

# Chapter 21

## Rearchitecting with Flow

*If you accept this fact – that the choices you make today will most certainly be wrong in the future – then it relieves you of the burden of trying to future-proof your architectures.*

*- Richard Monson-Haefe*

---

### Chapter 21 Table of Contents

Architectural Epic Kanban System.....	21-3
Objectives of the Method.....	21-4
Overview of the Architectural Epic Kanban System.....	21-5
Queue Descriptions.....	21-5
1. The Funnel - Problem/Solution Needs Identification.....	21-8
Sources of New Architectural Epics.....	21-8
Activities – Ranking the Epic.....	21-9
Work in Process Limits.....	21-10
Decision Authority.....	21-10
2. Backlog.....	21-10
Activities - Cadence-Based Review, Discussion and Peer Rating.....	21-10
Prioritization and Rating System.....	21-11
Weighted Rating and Decision Criteria.....	21-12
Pull from Transition to Analysis.....	21-12
Work In Process Limits.....	21-12
3. Analysis.....	21-12
Activities.....	21-13
Collaboration with Development.....	21-13
Collaboration with the Business - Solution Management, Product Management, Business Analysts.....	21-14
Work in Process Limits.....	21-14
Architectural Epic Business Case Template.....	21-14
Decision Authority.....	21-15
4. Implementation.....	21-16
Implementation Path A: Transition to development.....	21-17
Implementation Path B: Create a new team.....	21-18
Implementation Path C: Outsourced Development.....	21-18
Implementation Path D: Purchase a Solution.....	21-19
Work in Process Limits.....	21-19

Summary ..... 21-20

In the last Chapter, we introduced the role of systems architects, and system architecture, in helping teams evolve reliable, robust and scalable enterprise class systems in an agile manner. We described a set of principles that enterprises can use to govern this activity, while keeping the teams agile and the enterprise lean. We also described the business need for some amount of architectural runway, which is system infrastructure that exists to host features on the near term product map. Even then however, it's economic folly to attempt to *future proof* our solution, so occasionally, we run out of runway entirely, or our system becomes so unsuitable for the new market demands that we find ourselves needing to *rearchitect* the system.

Whether extending runway or rearchitecting the system, the 8<sup>th</sup> principle of agile architecture, *Principle # 8 – Implement Architectural Flow* reminds us of the constant need to keep all of our practices lean and flowing. In doing so, we can achieve the maximum velocity of value delivery and avoid the negative economic impacts of long queues, projects fits and starts, and thrashing across too many initiatives.

In this chapter, we'll describe a system that supports architectural flow and helps us match our wishes (new architectural initiatives) to our constraints (capacity of the development teams).

Before we do so however, it's worth repeating here that agile teams and agile programs are fully empowered to handle those significant architectural refactors and redesigns that are under their local control<sup>1</sup>. We don't need this additional layer of architectural drivers and governance (or even this Chapter) for that.

Therefore, the focus of this chapter is on large, cross-cutting architectural epics (as we defined in Chapter 20) that are likely to affect some combination of multiple teams, multiple products, multiple components, multiple services, and occasionally, even multiple product lines and business units. They imply a significant investment and they will impact a significant number of teams.

---

## *Architectural Epic Kanban System<sup>2</sup>*

In order to rearchitect such systems and manage architectural epics that rise to this global impact level, we need a process for reasoning about scope and return on investment, for performing analysis and prioritization, for analyzing capacity and assessing impact. We'll also need to keep all that activity visible to the stakeholders so they will know what we are working on, why, and when to expect results. We'll also need a systematic way to implement these large initiatives incrementally, in support of our basic agile and lean framework. To accomplish all this, we'll describe an *architectural epic kanban system*.

---

<sup>1</sup> Agile Manifesto Principle #11: The best architectures, requirements, and designs emerge from self-organizing teams

<sup>2</sup> Special thanks to the system architects and enterprise agilists at F-Secure Corporation for their substantial contribution to this chapter.

## Objectives of the Method

A lean, flow-based model for moving from architecture to implementation would accomplish three objectives:

1. *Make architectural work in process (AWIP) visible.*

Lean thinking drives us to make sure that *all work is visible*. As Reinertsen [Reinertsen 2010] points out, invisible, development work in process is WIP nonetheless<sup>3</sup>. Worse, since it can't be seen, it has “no natural predators” and therefore there is natural tendency to overload those involved in such work. (Since we can't see it or quantify it, and it seems like important stuff to do, why not do some more of it?).

Our kanban system must make AWIP visible so that it can be owned and managed responsibly. Architectural backlogs, queues and analysis work in process must all be visible, creating a shared understanding of current and future work and workload.

2. *Establish AWIP limits to control queue sizes and help assure product development flow.*

Limiting WIP helps us avoid the economic damage of large queue sizes, large delays, and the thrashing costs and inefficiencies of overloaded resources.

First, we'll limit local AWIP to only that work that the architecture teams can actually do, thereby assuring that the architects are not thrashing across too broad a workload – starting many projects - but finishing far fewer. That will increase the efficiency, productivity and quality outputs of the architecture team.

Secondly, in doing so, we will also be consciously limiting *global WIP*. This includes upstream, *portfolio WIP* (projects that drive new architectures) and downstream, *development WIP* (projects that build new architecture). In this manner, we'll match input objectives to implementation constraints, all across the enterprise.

3. *Drive an effective collaboration with the development teams.*

The tension between architecture and development is obvious. Eventually, architectural epics are going to be implemented by the development teams. It won't be helpful to surprise them with new stuff (“if we would only have known that sooner, we wouldn't have spent all this time ...”) or hold them accountable for implementing plans that they don't feel are actually workable. As F-Secure's James Cooper notes:

*Listen to developers – if they say there is a problem with the design, there probably is.*

---

<sup>3</sup> Principle of Product Development Flow Q1: The principle of invisible inventory: Product development inventory is physically and financially invisible

If our model drives effective communication between these teams, things will naturally flow more smoothly

4. Provide a quantitative basis for economic decision making

### Overview of the Architectural Epic Kanban System

So we'll need to implement a system that accomplishes these objectives with as much transparency and as little overhead as follows. Visually, such a system might appear as in Figure 21-1 below.

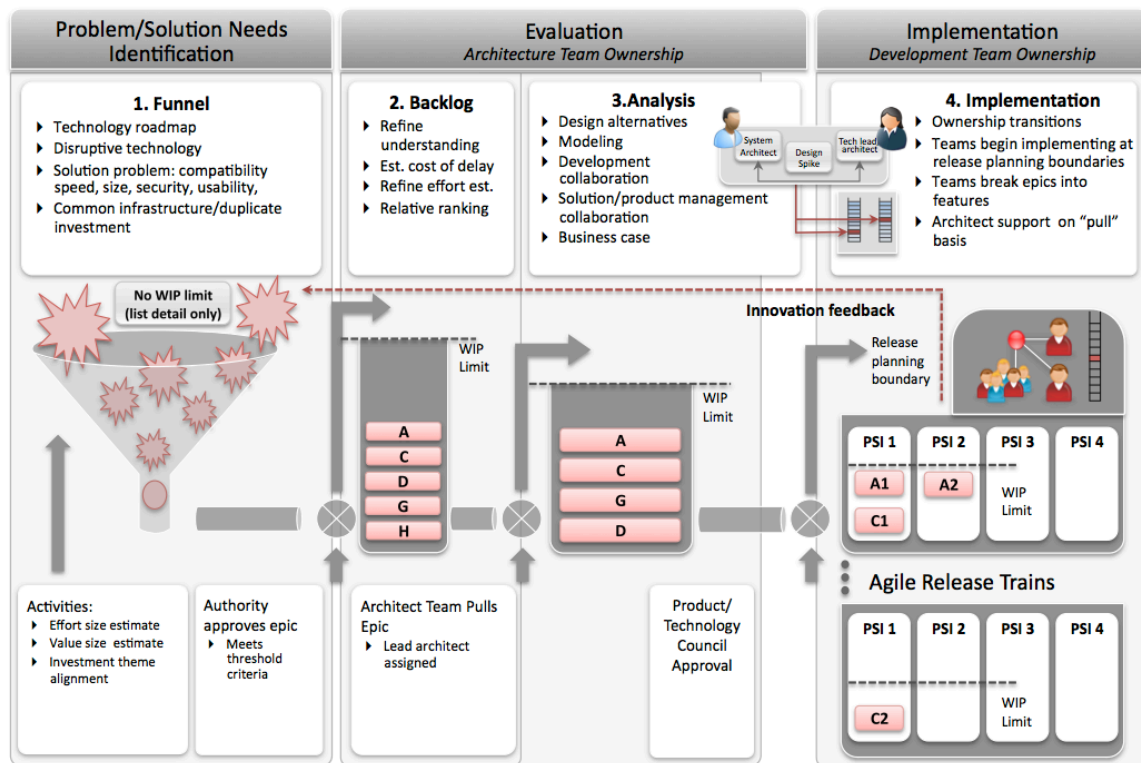


Figure 21-1 Graphic overview of the architectural epic kanban system

### Queue Descriptions

The assumption for the kanban system is that epics that are approved eventually go through a series of four queues, each characterized by different activities on the part of the architecture and development teams, along with correspondingly increasing levels of investment. The queues are as follows:

#### 1. The Funnel - Problem/Solution Needs Identification:

The Funnel queue is the “capture” queue. In this queue all new “big ideas” are welcome. They can come from any source. They need no business case or estimates. Tooling is trivial - a document, spreadsheet or simple list on the wall will typically suffice.

Since the investment of effort of items in this queue is minor, this queue is not WIP limited; all ideas are captured for consideration. Funnel epics are discussed on a periodic cadence established by the architecture team. Epics that meet the decision criteria are promoted to the Backlog queue.

## *2. Backlog*

Epics that reach the backlog queue warrant further investment of time. In this queue, epics are roughly sized and some estimate of value is established. Time investment is controlled to discussion level, and perhaps some very preliminary investigation. The epic may be elaborated to a paragraph or two.

Since the investment is increasing, this queue is WIP limited, primarily by the capacity of the architecture team. Backlog epics are discussed periodically. Epics are assigned a Cost of Delay (CoD). Epics that rise to the top of the queue are pulled into to Analysis as soon as space is available.

## *3. Analysis*

Epics in this queue deserve a more rigorous analysis, and require further investment. An architect is typically assigned as the epic owner. An active collaboration with development is initiated. Design alternatives are explored. Options for internal development and/or outsourcing are considered. A lightweight business case, with a *go* or *no-go* recommendation, is developed.

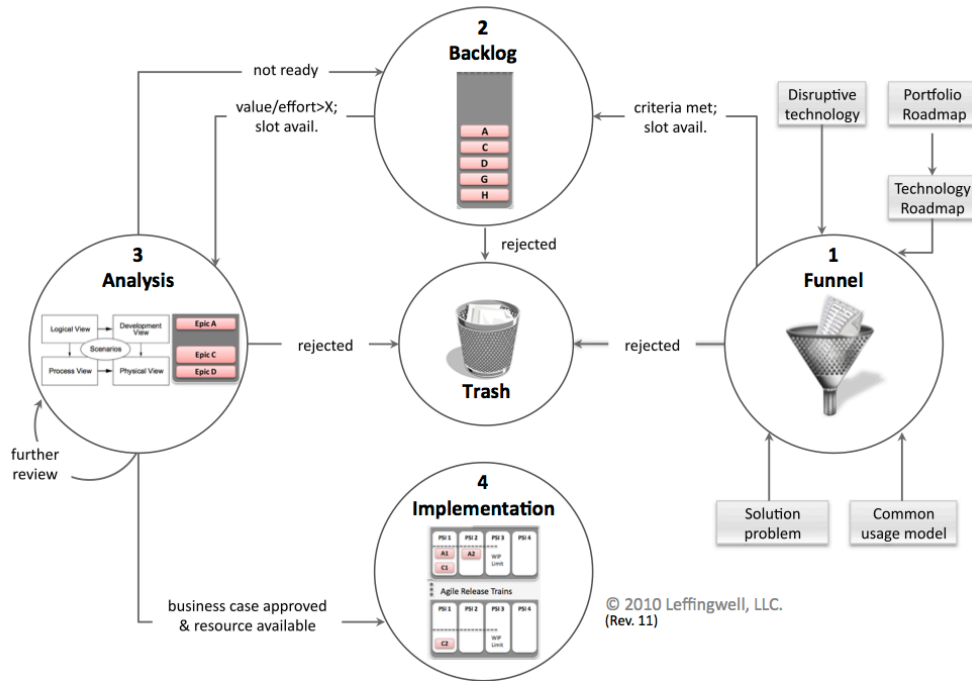
Items in this queue use scarce resources, so it is WIP limited based on capacity of the architecture and development teams. Promotion from Analysis to Implementation is an important economic decision for the enterprise that can be made only by the appropriate authority, based on the developed business case. Epics that meet the *go* criteria are promoted to Implementation.

## *4. Implementation*

In this queue, the primary responsibility for the epic is passed to the development teams. Architect resources remain available on a “pull” basis, i.e. the responsibility for implementation rests with the development teams, but the architect assists the teams and shares responsibility until the team has developed a sufficient understanding of the work required.

This queue is WIP limited by the capacity of the development teams.

While the overview description provides a broad sense of how the system works; it leaves as many questions as answers. In order to implement such a model, a more definitive description of the system behavior is described in the state transition diagram (Figure 21-1).



**Figure 21-2** State transition diagram for architectural epic kanban system

The state diagram illustrates all the paths for an epic and the decision gates that drive the epics down these paths. Details of the activities for items in the queue, the decision

criteria, and the decision authority are summarized in Table 21-1 below.

Queue	Activities to transition	Transition criteria	Next	Authority
<b>Funnel</b>	<ul style="list-style-type: none"> <li>▶ Estimate value</li> <li>▶ Estimate effort</li> <li>▶ Test against investment themes</li> </ul>	<ol style="list-style-type: none"> <li>1. Rank &gt;threshold</li> <li>2. WHEN Slot available</li> <li>3. Fails criteria</li> </ol>	<div style="border: 1px solid gray; padding: 2px; display: inline-block; margin-bottom: 5px;">→Backlog</div>  <div style="border: 1px solid gray; padding: 2px; display: inline-block;">→Trash</div>	Architectural Authority
<b>Backlog</b>	<ul style="list-style-type: none"> <li>▶ Assign Cost of Delay</li> <li>▶ Effort estimate refined</li> <li>▶ Establish Relative rank</li> </ul>	Ranked relative to other items Highest ranked item pulled  When age of item > limit	<div style="border: 1px solid gray; padding: 2px; display: inline-block; margin-bottom: 5px;">→Analysis</div>  <div style="border: 1px solid gray; padding: 2px; display: inline-block;">→Escalate or Trash</div>	Pull system  Architectural Authority
<b>Analysis</b>	<ul style="list-style-type: none"> <li>▶ Workshops, modeling, design alternatives</li> <li>▶ Development collaboration and cost estimates</li> <li>▶ Dev design spikes</li> <li>▶ Product/Solution management review</li> <li>▶ Implementation options</li> <li>▶ Market validation of value</li> <li>▶ Business case</li> </ul>	Business case with GO/NO GO recommendation  GO -> implementation  NO GO 1-> more elaboration needed  No GO 2 - reject	<div style="border: 1px solid gray; padding: 2px; display: inline-block; margin-bottom: 5px;">→Impl.</div>  <div style="border: 1px solid gray; padding: 2px; display: inline-block; margin-bottom: 5px;">→ Stay in queue</div>  <div style="border: 1px solid gray; padding: 2px; display: inline-block;">→Trash</div>	Product/Technology council

**Table 21-1** Architectural Epic Kanban System Table

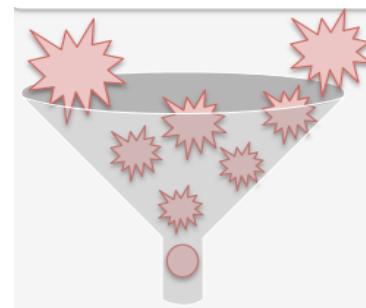
### 1. The Funnel - Problem/Solution Needs Identification

The “funnel” is the simplest queue. In this queue, all ideas are welcome; anyone can contribute. As with all future queues, when an item enters this queue, it is date stamped so the teams will be able to tell how long it has been in the queue.

#### Sources of New Architectural Epics

Business drivers for new architectural epics come from a variety of sources from within, and outside of, the enterprise. These include factors such as:

Technology roadmap. Driven by the product roadmap, the system architects typically manage a technology roadmap that they use to monitor and implement key new technologies over time. Examples include converting back office systems to service oriented architectures, “webifying” current customer facing applications, etc.



*Disruptive technology.* Some disruptive technologies may appear fairly suddenly and make their way to the roadmap in an expedited fashion. Examples include 64 bit chip sets, Single Sign On standards, Bluetooth, rapidly evolving wireless standards, etc.

*Common usage model and avoidance of duplicate investment.* In the larger enterprise, it is likely that a significant number of teams are developing code to solve the same problem for customers. This can be as simple as installation utilities, licensing mechanisms, access to common data sets, etc. If left to their own initiative, the teams will solve these problems in a way that suits the local team and their customers. However, that same customer is likely to be a user of a different product or service that will have implemented the same utility, but in a different way. In this case, we have two significant economic problems:

1. The enterprise is duplicating investment in common technologies and common code. Multiple teams are writing the same lines of code, but in a different way. That doesn't always make economic sense.
2. Worse, while simply trying to do the same function, customers are experiencing a wide range of user experiences. This causes confusion, higher training costs, lack of seamlessness for the user, and ultimately lower perception of quality and loss of customer confidence in the solution provider.

In these cases architectural epics are used to drive common initiatives that decrease investments costs and improve product quality.

*Problem with the Existing Solution.* Some architectural epics are driven by known problems with an existing solution. Examples improve increasing performance and reliability, enhancing scalability to support market success, or even building work-arounds for a patent challenge.

### **Activities – Ranking the Epic**

The primary activity of the architect teams for epics in this queue is a periodic review and analysis of the item, establishing three relevant parameters:

- 1) A preliminary estimate of the *size* (the relative “bigness”) of the epic. The effort estimate is an aggregation of a number of factors that may include:
  - Estimate of the cost and time to implement
  - Number of teams, programs, products potentially effected
  - Technical risk
  - Complexity
- 2) A preliminary estimate of the potential value of the epic, whether measured in customer retention, revenue, or market share value
- 3) A quick test to match the epic with the enterprises current investment themes.

Once the teams have agreed on a set of metrics that work in their context, it's a fairly easy matter to create a calculation spreadsheet to rank a funnel epic relative to its peers. One such example is provided in the table below:

Ranking Item	Scale
Effort Size	1,2,3,5,8,13, 20, 40 <sup>4</sup>
Potential Value	1,2,3,5,8,13, 20, 40
Alignment to investment theme	0, .25, .5, .75, 1
Epic Weight	= (Potential Value/effort)* Alignment

**Figure 21-3** Ranking criteria for epics in the funnel state

Once an item has been ranked and its value exceeds some threshold, (or perhaps simply shows greater value than each of its ranked peers), it can be moved to the backlog, assuming space is available.

If the item fails some threshold test, it is deleted. Items can also be deleted from the funnel when they have been in the queue too long (perhaps six months). This indicates that the item was either too ambiguous for investment, or perhaps simply didn't have high enough importance to warrant the attention of the team.

### Work in Process Limits

Since this is the “capture” state, there is no strict WIP limit associated with this state, although period pruning may be necessary to keep the list to a workable size (perhaps 50-75 items).

### Decision Authority

The funnel queue, and the final decision to move an item to the backlog, is managed by the appropriate architectural authority - often a chief architect or CTO.

---

## 2. Backlog

Epics in the backlog queue require a little more time investment, so they are treated with additional rigor.

### Activities - Cadence-Based Review, Discussion and Peer Rating

The architecture team, working as a group, reviews and discusses all epics on the backlog on a regular cadence, (typically every one to two weeks). During this review process, epics receive additional consideration and are further elaborated to advance understanding.

---

<sup>4</sup> We've arbitrarily used the modified Fibonacci series to indicate that uncertainty gets larger as the estimates get larger, but any appropriate scale can be used.

This includes the diligence required to further understand the epic, refine the estimates of effort and value, and to measure alignment with the current strategic investment themes.

In order to rank the epic relative to its peers, epics may be placed into a quantitative evaluation matrix, based on whatever rating system the team establishes for items in this queue. One such example of a template and rating system is provided in Table 21-2 below:

<b>Name</b>		<b>Date Entered backlog</b>		
<b>Version</b>		<b>Changes</b>		
<b>Description</b>				
<b>Stakeholder sponsors</b>				
<b>Prerequisites (if any)</b>				
<b>Teams, products, programs, markets affected</b>				
<b>Notes</b>				
<b>Ratings</b>		<b>Weight</b>	<b>Net</b>	<b>Comments</b>
<b>Effort Size</b>		1		(Scale 1,3,5,8,13,20,40)
<b>Cost of delay</b>				
<b>Business value</b>		1		(same scale)
<b>Time value</b>		1		(same scale)
<b>Information discovery value</b>		1		(same scale)
<b>Length of time in queue</b>		.5		(see below)
<b>Weighted Rating</b>				= (BV+TV+IDV+LT*.5)/Size

**Table 21-2** Template and rating system for backlog queue epics

### Prioritization and Rating System

In Chapter 13 – *Vision, Features and Product Roadmap*, we described a Weighted Shortest Job First (WSJF) prioritization system for rating features, which optimizes delivery value based on the economics of the Cost of Delay. In the template above, we’ve applied WSJF again with a few minor differences. The prioritization parameters include:

- Estimated size (see funnel queue size)
- Cost of delay – an estimate for the cost of delay. As with features, the cost of delay includes three components:
  - Business Value – an estimate of the size of the potential return for the business.

Note: This parameter is stated as *business value*, rather than as *user value* (as we applied to features), because many system level architectural refactors are driven by internal cost savings, maintenance concerns, performance and scalability, new

- business opportunities, or otherwise have a more indirect relationship to near term user value.
  - Time value. The way in which the value decays over time. A low rating indicates a stable situation; a high rating indicates a rapid potential decay on the business value
  - Information Discovery Value – the value associated with knowledge discovery and mitigation of risk
- Length of time in the queue. We’ve added this optional parameter to the list as a reflection on how long this epic has been in the system. This is intended to accelerate consideration and decision-making on items that have been in the queue too long, as they continue to require review and investment, and therefore drive overhead by some incremental amount.  
(Example 0-30 days=2, 30-60 days=5, 60-120 days=8, over 120=13.)

### **Weighted Rating and Decision Criteria**

These attributes are combined into a single weighted rating, which can be used to prioritize the epics relative to each other, based on the economics of a WSJF. The table above also applies an optional weighting system for the various attributes. Items with the highest rating rise to the top of the backlog for promotion to the next state.

### **Pull from Transition to Analysis**

Since the next queue is WIP limited, there may be no further decision criteria beyond the mature rating, and a pull system may be applied. In other words, so long as there is room in the Analysis queue, any available architect can pull the highest rated epic into the next queue.

### **Work In Process Limits**

Since items in this Backlog queue require additional investment, the queue is WIP limited to some integer number of epics (perhaps 20-25). Like all WIP limits, the limit can be adjusted over time based on the capacity of the architecture team and the response time desired for items to move through the queue.

---

## *3. Analysis*

In the two queues we have described so far, work is limited to discussions and preliminary analysis only. Investment is minimal. The artifacts are lightweight. A fairly large number of epics are manageable.

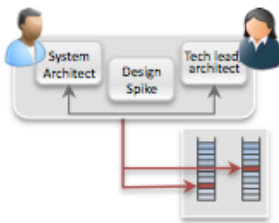
In the Analysis queue, however, material use of scarce resources (architects, team leads, product and solution managers, marketing/business analysts, development team members) is required.

## Activities

When an item is pulled into this queue by the architect team, the epic is time stamped and an architect is assigned to be the “epic owner”. The epic owner is the “chief engineer” for the epic and is responsible for defining and spearheading the analysis work that follows. Work in this state may include some or all of:

- Consideration of design alternatives
- Requirements workshops and other discovery techniques (Chapter 12)
- Impact analysis: development, distribution and deployment
- Evaluating internal resourcing versus outsourcing options
- Buy or build evaluation
- Architectural analysis and modeling<sup>5</sup>
- Refined scoping of market potential
- Collaboration with development
- Collaboration with business analysts, solution managers and/or product managers
- Market validation of value

## Collaboration with Development



In the last Chapter, we described “intentionally emergent system architecture” as a result of a *role collaboration* between the system architects and the development team. That collaboration is initiated when an epic appears in this state. Typically, this involves engaging tech leads and architects associated with the agile teams in the analysis and design alternatives work. In addition, the *development teams themselves estimate the development cost* for each area impacted. This provides higher fidelity estimates and a greater sense of buy in from the teams.

In addition, some number of development teams will likely be involved in technical spikes designed to determine feasibility and reduce risk of the initiative. In a mature state, it is not unusual to see teams investing as much as 5-10% of their resources in technical spikes<sup>6</sup>. Of course, architects are also likely to be doing technical spikes and evaluating various design tradeoffs in code. However, having the development teams perform the majority of the technical spikes has a number of benefits:

- 1) There are far more resources in the development teams than in the architecture team

<sup>5</sup> Yes, architects still model in agile development, though lighter weight approaches are often applied. More on this topic can be found at, for example, <http://www.agilemodeling.com/>.

<sup>6</sup> Agile architecture principle 3: When in doubt, code it out.

- 2) The development teams are closest to the current implementation; they have the best understanding of how to integrate the new architecture
- 3) The spikes give the teams time to explore the upcoming work and to socialize and integrate the new concepts into their backlogs, vision, designs and thinking

### **Collaboration with the Business - Solution Management, Product Management, Business Analysts**

In a like manner, the impact on the marketplace and on the business must also be understood. Therefore, a second active collaboration is initiated with those elements of the business who can best assess these factors. In addition, the final value estimate should come from the Solution Manager, Product Manager, or Business Owner - whomever is in the best position to judge the potential value. More importantly, this engages these key stakeholders in helping manage the impact of the epic so as to best achieve that value. Without their active support and buy in, the epic is likely doomed anyway and should be dropped from the queue.

### **Work in Process Limits**

As we mentioned, epics in this queue consume scarce and valuable resources, therefore this state is subject to rigorous work in process limits. The limit may be simply some number of epics, (or the number and size of the analysis work) and may be adjusted over time. However in our experience the size of the epics is less material to this WIP limit; the number of epics undergoing simultaneous evaluation is a more controlling factor. Typically, some small number of epics (perhaps 5-7) would be manageable in this queue.

### **Architectural Epic Business Case Template**

As we have described, the types of epics that have reached this queue are assumed to be large, and cross cutting – i.e. they typically affect multiple teams, products, components, and services. They imply a significant investment and they will impact a significant number of teams. Therefore, it is incumbent on the architects - in collaboration with the development teams and their product/solution manager-partners - to make a *go* or *no go* recommendation to the business. In addition, the recommendation should be presented in such a way that the business owners have the background data they need to make a final decision.

In order to do so, we recommend that the primary artifact of an epic in this phase is a *lightweight business case*, of one or two pages in length. As an example, an annotated template for such an artifact appears in Table 21-3.

<b>Epic Name:</b>	<b>Go or NO Go Recommendation:</b>	<b>Date entered backlog:</b>	<b>Architect Epic Owner:</b>
<b>Version</b>		<b>Changes</b>	
<b>Description</b>			
<b>Stakeholders sponsors</b>	(Identifies key business sponsors who will be supporting the initiative)		
<b>Products, programs, services affected</b>	(Identifies products, programs, services, teams, departments, etc. that will be impacted by this epics)		
<b>Impact on sales, distribution, deployment</b>	(Describes any impact on how the product is sold, distributed, or deployed)		
<b>Estimated investment</b>	<b>Story points:</b>		<b>Cost:</b>
<b>Weighted Rating</b>	(WSJF Rating from Analysis)	<b>Type of return:</b>	Nature and amount of potential return. Markets impacted, revenue, customer satisfaction, product line extension, customer retention, etc.)
<b>In house, outsource pr purchase</b>	(describes recommendations for where the epic is to be developed)		
<b>Estimated development timeline</b>	<b>Start Date:</b>		<b>Completion date:</b> (Estimated calendar date or number of PSIs)
<b>Incremental Implementation Strategy</b>	(Breaks initiative down into preliminary epics or sub-epics that fit the companies PSI cadence)		
<b>Reevaluation checkpoints</b>	(If the epic is large, identifies potential milestones or checkpoints for reevaluation)		
<b>Analysis summary</b>	(Brief summary of the analysis that has been formed to create the business case. Pointers to other data, architectural models, market analysis, etc. that was used on the creation of the business case)		
<b>Other notes and comments</b>			

Table 21-3 Lightweight business case template for architectural epics

When the architects feel that the analysis has been sufficiently thorough and that the business case is “ready enough”, it is presented to the decision authority for action.

### Decision Authority

Due to the scope of the effort, the stakeholders who have the ultimate fiduciary responsibility for the product line, business unit or enterprise must make the final decision as to whether or not to proceed with implementation. Typically, this responsibility rests with a Portfolio Management Team (Chapter 22) or *Product and Technology Council*, whether a formal or informal construct. Such councils typically include the chief executive for the domain, senior solution managers, line of business owners, sales and marketing representatives, the CTO, and senior development managers.

### *“No” is an Acceptable Answer*

However, even in the presence of a *go* recommendation case, the product council must take a broader view. For example, they must also consider the opportunity cost of the initiative (the opportunities lost to the business while this epic is being implemented) along with any other business cases at the same state of maturity. For example, if a product council is presented with 5, 10 or even 20, business cases with an estimated positive ROI (those with a negative ROI will not make it this far) that doesn't mean they could or should approve them all. After all, the I (Investment) comes before the R (Return), and the business must always work within its investment limits. In addition, the target is optimization of the overall portfolio, so individual epics must be considered in that light. So NO is an acceptable answer.

Indeed, one would hope that the business is managed in such a way as it is always presented with many opportunities for new investment, and picking the right ones means saying NO to many others. That is one of the many reasons we keep the business cases lightweight; lest the personal time investment of those who developed the case may be prejudicial to the decision. The enterprise should consider only the marginal, forward-looking investment, ignoring any sunk costs<sup>7</sup> and personal biases that may have occurred in analysis.

So reasonable decisions include *not* initiating the project, or in the marginal case, perhaps asking for some additional analysis. In this case, the business must recognize that due to the WIP limits in the Analysis queue, there is also an opportunity cost (other epics cannot enter analysis) in leaving the initiative in that queue.

### *“Yes” is Too*

However, many such initiatives will be approved or the organization will fail to innovate or keep the technology platform current for the longer view. So when approved, the epic is moved to *Implementation*.

---

## *4. Implementation*

While the analysis work of the prior two queues uses scarce resources, the real investment in cost begins once an architecture epic has been committed to implementation.

One major challenge in broaching this next state is an obvious one: “this new thing is big and no one is working on it now, so how exactly, do we intend to actually do it?” That drives the enterprise to a serious consideration of resources and implementation paths as part of the business case in the last state. In practice, there are three implementation paths to be considered:

- Path A – Internal development
- Path B – Create a new team to build some runway

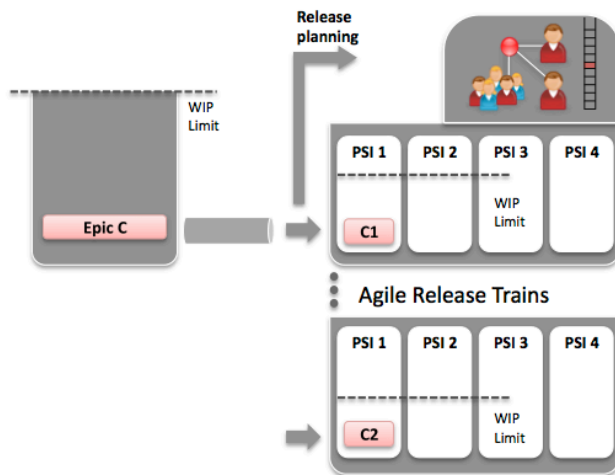
---

<sup>7</sup> Product Development Flow Principle E17: The sunk cost principle. Do not consider money already spent.

- Path C – Outsourced development
- Path D – Purchase a solution

### Implementation Path A: Transition to development

In this case, ownership of the epic moves to the development teams for implementation in the affected release trains as Figure 21-4 below illustrates.



**Figure 21-4** Epic C moves to development in two release trains

The sponsoring architect remains on the team in a “pull “ state, meaning that he or she provides whatever support the teams need to fully understand and implement the epic, but the responsibility now rests with the development teams.

Transitioning to development is the preferred case for a number of reasons:

- Timing – New development starts as soon as feasible within the context of the ongoing release trains for the affected products.
- Responsibility transitions immediately. Architects are freed up (except for the pull commitment) to work on other epics
- Work remains in house; team’s knowledge increases; companies core competence grows.

In this case, the new epics will be presented as part of the vision and architecture overview at the next release planning session. There, epics are likely to be split, and the teams consume the smaller epics by subdividing them into the architectural features and stories they will use for implementation.

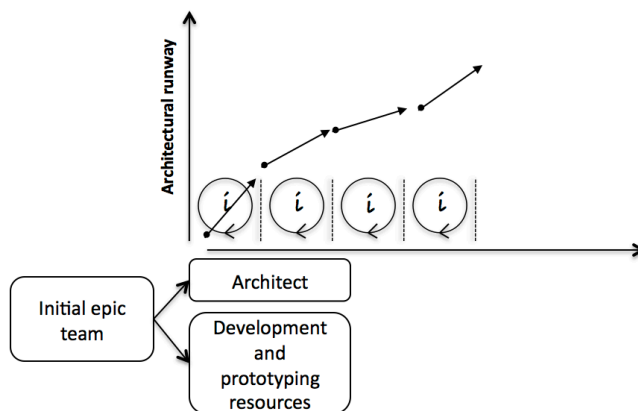
Thereafter, it is simply business as usual and the architectural epic flow system is complete for that epic.

### Implementation Path B: Create a new team

Though the above path is the preferred case, it is not always practical for a number of potential reasons:

- Sometimes the architectural epics represent a new technology or framework that must be built from scratch – or perhaps it still has significant risk associated – or is otherwise too immature or uncertain to warrant immediate disruption of the teams
- The development teams are so consumed by existing commitments that they cannot take on the new epic in a timely manner
- The epic may implement new technologies that are unfamiliar to the teams and/or may require creation of some new infrastructure (CI, etc.) that must be in place before development begins.

In this case, it may be beneficial to form a new team - consisting of the epic architect owner and some set of development resources – and charter that team to build some initial architectural runway in support of this epic as Figure 21-5 below illustrates.



**Figure 21-5** Building architectural runway for a new epic

Over time, the team may evolve to become a full-fledged development team that supports the new epic. Perhaps more likely - the new framework will eventually be transitioned to development teams for ongoing development and maintenance as in Path A above.

### Implementation Path C: Outsourced Development

There is a third path that must be considered as well. While outsourcing may be a less desirable strategy, there are situations in which outsourcing can be the most effective path to implementation of new technology. This can occur when:

- 1) There is simply insufficient internal capacity to take on the epic
- 2) There are outside sources who have more core competence in the new technology that and can therefore more quickly build the needed runway

However, as the epic likely represents some new core intellectual property - one that the development teams themselves must eventually understand and master – internal development or architect resources should be assigned to the project so that knowledge transfer begins immediately.

### **Implementation Path D: Purchase a Solution**

Purchasing a solution may also be a viable option under any of the following conditions:

- The product/service is needed to enable business but it is not the company’s core business
- There is a solution available in the marketplace that is cheaper to buy than
- When the enterprise can leverage the competence of the solution provider.
- When time to market is so critical that a more immediate solution must be delivered

### **Work in Process Limits**

The potential application of work in process limits for the implementation stage is not so obvious and is highly dependant on the path chosen. We’ll look at each below.

#### *WIP Limits for Path A – Transition to development*

In this path, WIP limits are highly subjective to the companies competitive and customer satisfaction context at the time of implementation. Rather than strict limits, it may be sufficient for the enterprise to know how much they are investing in new technology infrastructure for the near to mid-term, versus ongoing investment in feature delivery on the existing runway. That can usually be accomplished in the agile project management tooling, where features and stories driven by architectural epics can be categorized accordingly.

As a rule of thumb, we often see enterprises limiting new technology development to 10-15% of total investment. Yet, there are times when we have seen a program devote as much as 60% of the effort, spread across multiple PSIs. This decision must be based solely on the company’s context in the market at that time. It doesn’t really matter how the company measures or limits it, so long as the WIP is visible.

#### *WIP Limits for Path B – Create a new team*

WIP limits for this path are more obvious. Initiatives on this path consume scarce resources (architects, tech leads, senior developers) for substantial periods of time, and after that, *still* require a transition back to development for longer-term support.

Therefore, a typical business unit, product line, program or smaller enterprise can typically support only *one or two* such initiatives at a time.

### *WIP Limits for Path C & D – Outsourced Development and Purchased Solution*

WIP limits on these parts are controlled by two factors a) the availability of funding, and b) the availability of architects or others to effectively manage the projects and to provide a surface for information transfer.

Like Path B above, a limit of only *one or two* such concurrent projects may be suitable for the typical program.

---

### *Summary*

In this Chapter, we've described an architectural epic kanban system as an example of a lean process that can be used to reason about, analyze and make go/no-go recommendations for large-scale architectural initiatives.

This system is designed to make all architectural work in process visible, manage queue sizes of work awaiting implementation, and to provide a quantitative way to evaluate epics relative to each other. The final output of this system is a lightweight business case for the architectural epics that can be used by the appropriate authorities to make a final decision on any individual epic. Those epics that make the cut move to implementation and the systems architects work is largely complete, subject to whatever support may be required based on the chosen implementation path.

These last two chapters complete our discussion of agile system architecture. In the next two chapters, we'll move on to discussing agile approaches to managing the enterprises full portfolio.