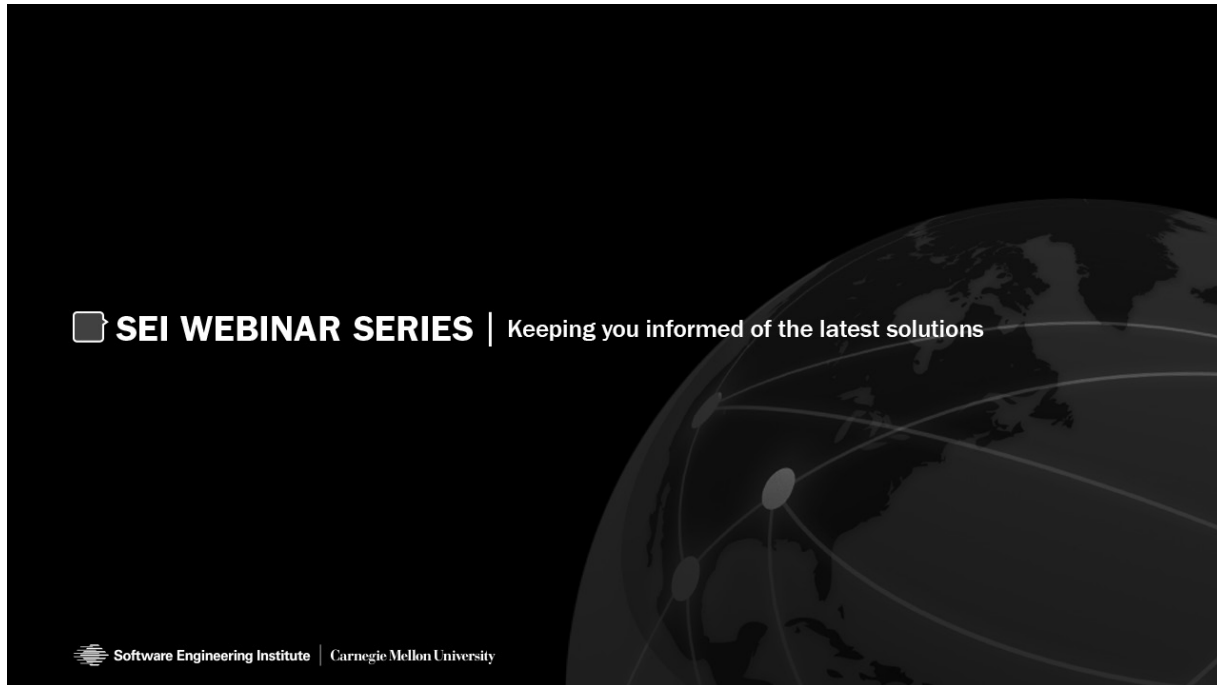


Leading a Successful Large IT Modernization Project

Table of Contents

SEI WEBINAR SERIES	2
Carnegie Mellon University.....	2
Copyright 2017 Carnegie Mellon University.....	3
5 Things You Need to Know for Leading a Successful Large IT Modernization Project	3
Polling Question 2	5
5 Things You Need to Know for Leading a Successful Large IT Modernization Project	6
Polling Question 2	8
Polling Question 3	9
Data Collection.....	11
Know where you want to be.....	12
Know where you are	16
Know what you need	20
Know how to move forward	28
Iteratively move forward	33
Periodically Re-evaluate Roadmap	39
Periodically Re-evaluate Roadmap	41
Periodically Re-evaluate Roadmap	42
Continuously Update the Modernization Artifacts	43
SEI WEBINAR SERIES	52

SEI WEBINAR SERIES



Carnegie Mellon University

Carnegie Mellon University

This video and all related information and materials (“materials”) are owned by Carnegie Mellon University. These materials are provided on an “as-is” “as available” basis without any warranties and solely for your personal viewing and use.

You agree that Carnegie Mellon is not liable with respect to any materials received by you as a result of viewing the video, or using referenced websites, and/or for any consequences or the use by you of such materials.

By viewing, downloading, and/or using this video and related materials, you agree that you have read and agree to our terms of use (www.sei.cmu.edu/legal/).

Distribution Statement A: Approved for Public Release; Distribution is Unlimited

© 2017 Carnegie Mellon University.

Copyright 2017 Carnegie Mellon University

Copyright 2017 Carnegie Mellon University

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[Distribution Statement A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

DM-0004538



Software Engineering Institute | Carnegie Mellon University

5 Things you need to know
March 13, 2017
© 2017 Carnegie Mellon University

[Distribution Statement A: Approved for Public Release; Distribution is Unlimited]

3

5 Things You Need to Know for Leading a Successful Large IT Modernization Project

5 Things You Need to Know for Leading a Successful Large IT Modernization Project

Stephany Bellomo

Felix Bachmann

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213



Software Engineering Institute | Carnegie Mellon University

5 Things you need to know
© 2017 Carnegie Mellon University

[Distribution Statement A: Approved for Public Release; Distribution is Unlimited]

**004 Presenter: And hello from the campus of Carnegie Mellon

University in Pittsburgh, Pennsylvania. We welcome you to the Software Engineering Institute's webinar series. Our presentation today is Five Things You Need to Know for Leading a Successful Large IT Modernization Project. Depending on your location, we wish you a good morning, a good afternoon, or good evening.

My name is Shane McGraw. I'll be your moderator for the presentation, and I'd like to thank you for attending. We want to make today as interactive as possible, so we will address questions throughout the presentation, and again at the end of the presentation. You can submit those questions to our event staff at any time through the Ask a Question tab on your webinar control panel, or the Chat tab, also on your control panel. We will also ask a few polling questions throughout today's evening, and they will appear as a popup window on your screen. In fact, the first polling question we'd like to ask is: How did you hear of today's event?

Polling Question 2

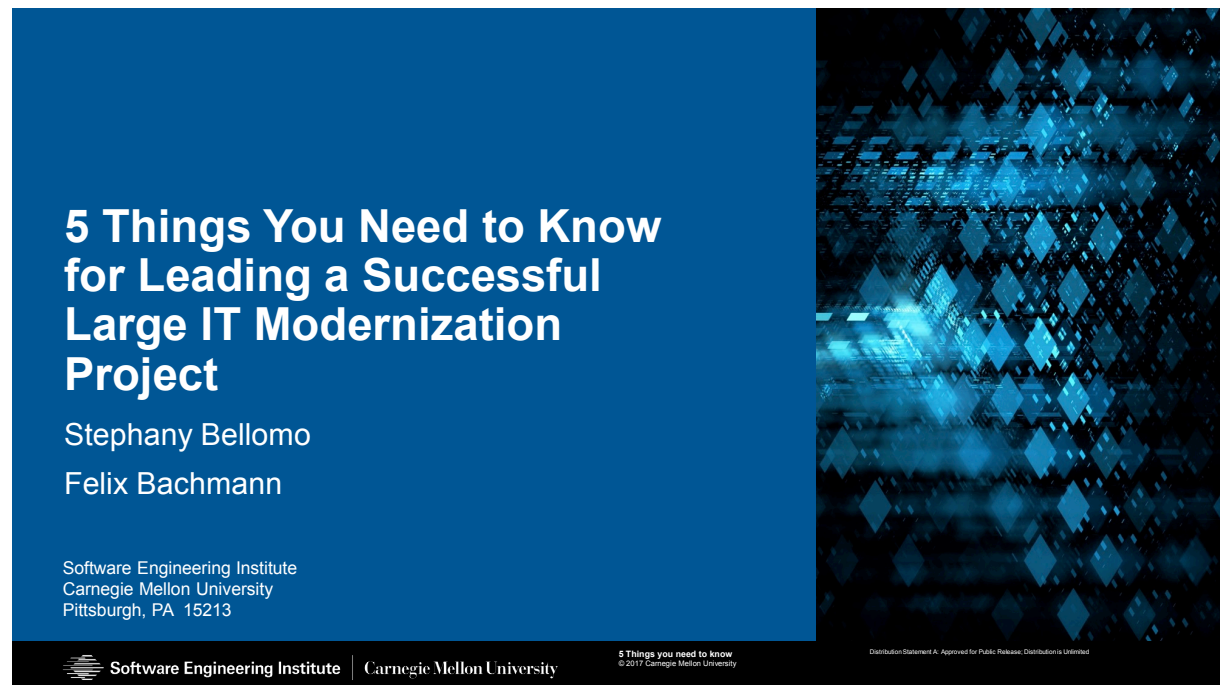
Polling Question 2

Have you ever been involved in a modernization effort?

- a) Yes
- b) No

**005 Another three tabs I'd like to point out are the Download Materials, Twitter, and Survey tabs.

5 Things You Need to Know for Leading a Successful Large IT Modernization Project



**5 Things You Need to Know
for Leading a Successful
Large IT Modernization
Project**

Stephany Bellomo
Felix Bachmann

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Software Engineering Institute | Carnegie Mellon University

5 Things you need to know
© 2017 Carnegie Mellon University

Distribution Statement A: Approved for Public Release; Distribution is Unlimited

**004 The Download Materials tab has a PDF copy of the presentation slides there now, along with other related work and resources from the SEI. For those of you using Twitter, be sure to follow @seinews, and use the hashtag #seiwebinar. And now I'd like to introduce our presenters for today.

First, our facilitator. Mr. Will Hayes is a principal engineer at the SEI and Will provides direct lifecycle management support to major software-intensive programs in government and military organizations. He also does research and consultation, and the application of agile methods in highly regulated settings.

Next, Stephany Bellomo is a senior member of the technical staff and

she teaches courses in software architecture and service-oriented architecture. Her current research interests include technical debt and architecting for dev-ops. She's an IEEE senior member and serving as a guest editor of the IEEE Software magazine in 2015 and 2017.

And lastly we have Mr. Felix Bachmann. Felix is also a senior member of our technical staff, where he's a member of the architecture practices group. He's a coauthor of the Attribute-Driven Design method, a contributor and instructor for the ATM Evaluator training, and a coauthor of the "Documenting Software Architectures: Views and Beyond". Will, Felix, Stephany, welcome. Will, all yours.

Presenter: Thank you, Shane, very much. As we pose the first polling question about our audience--

Polling Question 2

Polling Question 2

Have you ever been involved in a modernization effort?

- a) Yes
- b) No

**005 Let me ask Stephany and Felix to just chime in with: What do you mean by large-scale IT modernization?

Presenter: Well, I take that one. Okay, so when it comes to the modernization, what the name of course says is you want to kind of get something new, shiny, great, wonderful, because you are unhappy with what you have. But to make it clear what we really are looking for are more those projects we have of IT infrastructure. Typically you have 20 years old, 15 years old, even older applications running there which don't fulfill whatever you need to do for your business or for your mission, and you are kind of thinking in terms of you want to replace them or do something with them. So we are typically talking about many systems, usually big systems, and we are

talking about there's probably some timeframe where you have to live with the old and the new system in parallel.

Presenter: So I think our polling question was to how many have been involved--

Presenter: So we have 83 percent with Yes, 17 percent No, and we'll launch the third one now, just asking about the successful of that effort.

Presenter: And so the next one is about success.

Polling Question 3

Polling Question 3

On a scale of 1 to 5 can you rate the success of that effort? (5 being most successful)

- a) 1
- b) 2
- c) 3
- d) 4
- e) 5

**006 I know you've both been involved in a lot of projects that fall under this description. Could you talk about some of the challenges, successes, as people think about their experience? Could you reflect

on the good, the bad, and maybe some of the ugly?

Presenter: So I think some of the challenges that we run into, and one of the themes that we're going to talk a little bit about that are common across many projects, is that there are technical and nontechnical aspects. And so I think what we'd like to do today is elaborate a little bit more on both sides of that coin. Felix, do you have some other thoughts on--?

Presenter: Yeah, I think that is probably our most important observation that we had with those kind of projects. Since we are not really talking about the minute change somewhere in the technology, we talk about bigger change. So there is, without the right organization support, many of those modernization projects are actually getting into trouble, and that's the topic we would like to elaborate a little bit here for the next hour.

Presenter: So let's get a sense from the audience of what kind of success--

Presenter: Right, so we had 39 percent rated it a three; 32 percent a four; 13 percent five; 13 percent two; and 3 percent at one.

Presenter: So we're favoring more successful-- middling to more successful.

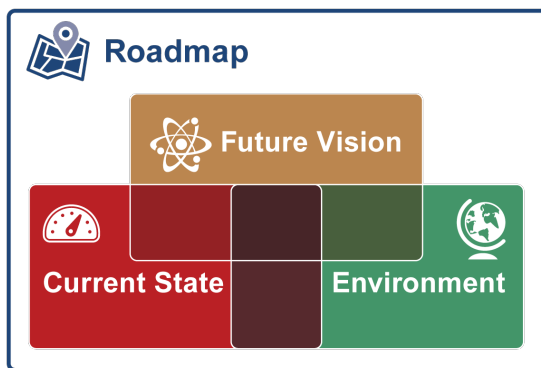
Presenter: Sounds like it. Okay, good.

Presenter: Good.

Presenter: So let's start talking about the five different things that lead to success.

Data Collection

Data Collection



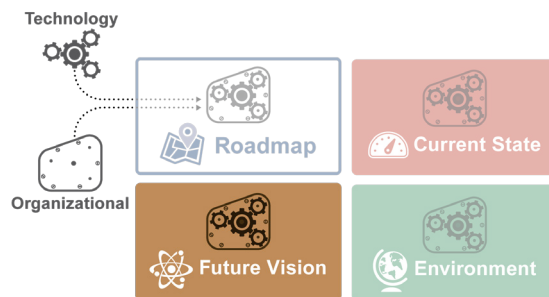
- Know where you want to be
 - “Realistically”
- Know where you are
 - “Honestly”
- Know what you need
 - “Affordably”
- Know how to move forward
 - “Cautiously”

**007 Presenter: Yeah, well, okay. Let me start it in the following way. Of course probably everyone here in the audience knows that if you want to change anything-- so modernization is some kind of change-- at the very minimum what you need to do is you need to have some kind of pretty clear picture of where you want to be. You need to have some idea of where you are today, and then of course you would have to go forward and move from today wherever you want to be. So those are the obvious ones in there.

There are a few things that come in addition that we usually see they're overlooked, or maybe not overlooked but sort of treated as not a second-class citizen. So that's kind of where we want to dive a little bit more into the details, by-- let it go to topic by topic.

Know where you want to be

Know where you want to be



It is easy to list all the great cool things you want to use in the future

- Introducing new technologies. Such as service oriented or perhaps micro services
- Utilizing new hardware, such a multi core
- Ability to support fast and agile feature development

What factors hindered you in the past to always get what you needed?

- New technologies come have a learning curve
- Is there an agreement on the requirements
- Will modernized system require the business processes to change

**008 Presenter: So in the area of knowing where you want to be, what are the important considerations there?

Presenter: Yeah, again, from the technical perspective, it is so easy to say, "Oh yeah, I want to have this great, wonderful, new shiny thing, that whatever comes up, because it really looks so good and it really sounds like whatever problems I have, those problems will be solved by just doing that." So there are a

lot of things that actually can be done, but instead of just saying, "Oh yeah, I want to have that shiny thing," so let me give you an example. So, cloud for example.

So I hear, "Cloud, oh yeah, we can save a lot of money. We don't have that much responsibility anymore. We're just putting it in the cloud and everything is just fine." Yeah? Yeah. Now, fact is, that is not that easy. There's a lot of work to do, and you are not really sure that whenever you go to the cloud, for example, does it really fix the problem that you encounter today?

Presenter: So it's not just about the shiny thing, it's the context in which it will be used as well.

Presenter: Right.

Presenter: Stephany, can you elaborate?

Presenter: Yeah, there are some other areas too to consider. So everybody does tend to focus a lot on the future vision and the shiny-shiny, we call it-- the cloud and the micro-services, etcetera. But there's also factors to consider for the future vision that are needed to enable that, and some of them are things like you need to have the right skill sets, and you also need to have some people things, like agreement on the requirements-- everybody has to agree on the future vision, and a lot of times that's not necessarily the case; and another big aspect is just

needing to be ready to make sometimes some pretty significant business process changes. So, for example, a lot of people like to use-- move to something like an ERP, an Enterprise Planning Resource tool, or SAP kind of things, but at the end of the day those are great products for the right use, but if your business processes don't align with those things, it can be a big change for you and the organization has to be ready for them.

Presenter: So it's really a sociotechnical challenge--

Presenter: As well.

Presenter: And the engineering solution is just a piece of the larger puzzle.

Presenter: Mm-hmm. Mm-hmm.

Presenter: I think yes. That's exactly it. I mean, the warning we would like to put out here when it comes to vision is just: Get real. So think about things within your organization-- so no matter how the organization is-- you probably know your organization the best. Think what you can actually really achieve. Trying to put changes in there that look great and wonderful, but if no one in your organization can actually deal with that, if they are not willing to change the process, for example, then you will fail. You get that shiny thing, it won't work, you end up with two systems now that you have to--

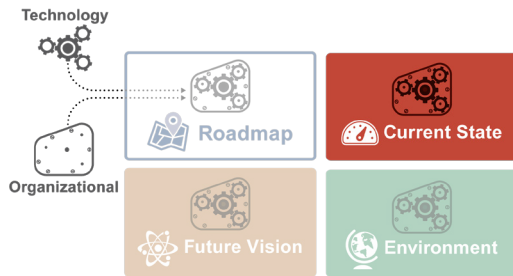
Presenter: Many people would think of the technology as a support to the workflow, and this sociotechnical entanglement, if you will, those folks might say, "First define the workflow, and then engineer the tool to meet the needs." Has that been your experience, that that approach works?

Presenter: Well, I would say actually both approaches work, as long as we keep in mind that we have to change both sides of the equation to make it happen. If that is clear, then it doesn't matter from which side you start. If you have some kind of new technology which requires you to change your organization, your approach, and you are willing to do the change, then of course yes, it does work. But, especially with the Organization you typically are working with, it makes more sense to look at the processes themselves first, and then start thinking about how can we actually map that into the technical future of the organization.

Presenter: So maybe part of that is looking at the current state then.

Know where you are

Know where you are



Conway's Law

Organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations

Conway, Melvin E. (April 1968), "How do Committees Invent?", *Datamation*, 14 (5): 28–31, retrieved 2015-04-10

Determine which of the implemented solutions caused you to do the modernization.

- Old design principles, such as let the database run everything?
- Plethora of outdated technologies?
- Dependencies that inhibit scaling?

What went wrong in your organization that led to these issues?

- Skill set of your employees?
- Competition between "siloed organizations"?
- Prioritized funding decisions favoring new features?

**009 Looking at what-- you have to understand where you are today and where you're headed in the future.

Presenter: Right. And there, when we're sitting down-- and we're talking always the same thing-- you have a technology and you have to think--

Look at your organization, and many of you in the audience, you probably heard about the Conway's Law, which-- right now we just looked at the date saying it's now 50 years that that law was first stated, by Conway, which says that basically organization and the system that support those organization align, and to make it even stronger, they have to align. If they don't align, your system will not work. So that means, for the second step, looking at the current state, you need to take

into account that you are so far, the whole time, under that Conway's Law, and you ended up with whatever you have because of it. So of course now again from the technical perspective, you would look into your existing system and trying to identify, "What is it that I don't like anymore? What is it in the system that makes my life difficult?"

Again, giving an example here, many of the older systems I actually implement within the database. So, stored procedures in the database. Now, in the past, when we started doing this, 20 years, 25 years ago, at that point we were pretty much resource-constrained. So to make big system perform well and fast enough, it was basically our only chance for achieving this-- put everything into the database. It makes it fast. Great. A side-effect of that, of course, is we created a maintenance problem. That is the problem that hurts us now today. So one possibility would be you look at the system and say, "Okay, we learned that having everything a stored procedure in the database is not a good idea. We cannot utilize modern technologies if we keep doing this," so you want to remove this. Great.

But, again, now back to the Conway's Law. Think about that over the past 20, 25 years, you and your organization created people who know exactly how to deal with those. Those are experts, and they love their job. So if you say, "We don't

want that anymore," what do you think those people will do? They will be not very happy. So at that point, from the relationship, look at your organization. What is it in your organization that led you to the point where you are? And if there is something in the technology that you would like to change, think about it instead-- impact that your organization has, whatever got you there. Can you actually remove this and change this? If yes, great. If you think it is impossible, well then, why would you think that the change that you now propose, what you don't like, will work in the future?

Presenter: So Conway's Law would alert us to be mindful of the potential disruption of changing the communication channels supported by IT because the communication channels that exist in the organization have gotten used to them. Does that mean that we have to keep this mirror between the IT system and the organizational structure? Can we hack Conway's Law?

Presenter: Yeah, let me answer that. So I think one of the issues we run into when we look at multiple systems, multiple IT systems that need to interconnect, is we find out that there are a lot of stovepipes, and that we find out that the different programs or projects have not been speaking to each other in a long time, and when we dig underneath we actually see that the systems reflect that, and therefore

they won't necessarily be easily interoperable. They might not have APIs, they might not have any kind of capability to share data between them. So it's a different result of, to some extent, the way the organization is structured, and the other byproduct of that that can be seen in the current state is you might have redundant capabilities. So because you have all of these different stovepipes everywhere, everybody builds a full stack everywhere that they need a system, and therefore you have duplicative capability, maybe duplicative features, but also cross-cutting functionality like access control, event handling-- things that can be handled more economically across multiple systems or handled all the way through each stack. So I think that's also one thing, where you can see that the organization, if they're not willing to change organizationally, it's very hard to think about how do we restructure the systems to go against that kind of pattern.

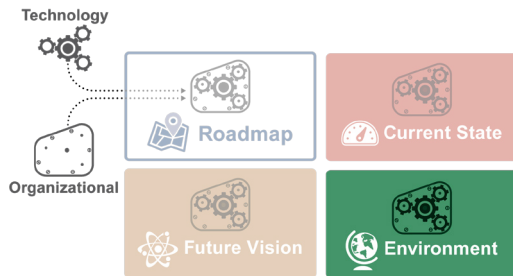
Presenter: So it sounds like there may be local optimizations that have entrenched over time, and the systems and the organizational structure kind of collude to keep that suboptimization system level for optimization department level, and you're really talking about changing both simultaneously, aren't you?

Presenter: Yes. Yep. In a lot of cases.

Presenter: So what more in the roadmap?

Know what you need

Know what you need



For the endeavor to be successful you need tool support and adjusted processes.

- Automated regression testing?
- Appropriate documentation / models?
- Synchronization of old data model with new data model

If you do not have that support now, what were the reasons?

- Right skill set of the people doing the modernization
- Necessary training in place
- Right contractors selection?
- Enforcement of essential principles?

**010 Presenter: Yeah, so the next point there-- actually you would like to spend a little bit of time here, because that's one of those topics that is very quickly overlooked. So after you kind of know where you want to be and you figure out what is it actually in the current system that you would want to change to get there, then we all feel good about it and say, "Okay, let's do it." So, but what we also see there is something beside of this, before we can even start, something that you need to set up. You have to create environment for that period of time, which may be, depending on what kind of systems you have, between somewhere three years, maybe five years, when you transition from wherever you are to wherever you want to be, to make actually that transition happen.

So, simple example. What will happen over time is you will take, function by function-- so business function by business function-- and move them over from the old existing system into some newly structured system in there. While you are doing this, you want to make sure that you don't break anything, because during that period of time old and new has to work together, and make sure that the users that you have can still use the old function for everything else that's not moved yet, and can use the new function and whatever they do with the new function will not impact the old one.

So what is on the table is regression testing-- automated regression testing. I know it is an old topic. We know that, of course, yes, we should have automatic regression testing. Unfortunately, almost every time when we come to an organization, ask that question, "Do you already have automated regression testing in place?", the answer is, "Well, not automated. We have our users doing it." Well, if you don't have that in place, then you will suffer a lot during that transition period, because every time you do something new, something will break and your users will be very upset. So one thing in the environmental infrastructure that needs to be there is to make sure you have your automated regression testing. If you don't, set it up first, and then use it continuously, just as an example.

Documentation is another example. Many issues that we encounter, you may encounter, is that the problems that you have is because you don't really have a proper documentation for it. Just over time, over the past 20 years or so, you just add to it, and no one actually knows where stuff is and how stuff works. Maybe one of the decisions that you had was, "We need to fix it." We don't really want to have all the documentation-- by all means, no-- but the right documentation. But if you are not used to it, then you need to set up a definition of what is it that you need to be done; you have to have a tool environment where you can do that very easily, and such. So there are environmental tools that you need to put in place to make the whole transition happening.

Presenter: And there's also-- I'd say, Will-- there's also focus on the sort of organizational and nontechnical aspects of that environment too. So what we're talking about is putting in place this environment that allows you to move from the current state to the future state. So we know there's the technical that Felix just talked about-- having automated regression testing, having strong documentation models of the future-- the current and the future-- those kinds of things, tools to support that. But you also need to think about, "Well, do I have the right skill sets?" People who, say, want to move to the cloud. "Do I have that?" You've got the right acquisition strategy.

But also, I think a really important piece that we've run into quite a bit is the notion of having governance over sort of common principles, and when I say that, I don't mean software engineering principles and practices, like, "Just follow good coding standards." What I mean is once you have identified a future vision or a future state that you want to move toward, you need to define some principles or some guideposts, or guiderails, to kind of keep you inside that boundary as you move forward, because what happens is you go out and chart out this great new vision, and as developers are making day-to-day decisions, they aren't aware of what actions they might take that might run counter to that vision, and what happens is over time you start to just diverge too far from it. So it's very important to establish some set of principles and some governance structure.

And so an example of that is a common place for large-scale IT systems to start is to create a set of data services because it allows you to encapsulate your application layer from your data layer, and what that does, it allows you to modernize the application layer separate from that without a ripple effect. That's one tactic sometimes you can use. If, for example, you use that tactic and the developers don't have a principle or guideline that says that they should not directly connect databases, they may go in and just do that, and then when they do that, there you are running counter to that vision and

that goal, and it makes it difficult to get there. So I think part of that environment includes having a set of principles that are specific to your future vision to keep people sort of moving in that direction.

Presenter: So those principles are essential to the architecture, right? Oh, you have a good question?

Presenter: Oh, cool.

Presenter: Yeah, I was just seeing here. So there's actually a very good from-- I think it's David. So saying, "You also need to put an incentive structure into place to support the whole transition." Yes, absolutely.

Presenter: Yes.

Presenter: So that would be one of the topics that you need to think about when you create the environment: What can you do to give the incentive to the people to actually follow and provide the new things? Absolutely. Very good question, thank you.

Presenter: So there's an architecture flavor to what you're saying here, including the incentives. We need to align the different communications with a common view governed by principles. Can you portray this with an architecture background?

Presenter: Yeah, I think. So the issue here is-- so as soon as you get some idea about what you try to

achieve, you need to have a tool that helps you to get structure into the whole modernization project, which means the technical structure, but also the organizational structure. You need to have something. So kind of like a skeleton where you can put the meat around it to actually then hopefully build some nice human body. And that guiding structure actually is the architecture. Going back to Conway's Law, there is an alignment between organization and the technical system. If you can capture whatever makes sense in your organization in an architectural structure in there, that also maps to your organization, where you can really say that, "Okay, yeah, so that structure, I can see that also in the organization, and the organization will support it in some way," then you have the internal schedule that you can use for making your technology decision, making your architecture decision. So Stephany was talking about some kind of governance infrastructure. You can align that around that architecture. So your testing environment or your tools environment can all be around that architecture, so you start creating the first structure that allows you then, little by little and in steps, to build out until you are then, after a certain amount of time, achieve what you want to achieve.

Presenter: So tools really can be significant here. The use of application lifecycle management tools is very common in industry, and I've heard of organizations describe

the process of bringing someone new onboard, not as a, "Sit in this conference room, view these slides, listen to this training," but go into the tool, and understand the architecture, understand the technical decisions that have been made. Can you discuss how that kind of engagement using tools assists in this process?

Presenter: Yeah, that's actually-- for me actually always-- yeah, I should say "funny" in quotes. It's amazing to watch people. So, as the organization we are typically working with, when it comes to dealing with architecture, I wouldn't say they're that mature. So for many of those organization, architecture is still those boxes and lines that you draw on the whiteboard and many people talk long time about it, and then when you are done, then you do the real work. So, what I just said, that you actually use the architecture as a skeleton to build your system out, means that you need also get a little bit more sophisticated with architecture. So, and here we are talking about something that you use as a communication tool, which means it has to be explicit. It cannot be in the brain of some people, because those people may not be available when you need them. So it has to be documented in some way.

But we also know putting everything in a text document, that is the same thing like just saying-- digging a hole and putting it in there and building a grave for it. Won't work. So you

really have to create a living model, and that's where the tools come in. So if you build the architecture using a tool-- and there are enough tools available for doing that-- and you take it serious-- so you build the model and you...and you communicate that model throughout the process-- that is one of the important factors for success for a project. And yes, I know architects have a hard time at the beginning to get used to it, because it is work. You have to be precise, not just like boxes and lines on the whiteboard, which is sort of like you can interpret anything you want to. You have to be precise, it has to be consistent, and all of this, and that is tough. That is hard. But you have to do it.

So after we guide the architect through, after they get to the point where they actually have a model, we can actually use it to communicate. It is amazing to see them watch and say, "Wow. All of the sudden there were questions; I was able to answer them. Within five minutes that topic is done. Here, we can move on to the next problem." We don't have hour-long discussions about things. So yeah, tool, absolutely.

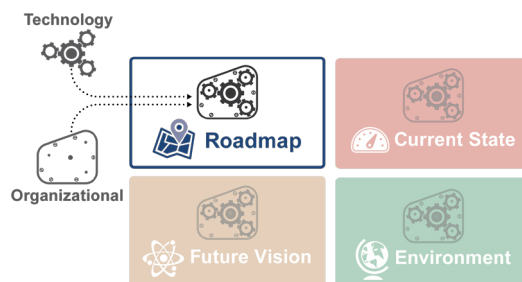
Presenter: So this is another aspect of connecting the new shiny thing to the real environment in which people have to operate and continue to provide value to their customers.

Presenter: Yeah, let me move this forward a little bit here. Let's talk a

little bit about this next topic,
because this rolls right--

Know how to move forward

Know how to move forward



Technical Roadmap

- Use of new technologies is risky.
 - plan for feasibility studies / Prototypes
 - off-the-shelf product integration
- Plan for backing out of a technology in case it shows to be inappropriate

Strategic roadmap supporting technical roadmap.

- Set up a steering group
- Set up the funding model
- Set up review boards
- Set up investment management

**011 We're going to segue back to this topic in a second and talk a little bit about the incremental architecture. So why don't we talk for a second about the roadmaps, where basically one of the things that we suggest is to develop a roadmap that includes both the technical and a nontechnical flavor to it. So in the technical aspects, where it's the typical phasing and tasking that you would expect to have a roadmap, a technology roadmap, that defines your future architecture. So that's where you would have your technical architecture, your future vision. But then we'll also want to be thinking about some of the nontechnical aspects. And so we suggest that you also have along with that a strategic

roadmap, we call it, which basically allows you to think about some of the other mechanisms that you need to have in place to move that forward.

So, for example, some of the things you might want to have in the strategic roadmap are things like a steering group that helps you to make decisions across multiple groups, because one of the problems that you run into in these complex environments is, like we said with Conway's Law, you end up with these groups that are stovepiped everywhere. You need to kind of bring them together and start to be able to make common decisions. So one of your first common decisions might be, "Do we actually agree on a common vision?" Which is sometimes something that takes quite a bit of-- and in order to do that, you really need to have the level of what Felix is talking about. So one of the things you can run into is that if you're looking at a bunch of PowerPoint slides that are very roughly thrown together, it's really hard to say, "Well, do we agree on this vision, and what is this going to buy us from a quality attribute standpoint, performance, security, etcetera?" It's just too vague. So you really need to get down to the level where you can actually define that future state pretty well in a model in order to have good, intelligent conversations about it, and then to agree on, "Do we agree as a group that we're going to support this?" And then as somebody else was just chiming in, then you need to

think about, "Well, how are we going to fund this and incentivize it?"

So it usually can cost more; it can be painful, especially if you're building common components, so you have to think about if you are building components that are going to cross multiple programs, are you going to have each program fund a little bit of it? Are you going to have some other structure? How can you basically fund common components, is a huge challenge that people run into, especially when they want to be able to leverage the ability to share common infrastructure across multiple systems. So those are a couple of the things to think about from a roadmap and a planning aspect. I think it's important to think about how to focus on the technology but then also to focus on the nontechnical.

Another one is the architecture review boards. Again, if you're making decisions as you're going through the development and the implementation and the architecture, and then you're evolving other features at the same time-- like Felix said, it's not a case of it being-- you can't stop and just start doing an architecture project and migrate it. "If you build it, they will come." That doesn't work. We know that doesn't work. So you have to be basically developing your architecture alongside of your features, and therein is a big challenge. So as people are developing features, making new features with the

existing systems, and you're focusing on trying to build infrastructure for the new, you have to weave those together. That means that there needs to be an architecture review board that crosses multiple programs and projects that allows you to determine whether a decision you're making today is going to impede or enable your future architecture. So you can't just focus on one or the other; you have to be focusing on both of those at the same time, and your transition plan, like Felix said, from the beginning. So those are some of the other areas to focus on the nontechnical.

Presenter: I was just going to, from the technical side, just throw one thought in there, and that is be aware of whenever you do the transition from today to tomorrow, you are in a high-risk area. You are not running a project that you did many times during the past 20 years. Those are big projects, they have a high probability of failure, so take that into account. Don't think you can just put in some kind of timeline to do all of those things and it'll be just fine. Yeah? The positive way. Think more in terms of: Every time, every step that you go along probably will fail. So take that into account, and that means-- and plan for it. So plan for feasibility study, prototypes or something.

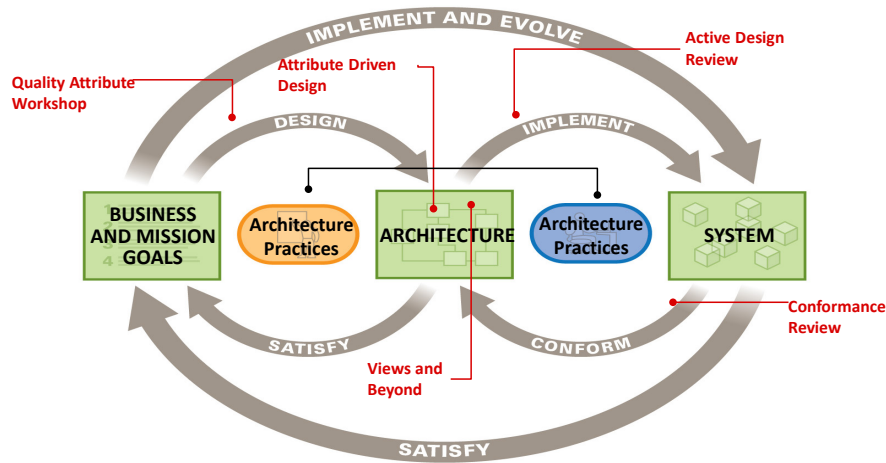
For some organizations, there was also a way that they looked at all the steps that they want to do in the new system and asked the question,

saying, "If this step would fail, how easy is it for us to back out again of it?" And they preferred mechanism that helped them to go a little bit further, see how it goes, and if it doesn't work, go back again. So those are all criteria to take into account when creating a roadmap. It is not just, "Oh yeah, here's a bunch of activities. See who is available, and then just put it on the timeline."

Presenter: So as we think about the tactics and the intermediate stages that we want to progress through, it is not limited to the deployment of new technology. It is about the human resources that operate that technology who need to recover gracefully if the technology doesn't work, or who bring you new requirements if the technology changes the way they do their work. It's really a complex, as we said, sociotechnical challenge.

Iteratively move forward

Iteratively move forward



**012 Presenter: Yeah, it is, but I think one of the things that I think is important on the technical side is-- and I think you alluded to this earlier-- is the importance of doing this incrementally. So one of the things that we feel is very important is to focus on doing and developing your architecture and delivering your architecture, pieces of your architecture, incrementally as well. So like I was saying with agile, we've gotten pretty good at identifying smaller batches, we have stories, we develop features, we put those into sprints, but a lot of times what people still do with architecture is, like Felix said, they go off and they try to do that as an ivory tower project. It's off to the side and has some kind of pilots, concept pilots, but there's no real integration between the two, and what we've

found, at least in the environments we work in, is there's not the funding and the business doesn't have the tolerance for that kind of thing. You need to be able to identify that future architecture vision and then figure out how to weave those features in as you're delivering other features, and that is really the art and craft of basically trying to manage the dependencies to make that cycle work, and I think one of the things too that's important to do when you're developing architecture incrementally is to basically have a pretty tight cycle for being able to identify the design that you're going to develop in an increment, be able to go in and do a very quick review, design review-- you can't do a CDR that takes five days. Forget that.

You need to be able to do that very quickly, efficiently, whiteboard session-- done-- and then you need to be able to develop that piece of architecture along with your other features, make sure it conforms with-- that the implementation, the code, conforms with the design, and then you need to make sure that all of that is documented from the beginning, and then as you go back, what you've learned, you've integrated it back into your design model. So it's important to have a very tight sort of process there for doing all those things, because no longer can we have these long, drawn-out-- they never worked in the first place, but people would try to have these long, drawn-out

architecture tasks that just never got integrated, and--

Presenter: And this is really the fifth element. The astute observer would have noted that you had four boxes in the graphics used till now, and this is the fifth element.

Presenter: This is the fifth one.

Presenter: I just want to go there, is a question here, very interesting, right for that topic, asking for: Are there any strategies to convince the business, a business unit, that treating architecture as a first-class citizen actually provides value? So what Stephany just said is-the... really is that you have to show value, and you have to show value very quickly. So you cannot try to convince a business unit and say, "Okay, give me one year time, and I will provide to you this document here that will solve all the questions that you have." No business will go for this. You have to show that you do your first increment of the architecture-- so focusing on one topic, one topic only, saying, "Okay, we'll fix this." In the architecture you describe how to fix it, and what else is involved, including the organization, what they need to do, and then you integrate that piece and go to the developers and say, "Here's how you do it." So you align the architecture into the code, and make sure whatever the developers do is actually aligned to the architecture back again. So to try to solve one problem, you show that now very

quickly you can use that actually as a communication tool. You can guide your development to implement the right thing so when they put it out, it actually works and solves the problem.

Presenter: Yeah, I agree. I think that the showing value-- you're right on with the value-- and I think one of the things that we've had to develop a skill to be able to do is to work with the business and figure out what projects are coming down the pike that can actually be nice candidates for your architecture component. So say, for example, you have data services as one example. Maybe there is a need, a project coming down the pike, where you need to share data externally securely, or maybe you need to share data within your organization. We will latch onto that project and try to move it forward by getting that architecture component in there for them so that the value is that that project actually gets delivered with business value at the same time as delivering version 1.0 of the architecture feature that could be shared, and that's a tricky thing to do. It requires working closely with-- the business and the architect can't be far apart. We have to work very closely and collaborate on figuring out where the windows of opportunity are for doing that kind of thing.

Presenter: So you reminded me of a story. I once interacted with a CIO who was very successful, and he talked about all the expensive art he

had on his bookshelf. These were documents full of technical drawings that were inanimate, and what you're talking about is how quickly we can go from strategic to tactical to realizing what's described there, and that getting to what people need to use and what people benefit from is one thing that makes the architecture a first-class citizen.

Presenter: Yes.

Presenter: Right, but also, back to that comment, only if you really do that iteratively.

Presenter: Mm-hmm. That's true.

Presenter: So the way how this-- the one slide that we just showed-- is built intentionally to really show that we are not doing a one-step approach here, which you probably had in your experience that it doesn't work. We have that all over the place. It doesn't work. So don't even try. So there are circles in there, iterations, between the business-- because the business-- we're talking a longer period of time-- the business will change too during that time, which means we need to do some adjustment in the architecture, and which leads to adjusting the system. We typically ignore the cycle between architecture and system. There is still-- in many organizations, there is still a big wall in between. So architects also are the ivory tower people. They get all the big bucks, and we poor people here that do implement, we don't get

anything and have all the work. That wall is still there. But the work actually has to be done by both.

Presenter: And an iterative approach really keeps the speed of communication going too, so that people who think strategic thoughts have to come to a cadence that's new, perhaps. It allows them to have strategic thoughts more frequently, I would suspect though, instead of just one time.

Presenter: Well, it does. It does, but it also helps to be a little bit more grounded. So of course the developers will say, "What a great and a wonderful idea that you had. Sounds good, but it won't work."

Presenter: Where's the practicality?

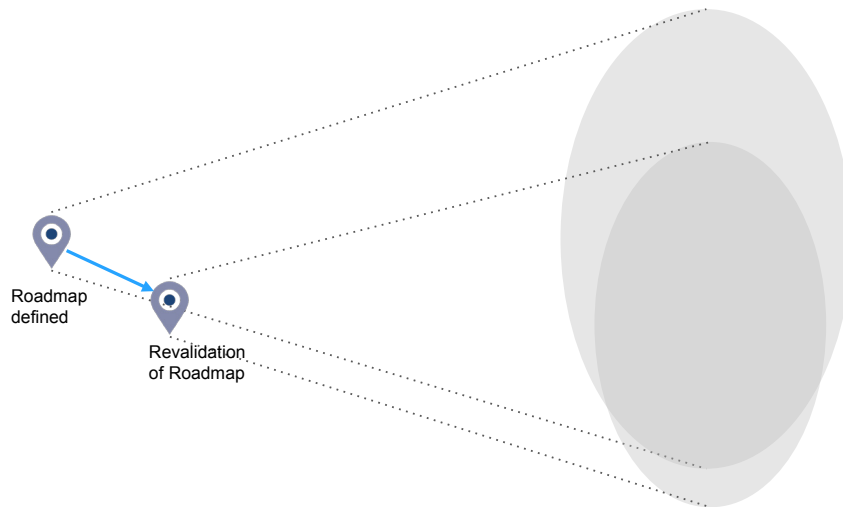
Presenter: That's right.

Presenter: As an architect, you have to deal with it. So either convince that person that that person's wrong, or fix something, and you need to be convinced to do something right.

Presenter: So, roadmap, I think.

Periodically Re-evaluate Roadmap

Periodically Re-evaluate Roadmap



**013 This is what we're into now, the next series. Why don't you go ahead and walk through this?

Presenter: So we're are trying to put all of this together, what we just said. So after you did all the work, all those first four steps-- you did a great job, you looked at where you want to be, you made a very appropriate selection, saying, "From all the possible things that we could do, we only focus on those things because that would make our life a little bit easier." You had a very close look to what you have. You really identified the key issues that you really want to solve. You put all your environment into place. So you have your regression testing, you have your steering group, you have everything in place. So after you did all the work, which can be-- it's a lot

of work-- there you are at that point and saying, "Okay, now I know it. Now I know exactly where I want to be, I know exactly what I have, and so just let's now go on and move and just do it."

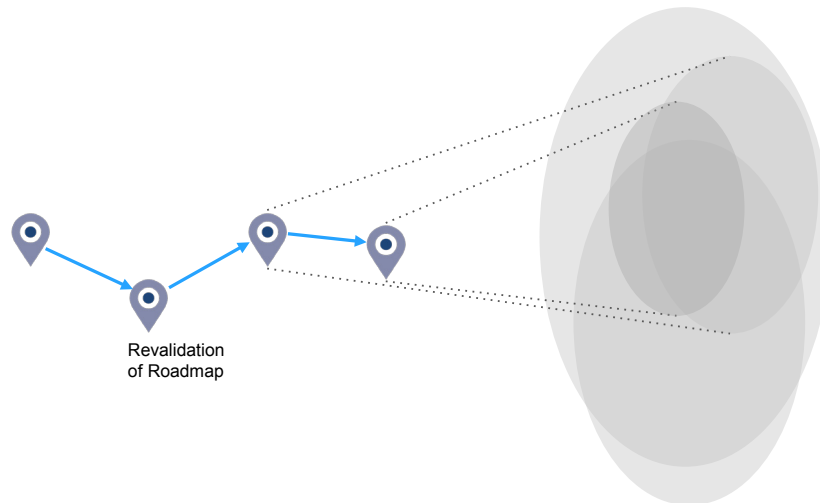
So, but fact is, as we already started talking to, is whatever you know now is a very high risky area. You don't really know where you will end up with. You're also talking here a longer time. So it might be a good idea-- we talked about iteration-- it might be a good idea from the whole effort perspective every now and then-- so let's say maybe after half a year or year, depending on how long the project is-- to stop. Look at what you achieved. See what you learned. You worked now for, let's say, a year. So you learned a lot of things, and you may have devised whatever you think the future should look like from the issues that you may have identified a year ago may not be the real ones. You may have identified others, and such. So you revisit that and make some course corrections.

Presenter: A new bearing.

Presenter: Yeah. So therefore now at that point the risk that you still have is hopefully a little bit smaller.

Periodically Re-evaluate Roadmap

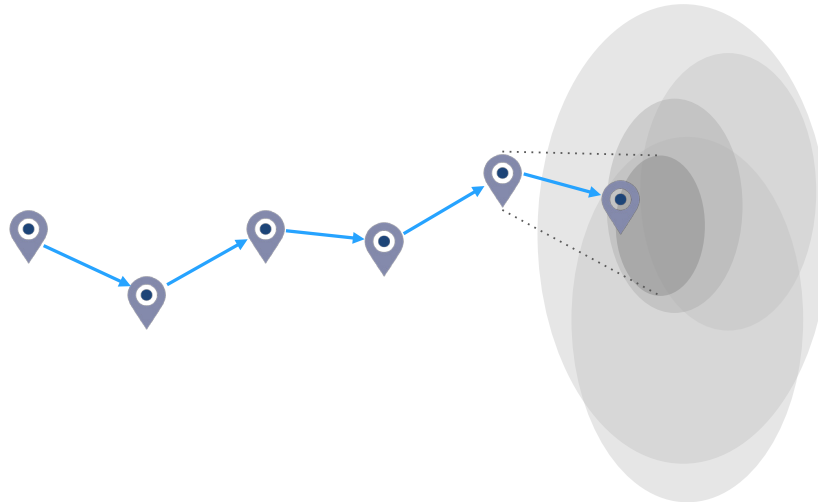
Periodically Re-evaluate Roadmap



**014 But there is still a pretty big area of risk that you don't really know where you end up with.

Periodically Re-evaluate Roadmap

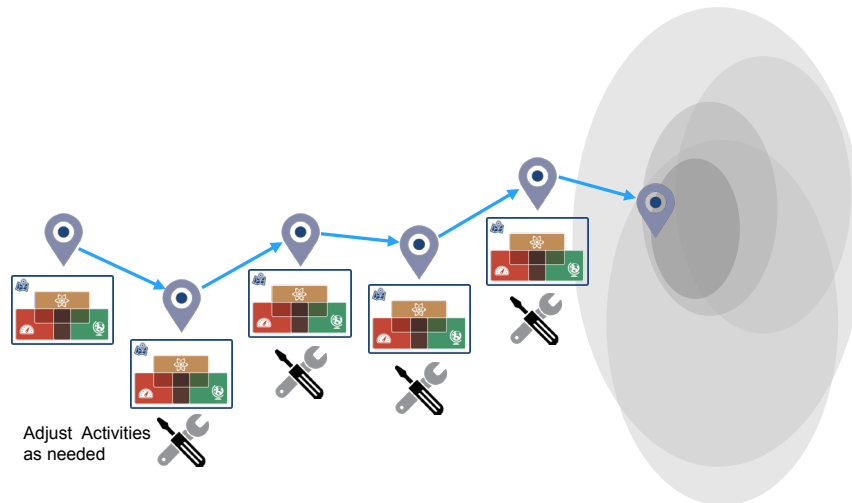
Periodically Re-evaluate Roadmap



**015 So which means you repeat that process after-- periodically. Maybe it's each year, or each half year. So at some point. Make the-- revisit that problem, see where you are, and come closer and closer to where you actually end up with, and then hopefully end up at the place where you would like to be. But because of this, you also need to keep in mind--

Continuously Update the Modernization Artifacts

Continuously Update the Modernization Artifacts



**016 That for each of those points in time, we want to look at do you have to make course correction or not. You have to look at all the artifacts that you created the first time and you have to make some adjustment to it. It actually might be a good idea-- so while you are actually doing the transition here to the new place-- that in parallel you actually always look at your artifact that you have-- your future vision, your current state, your environment, all the infrastructure that you put into place-- if you have to adjust something. Because you know the time will come when I have to look at that again and make another course correction. So that is the fifth point, to execute this in iteration. Don't believe the first roadmap that you have. It's a good first cut. You have to do it, because it gives you an idea

of how to move forward. But you can also be assured that roadmap will be wrong. So therefore every now and then you have to revisit, make a course correction until we get to the point we really want to be.

Presenter: So reexamine both the social and technical as well. So you'd be looking at what's been the benefit accrued from this initial release, perhaps; how does it change our workflow; what new needs for technology does that create; and you really want to reenvision where you're heading.

Presenter: Right, and I think somebody pointed out here that the business case-- making the business case from the beginning of maybe wanting to do some architecture work is an important thing to do, and it also does need to be revisited. So if you make a business case that, say, you're going to put in a security infrastructure or something like that, or maybe you're going to improve performance, you're going to put caching in or something like that for critical services, you do that, you're going to want to make sure that you do go back and see if actually, once you implemented that, did you actually reduce the cost, or did you reap the benefits. So I think there's that, of establishing those business cases, which is-- whoever submitted that is absolutely right. You have to have that from the beginning to actually sell it, but then making sure that you're actually getting the

benefit. I thought that was a good observation.

Presenter: So I think Shane has a comment from a viewer.

Presenter: Yes, just two in the Chat. First one's from Daniel, asking, "We have an interesting case where we've used an automatic migration of our legacy system in one big bang. It was a huge project and the switch was very successful. However, now we need to figure out how to live with automatically migrated system." So, any comment to that?

Presenter: Yeah. So-- and I don't know what the migration was. I assume that that means you migrated from one language to another one. So maybe it was Cobol to Java or something. So I don't know what it is. But in general, keep in mind that, yes, all those automated tools will help you to switch from one technology into the other. But it will not change anything in terms of how your system works. So there are no functional changes. So if your problem is only that you're using a language or maybe a framework or something that is not supported anymore and therefore you need to go to something else, then yes, it's a good approach. But as you discovered, you now have code that was touched by some automated tool with changes in there and there is no guarantee that for a human being that is actually an appropriate representation that you can actually read and understand it

and do something with it. You may end up with solving the first problem but ending up with another problem that you cannot maintain the code anymore.

Presenter: So it's rare that the transition being desired and attempted is purely a technical one.

Presenter: Rarely.

Presenter: Yeah, and so what-- I don't know what this migration was, but one of the problems that we run into is that people see-- they fund projects, even an architecture project, with a start and a stop, and so there is no funding for the remaining cleanup that needs to be done, and often there's a lot of cleanup-- and maybe there's ongoing work to sustain and maintain that architecture component-- but it's a very tough business case to sell, in a lot of cases, to actually sustain them, or to finish the migration. So maybe it's technical, maybe it's not technical, maybe it's a mix of things that are left hanging out after this migration, but I think a key part of it is trying to get it in the plan that you're going to deal with the technical all the way through and continuously maintain that as well as the people side of it.

Presenter: So while you were finishing up, Felix, Daniel did get back saying it was a language. It was EAE LINC to I think C-Sharp is the language, was his follow-up. Then we had another comment from

Carl saying, "This reexamination requires honest reflection. How can we ensure that this process is handled honestly? It's not easy for architects to admit to the team they were wrong."

Presenter: Oh, yep. You've got a very sore point here. So, the best solution that I can-- that we actually see and actually that's our advantage here of being is it would be a good idea to get a third-party in there that facilitates that reflection process. Ask tough questions, ask everyone, maybe some interviews, to get the collection, saying, "Okay, what went well? What didn't go that well?" And for everything that didn't go well or did go well, then ask really the question to all of them, saying, "Why is it so? What happened?" So if you can get-- and it does not necessarily have to be an external organization. So in your own organization you find a group which are not involved in that project but do similar things. Might be a good idea-- maybe talk to them and get them in as interim facilitator. Yeah, but you're right, if you cannot get to an honest reflection, it's the same like saying, "Oh yeah, yeah"-- we close our eyes and say, "The first is still fine. We just follow it." And who knows where you end up with?

Presenter: Yeah, and I think too it's the perspective. So Google and Amazon and whatnot, they've turned to looking at failure as success. So what you learn from failure in a lot of cases, they incentivize more than

success, according to what I read. So I think we need to change the way we think about things too, because sometimes-- okay, so maybe a solution didn't turn out to be the best solution, but you learned a lot about your environmental constraints and it may be people stuff, it may be technology stuff-- for whatever reason that solution didn't work. However, there are probably things that were successful there that you want to build on. Most of the time we see successful efforts that we can build on. And then there are places where you don't want to go down that path again and pick it up, because it is a learning process, and I think if we don't treat it that way, for whatever reason-- in agile and in product development flow and things, we've learned in lean thinking that we need to do that for features. We need to put them out there and let the users react. However, for architecture, we still seem to hold a binary-- you succeed or fail. And we also need to-- these are big decisions, big changes, and so a lot of times you need to be able to work with the management to say, "Okay, we don't want to have a huge outright failure, but we are going to learn in a pilot, and we're going to take that forward, and just know that some things will go forward, some things won't. But it's not a huge failure if you--

Presenter: It seems like the iterative approach is really essential here, and if you're able to do small iterations the consequence, the

adverse consequence, of an unfortunate path is limited to a greater degree if it's a small iteration.

Presenter: Yes. Yeah.

Presenter: I also think there's a mindset behind it.

Presenter: Mm-hmm. I agree.

Presenter: So if you're an architect and you come up with a solution, the worst thing that can happen to you is that you actually believe what you have. We have a way better approach when saying, "Okay, it's a good first guess. So there is a good chance that's the right one, but it's just a chance. We don't know. And I as an architect don't know about it." If you as an architect get that mindset in there, that opens up and saying, "Okay, so now I know that I'm not having right solution. I need to put a mechanism in there that helps me discover if I have the right solution," which gets the iterations in there. Feedback on those. So it's a mindset issue.

Presenter: So there are learning cycles.

Presenter: Mm-hmm. Mm-hmm. Mm-hmm.

Presenter: Great.

Presenter: Okay, before I turn it over for any last words-- we're done in about three minutes-- I just wanted to remind everybody-- a

couple questions in the queue asking if the slides from today are available and if this is archived. The slides are available now in the Download Materials tab on your webinar interface. The event is archived, so you will receive an email letting you know when that's available tomorrow. A reminder to everybody to please fill out the survey upon exiting today's event. We appreciate your feedback, your thoughts on today's presentation. And then lastly, we wanted to let everyone know about SATURN 2017, and SATURN is our SEI Architecture Technology User Network Conference. It's an annual software architecture conference. This year it's taking place May 1 through 4 in Denver, Colorado. The great lineup of speakers and program is now available on the SATURN website, and from registering from today's event, everyone will get 15 percent off for attending SATURN, and we'll send out that discount code out through email, so we hope to see some people there. So, last thoughts. Will, anything you want to-- we have one more comment in the queue we can wrap up with unless you guys just want to close it out with something else.

Presenter: Well, let's hear the comment.

Presenter: Okay, from Tito asking, "In Agile, hardening sprints can be used to tackle the problem of cleaning up code and maintaining the architecture, documentation, etcetera, instead of leaving it to the end of the project implementation."

Presenter: So we can explicitly account for the need to do that, and as we look at the balance between the social and the technical, I think opportunities to harden the workflow and make sure that people's work instructions are sufficiently updated. That may be something that comes into play.


Presenter: Right, right.

Presenter: Thanks very much for speaking today. This is a really important topic, and more and more modernization issues are going to be coming to us. So, glad to have you.

Presenter: Well, the older the IT system that we all use, again, the more you have the problem to modernize them. Yes.

Presenter: Great. Excellent talk today, folks. Thank you very much. Folks, that's all the time we have for today. Again, please fill out that survey upon exiting today's event, as your feedback is always greatly appreciated. Thanks everyone. Have a great day.

SEI WEBINAR SERIES

 **SEI WEBINAR SERIES** | Keeping you informed of the latest solutions

 Software Engineering Institute | Carnegie Mellon University