Agile Metrics: Three Secrets to Success

Table of Contents

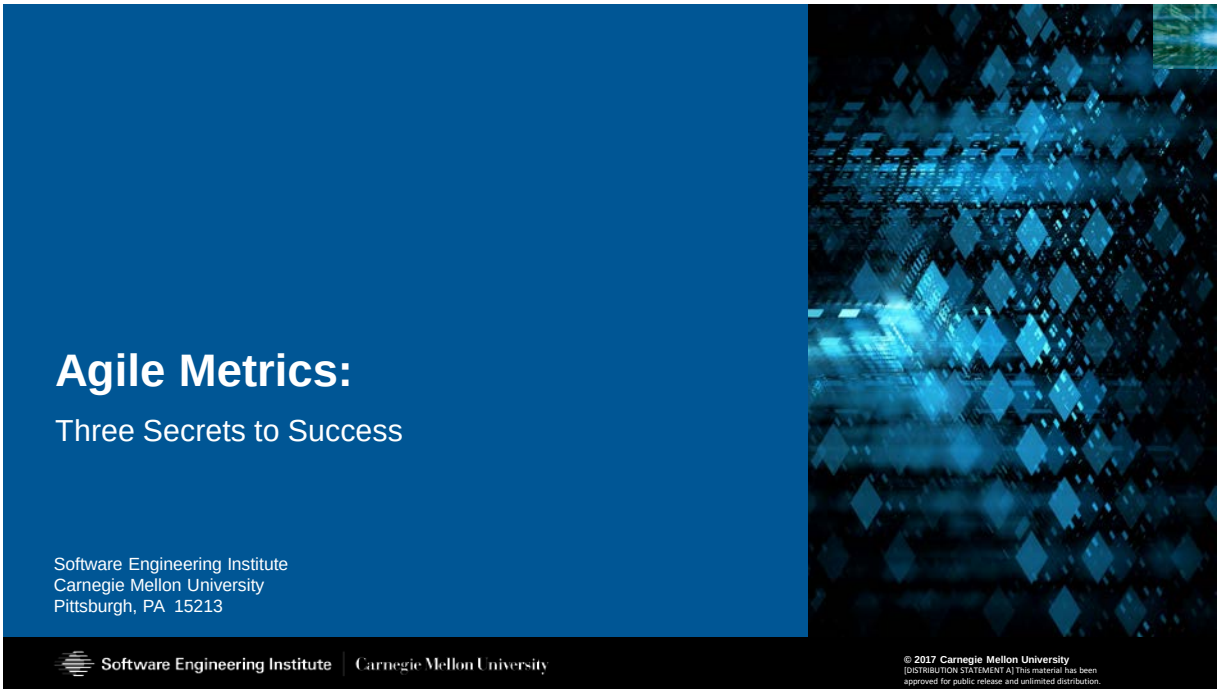
SEI WEBINAR SERIES | Keeping you informed of the latest solutions

Carnegie Mellon University

Carnegie Mellon University

Copyright 2016 Carnegie Mellon University

Agile Metrics: Three Secrets to Success



**004 Presenter: And hello from the campus of Carnegie Mellon University in Pittsburgh, Pennsylvania. We welcome you to

Virtual SEI. Virtual SEI is our new streaming platform where you can watch live events or access on-demand videos discussing our latest cybersecurity and software engineering research and best practices. Our presentation today is Three Secrets to Successful Agile Metrics by Will Hayes. And depending on your location, we wish you a good morning, a good afternoon, or a good evening. My name is Shane McGraw. I'll be your moderator for today's 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 you can submit those questions through the Q/A tab, or the chat tab, on the page interface. Now, we will also ask a few polling questions throughout the presentation. But they will only appear as a slide within the video window. And we ask that you put your response into the chat tab. So, you'll need to type in your response to that polling question into the chat tab as we go along.

Also, a link to today's-- a PDF copy of today's presentation slides are in the chat area now that you can link and download those slides. And also, with this new platform, we ask that you fill out a survey upon exiting today's event. And that survey link will be added to the chat tab as well. And we greatly value your feedback there. For those of you using Twitter, be sure to follow @SEInews and use the hashtag #seiwebinar.

And now, I'd like to introduce our presenter for today. Will Hayes is a principal engineer on the Agile in government team at the SEI. Will currently supports major programs in DoD and other government agencies that acquire software from contractors applying Agile methodologies. And throughout his twenty-six-year career at the SEI, he has supported numerous commercial, government, and defense organizations providing consultation and coaching for a wide range of roles from engineers to CEOs. And now, I'd like to turn it over to Will Hayes. Will, all yours.

Presenter: Thanks, Shane. Thanks, everyone, for tuning in. We've got lots of neat stuff to talk about. And we're going to do our best to involve you at every turn. So, first, an official disclaimer, I'm not here to provide a dashboard for project managers to use to manage Agile projects. We're really wanting to talk about the considerations that let you make the right choices and how you know that the choice you've made was a good one.

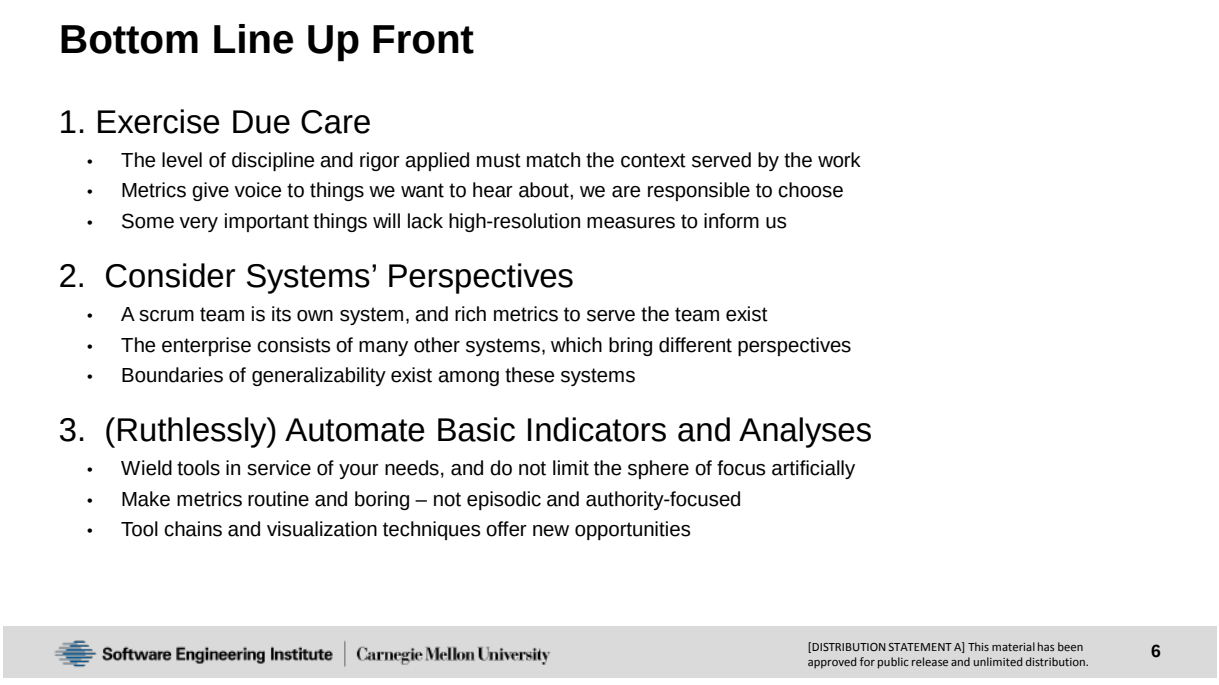
Agile In Government

**005 As a backdrop, our focus on the Agile in government team really is how these concepts are playing out in the government ecosystem. Those of you who work in this ecosystem understand that there are some unique aspects of being successful in this realm. There also are some extremely demanding challenges that are placed on people who are successful in this realm. As well, the concept of an Agile government, that government personnel implementing these concepts from Agile, is relevant here. But that's not the central focus of what we're talking about here.

One of the ways you might summarize what you would aspire to come away with is what would the role of an Agile government product manager be. The notion of a product manager is well understood in a

commercial setting, as well the role of a product owner is well understood in the context of individual teams. We're starting to see now with many of our government clients that product management is a role that is becoming more and more obvious. And so, as you think about the concepts that we talk about today, have that perspective at least in the back of your mind. And I know from looking at the sign-ups that there are folks in the audience who do have that role and that you are facing challenges that are perhaps new to a job like that. And we're keen to hear from you if you could drop us a line. So, Agile in government, that's our backdrop.

Bottom Line Up Front



**006 The three main points, right up front we want to talk into these. And we'll cover them again at the end just to make sure we've got

them all throughout. So, first, as we think about why and what you would want to measure, you take on an obligation in roles where metrics are involved to exercise due care. Often, we hear methodologies or particular approaches to software development espoused, very good approaches, many of them we can learn from. But sometimes, the audience will limit their view according to what is in the off the shelf version of that methodology.

As we know from the history of progression in Agile approaches, many of these concepts evolved from a commercial setting. And some of the demands that are placed on those of us operating in a government setting may not coincide very well with the presumptions that occur there. And so, as we think about what due care represents, we really need to think about the environment in which the system we're working on operates as a basis for deciding what is the level of precision, what is the level is the level rigor that's required.

Just because you've heard somebody say that Agile methods are more informal, doesn't mean that the use of those methods to develop life critical systems can afford to be informal. And so, you must think about mission criticality, life criticality, those sorts of things, as you consider what is the sharpness of the lens we need to have as we measure progress, as we measure success in the application of these techniques.

Secondly, the perspective of different systems, those of you who've worked in the government setting for a long time understand there is no software reliant system that doesn't operate in a system of systems context. Every system has connections to everything else in our domain. And in a similar way, as we think about the personnel engaged in the work, they too are operating within a system of systems concept. And as we look at the application of things like Scrum, there is a very nice description of a system for a Scrum team described there. And there are metrics, well understood metrics, that have been used with great success within a team setting. And we would certainly want to keep using those things that are successful. But to presume that our sprint burn-down charts would be something that we would present at a congressional oversight hearing is folly. And so, there are similar sorts of boundaries that we need to think about in terms of the systems of interacting people in roles of responsibilities that exist.

And then finally the third point, there's bit of a tongue in cheek mention of ruthlessness here. And the joke that goes with this is my bathroom scale ruthlessly tells me how much I weigh every single morning. So, should I be attributing such human emotions to a machine, the bathroom scale? Certainly, not. You can see the folly of that. However, when we think about metrics and the way measures are used in the kinds of settings that

we're talking about, we often have fears and trust and similar concerns associated with the way measures are used.

As we see developments in places like DevOps, and things like DevOps, and the great progression of insight that we can get when we instrument things well, the benefits that could accrue from very inexpensive, very timely measurement, those things can be attenuated by these more cultural or socio-technical issues that arise in use of measurement. And so, we want to make sure that we address those concerns as we think about metrics in Agile in government setting.

A Familiar Problem

A Familiar Problem

Data can shine a light on important things.

If we don't focus on the right thing, we won't get what we need.

Due Care is context-dependent, and should not be left up to the advocate of a particular methodology.

**007 So, diving a little bit into each of those concepts, many of you likely are familiar with the notion of

searching for information, or-- there's an old story, sorry. A fellow comes upon someone stumbling around underneath a streetlamp in the middle of the night, looking somewhat distraught. And that fellow says, "What are you doing there, buddy?" He says, "I'm looking for my keys." "Well, where were you when you last saw them?" "Well, I was down that dark alley down there." "Well, then why are you looking out here?" "Well, of course because the light is so good here." And this story is kind of an interesting way to shine a light on the way we sometimes approach metrics. We often try to make the most of convenient and available data without really recognizing the limitations of the utility of that data and the fact that they may be out of focus or totally off-target for the decision that needs to be made.

And so, as we consider what it takes to exercise due care, we need to think about the affirmative obligation, and I chose a specific term that comes out of the finance industry there, an affirmative obligation we take on to assure that we're focusing in on the place where the information can be found as opposed to merely using convenient data. So, this notion of searching and taking the responsibility to collect the right information and to be open about the level of information and the certainty you have, those are essential for us being successful as we consider the other two of the three main points, as well.

Multiple Intersecting Systems

**008 And so, here we're moving into notion of the ecosystem. And this is a marvelous picture that our colleague Kurt Hess drew for us for one of our early Agile and metrics technical notes. Here we see a system of systems where the gentleman in the middle, the captain in the picture, has an obligation to understand the progression and the success of the development firm that's supplying to this military organization in this case. And so, you see him pondering these different representations of metrics that tend to come from Agile teams.

And in your upper left, you see a gentleman wearing a necktie intended to depict a commercial provider who is contracted to a military branch in providing capabilities that will be fielded to

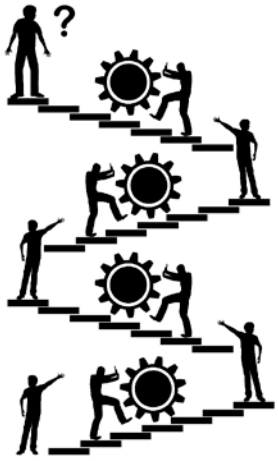
support the war fighter. So, the captain has an obligation to understand what's happening in that largely commercially driven world, although many of these providers have a keen and deep understanding of the military setting and how the products they work on are used. But that captain needs to understand so that he can then provide information that is needed by the other two folks in the picture. So, the colonel, to his right, has an obligation perhaps as the material leader in this system program office to make sure that the program is going in the direction and at the speed that it needs to go. Not just the work of an individual provider, the one depicted in the upper left, but he needs to balance the contributions of many different parties. And the captain understands his role there.

Then the general depicted in the upper right quadrant, she can ill afford the time to look at sprint burn-down charts or to understand things associated with individual defects that might occur, though she often gets pulled into conversations like that when the visibility rises. But that captain really needs to supply a different kind of information to go up the chain to that general than he does to his colonel. And so, understanding these different needs and the interactions among them is really, really essential to having effective and efficient approaches to measuring progress.

There's a contrast between project management goals and product management goals that are found in this picture. The project management goals, primarily the focus of the gentleman in the necktie in the upper left, don't necessarily find their way to the general in the upper right. And in fact, it's bad news when that happens, when the general has to look down in to focusing on the taking care of business, the hygiene issues, are we taking care of the business that needs to be done routinely. We want the general to be championing the cause of the whole enterprise that's represented in the system program office. And so, there are roles relating to sponsorship, roles relating to being a stakeholder, and differences in tactical versus strategic that all come into play as we think about the interacting systems where we need to apply due care to collect information to feed the needs of decision makers that are a part of these systems.

Barriers to Automation

Barriers to Automation



Metrics often focus exclusively on:

- Appeasing an authority role
- Demonstrating competence
- Validating the chosen path

This may engender trust concerns, and often conflicts with the concept of an empirical process – one where we learn from looking at facts that inform tactical/strategic options.

**009 And then finally, perhaps my favorite of the three to talk about, there are a history of cultural and socio-technical issues associated with the way we do measurement in a business setting. Often, we are focused on appeasing an authority role to assure that they understand that we are competent. And a very keen point made by many Agilists is often our metrics focus on validating the single chosen path, a path chosen long in advance of the work occurring. As we see the benefits of Agile approaches growing more and more across the government setting, we understand now that the intent is to have short learning cycles so that we can pivot and move in a direction that's more beneficial to the program overall rather than blindly following a plan that was made based on less knowledge than we have at this point.

Because of this tradition of appeasing authority, focusing on competence, and making sure we're validating the correct path, we tend to have concerns relating to trust. And we tend to see conflicts in the mix here. Those things are barriers to automation because the interpretation of an individual item of data is so context dependent, is so dependent on who's in, who's out, what's up, what's down right now. Whereas, my bathroom scale doesn't really care. It doesn't care that hey, it's a weekend. I'm not supposed to be as active as I am during the week. Give me a break scale. Give me a lower weight. That's a silly thing to ask. And if you think about the way we constrain our measurement approach, the kinds of barriers that exist for automation, these are really legitimate concerns that we need to address.

And many of the efforts that we're seeing among champions for Agile methods try to address these things. Some of them address them by trying to draw a strong boundary around the team to protect them from such influences. Others address them by showing a clear path and an enterprise awareness such as we see in the disciplined Agile context.

Polling Question #2

Your Role

1. Government employee working in a program office
2. Contractor working in a government program office
3. Employee of a firm serving a government program
4. Employee of a firm doing commercial work
5. Coach/Advisor/Consultant for government
6. Coach/Advisor/Consultant for industry
7. None of the above

**010 So, with that, we've got a poll question here, a polling question, to try to get a bit of a gauge on who our audience is. And so, I would ask you to sue the chat window or the Q/A tab, whichever is visible to you. And simply enter the number that best describes who you are. Now, let me see if Shane's got a comment or a question from the crowd, yet.

Presenter: No, just a reminder to folks that the slides are available you'll see in the chat window to download a PDF. That will open up a new tab. And you can walk away with those today. People from various locations joining us, Will, but no actual audience questions at this time. But we'll give them a little bit of time to chime in since they're having to write it in. So far, I've got, let's see, three, five, six, one. So, we'll

tabulate them as they're coming in. But we'll turn it back to you to--

Presenter: So, I had a chance to review the registration list. There were some four hundred folks, some old friends I hadn't seen for a long time. I hope you don't mind my gray hair. It's been some years for some of you. And it's interesting to see the range of roles that are participating. We have people that are engaged in law enforcement roles. We have folks that are engaged in commercial large retail chains, a good crowd from Sandia, some folks from U.S. Air Force as well. Glad to have you.

We will use these polling questions as a break in between segments. So, if you have questions come up, try to get them into the window before the polling question comes up. And that way, you'll be in the queue. So--

Presenter: We have a very diverse audience. Yeah so, I'm seeing every number and very infrequent switches. So--

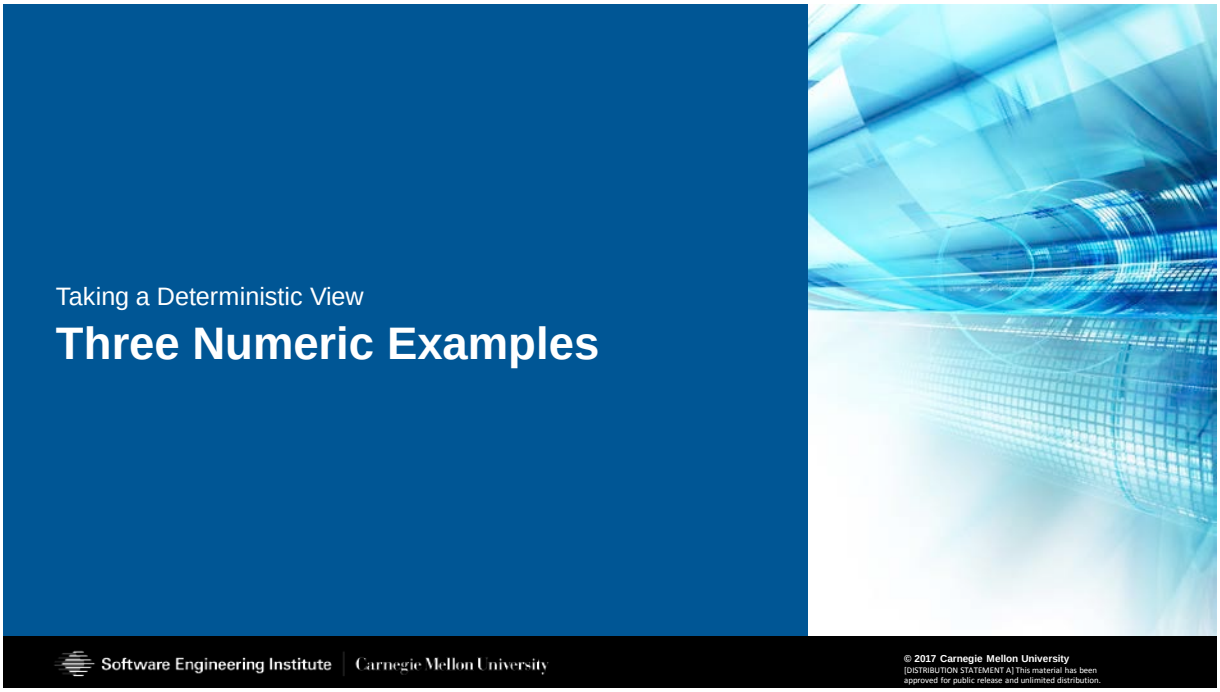
Presenter: Great.

Presenter: No tabulations, we'll just turn it back to you.

Presenter: So, everyone didn't say none of the above. We don't have a bunch of mystery folks.

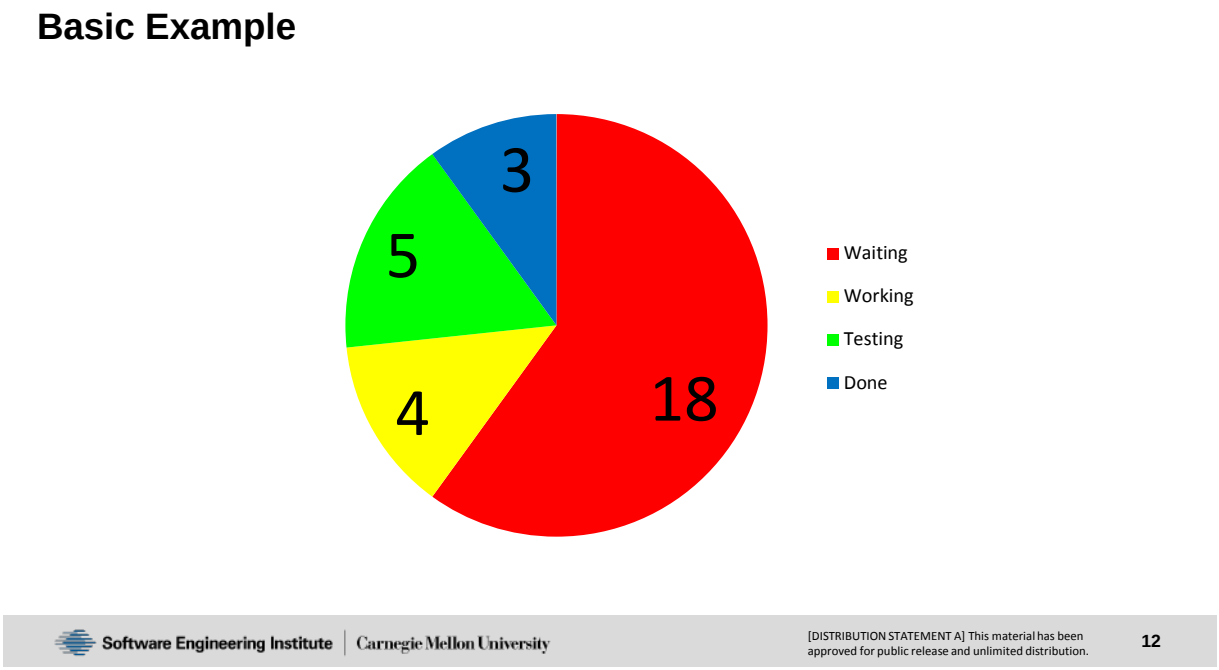
Presenter: That's right.

Presenter: Okay, terrific.



**011 So, let's dive into a numeric example. What I want to use-- what I want to do is use a colorful and kind of a catchy image to thread throughout some specific examples. And so start with this pie chart.

Basic Example

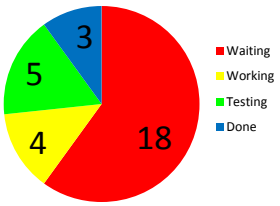


**012 And those of you who've been in Agile classes I've trained, you might have seen this before. But here is a simple pie chart reflecting the status of thirty different things. And these thirty different things go through four different states. We start with waiting. There are eighteen things that have not been started. There are four things that are in the process of being worked, five things being tested, and three that are done. And what I'm going to do is use this graphic and this set of numbers for a set of contrasting examples. But the point is this is a single snapshot in time. This is how we understand the status of each of the thirty items at this particular point.

IT Modernization

**013 And so, first let's talk about it in the context of an IT modernization effort. What I want to allude to here when I use the term IT modernization is systems that collect, organize, analyze, store, protect, and communicate data and information. So, these are software reliant systems that primarily manage data in the ways that I listed.

IT Modernization Example



These are 30 *RICE* objects* that define the scope of work for one or our vendors.

They will be folded into a series of three releases, which will integrate the work of multiple vendors.

		Size Breakdown			Planned Release		
Object Type	***Count***	***L***	***M***	***S***	***R1***	***R2***	***R3***
Reports	3		2	1	1	1	1
Interfaces	4		4		2	2	
Conversions	3	1	1	1	3		
Enhancements	20	12	5	3	2	6	12

* note: CEMLI might be more familiar for those in this domain. RICE was chosen for the sake of brevity…

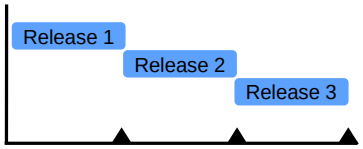
**014 And so, that single point in time status provided by the pie chart might play out in this way. We have thirty objects here. I'm using the acronym RICE for reports, interfaces, conversions, and enhancements. And those of you who work in this domain know that this is a somewhat older language. There's a newer term that's in vogue. I have a little footnote there. But RICE allows me to have a smaller table on the charts. So, I chose that one.

And so, these might be objects that are being created by a provider who will then supply these into a larger program of modernizing some IT system that we're responsible for. We might have data of the type depicted in this table here. And we would likely have much, much more about these things than is depicted here.

So, I just want to make sure you're aware of that. There may be three releases we're planning. And the first release contains eight items. And so, three of them are done. And five of them are in test now. And we may be looking at this pie chart for a status on how we're doing on this first release. We could see there are size breakdowns. And we can see how many of each type of object. So, we might have data of this type for a plan for an IT modernization effort.

Managing Three Planned Releases

Managing Three Planned Releases



Common Focus for Metrics

- Size
- Effort
- Quality

Goal:

- Predict release performance

Questions:

- Is the work larger/smaller than estimated?
- Is the work taking more/less effort than we estimated?
- Will the quality of the delivered products be acceptable?

Metrics:

- Estimated vs. actual effort
- Planned vs. delivered products
- Estimated vs. actual size of products
- Defect counts and profiles
- Measures of performance

**015 And breaking that down a little bit further, if we think about traditional ways that we might define measures, very handy, very useful goal question metric approach, we might have the high-level goal of predicting release performance, which plays out in a variety of ways. Our focus might be on size, effort,

and quality. Those are very common areas of focus for metrics. And so, the questions we would have relate to are things larger or smaller than we estimated. And who would have such questions? Would the general in the picture have such questions? We probably want to spare her the time that it takes to focus on that. Certainly, the provider working on this needs to understand have we sized things up well, and are we getting ourselves into a deeper pool than we'd anticipated. Or is it simpler or smaller than we'd anticipated? Those are things they need to understand.

They need to understand progress. And as we think about the level-- the rate at which things are proceeding, the captain in the center of that picture now really needs to understand in order to be able to convey information to the colonel who then puts this in context of a larger set of providers.

Will the quality of the delivered products be acceptable? That seems to be a concern for everybody in that picture. The kind of information that they would use to judge the result or the answer to such a question may vary, though. And so, we have some examples of metrics presented below there.

There may be measures of performance, just jumping to the bottom of the list quickly. There may be things about the capacity of the system, the speed at which

processing occurs, the access to diverse data sets required to perform the function. Such things would likely rise to the level of concern for the general if performance was lagging behind what was expected. But certainly, defect counts in profiles of which types of defects, those aren't metrics we would necessarily present to the general, and perhaps not to the colonel. And whether or not the captain needs to be concerned with these has something to do with the status of the program.

And so, with the traditional approach to decomposing from goal to question from metric, one path, and just one path through that process, might lead us to this set of information being sought. Many others would be of interest. And there are likely some that are unique to the context in which we're talking about adding metrics. And so, these may or may not fit you.

Understanding Benefit of IT Modernization

What combination of choices leads to improvements in things like:

- Amount of exception-handling
- Users finding the correct path through the system on the first try
- User migration to a new system

Can we iterate and experiment with functional changes as well as technological changes, to improve performance of the IT-enabled service?

**016 But one of the things that Agile processes bring to us, this picture of the plan-do-check-act cycle on the left, what we're trying to do when we're successful in using Agile methods is to shorten the time between when we have an initial concept to when we have a demonstrable capability potentially field-able. That cycle time is something we're trying to shorten. And at the end of the day, the product management focus we might have in an IT modernization setting might be more like the list of bullets we see here. The enterprise that is relying on this information technology, how much of the processing they have to do has to be handled through exceptions? What kinds of manual paths exist? And how often are they utilized versus the information technology is able to

encompass that part your concept of operation, your workflow. Changes in that performance are very, very meaningful to the enterprise.

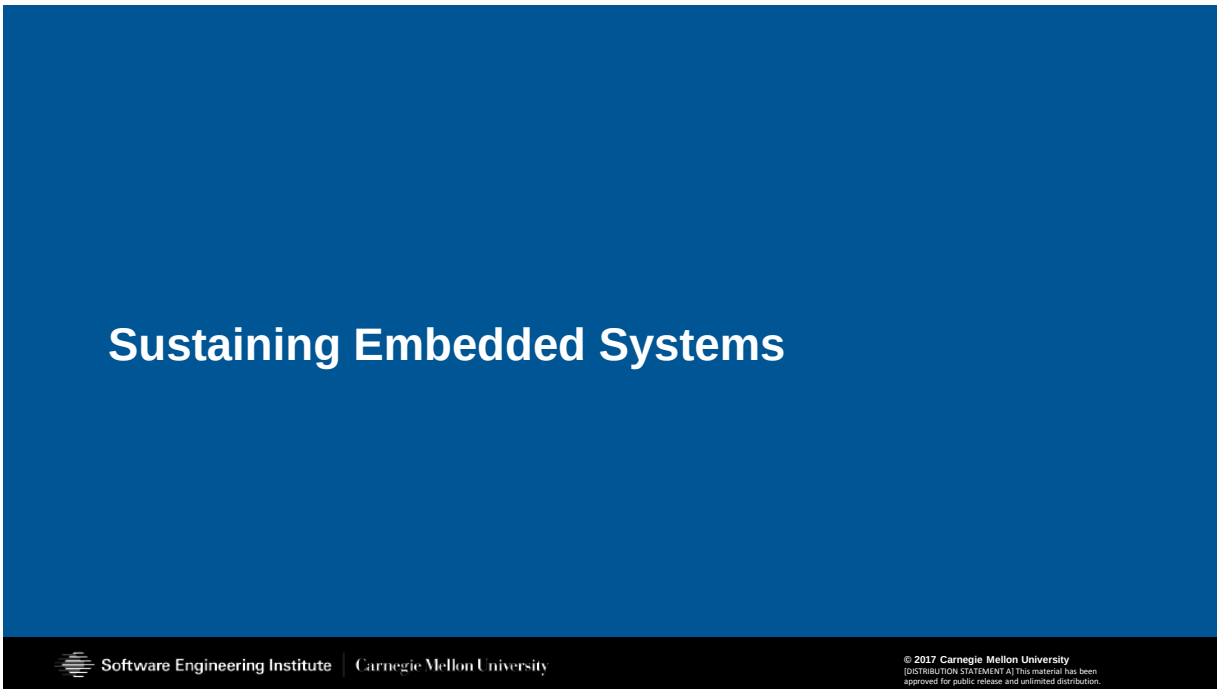
Users finding the correct path through the system on their first try, those are things that really can make the workflow work much more efficiently. You could think of those as requirements to levy on the software development firm. But there may be other considerations if we're able to adopt and be successful with Agile processes. As we think about migrating to a new system, the users' level of involvement in specifying and reviewing what that new system is might have a lot to do with their propensity to migrate to it.

And finally, one of the exciting things we're seeing in some of the clients we're supporting, a very powerful consequence of succeeding at Agile, whereas in the past we would have had a workflow that was defined and we would specify requirements for information technology to support that static definition of a workflow, now we're able to see iterative development and refinement of workflow, create a little bit of technology, look at how it works in the workflow, adjust the workflow to pursue a new opportunity, custom develop another iteration of technology that supports that newly defined workflow. And with a collaboration between functional, technical, and acquisition organizations, we're able to see this plan-do-check-act cycle play out in a

very different way. And so, in that setting, Agile is not merely a software development activity. In that setting, Agile is an approach to understanding needs and trying to fill those needs that plays out on a number of different fronts.

And so, this third slide in the sequence-- and there will be two more sequences. This third slide in the sequence is really intended to highlight the perspective that a government product manager might have and the opportunities they might pursue.

Sustaining Embedded Systems



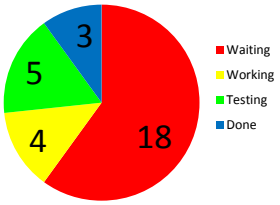
**017 So, we'll change to a different example now in the setting where we're sustaining an embedded system. An embedded system, it's basically a component of a larger physical electronic or mechanical

system that is created through software. And often, there are real time considerations that this is a system whose real time operation has fairly dramatic benefits or consequences to the mission that the enterprise is serving.

So, we'll take it to that pie chart of thirty things again.

Sustaining Embedded Systems Example

Sustaining Embedded Systems Example



These are 30 *Must-Fix Defects* which limit the operational utility of the system in the field today.

There is a strategy for patching the fielded system based on logical groupings of the defects.

Sample of Fields in the Defect Database

Name	Description
FindActivity	lifecycle or mission activity that uncovered the defect
FindDateTime	date and time when the defect was discovered
TestID	If found in test, the ID# of the test that exposed the defect
FeatureBlocked	user capability that does not function due to the defect
SysComponent	configuration item or other component containing the defect

**018 But instead of RICE objects, maybe we think of these as must fix defects, DRs, deficiency reports as you might know. And so, we might be looking at a snapshot in time with how far the team has gotten in resolving those thirty defects. Three of them are finished. Five are in test, four in work, and eighteen waiting to be started.

We have a table here that is just a small piece of what we might see in a defect logging system. So, we have things about the test that uncovered it or the event that occurred that led to us discovering this defect, when it was injected, and so on, and so forth. Many other things might be known.

Fixing Fielded Defects

Fixing Fielded Defects

Common Focus for Metrics

- Cycle Time per Fix
- System Availability/Function
- Quality

Goal:

- Timely resolution of known defects

Questions:

- How many defects remain to fix?
- How many defects have been fixed?
- How many fixes have been deployed?
- How many fixes had to be redone?
- How fast are we fixing things?
- What functionality remains blocked?

Metrics:

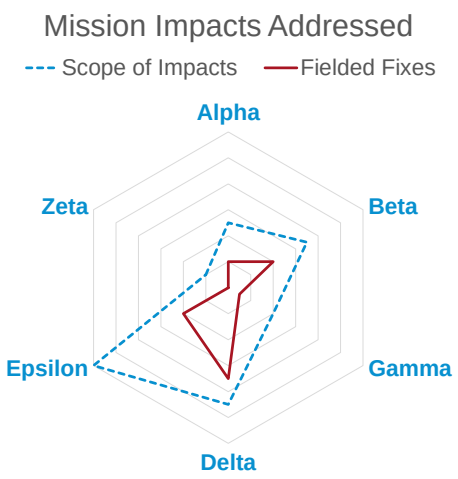
- Tally of defects remaining/fixed
- Number of fixes per month
- First pass fix rate
- System down time
- Revenue/mission loss due to quality

**019 Applying the goal-question-metric concept again with a different graphic on the left here. We might be focused on cycle time per fix, system availability, and the mean time between failures, the amount of down time, or what sorts of functionality are enabled or disabled because the defects still exist, and certainly the quality. And so, our goal might be a timely resolution of known defects.

We might have questions of the type we see here, often quantifying what the status is today, how today's performance matches against our plans from the past. And so, we might have a metrics along the lines of tallies, numbers that are fixed per unit of time, a first pass fix rate. So, how many times do we need to pull the patch back because there was a new error introduced? That's a very powerful metric we've seen in a lot of our clients. We might be very concerned about system down time. And that might have a lot to do with the choices we make in fielding these patches. And certainly, revenue or mission loss, threat to the enterprise that comes about from the persistence of these defects, these are all metrics we might need to know about.

Again, who in that picture needs to be thinking of these things? If the general is spending time watching the tally of defects that remain to be fixed or have been fixed, she's probably not able to focus on the much more important, much more broader, longer term things that she's charged to attend to. And so, was we think about the strategic and tactical differentiation, as we think about project management goals versus the goals of the enterprise that is responsible for this embedded system, we really need to consider what is the audience for different metrics.

Enabling Mission Threads with DR Fixes



The impact of fixing defects is charted for six (6) mission threads.

Looking at the area inside the blue dotted line:

- Epsilon has the greatest number of DR impacts
- Zeta has the lowest

Looking at the area inside the red line:

- Fielded fixes have benefitted Delta the most
- Zeta the least

DR = Deficiency Report

**020 And so, taking this to the product manager perspective, one example might be what we see here. So, this is a Kiviat diagram or a bull's eye chart as some folks refer to it. And what we've done here is to array on these different axes, these are six different axes, different mission threads. And so that's a way for us to understand the intention of the system's usage, the different avenues where utility is found. And we could think about the threat to succeeding in that use of the system as depicted in the different axes here.

The dotted line colored blue represents the number of defects that map to each of those mission threads. And so, mission thread-- the one that's labeled Epsilon has the largest number of defects in that pool of thirty that relate to it. Whereas,

the mission thread labeled Zeta has perhaps the smallest number. And so, those are the places where fixes are needed.

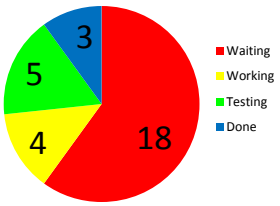
The line in the solid red now depicts where we are today, that status that we looked at from the pie chart. And so, we have addressed a largest percentage of defects that relate to mission Delta. But mission Epsilon that seems to have the largest population of defects tied to it has received perhaps less attention. And that may be quite intentional. Mission Epsilon might be something that has an alternative manual workaround that's tolerable. Whereas, mission thread Delta is an extremely time critical, mission critical thread which we can't afford to be disabled.

And so, as we look at the progress depicted in a simple pie chart, there's really a much more dynamic, much more involved story behind it. And as a government product manager-- I'm pushing that title I suppose, though that's not my intent, we might have concerns relating to the larger picture not just the performance of this software fix effort that's underway.

R&D Pathfinder Projects

**021 Now, to the third and final example using the same pie chart, there's this notion of a pathfinder project or a research and development effort. Sometimes, we see in major system programming offices risk reduction efforts that are chartered in order to quickly get to a proof of concept to make sure that a larger bit of work is feasible in the long run.

R&D Pathfinder Projects Example



These are 30 *requirements* to meet in order to establish a proof of concept for a new product offering.

A prototype satisfying most, if not all, of the requirements will be used to assess the potential market for the concept.

ID#	Priority	Requirement Text	Success Criteria
1	H	… text statements	… text statements
2	H	… text statements	… text statements
3	M	… text statements	… text statements
…	…	…	…
30	L	… text statements	… text statements

**022 And so, as we look at a simple pie chart, this may be a set of thirty requirements that, once implemented successfully, act as a proof of concept for something. In a commercial setting, this might be a new product offering. In a product line sense for a military command, this might be diversifying the range of mission capabilities that are available on a given platform. And so, quickly getting to a proof of concept, achieving knowledge point as we sometimes refer to it, is a very important priority in this sort of project. And so, we might say that three of the thirty are to the point where we have information on them. Five of them, we are proving them out, and we are in the final stages of testing. And four are in work. Eighteen are ready to be picked up. A

single snapshot in time, I'm going to remind of us all of that.

We might have data of the type vaguely described in the table here. We might have prioritized high, medium, low. Certainly, we would have ID numbers if they're stored in a database. And we might have descriptions of what the requirement is, what sorts of success criteria we're striving to achieve.

Building a Proof of Concept

Building a Proof of Concept

Common Focus for Metrics

- Requirements Satisfaction
- Test Cases Passed/Failed
- Technical Performance Attributes

Goal:

- Effective demonstration of capability

Questions:

- Is each requirement achievable?
- Which are the most challenging?
- How confident can we be about production feasibility?
- What are the bases for estimating total lifecycle cost for this product?

Metrics:

- Count (or %) of objectives achieved
- Number of business case questions answered
- Effort expended

**023 And then taking it to the next picture, what we're trying to do with this very cute graphic in the top left, we're taking the idea and expanding it into all the possibilities that it offers. That's the purpose of that image. Our focus might be on the extent to which, or whether or not we have satisfied individual requirements. We might have a set of

test cases that are tied to that success criteria that was on the table in the previous slide. And we could understand which test cases are presently passing, which are presently failing. In a test-driven development sense, we might be looking to manage progress in that way.

And certainly, technical performance attributes as we're looking at a proof of concept for a new capability, what aspirations do we have for how effective it is in light of the workflow, or mission that it needs to serve? And so, the goal would be to effectively demonstrate a capability. We might have questions about the achievability, which requirements are most challenging. We might have things that help us understand what would be the long term, total lifecycle cost relating to adding this product to our offering or adding this capability to a platform. So, metrics might be fairly simplistic in some cases, counting or percentage of the objectives that we've achieved, number of business case questions that have been answered, so how much insight do we get about feasibility, how much insight do we get about the range of utility for this new capability we're trying to field, and the effort expended.

And so, let's go back again to that picture of the different systems of people. Which of these rises to the attention of the general? Which of them are strictly the parlance of the captain? And which of them are really the business of the commercial

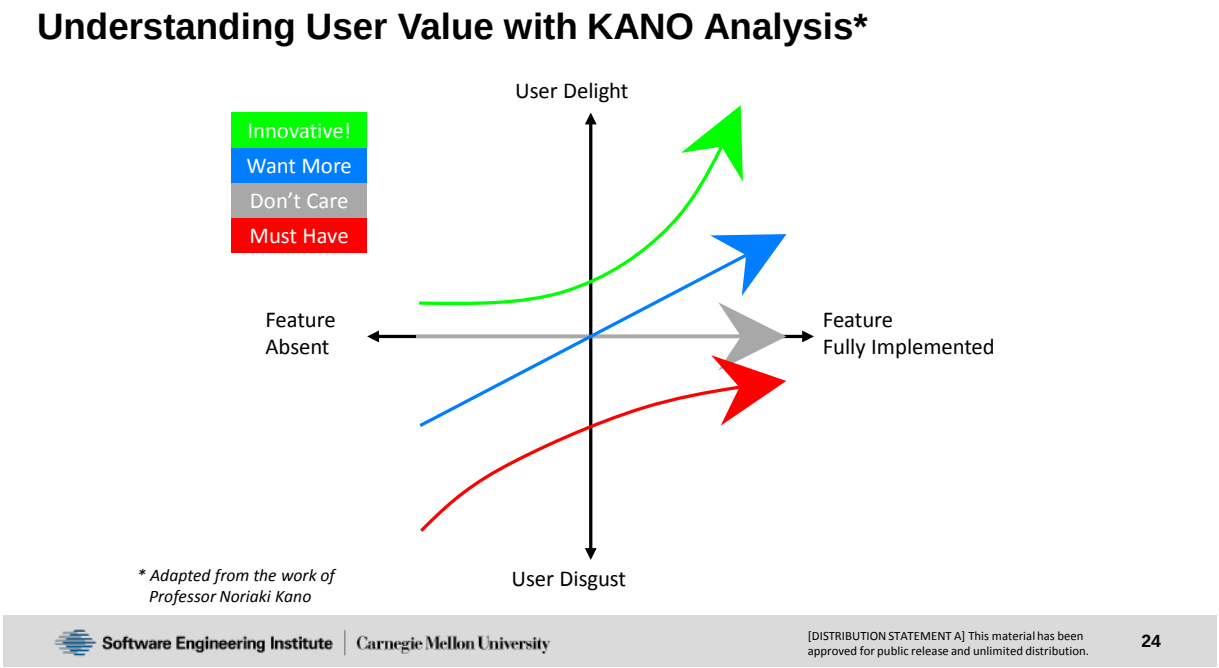
provider that is doing the technical work for us? We really need to think about that. And the answer, especially in this case, will be very different depending on the context.

So, some commercial providers for a government audience are very engaged in a product line approach. And so, when you go to them, you are looking for a new instance of something they have well-established experience and expertise in. And so, their level of interest, and their level of coverage for these concerns may be much greater. And your ability, as a government person, to get into that detail may be more limited because they're much closer to, although they're not actually, a shrink-wrapped software provider. Their level of competence in the market might be much higher.

In contrast, you might be in a setting where the government is really taking a lead to push the technological boundary. And they're having a much more active engagement in the process of doing the work, the process of understanding performance. And so, they may more likely be partners in that setting. And so, the kinds of roles, the kinds of people that would be interested in these things would be quite different.

And so finally, going to the product manager perspective.

Understanding User Value with KANO Analysis*



**024 If you've not heard of KANO analysis, I really encourage you to look it up. There's lots to be learned from the work of this Japanese professor. And the graphic here really tells a good story once you understand it.

So, depicted in the red arrow, this is something, when fully implemented, this user will accept your system as being acceptable. When those requirements are absent, as shown where the arrow begins, the user will find your system unacceptable. So, these are must haves. These are things that are barriers to entry if you don't have them.

In contrast, the green arrow shows things that wow, this is new. This is going to offer me something I didn't realize I would need. These are things that really delight users. And

so, investing in having those has a different impact on the utility and the perception of the system you're implementing and deploying.

And in addition to those two, we have some depicted in the blue arrow going diagonally that are nice to haves. We would like more of these. We would like more bandwidth. We would like faster speed. But at this point, where the floor is on-- well, I probably should not exaggerate that. In some settings, we have enough. And more is better. But it's not the same as adding a brand-new thing that really changes the utility of the system. And then there are, as depicted in the gray line that grows horizontally, there are things people don't tend to express a lot of concern one way or another about that are just there, maybe ho-hum.

With an Agile approach, you have the opportunity, on the government side, to evaluate what you're asking of the development organization in light of these kinds of differentiations. You might be able to invest in running focus groups or have access to a user community where you can understand the differences among the thirty requirements here in terms of what their role is in the perceptions of the system's users. And perhaps more obviously, as you do demos at the end of iterations, these kinds of differentiations of the work being done could be very powerful in helping you, as a government person, a person playing a role in a government program

office. It would help you to differentiate where your priorities ought to be.

So, make sure that you have the must haves in place before you start to say that you're done. Make sure that you leave budget for these really innovative things that are going to dramatically change the mission capability of the force that you're supporting. Okay, so those are three examples where we've tried to weave the same picture of thirty things. And I'm going to come back to that in a minute.

But once again, we're going to collect some data from you.

Polling Question #3

Polling Question #3

Which of the examples is the best match for your context?

1. IT Modernization
2. Sustaining Embedded Systems
3. R&D Pathfinder Projects
4. More than one of the above
5. None of the above

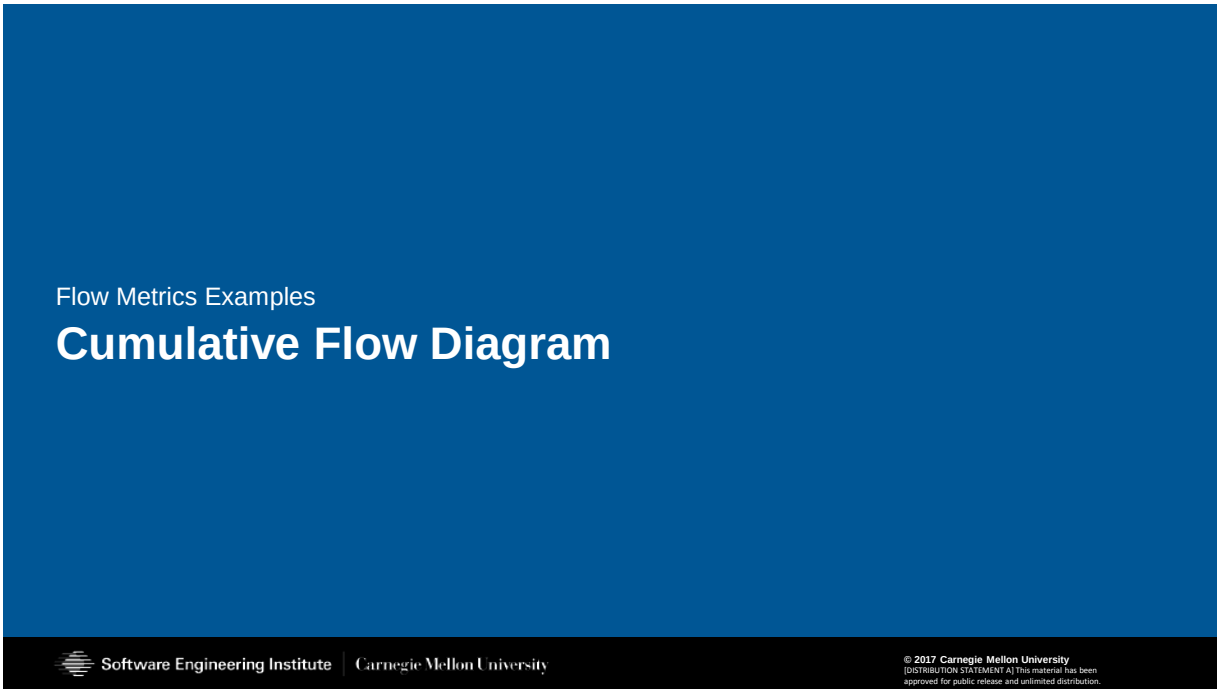
**025 And so, a polling question, which example best matches your context? This helps us understand how to tune the

message, perhaps. And so, we'll see if Shane's got a question to ask us.

Presenter: So, you are doing such a good job explaining everything, we don't have any current questions. So, we're going to give the people some time to file in this information to us. And we'll just keep running with you.

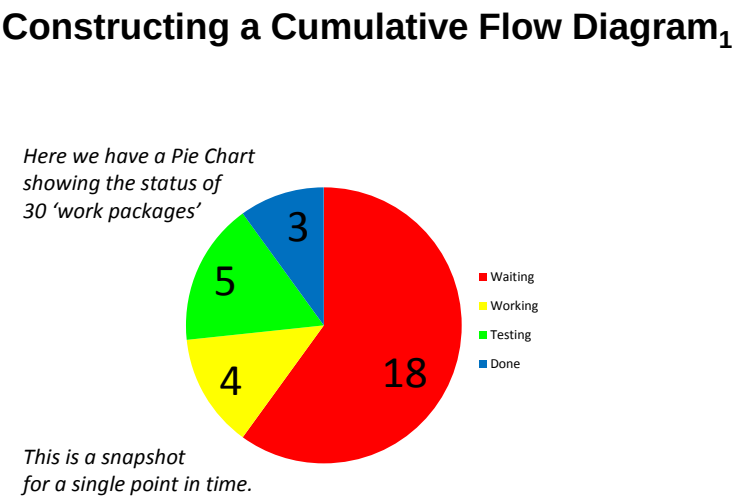
Presenter: Okay. So--

Flow Metrics Examples Cumulative Flow Diagram



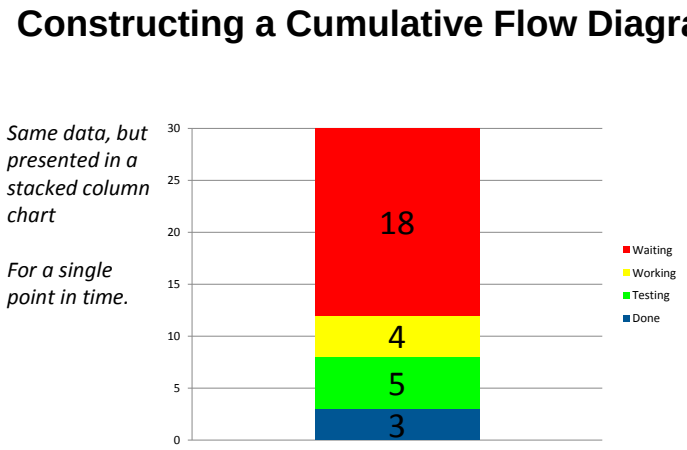
**026 Now, I want to switch gears and tie to that pie chart. But I want to talk to you about what a cumulative flow diagram is. And this is something, if you've got a lot of experience in Agile, this is something you've no doubt seen.

Constructing a Cumulative Flow Diagram[1]



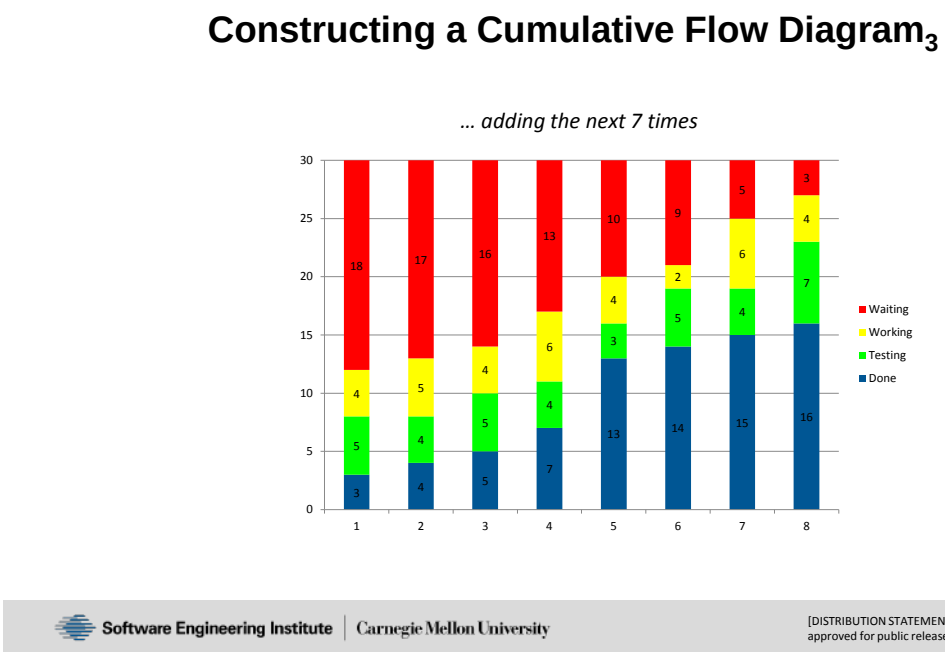
**027 So, I'm going to build to it though. So, here's that pie chart again with thirty items showing different status, a single point in time. And we're going to stay on the screen here.

Constructing a Cumulative Flow Diagram$_2$



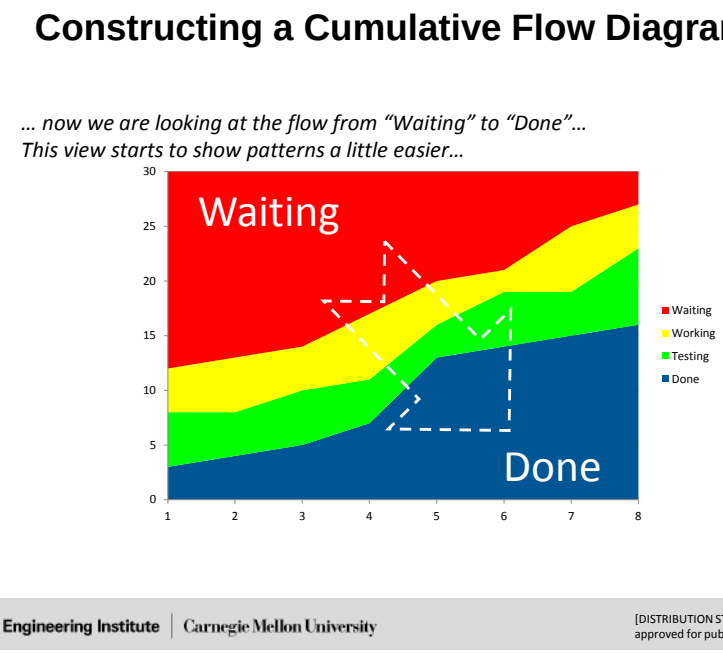
**028 If we take that same data and present it as a stacked column chart-- again, a single point in time, same numbers. We're going from waiting to be done-- waiting to be started to done, from eighteen down to three.

Constructing a Cumulative Flow Diagram$_3$



**029 If we add now a number of other points in time, perhaps this is eight different status reports that we've gotten, and we can start to see a pattern moving from upper left in red to bottom right in blue.

Constructing a Cumulative Flow Diagram4



**030 Now, we have a cumulative flow diagram. The cumulative flow diagram allows us to focus in on this band depicted here in yellow and green in the middle where work is being done, and understanding the flow of work over time. And there's a whole host of metrics that are associated with flow. And some of you in the audience may know about that. Let me check to see how we're doing.

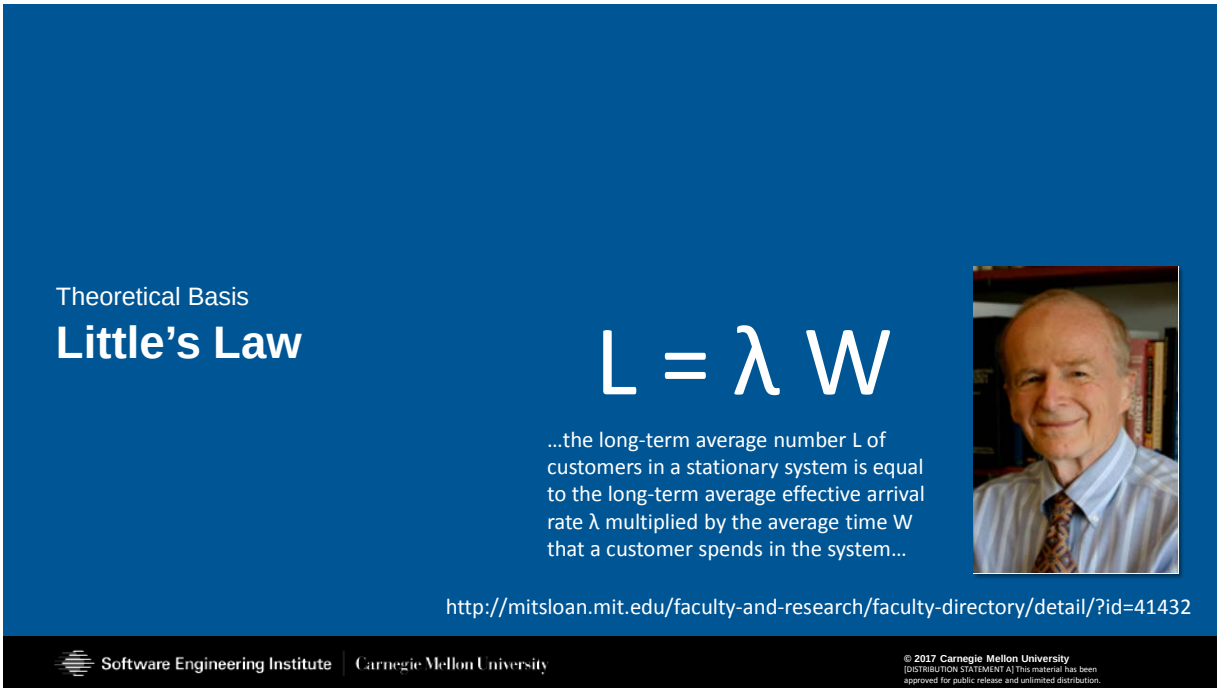
Presenter: So, it looks like, Will, the majority are putting in number four, more than one of the above.

Presenter: Oh great, okay. So, I'll have to talk to more than one example as we go.

Presenter: Yes.

Presenter: Terrific. So, this cumulative flow diagram has some really nice features.

Theoretical Basis Little's Law



**031 And they derive from something that may be familiar to many of you, out of queuing theory this law called Little's Law. And this photograph is from Dr. Little's staff page at MIT. And you see the URL there. Dr. Little is credited with being the first one to concisely and precisely define this law that people believe has existed. It's not necessarily a discovery, depending on who you ask. But he is the one credited with showing concrete proof that this law in a queuing system is something we can rely on.

And so, the law is shown there. I toyed with the idea of trying to quote unquote teach Little's Law in a

ninety-minute webinar where I had other topics to cover and thought better of it based on feedback from colleagues. But we're going to talk through some of it and reserve the rest for a more complete coverage in the context of a course.

So, one of the places you'll see Little's Law come into play for Agile metrics is a book by this gentleman, Dan Vacanti.

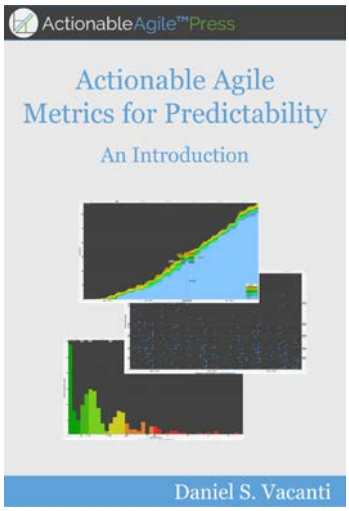
Little's Law in Agile Metrics

Little's Law in Agile Metrics

Three Metrics Emphasized*:

1. **Work In Progress** (the number of items that we are working on at any given time),
2. **Cycle Time** (how long it takes each of those items to get through our process), and
3. **Throughput** (how many of those items complete per unit of time).

** Excerpted from page 13 of the book depicted on the right.*



**032 Really accessible book. Little's Law and queuing theory may appear intimidating to some. But the way it's addressed in the Agile community, it's really much more approachable.

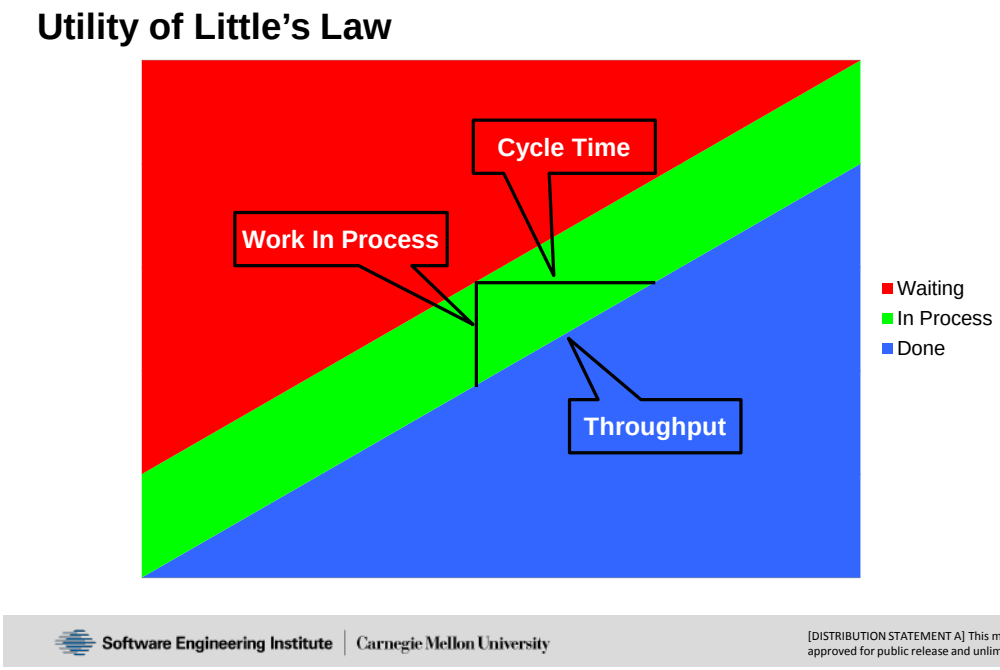
Fundamentally, what we're saying is that the average cycle time is equal to the average amount of work in progress divided by the average

throughput in that system. And so, if this law holds-- and there are a set of assumptions that are necessary in order for us to say that this law does hold. And there's a lot to cover in discussing what those mean for our setting. But if this law holds, what this is saying is because we know the relationship between these three things, if we know two of them, we can use that information to do the forecasting for the average value of the third one going forward. But that's only if we have evidence that Little's Law holds in the queuing system we're working with.

This is the kind of understanding of queuing theory that we use when we go to the grocery store. You all know there's a line where, if you have nine items or less, you can get through that line. And the folks that are shopping for the scout troop, and two carts full of s'mores ingredients don't go in that line. And so, the work in process there, and therefore the cycle time, that's going to be different. And so, how long does the nine items or less line need to be before you're willing to go stand in the other line with the scoutmasters? That's based on your understanding intuitively of Little's Law. Lots more to be said, but we'll leave it at that.

Okay. So, with work in process, cycle time, and throughput it turns out there's a lot you can do.

Utility of Little's Law



**033 So, let me tie this back to the cumulative flow diagram. It turns out those are things we can see rather readily in a flow diagram of this type. I've got an overly simplified one here where we just have one work in process state. So, the average height of that green band, that's the amount of work in process, or WIP as we call it. The average width of that green band measured horizontally is the cycle time, how long things stay in that queue. And then the slope of the line that leads us into the done state, that gives us the throughput, the number of items being completed per unit of time.

There are a number of other assumptions that we need to make before we take these values now and apply Little's Law. Some of them relate to the age of the items in work

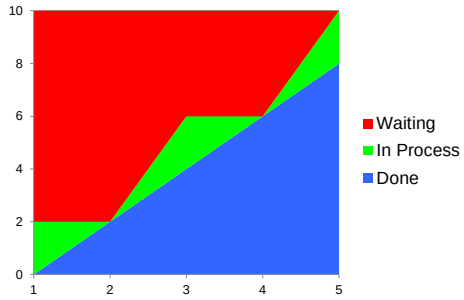
in process. It's presumed that they are neither inc-- the average age is neither increasing nor decreasing. It's presumed that everything that enters the work in process state exits. And those are things we can't see from the chart.

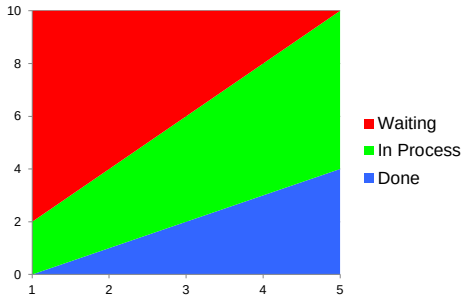
What we can see from the chart is that the rate at which things enter in process and the rate at which things exit in process are the same. And the amount of work in process at the beginning and the amount of work in process at the end is relatively similar, or in this case, I know it's identical because I created the chart. But the average WIP is the same at the beginning at the end. Those are presumptions.

If those things hold, then these values that you see on the screen turn out to be very, very powerful. And really the diagnostic use of this methodology comes about from observing violations in the assumptions that underlie Little's Law.

Exercise: What is Going on Here?

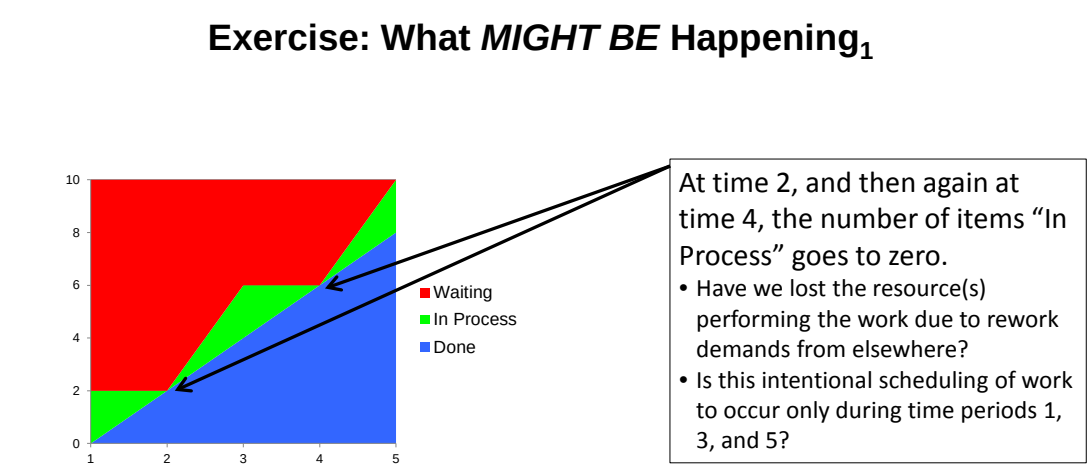
Exercise: What is Going on Here?





**034 So, let's show you a couple. So, on the left and right, we see two different examples of potentially non-predictable systems. And we would conclude that in both of these settings, it's obvious that Little's Law does not hold at least for the time period depicted here. So, what might be happening?

Exercise: What MIGHT BE Happening$_1$



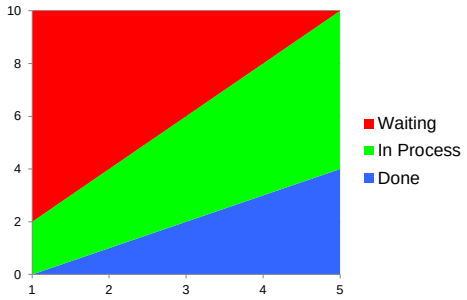
**035 The one on the left, the work in process goes to zero at two different points. And from an operational perspective does that mean that we've lost resources that are pushing stuff into in process? Or does that mean we have planned for work to go to zero at those two points because this is an activity that fills a gap and takes advantage of people's availability? In any case, we would conclude that using those parameters associated with Little's Law wouldn't be a basis for making forecasts for average future performance in that area. And it's pretty obvious why.

Exercise: What MIGHT BE Happening$_2$

Exercise: What *MIGHT BE* Happening$_2$

The number of items that are "In Process" is growing over time.

- The rate at which things enter "In Process" is greater than the rate at which things leave "In Process."
- Are people moving onto new items without completing their work?
- Are new resources being added, who start new work at each time period?
- Are things moving into the "Done" state quickly enough?

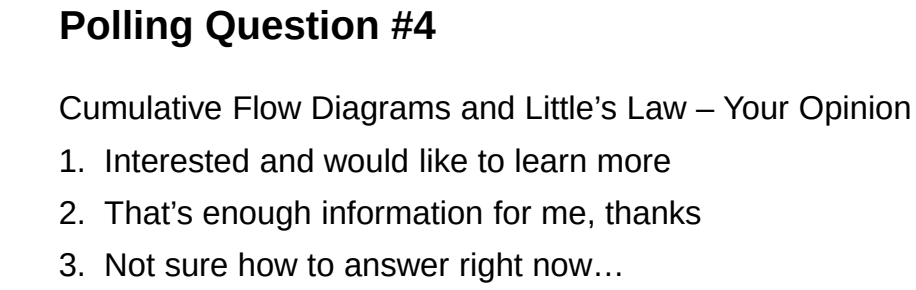


**036 Similarly in this case-- and this one might be more typical of what you'd see in terms of violations of Little's Law. Here we have the amount of work in process growing over time. The rate at which things enter in process and the rate at which things exit in process are not the same. Is this because we're growing the number of staff that are working in this activity over time, which is perhaps a good thing? Or is this because people were pulling tasks off the Kanban board, getting stuck, not able to find help, putting them aside, and starting new ones? I think that's pretty commonly thought of as an anti-pattern. That once you pull something and start on it, you really want to push it all the way through the end. And it's an exception when you have to stop.

And so, these are ways that we might use cumulative flow diagrams, overly simplified examples to be sure. But we've got clients who are finding ways to use this in understanding the influence of different events along the timeline shown in the horizon.

So, we have a polling question relating to flow metrics and cumulative flow diagrams.

Polling Question #4



**037 And basically, we're trying to gauge the audience's appetite for these things. And so, as people type in their response to that, any questions queued up?

Presenter: We do. From Roland wanting to know, "Should we develop measurement systems for our programs that measure business outcome achievements versus purely internal delivery metrics?

Presenter: So, I would think you would want both. I think the perspective you have in terms of your role in the enterprise governs what your keenest interest is. But understanding that you're able to operate in a consistently positive pattern, that is you've got delivery happening at a pace that the business needs, and you're not flooding the business beyond what they can tolerate in terms of delivery rate. So, we can tolerate much more in terms of updates on our phones than we can tolerate in terms of updates in our automobiles. We really would not wish to have our automobile software updated very frequently. It might cause us some concern, whereas we seem to be happy with it on our phones. And so, understanding that from the perspective of matching what's happening in the internal cadence with what the market appetite is, those are I think important to balance. We got anything more?

Presenter: We're looking at about eighty-five percent, if I had to guess, that are putting in number one. So, they're interested.

Presenter: Marvelous.

Presenter: Yeah.

Presenter: Okay. So, by popular demand--

Cumulative Flow Diagrams – Beyond Basics

Vacanti elaborates on Little's Law and "Flow Debt*" using CFDs.

Hyman Minski popularized these terms for types of debtors:

- *Hedge,*
- *Speculative, and*
- *Ponzi.*

Patterns of flow can help you identify which category best describes the prevalent decision making style in your project.

Ever been on a project that was trying to do so many things that none of them ever got finished? Is that a Ponzi project?

* Page 144

**038 I have just one more. And Vacanti's book really does a nice job of pulling in some clever reference to this bit of work in the finance industry from Hyman Minsky, who talk about different kinds of debtors. So, hedge debtors are people who can make payments that cover the principle they owe as well as the interest on that principle. And so, those are folks who diligently pay their mortgage like many of you out there. Then we have speculative borrowers who will maintain and pay the interest but hold off paying the principle because they're waiting for a payoff. And then finally we have Ponzi debtors who have to keep borrowing and really don't make progress on working down their debt. Their debt grows more and more.

As we think about debt, Vacanti's approach to talking about this is in terms of flow debt. And so, some of you may have friends who work on projects where the management regularly comes to you and says, "You had three projects you were splitting your time across last week. Well, we just had a big win in the market. We've got a new project that is going to take our company into the next century. Now, you've got five projects to split your time among." And the more of those visits you have, the larger the number of things you have to split your time across.

And pretty soon, nothing can ever escape that factory because we're too busy trying to serve all of the demands. We're trying to get so many things done. We're trying to work on so many things. None of them can ever be finished. And is that a Ponzi project?

Well, it turns out that if we have a system that we can analyze for flow, and we have a way of evaluating the feasibility of Little's Law in our setting, we can differentiate the kind of flow we have along these lines. And so, much more to cover in this, but I'm just going to tease at it a bit so we can make sure that we cover a number of other things. So, see if we got any comments from that.

Presenter: Just a comment from John saying, "Although Tesla updates their auto software at the same rate, more often than buying a new car." Just a quick comment.

Presenter: Yeah, so Tesla really is pushing boundaries here. And some of my customers in the DoD realm would like to go to Tesla and understand how they manage, especially with hardware, how they can manage such quick turns. That's a really big challenge. And so, you can imagine the business decision that went into Tesla's choice to push updates out in real time like that. That was a very careful and frankly very aggressive choice they made. And so far, it seems to be doing well for them. We're probably not going to do that with our fighter jets though, huh?

Presenter: Another one from Barbara asking, "Little's Law would only apply with a large pool of workers to smooth out absences. Do we need uniform Scrum practices across teams?"

Presenter: Yeah, so Vacanti has a nice video that addresses some of the misconceptions, not that you've misunderstood. There are things that you would think naturally follow from the pattern we've shown. And one of them that I've had a healthy debate with one of my colleagues is the size of the work items must all be the same size in order for this to work. It's a similar kind of concern as the staff available must be the same. You would imagine that when such conditions are met, it may be easier to judge that Little's Law does apply in our queuing system. But from a mathematical perspective, we need the assumptions to be met. And

however they're met is immaterial because my scale doesn't care. It's going to tell me how much I weigh. It is an objective fact. And so, as long as the entry rate and the exit rate are the same, and the work in process at the beginning-- the average work in process at the beginning and the end are the same-- I am ending up teaching Little's Law despite my decision not to. These things-- this law is robust to a variety of settings. But you could see that if you have a great deal of fluctuation in staffing levels and other things that are really unpredictable, it could certainly threaten the feasibility of saying Little's Law applies.

Presenter: And then from Roland chiming in, "Portfolio WIP," I'm assuming work in practice, "Is the biggest issue that I see in my clients, way too many simultaneous projects all competing for the same resources."

Presenter: Yeah, and so we're seeing some influences in the government settings where Agile is applied. People from a commercial background are coming in to help break log jams. And one of the things that's really a problem is the needs that we have for software driven capabilities are not diminishing. They are very rapidly and very broadly expanding. And so, it is a real struggle for us to rely on providers who already are doing beyond their intended capacity. We have very compelling needs for them to meet. If you think about the software

systems that are supporting the people who are experiencing severe weather issues right now, they rely on communication channels. They rely on distribution channels. These are really, really essential. And government personnel responsible for making those robust and more powerful have no end of appetite for serving their constituents better. Prioritizing, a government product manager's role in prioritizing is perhaps the hardest and harshest at times. You've really got to manage that flow and not overwhelm the system with demands. A really important point, thank you for bringing it up.

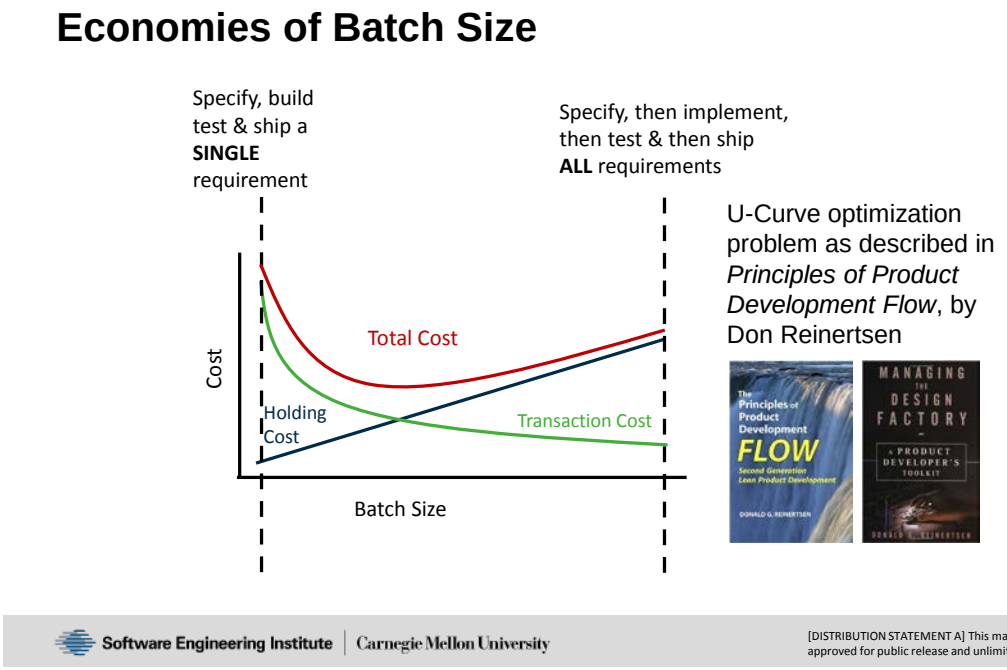
Presenter: We are caught up in the queue.

Presenter: Okay, great. So, let's move into a deeper coverage of lean economics.

Influence on Modern Agile Practice Lean Economics

**039 And this is a really neat topic as applied to software intensive systems. There's a lot of work in lean manufacturing. Much has been learned there. There are some authors that are bringing these into the knowledge work domain.

Economies of Batch Size



**040 And so, this is a graphic that's really influenced my thinking a lot in considering how Agile methods work in the settings where we're familiar, government settings, ecosystems with a great deal of complexity and a lot of different players. This U-curve optimization problem as depicted here posits that the total cost of the work you're doing is a composite of two different kinds of cost. One is called holding cost, and the other is called transaction costs. So, transaction costs are, for us, much easier to understand. That's the cost of getting the work out the door, the people who need to do the work, the test frames, the test cases, the environments that need to be used, what it takes to do the work.

And if we think about doing work in larger and smaller batches, we tend to favor doing work in larger batches because the infrastructure that we want to use can be amortized more effectively if we take that approach. And so, if this test frame can be shared by a larger number of people or can be used for a larger number of tests, then we're really doing well to define a robust capability that has a broader use.

The really tricky part for us is the holding costs. Holding costs, for us, are the fact that software requirements are a perishable opportunity to satisfy a user's need. I need you to think about that carefully because it's fairly profound in our setting.

If you think about automobile manufacturers or people who produce hardware, holding costs are the warehouse you need to store the parts and the sub-assemblies into. And so, you need to pay for the warehouse. You need perhaps hire security to watch over it. But even more, you have to make sure that you don't put too much in there for fear that the sub-assembly that you use now will be obsolete in next year's model. And you will have a lot of excess parts sitting there in a warehouse offering no value. That analogy brings us a little closer to the software domain.

The problem for us is it's hard for us to see holding costs. In fact, we've grown accustomed to, we've

habituated to, the reality that requirements change. And we often say, "We need to develop and deliver software faster than the user has an opportunity to change their minds." And that's because we tend to work in very large batches. And so, we don't see holding costs because those are carried around in the minds of the engineers who think they have a solution to a requirement not realizing that technological evolution is conspiring against them rendering that requirement obsolete before they have a chance to implement it. And so, as we think about the size of the batches we're working on, we're really relating that to the amount of time, the delay that occurs.

So, to make the point a little more clearly, let's show a couple of contrasting examples. If we decided to implement, that is specify, build, test, and ship, a single requirement, well the chances are much better with a single requirement that it wouldn't be overcome by technological evolution. And if we could get it done in a week or two, we might be able to even keep up with the fickle preferences of a user base. But the amount of investment we would need to make in creating the infrastructure in which we could do such a thing, to simulate the rest of the system without building it, is actually quite infeasible in most settings. We wouldn't choose to do that.

In contrast, if we specify, then implement, then test, and then ship,

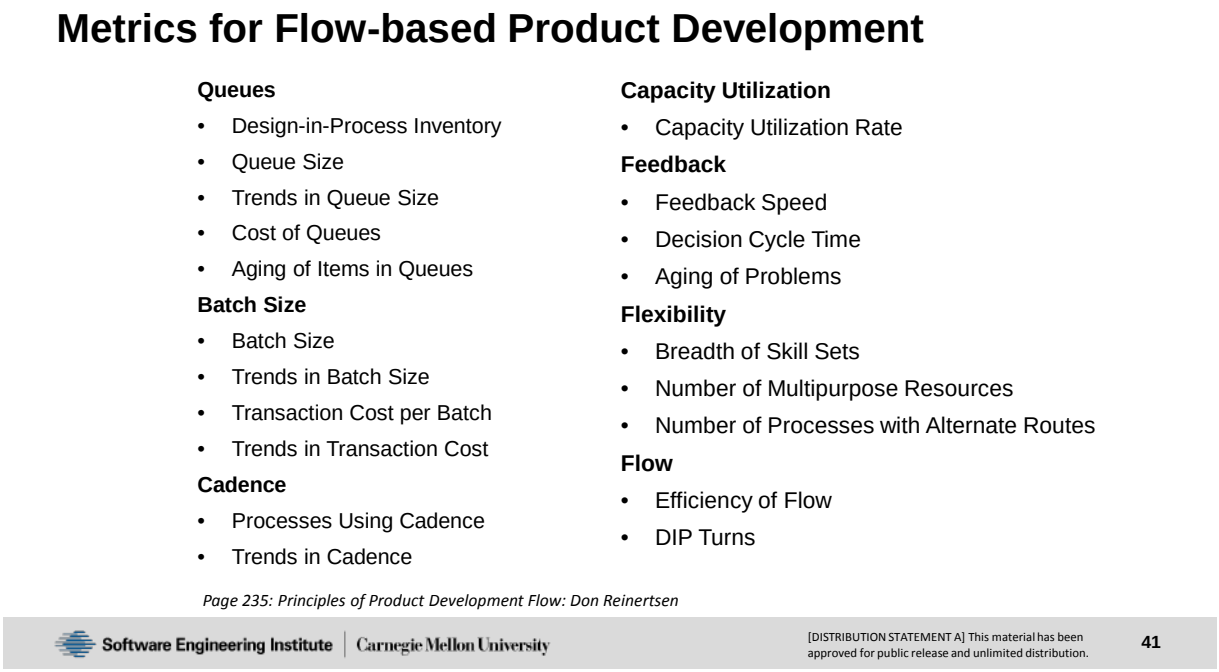
all requirements, well we've done the best job of amortizing our costs across the infrastructure. We perhaps could make the argument that that's a more efficient approach to take. But we've also increased the likelihood that some of those requirements are no longer germane to the operating environment where we want to ship the capability. In fact, this timeline reflected as a batch size for a proxy here, this timeline is measured in government settings in years, not weeks and months. It may be five or even ten years before some capability is realized. And to cause people to wait for that capability that long, we're inviting change in the demand as we do that.

And so, the argument from leading economics here is that we need to move toward smaller batch sizes. And this is certainly resonant with what we see for a team level processes like Scrum, XP. People are focused on doing things in smaller batches.

The author that many of us go to for this kind of coverage is Don Reinertsen, who-- two of his books are depicted here. "Managing the Design Factory," the one on the right is perhaps where you want to start. But please don't hold back. Go into the "Product Development FLOW book." Some of my friends make fun of me for having read that multiple times. It is a very rigorous treatment of issues of the type we are only alluding to with the graph on the left. And it covers things in a scientifically sound way and really gives a

methodological foundation for many of the choices we make in designing Agile processes and therefore the metrics that we'd want to have.

Metrics for Flow-based Product Development



**041 And Reinertsen offers a number of other examples. And I've got the page number referenced here. If you want to pursue this further, I really encourage you to pick up those books and Vacanti's as well.

Polling Question #5

Experience with flow-based metrics?

1. Never heard of it before
2. Yes, I've read about it or seen it before
3. Yes, I have used them in my own work

**042 Now, we've got one more polling question. And this is your last opportunity to type in a response to a poll. So, flow-based metrics, heard of them? Read about them? How many of you are using them? We'd really be keen to know how popular they are out there.

Presenter: And while you're doing that, folks, just a reminder that we ask that you do fill out the survey upon exiting today's webinar. I will add that URL back into the chat. We appreciate your feedback on the new platform and how we can improve things. Also, some people asking for the location of the slides, if you scroll to the top of the chat, you'll see a link that will open a new tab on the SEI website where you can download a copy of the presentation slides.

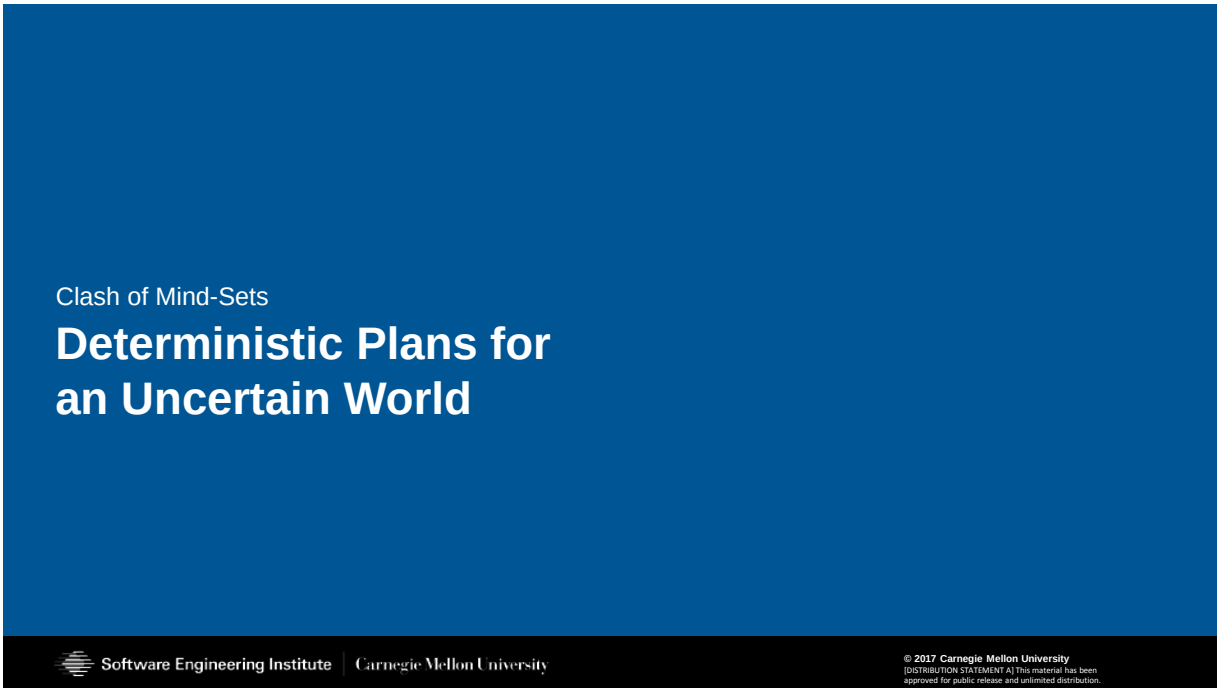
So, it looks like we're getting responses all across the board so far, Will. But it looks like number one is our--

Presenter: Okay.

Presenter: Most prominent answer, but there is quite a mix of two and three as well. But one looks like the prominent answer, never heard of it.

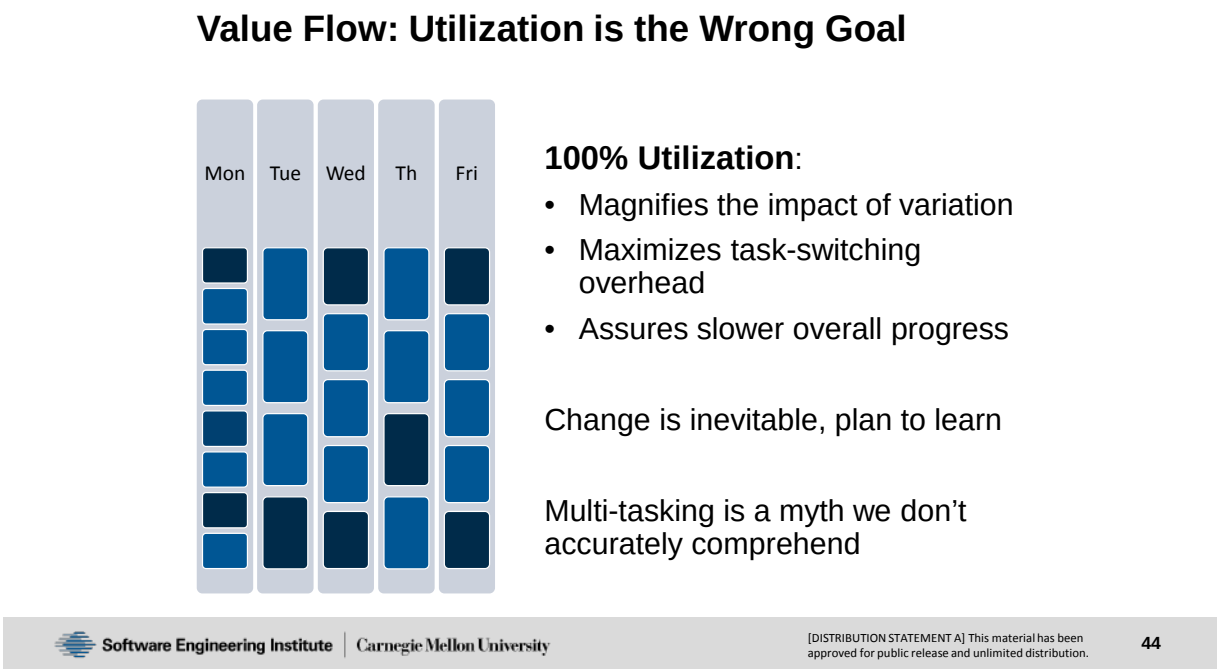
Presenter: Great, great. So, we've got an audience that perhaps can consume some of the information we're preparing. In the Agile in government team, we try to maintain our own backlog, try to use community input as a way of prioritizing. And so, your responses to polls like this really do help us. So, any questions queued up?

Presenter: No questions. But again, now I'm seeing lots of twos coming in as well. So, we're getting a mix with the poll question. But we'll kick it back to you.



**043 Presenter: Great. So, let's now turn to a conversation about mindset. And a deeper exploration of what determinism is. And even the term stochastic is one we're sparing you from. But that's something we'd certainly like to cover in more detail in a course. And so, let's start with this picture.

Value Flow: Utilization is the Wrong Goal



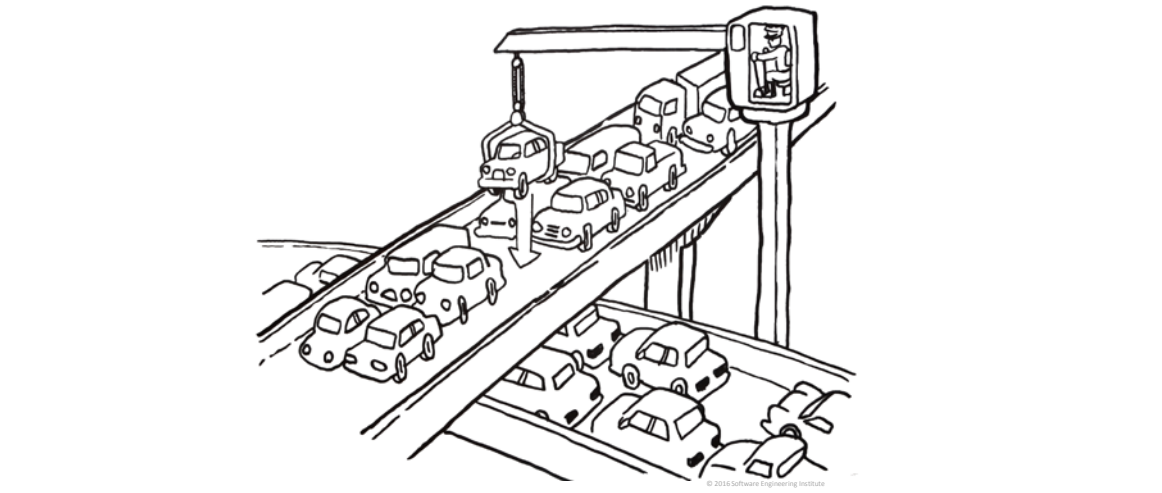
**044 So, many of you, if you open your Outlook calendar or whatever calendar you prefer to use and looked at your week, it might look like this, that each of these colored objects is a meeting or a working session you're obliged to attend. And you are scheduled wall-to-wall, morning, noon, and night, each day of the week, and perhaps into the weekend for many of you. And what happens if the first meeting Monday morning runs late? Do you end up paying for it all week? And so, when we schedule our work to this level of finality, this level of fidelity, we're really setting ourselves up for a difficult situation to manage.

One of the realities that we see in very large-scale government programs is when things start to go wrong, we turn up the volume on

and we turn up the resolution on what we're looking at. And so, we have these very large programs that are estimated with millions and billions of dollars and months and years at least. And so, we rely on estimation algorithms that recognize the variation that occurs at the lowest level. And some things take longer. Some things take shorter. But we don't expect there to be a strictly algebraic relationship between the fifteen-minute tasks individual engineers do and the total budget measured in millions.

However, when that total budget that runs in millions starts to be a concern, we turn our attention to making sure the fifteen-minute tasks are managed and that we have no gaps between the fifteen-minute tasks. We're trying to fill in to make sure we're utilizing to the maximum extent. And we don't really know, I would argue, whether or not we're overcorrecting, whether or not our tinkering with those variable processes at the lowest level is really causing the system overall to go out of balance. It's as if we are standing by the side of a very busy highway observing that there are thirty-five extra feet of pavement between two cars and insisting that we insert another car there.

Maximum Utilization is Counterproductive



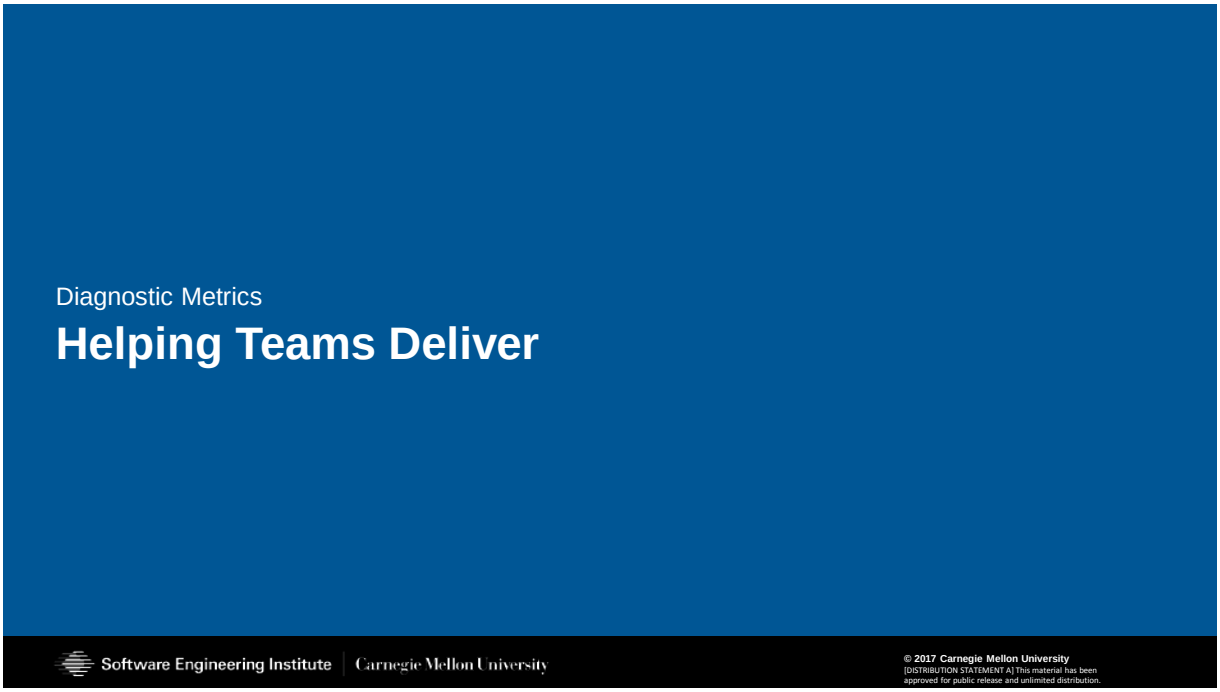
**045 None of us looks at this picture and says, "This is a way to help everyone get home faster."

Each of us has the experience of sitting there in one of these. Well, some of you in Los Angeles more so than others perhaps. The possibility of being able to have a smooth ride home when you're constantly moving your foot between the gas pedal and the brake pedal, between one project and another that you have to do, we know that things go slower. They necessarily go slower when we make them this crowded on the road. It's the same with the way we manage our time.

And so, if the contracts that we let, the metrics that we ask for, reinforce

this view, we may in fact be working against ourselves. And this is an insight that comes about from a lot of good Agile thinking and the authors that we have out there. Something we could have always known but we'd not really thought about it this way because many of our metrics focus on planned versus actual, not planned in the context in which we need to work. When we ask an individual engineer to estimate the size or the duration of a piece of work, they're not considering how busy the highway is, how much other work they're doing as they're doing that estimation.

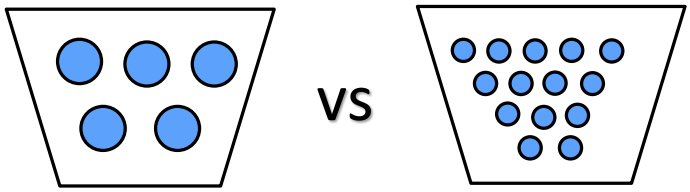
And some of us try to compensate for that by padding their estimates or by having some sort of informal transformation that tries to account for this. But this phenomenon, this things that slow flow, this is not actually measured. And so, the reason we emphasize cumulative flow diagram and the reason we talk about Little's Law in the context of Agile metrics is because that is a very good way for us to access, in concrete fashion, this phenomenon. Having that lens as well as the lens of individual objects that we're working on, how big are they, how long do they take, the confluence of those gives us a much richer view of what it is we're working.



**046 So, let's think about diagnostic metrics that we might use to help teams think about how they're delivering.

Batch Size Analysis – Story Size Focus

Batch Size Analysis – Story Size Focus

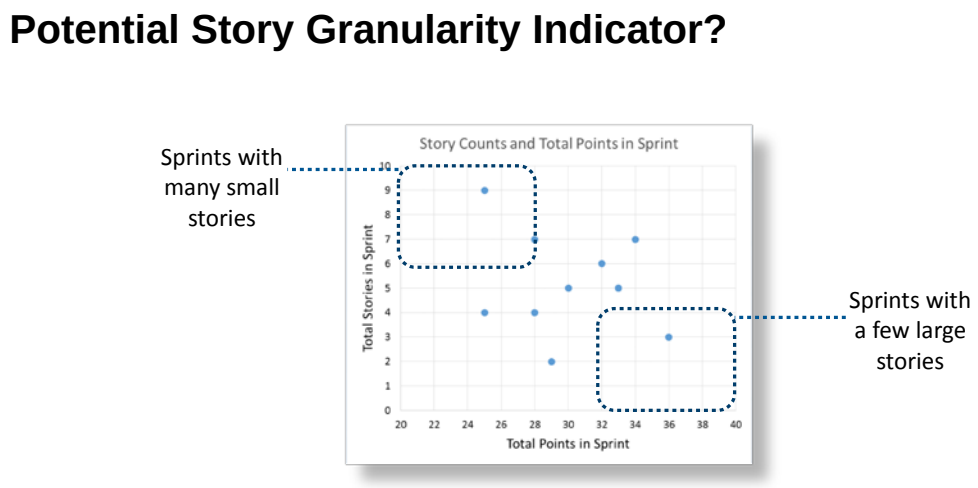


Splitting stories requires engineering judgment

**047 As I said earlier, batch size is a primary consideration in software development, in Agile software development. So, we're trying to get from this picture on the left more like the picture on the right, working a large number of small things as opposed to a small number of large things. But it's not so simple as to rip the specification, and you take pages one through three. Shane takes pages four the seven. And I'll take eight through ten. We can't arbitrarily split things up. This is an engineering consideration. This is not just about who writes which lines of code. This is about what is the minimum viable version of this. What do we need to have in place in order to show that this is working? And what is a reasonable boundary that allows us to rely on interfaces as opposed to having to have a tightly coupled

product that we're looking at. And so, as we think about a story-sized focus, there may be ways to diagnose.

Potential Story Granularity Indicator?



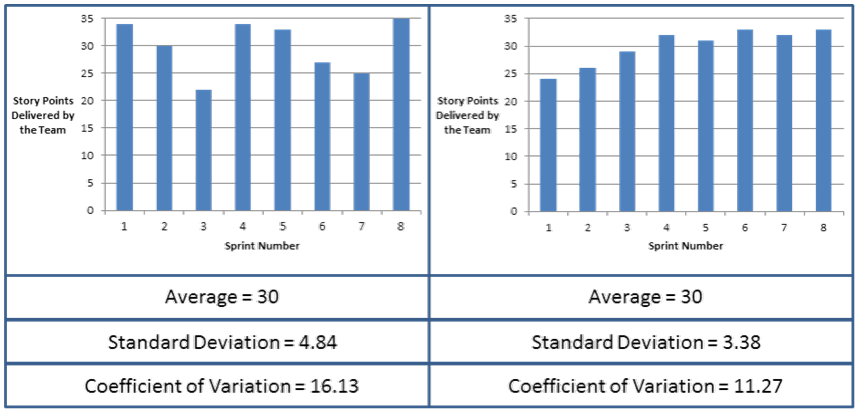
**048 And so, this graphic is intended to show many different sprints. So, each dot inside the scatter plot is a particular sprint that a team completed. Along the horizontal axis, we have the total number of story points. And it doesn't matter what scale, same team, same rough period of time. On the vertical axis, we have the number of stories that are in the sprint. And so, we might be able to identify, hey, here's a sprint where we had a few large stories. In contrast, in the upper left, here are sprints where we had a large number of small stories.

I don't propose to create a particular threshold for you to abide. That may

be something a team decides to do. I don't propose that you should have this is the maximum story point size recommended, although teams do choose that. There are many implications for how you could get to a smaller or why you must stay to a larger depiction of the capability. But understanding which extreme you are in on any given sprint, especially if we have data of this type, and then we can go and look at well the results from that sprint depicted in the lower right, are there undesirable outcomes that have occurred. And can we trace that cause back to the fact that the stories weren't decomposed to a lower level? And so, we were relying on a smaller number of people doing a longer piece of work versus people who could finish rapidly and get feedback and pivot to address work differently the next time. And so, this sort of diagnostic look at how work is flowing through your system could be beneficial.

Coefficient of Variation – Analysis of Velocity

Coefficient of Variation – Analysis of Velocity



Average = 30	Average = 30
Standard Deviation = 4.84	Standard Deviation = 3.38
Coefficient of Variation = 16.13	Coefficient of Variation = 11.27

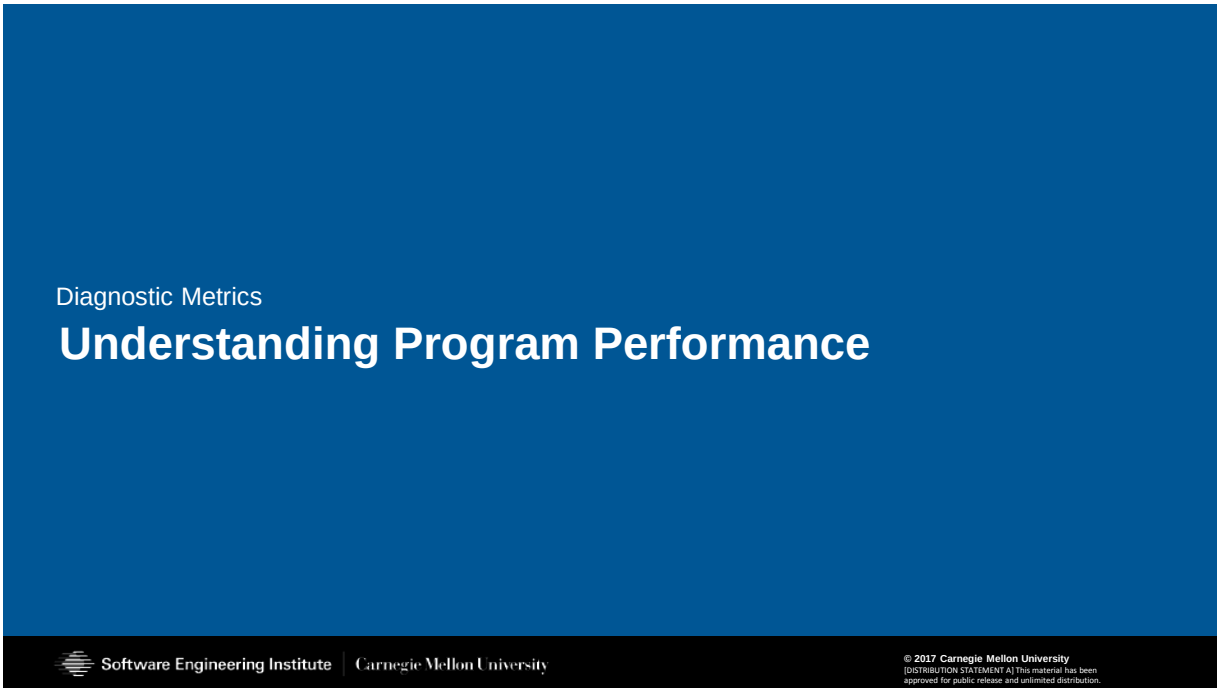
**049 Similarly, we might look at the performance of maybe the same team across two different periods of time, or maybe this is a comparison of two teams. It is a fictitious example. So, it needn't be one or the other. But what we're depicting here is a series of eight sprints by two teams, or the same team on two occasions, where the same number of story points were delivered. But the experience of the team from sprint to sprint was different.

On the left, we see a rise and fall of velocity, that is the story points delivered per sprint. And there is something that occurred in the third sprint that led that bar to be much lower than what occurs on the next sprint in sprint four. Whereas, on the right, we see a much more consistent pattern compared to the left of velocity.

And so, the teams or the team having these experiences might feel that the experience on the right is a much more predictable, much more steady reliable pace of work that I can count on. Whereas, the depiction on the left, there are things that occurred at different times that caused us to deliver less. Maybe it was intentional. Maybe people were absent. Maybe there was more challenging work. But in any case, as a diagnostic measure, we might say that a team would aspire or prefer to have the pattern on the right because it's a much more predictable way of working.

Much more needs to be understood before we can make conclusive statements from this type of thing. But this view into performance could be a valuable view.

Let me see if any questions, can you?



**050 Presenter: Just a comment from Roland earlier asking-- saying, "DevOps automation can help drive down marginal costs of deployment toward zero, but of course there are substantial investment required."

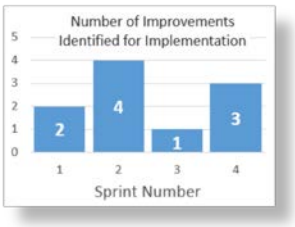
Presenter: That's a really keen point. And I should have made it when I was talking about the U-curve optimization because with DevOps, you're really changing that transaction cost curve. You're making it less expensive to quickly deploy. And so, you can actually entertain increasing batch size. Certainly, you can more frequently push code to commit. And so, very strong point, thank you for bringing it up.

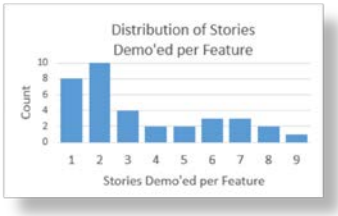
Okay so, understanding now at a program level some ways of doing diagnostics. And I have a scattershot of examples here.

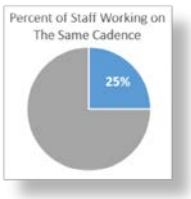
Indicator Examples1

Essential Process Attributes

- Cadence
- Synchronization
- Short Learning Cycles
- Reduction in Batch Size
- Iterative and Incremental Delivery







**051 If we think about the empirical process control, when I went to Scrum master school with Jeff Sutherland, one of the things he emphasized a lot was we manage through observation and experimentation. And so, it's important that we have feedback from the work that we do. And some of the priorities that allow us to rely on that feedback relate to cadence. That is a steady pattern of this is the duration for our iterations, or these are the periods at which these types of activities happen. Having that as a metronome to rely on is very valuable.

Synchronization, when multiple teams need to contribute together to build a product, then having more frequent synchronization points, continuous integration being a chief

illustration of this, allows us to have a more predictable outcome, shorter learning cycles, reduction in batch size, which we've talked about, and making sure that we're delivering incrementally and doing work iteratively.

So, if we look at the graph on the top right, that shows, for this program, how many improvements that come from our retrospectives have actually been deployed. So, this is beyond the lessons learned report. This is the lessons implemented, things that we have decided to do. If we don't see a learning process going on, if we don't have time to improve on the way we're working, we're not benefitting the approaches that we are espousing to use.

Bottom left, this is kind of an interesting thing. I've not yet seen this in operation. I'm putting it up here, perhaps as a trial balloon. If we think about the way we decompose features into different user stories, the more spread out those user stories are across demos, the more user input we're getting on the feature. So, if all of our features are implemented in a single sprint, then we have just that one sprint demo to get feedback on how we did. Whereas, if we're able-- and there are many things to consider in whether or not this is even feasible. If we're able to demonstrate little pieces of that feature over time, our understanding of what the user's true needs are can grow more readily than if we only have one shot at it.

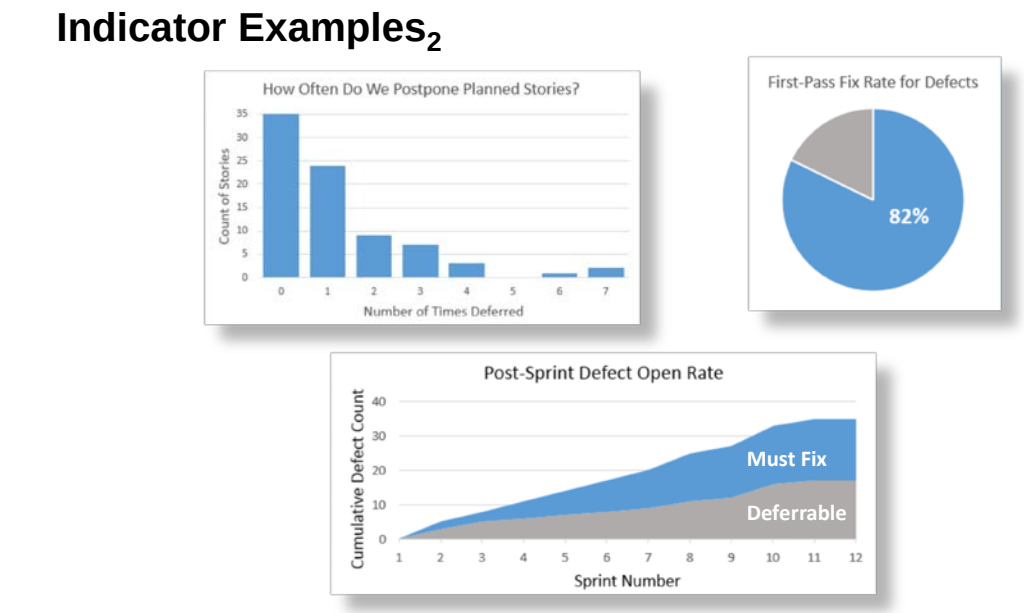
And so, stories demoed per feature is what we're showing there on the bottom left.

And then the pie chart in the middle is what percentage of the people contributing to this program are operating on the same cadence. And therefore, how frequently are we able to synchronize everyone? And how completely does that synchronization occur?

And the on the right on the bottom is the distribution of story point sizes. So, have we got a program that's largely composed of very large stories? Or are these things decomposed into smaller pieces? And does that help explain the rate at which progress occurs? Does that help explain, perhaps, dramatic or the absence of dramatic blockers that keep us from making progress? These are things worth looking at to diagnose what's going on.

And three more to show you.

Indicator Examples2



**052 And so, top left is how many times we've deferred something that we have taken into the work in process queue. So, the tallest bar on the left shows stories that were implemented the first time they were committed to. That little bit of a bump out at seven shows three or four stories that were deferred as many as seven times. Why is that? That's worth understanding. Are we waiting for GFE to come in, and we keep missing the window for demoing this because the GFE keeps slipping? Not to blame the government.

And then top right, we have the first pass fix rate. And the on the bottom, we show defects being discovered following the close of a sprint, during subsequent integrations perhaps. Some of those defects are maybe

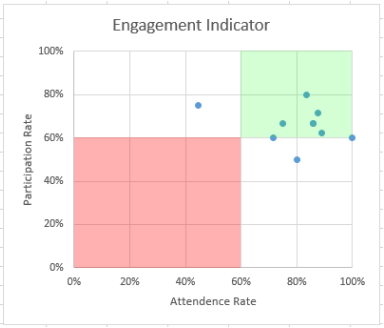
cosmetic and things that don't threaten mission. And so, they are deferrable. And others are must fix because the next bit of capability we want to add to the system can't be demonstrated until this blocker is fixed. Tracking the growth of that hill over time as well as the differentiation between must fix and deferrable, those could help us understand program performance.

Adopting New Approaches Assessing Engagement

**053 Finally, as we think about helping organizations to adopt and gain benefit from Agile methods, those of you who are coaches, this is something we've emphasized a lot. And we've seen a lot in a literature that engagement is what we're after. We want folks to be motivated and involved in the work we do. Yet, we don't have readily available metrics for it.

Simple Indicator, Powerful Analysis

Ceremony	Expected	Attended	Engaged	%Attend	%Engage
1	5	5	3	100%	60%
2	7	6	4	86%	67%
3	8	7	5	88%	71%
4	4	3	2	75%	67%
5	9	4	3	44%	75%
6	6	5	4	83%	80%
7	9	8	5	89%	63%
8	7	5	3	71%	60%
9	7	6	4	86%	67%
10	5	4	2	80%	50%



Subset/aggregate data to look for trends across:

- Particular event types
  - Are ‘standups’ not working?
- Pockets of staff
  - Have we alienated ‘release managers’?

**054 So, a very simple one we proposed and were using with a client of ours, what we're looking at here is a spreadsheet I had on my desktop. And ten different ceremonies, that's why there's ten rows there. So, those might be daily standups. They may be demos, retros, what have you.

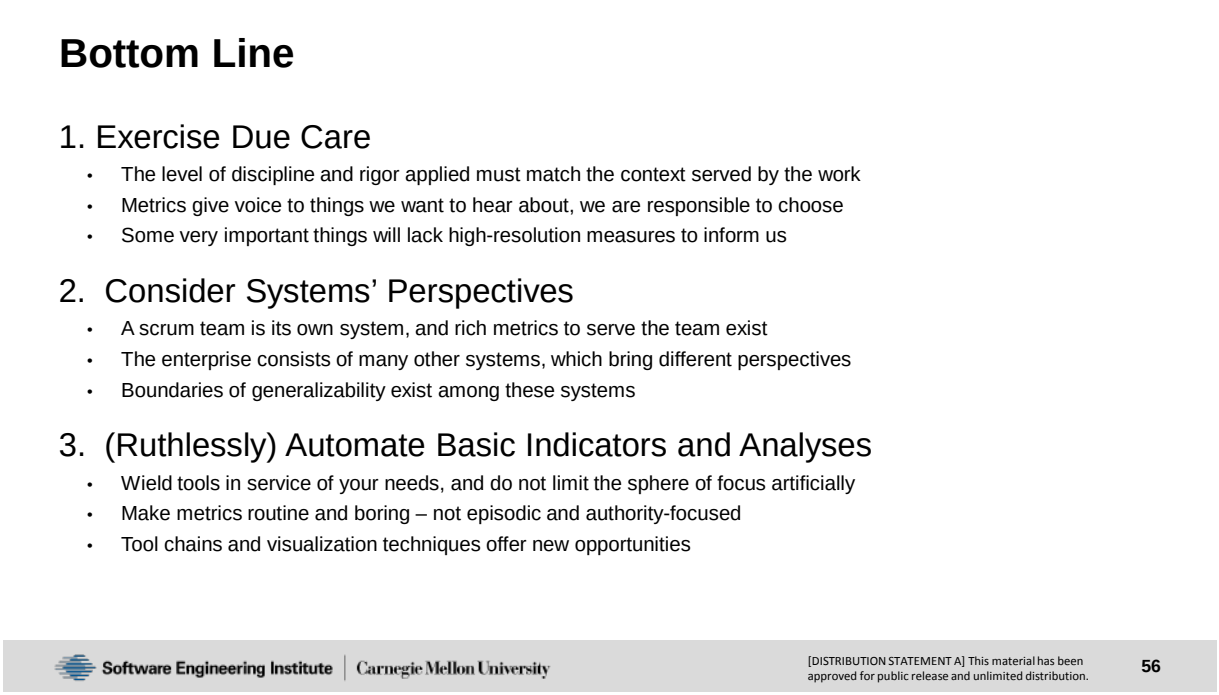
The number of people we were expecting to attend that event is shown in the next, and then the number of people who did attend, and then finally a judgement from whoever was running that even as to how many were actively engaged. And by arraying a scatter plot, as we see on the right here, with attendance rate on the horizontal and engagement on the vertical, what we want to see is full attendance/full engagement. And so, we want to be in the upper right quadrant in the

green area. And we want to stay out of the lower right quadrant-- the lower left quadrant where we may have lagging attendance and those who do attend are not as engaged as we would like them to be. If we can use coaches to track this over time and understand how our attempts to influence their level of engagement affect this performance, this could be a valuable indicator.

In Closing...

**055 So, I think we're doing great on time. I've got a little bit for conversations at the end.

Bottom Line



**056 So, in closing, the bottom line, exercise due care. You take on an affirmative obligation to focus your metrics in a direction that yields the information that's needed for the roles you're serving. We need to match the information with the right roles and responsibilities in this system of systems that we're operating. And finally, we need to try to remove the drama from metrics. Metrics should no longer be a defensive. Can I show these data? And now, go back to work. Metrics should be just as my bathroom scale, information that we use for a mission that we're on.

Presenter: So, Marcella wanted to know-- I know we can't endorse tools, but asking, "Is there a tool, commercial or otherwise, that can help organizations implement these metrics?"

Presenter: So, there's a lot of conversation about tools. Alex Yakima had a very nice view on it. One of the things he suggested was if the tool seems to reinforce the notion of hierarchies and seems to try to match a particular organizational structure, that tool may be more limiting to you than it is enabling. And so, many folks who have been doing this a while prefer tools that are much more customizable.

Having said that, we get a lot of communication from vendors that are trying to create an omnibus development environment. And so, they offer some very nice features. And that-- I'm not going to name a name and endorse a particular tool. So, forgive me. But in many organizations, when a new person comes on board and joins the team, they don't sit in a conference room and suffer death by view graph. They're not given PowerPoint slides. They're told to go into the application lifecycle management tool and look at the chats that the team you're on have been having, follow the evolution of this story. And you learn about the product in that way. And so, having that information captured in an easily accessible form that is intuitive and helpful to the engineers is a very powerful tool, a very powerful thing. And having said that, many of us love putting spreadsheets together and using macros and creating kluges that only we can manage. So, there's the other end of the extreme.

Presenter: So, we'll wait about another minute, see if we get any new questions. In the meantime, some reminders, our next event is going to be in October. We don't have an exact date, yet. I think in the console, you'll see October 4th. That data may be in flux. But the topic will be four valuable data sources for network security and analysis. And that will be by Tim Shimeall. And we will send out an invite for you to share along.

Once we do conclude today's event, obviously, the live event will end. We do encourage you to stay on Virtual SEI and check out the rest of the content that is available on the on-demand content. We hope that you'll share with colleagues, share within your social networks and provide feedback to us that way.

And it looks like you're going to be off the hook, Will, because I think the questions have dried up. So, thank you very much for an excellent presentation. We appreciate everybody's time today, joining us here today. We will send a follow up email tomorrow. Again, the location of the slides are in the chat. Please fill out that survey. We appreciate your feedback. But the email we send tomorrow will be, again, the location of the archive of this, which will live on this site probably as soon as this evening. The people that missed the event, you can share and watch it on Virtual SEI.

So, once again, everyone thanks for attending. Have a great afternoon.

Presenter: Thank you.

SEI WEBINAR SERIES | Keeping you informed of the latest solutions

**001 Presenter: Great job, Will.