.

Gardner: You know, Paul, I've heard a little different story from some of the actual suppliers of legacy systems. A lot of times they say that the last thing you want to do is start monkeying around with the code. What you really want to do is pull it off of an old piece of hardware and put it on a new piece of hardware, perhaps with a virtualization layer involved as well. Why is that not the right way to go?

Evans: Now you've put me in an interesting position. I suppose our view is that there are different strategies. We don't profess one strategy to help people transform or modernize their apps. The first thing they have to do is understand them, and that's what Steve's tools do.

It is possible to take an approach that says that all we need to do is provide more horsepower. Somebody comes along and says, "Hey, transaction rates are dropping. Users are getting upset because an [ATM](#) transaction is taking a minute, when it should take 15 seconds. Surely all we need to do is just give the thing more horsepower and the problem goes away."

I would say the problem goes away -- for 12 months, maybe, or if you're lucky 18 -- but you haven't actually fixed the problem. You've just treated the symptoms.

At HP, we're not wedded to one style of computer architecture as the hub of what we do. We look at the customer requirement. Do we have systems that are equal in performance, if not greater, than a mainframe? Yeah, you bet we do. Our [Superdome](#) systems are like that. Are they the same price? No, they are considerably less. Do we have [blades](#), PCs, and normal distributed service? Yeah.

The point is that we don't have a preconceived view of what this thing should run on. That's one thing. We're not wedded to one architectural style. We look at the customer's requirements and then we understand what's necessary in terms of the throughput TP rates or whatever it may be.

So, there is obviously an approach that people can say, "Don't jig around." It's very easy to inject fear into this and just say to put more power underneath it, don't touch the code, and life will be wonderful. We're totally against that approach, but it doesn't mean that one of our strategies is not re-hosting. There are organizations whose applications would benefit from that.

We still believe that can be done on relatively inexpensive hardware. We can re-host an application by keeping the business logic the same, keeping the language the same, but moving it from an expensive system to a less expensive system.

Freeing up cash

People use that strategy to free up cash very quickly. It's one of the fastest ROIs we have, and they are beginning to save instantly. They make the decision that says, "We need to put that money back in the bank, because we need to do that to keep our shareholders happy." Or, they can reinvest that into their next modernization project, and then they're on an upward spiral.

There are approaches to everything, which is why we have seven different strategies for modernization to suit the customer's requirement, but I think the view of just putting more horsepower underneath, closing your eyes, and hoping is not the way forward.

Gardner: Steve, do you have anything more to add to that, treating the symptom rather than the real issues?

Woods: As Paul said, if you treat this as a symptom, we refer to that as a short-term strategy, just to save money to reinvest into the business.

The only thing I would really add to that is that the problem is sometimes not nearly as big as it seems. If you look at the analogy of the clone codes that we find, and all the different areas that we can look at the code and say that it may not be as relevant to a transformation process as you think it is.

I do this presentation called "Honey I Shrunk the Mainframe." If you start looking at these different aspects between the clone code and what I call the asymmetrical transformation from handwritten code to model driven architecture, you start looking at these different things. You start really seeing it.

We see this, when we go in to do the workshops. The subject matter experts and the stakeholders very slowly start to understand that this is actually possible. It's not as big as we thought. There are ways to transform it that we didn't realize, and we can do this incrementally. We don't have to do it all at once.

Once we start having those conversations, those who might have been arguing for a re-host suddenly realize that rearchitecting is not as difficult as they think, particularly if you do it asymmetrically. Maybe they should reconsider the re-host and consider going to those context-core concept and start moving the context to these well-proven platforms, such as the ETL tools, the reporting tools, and [service-oriented architecture \(SOA\)](#).

Gardner: Steve, tell us a little bit about how other folks can learn more about this, and then give us a sneak peek or preview into what you are going to be discussing at the upcoming virtual event.

Woods: That's one of the things that we have been talking about -- our tools called the [Visual Intelligence Tools](#). It's a shame you can't see me, because I'm gesturing with my hands as I talk, and if I had the visuals in front of me, I would be pointing to them. This is something to really appreciate -- the images that we give to our customers when we do the analysis. You really have to see it with your own eyes.

We are going to be doing [a virtual event on November 3, 4, and 5](#), and during this you will hear some of the same things I've been talking about, but you will hear them as I'm actually using the tools and showing you what's going to happen with those tools, what those images look like, and why they are meaningful to designing a transformation strategy.

Gardner: Very good. We've been learning more about Application Transformation: Getting to the Bottom Line, and we have been able to separate core from context, and appreciate better how that's an intriguing strategy for approaching this legacy modernization problem and begin to enjoy much greater economic and business benefits as a result.

Helping us weave through this has been Steve Woods, distinguished software engineer at HP. Thanks for your input, Steve.

Woods: Thank you.

Gardner: We've also been joined by Paul Evans, worldwide marketing lead on Applications Transformation at HP. Paul, you are becoming a regular on our show.

Evans: Oh, I'm sorry. I hope I am not getting too repetitive.

Gardner: Not at all. Thanks again for your input.

This is Dana Gardner, principal analyst at Interarbor Solutions. You've been listening to a sponsored BriefingsDirect podcast. Thanks for listening, and come back next time.

[Listen](#) to the [podcast](#). Find it on [iTunes/iPod](#) and [Podcast.com](#). Download the transcript. [Learn](#) more. Sponsor: [Hewlett-Packard](#).

Gain more insights into "Application Transformation: Getting to the Bottom Line" via a series of HP virtual conferences Nov. 3-5. For more on Application Transformation, and to get real time answers to your questions, register to the virtual conferences for your region:

[Register here](#) to attend the Asia Pacific event on Nov. 3.

[Register here](#) to attend the EMEA event on Nov. 4.

[Register here](#) to attend the Americas event on Nov. 5.

Transcript of the second in a series of sponsored BriefingsDirect podcasts on the rationale and strategies for application transformation. Copyright Interarbor Solutions, LLC, 2005-2009. All rights reserved.