A Program Backlog Story with Patterns

Expanding the Magic Backlog Pattern Collection

Lise B. Hvatum
lhvatum@acm.org

Rebecca Wirfs-Brock
Wirfs-Brock Associates
rebecca@wirfs-brock.com

ABSTRACT

This paper extends our Magic Backlog Patterns collection with three additional patterns for managing the work of a program– or rather how to deal with coordinating the work of projects which are part of a larger program and where there may be dependencies and shared deployment. While teams within a program may work fairly independently, their work still needs to be coordinated to produce a product. These three patterns, which represent alternative strategies for structuring the backlog of work, are introduced through a story of the correspondence of a business analyst as her hypothetical program moves through different backlog management strategies.

CCS CONCEPTS

• **Software and its engineering → Requirements analysis** • Software and its engineering → Software implementation planning • Software and its engineering → Software development methods

KEYWORDS

Requirements engineering, requirements analysis, agile, product backlog

ACM Reference format:

Lise Hvatum and Rebecca Wirfs-Brock. 2018. Program Backlog Patterns – Expanding the Magic Backlog Patterns. 23rd European Conference on Pattern Languages of Programming (EuroPLoP), EuroPLoP 2018, Jul 4-8 2018, 18 pages. https://doi.org/10.1145/3282308.3282338

1 Introduction

This paper extends our Magic Backlog Patterns collection [1, 2, 3] with three patterns for managing the backlog of work for a program. Often programs are formed from pre-existing projects and project teams, which have established ways of managing their backlogs. The challenge when forming such a program is how to best to support individual projects' established ways of working while facilitating the coordination of work and a consistent view of the overall program's progress. Sometimes, programs are formed anew. Even in this case, the question of how best to manage and to coordinate the work of various project teams working on different aspects of the overall program, still needs to be addressed. The three patterns described in this paper represent alternative strategies for structuring the overall work for a program and offer guidance for those struggling with how best to manage their program backlog, given their program's history and unique context.

The structure of our paper is as follows: after providing a brief overview of agile requirements, backlogs, and programs, we present a story in three parts that motivates each backlog pattern. After each part of the story, we present a particular program backlog management pattern which, given the current context of the story, appears to be a good fit to the current situation. After a discussion, we draw some conclusions. An appendix summarizes each pattern in our collection.

EuroPLoP '18, July 4–8, 2018, Irsee, Germany

ACM ISBN 978-1-4503-6387-7/18/07 $15.00
https://doi.org/10.1145/3282308.3282338

2 Background

The requirements for a software product are generated through various elicitation techniques, and are further detailed and analyzed using methods like story mapping, use cases, and workflows [2]. There are a number of useful publications [4, 5, 6, 7, 8, 9, 10] that provide methods and techniques for how to elicit, analyze and process information to reach detailed software requirements. Most software projects use an ALM (Application Lifecycle Management) tool like JIRA or TFS to manage the requirements and the work that is needed to implement these in the "Product Backlog."

The term Product Backlog is part of Scrum terminology but used in general by agile processes:

"The Product Backlog is an ordered list of everything that might be needed in the product and is the single source of requirements for any changes to be made to the product. […] The Product Backlog lists all features, functions, requirements, enhancements, and fixes that constitute the changes to be made to the product in future releases." --- *Scrum Guide [11]*

Most agile process descriptions are sketchy on how product backlogs (for short we use the term "backlog" in this paper) are developed and maintained, and so give little guidance to the product team for this activity. Still, having a good backlog is really important because it contains the fundamental information that drives the development activities. Our work is aimed at filling this knowledge gap with patterns that provide practical advice on how to build a good quality backlog for a large and complex product using an ALM tool. This paper is focusing on backlog patterns for software programs.

So why Magic? We initially used the term "Magic Backlog" in our earlier writing because of the lack of attention that agile process descriptions pay to the creation of the backlog – so it appears as if by magic. Feedback we got on our first backlog pattern paper made us realize that the term has another meaning: done well with the right contents and structure the backlog can do magic to support the team. Readers familiar with the children's stories of The Magic School Bus [12] will recognize the connection. If you need a submarine, the school bus will transform into one. If you need a microscope, or a fully equipped biology lab, there will be one in the bus. With careful design and preparation of your backlog, it can be as magic as the school bus, supporting your current needs. It provides a technical view of the product to the development team, while the Product Owner can see a business view. It keeps the current plan for the project manager, and the testing structure for QA. It helps you know where you are and where you should be going next.

Our patterns define role-based activities and responsibilities. We expect individuals to be moving between roles depending on what they are currently doing. A product owner could double as a tester. A project manager might do the work of a business analyst, as well as development and testing. One role that especially needs clarification is that of the analyst. A Scandinavian proverb states that, "A beloved child has many names," and this is true of this role. Business analyst, product analyst, and requirements engineer are frequently used. In essence this role is an expert on requirements engineering, i.e. the elicitation and management of requirements. The analyst is not a domain expert, but a skilled resource who knows how to drive elicitation efforts, how to analyze and structure the outcome, how to translate from the business domain to the technical domain, and how to administer and maintain a requirements collection for a product. The role of the analyst often falls on a product owner or the project manager, and sometimes is distributed among the development team. But as products are getting larger and more complex, there is an emerging analyst profession, and more frequently teams include dedicated analysts. Just as the agile tester is integrated into the team, so should the analyst be. The primary audience for our patterns is the analyst.

3 Programs and Backlogs

A project is typically well defined, with a singular purpose and result, and with a single, unified team. A program is a larger endeavor. It consists of a collection of projects, which collectively produce a desired business solution. Project related sub teams within a program may work fairly independently, yet their work still needs to be coordinated. While each project in a program may have a valuable deliverable, the overall value to the business is achieved only when all the projects in a program deliver on their results [13]. So a program's success is more than simply the cumulative successes of its individual projects. Different projects within a program are complimentary and help the program achieve its overall objectives. Even so, on complex programs, there are likely to be overlaps and dependencies between projects. The overall objectives to be accomplished by a program are only achieved by coordinating the work of various projects, resolving any conflicting or competing priorities, and prioritizing the work.

One way to coordinate the work of various program teams is to manage and coordinate their work via the product backlog (or backlogs). The purpose of our work is to provide knowledge around product backlogs, independent of any particular development process. In our writing we try to be as process agnostic as possible.

Still, we have an expectation that the process applied to manage a product backlog is a form of agile/lean, and this expectation will decidedly influence our advice on how to structure and manage the backlog. In particular, we have been influenced by Johanna Rothman's writing on program management, the Nexus Framework for Large Scale Scrum and other Scum of Scrum models [13, 14, 15, 16, 17]. Our focus is on how to best utilize the tooling, and how a good handle on the tooling can help make the workflows of the team more effective. What our work excludes are patterns for team collaboration and agile practices (there is a wealth of books, blogs, and training available on this already), and we deliberately focus on the tooling.

To explore the unique challenges of creating and maintaining program-level backlogs, we tell the story of the Bluebird program via a series of emails, notes, and conversations between Caroline, a business analyst and the main character in our story, and various team members, friends, colleagues, and consultants. The story highlights the choices and challenges a program has with keeping their backlog useful to coordinate and plan multiple projects collaborating to deliver a joint solution. As the story develops, we pause to capture the program patterns emerging from the experience of the program. An overview of the patterns collection with some history of our earlier work is found in appendix A.

In this document, we use small caps for patterns, and small caps italics for patterns in our collection that are not yet written. If we refer to a pattern outside our collection it will also be in small caps but with a reference attached to it.

4 Dear Dixie – Introducing the Story

We follow the story of Caroline through a set of e-mails, text messages, diary entries, and personal notes. Caroline is a business analyst on the Bluebird project. This project has been running for almost three years. During this time, Caroline and her team have been through the "storming-forming-norming-performing" [18] process, and have reached a way of working together that enables them to perform well. There is a high degree of trust between team members. For the last six months they have steadily delivered

software. Most of their builds pass the extensive automated test suites. They normally update their production system weekly if not more frequently.

The product backlog for the Bluebird project also went through the "storming-forming-norming-performing" process. At the start of the project Caroline was unfamiliar with TFS, the tool selected by the company as their ALM tool standard, and this was the first time she was working as a business analyst on a project this size. After a few months using TFS, the Bluebird project backlog had grown into a confusing mash up of technical and feature user stories. This was mixed up with tasks that represented project work like organizing meetings and training and as such did not contribute to the product. Team members freely added contents, and items of the same type were inconsistent with wide variation in granularity and contents. Gathering metrics out of the contents was difficult, and the QA team members struggled to support developers with usable feedback. After several rounds of changing the structure of the backlog and maturing the processes around use of the tooling, Project Bluebird's product backlog now has a well-defined structure. It is easy to understand the product features (see FRAME) as well as technical and quality aspects (see VIEWS). The contents follow a clearly defined structure (see CONNECTIONS) and there is understanding among the team members about who will modify information and how (see *RULES*). Although the team keeps tweaking and improving their backlog, it now is in a state where it clearly supports the development effort in planning, implementation and verification of the product. Management perceives the Bluebird project as performing well.

When we enter this story, Caroline is about to get a new challenge. Her company is merging with another company, and a combined product strategy means that the Bluebird project along with other projects will be formed into a program to deliver an end-to-end customer solution. Caroline is a social and well-connected person, which helps her in her expanded role as business analyst for this new program. She reaches out to people both inside and outside the company to share ideas and discuss program challenges. She takes care not to expose information that is company confidential, but instead focuses on sharing practices around requirements and program backlog management. In particular, Caroline is communicating with these people:

Dixie – a senior business analyst that Caroline worked with in the past. When Caroline started her first job, Dixie was her mentor. Over time, they have become close friends.
Ross – the program manager
Frank – the product owner
Cindy – the Bluebird project manager
Ivan – a project manager from the company that Caroline's company is merging with
Kate – the program level architect and Ivan's best friend
Martin – an independent agile coach who has trained and coached at Caroline's company, and who is always willing to listen and provide advice and pointers to resources. Martin is close with Scrum.org and knowledgeable about Microsoft and VSTS.
Tommy – Caroline's boyfriend

5 Program-level Backlog Patterns

In this chapter we let the story evolve and lead us to three different solutions we have found when looking at how program backlogs are structured. Each section culminates in a pattern that captures the solution used in the associated part of the story.

Caroline's story – part I

Dear Dixie, May 7 2016
Today was my work anniversary – it is hard to believe it is 3 years since I joined the Bluebird project as their Business Analyst. My manager, bless her heart, invited the team to lunch to celebrate. Thinking back, we have come such a long way. Remember all those half-finished sprints? And how the team demo was a hack and when you tried to test it turned out it would only run OK on the developer's laptop? And how lost we were on organizing our backlog and utilizing the TFS tooling?

I will never forget that silly training we went to. We learned next to nothing. The trainer was remote so we spent the time talking to each other and having fun. Maybe we should have paid more attention to how to create those reporting queries. But the trainer did not explain anything about the whys – only technical details about using the tool. And since we had no significant data in our backlog yet, it was pretty hard to be interested. Looking back, we now know it is so important to decide how we structure and work with our backlog to support the team and provide us insights with fancy dashboards and cool graphics.

Anyway, just wanted to say hello to you old friend. Let us meet for lunch soon!
Your friend, Caroline

Hi Dixie, May 15, 2016
So good to meet again. Corelli's is my favorite lunch place!

I am back from a big reorganization meeting, and I just had to let you know what is going on. Things are crazy! Only yesterday, I was tooling along and feeling in control of my work and our nice backlog and our progress and everything. Well, "how long was Adam in Paradise?" as my Mom used to say. We are merging with another company and our project is becoming part of a major program with several projects joining to create a full-blown end-to-end client solution. Since our Bluebird project is the most mature and management has seen our successes delivering often and with high quality, we are to be the model for how the entire program is managed. And not just that, but I'm now the Business Analyst for the whole program!

I need to run as we have meetings all afternoon to figure out the impact and what we are to do. I'll be in touch and let you know how things are going.
Hugs, Caroline

(Text to Tommy) May 16, 2016
I will be home late tonight. Sigh. We have all these meetings to come up with a strategy for how to combine the roadmaps and requirements for the new program. Don't wait up.

Hi Cindy, May 16, 2016
Here are my notes and action items from our project planning meeting today:
- Investigate overlaps in requirements with other projects
- We decided not to change anything before we know more about how the program will be run
- Reach out to Ross about planning
Caroline

Hi Ross, May 20, 2016
I appreciate you making sure that I get involved early in the planning. I know it will take time to align various stakeholders and get clarity around various roadmaps and requirements. For now, I will look into how the various projects manage requirements and backlogs and get back to you with what I find.
With regards, Caroline

(Text to Tommy) May 23, 2016
Hi, let's go out tonight! I need a break – work is crazy!

Dear Dixie, May 25, 2016
I wish you were here to help! I have started investigating how we can practically bring together the requirements from the different projects. And the product owners are working on this from the business perspective. The more I look into the various projects, the more variation I find. I had no idea there were so many ways to do backlogs – correction: so many ways to mess up backlogs. I told Tommy the other day that maybe I should find an easier job. He thinks I should stick with this one as it is an opportunity to show what I can do. As long as Ross supports and gives me the authority to make changes I'll keep trying.

Have you ever dealt with this before – having to merge backlogs? Happy for any advice!
Yours, Caroline

Dear Dixie, May 27, 2016
Sorry, I realize that I did not give you enough details for you to comment. Do you really think that if the backlogs are so different and have a lot of content/history that is reasonable to let the projects keep their individual backlogs? That is an interesting idea I will discuss further with Ross.

Our situation is pretty complicated—not only are the backlogs structured quite differently, but they are maintained in different instances of the tool. Every project is using TFS, but some are in the cloud, others on premises, and versions are different. It will take time before they can all migrate to a common instance, which we need to do before we can even consider combining them into a single backlog.

And I promise, I will not just try but "give it my best try" as you always said! And thanks for the reference to Martin. I think we need some training and coaching help at this point.
Caroline

Hi Ross, May 29, 2016
Please find attached an overview of all the projects showing the current tooling, backlog structure, and team process (Scrum, Kanban, hybrids). There is so much variation. It will not be easy to come to a common solution quickly. If and when we decide on a common solution, projects will have to convert what they have into a new, consistent format. Only then will we get full benefit of a consolidated approach.

I reached out to an old friend who is an experienced BA for advice. She suggested we don't merge backlogs, at least not yet. Instead we could create a program level backlog to support planning and program level verification. She also recommended that we look into scaling Scrum since most of the projects use Scrum now as their development methodology. If we do want to scale Scrum, she recommends a very experienced coach and trainer Martin, who has helped her company recently.
Let me know when you have time to discuss, Caroline

(Text to Tommy) Jun 15, 2016
Will be home late again tonight. I'm helping QA produce metrics reports from all projects.

Hi Dixie, Jun 29, 2016
Sorry for the radio silence. We are humping along. I am tired. Ross felt it was too early for training. The focus now is on the business side and trying to align the projects and come up with program goals. Meanwhile, we are struggling with planning and QA and product-level metrics. I feel sorry for the guys doing system integration testing. We are kind of getting the end-to-end solution together, but it is hard for the system testing team to know what cross-product features should be working when each of the projects are developing and testing individually but never checking how they function in the overall product. Part of my responsibility is to produce metrics from the individual backlogs that show overall product status. This is a time-consuming, error-prone, manual process. We pull data from several sources and merge it. I am not even sure we are comparing apples with apples with these different backlogs.
Caroline

Hi Dixie, Jul ,1 2016
Thanks for the advice! I cannot understand why I did not take action on this before. Of course we need to make sure we share some common definitions even if the backlogs are different. Thanks again, Caroline

Hi Ross, Jul 1, 2016
As discussed, I need your help to ensure that our program level metrics make sense. When we pull data from each project backlog on completed users stories and bug statistics, we need each project to apply the same definitions for user stories being "done" and for bug criticality. I have attached drafts of the common definitions. If you agree with them, could you please send these out and ask that these definitions be applied consistently going forward?
Thanks, Caroline

Hi Martin, Aug 3, 2016
I got your name from Dixie. My company is interested in some help with scaling Scrum and handling program level backlogs.
Best regards, Caroline

Hi Ross, Aug 24, 2016
I agree. The 1-week workshop with Martin went very well and we have a good plan for moving forward. His proposal of creating a new backlog for the program level planning and monitoring is a promising way forward. Fortunately we will also avoid merging all the project level backlogs for now, although this does mean we still need frequent communication across projects. I'll start working on a program level backlog right away.
Caroline

Hi Ross, Aug 30, 2016
Here is a first pass at the structure for the program level backlog. As you see, it is purely focused on product features. We only added top nodes for user documentation/online help and for crosscutting quality requirements. Let me know what you think. We should have the first program-specific user stories ready for our program-planning event next week where we can try out the full flow from program level backlog to verified feature.
Caroline

At this point, the Bluebird program has decided upon the following solution – keep the project backlogs as they are and add a program level backlog to help with the overall system-level planning and integration.

Pattern: Pragmatic Program Backlogs

Your program is being newly formed to coordinate the work of several existing projects. You have the goal of presenting a more coordinated, consistent view of your products to your customer. You need to be able to make plans and answer questions about the overall status of work in progress.

You desire to create a view into the ongoing work of the various projects so that you can understand and manage dependencies between them and better coordinate deployments and feature releases. Several projects will need to be better aligned with the overall program objectives to improve the feature delivery in the end-to-end solution. New projects may need to be started up to fill some shortcomings and gaps in current products. All in all you have a number of backlogs at very different levels of maturity, likely structured in ways that are hard to consolidate, and with lots of contents that constitute the history of each project and so their data has to be preserved.

How do you manage a program-level view of the work of individual projects with highly different backlog implementations, so that you can better coordinate their work?

Overall, there is little consistency between projects and the way they manage their backlogs. Some projects have been running smoothly for a while and could potentially be models for how other projects should operate. These project teams have learned how to use the Application Lifecycle Management (ALM) tool in sophisticated ways and can easily answer sophisticated questions about their progress. They may even have a PIPELINE and a process to FUNNEL new innovations into their individual backlogs. Other projects, even though they may use the same ALM tool, have backlogs that contain items of highly varying granularity and content quality. These project teams largely use their backlog as a giant prioritized To-do list, and have trouble tracking progress or knowing when anything is done. Still other, smaller projects have only just begun to manage their backlogs electronically. Some project backlogs contain hundreds of backlog items.

Ideally, you desire a single program backlog shared by the projects. But practically, this isn't possible without a lot of rework restructuring existing projects' backlogs and developing a consistent set of practices around how they are managed. This rework may cause projects to slow down at a time when they cannot afford to lose momentum. Changing their backlog structures and contents may seriously disrupt and change existing work processes and thereby frustrate and even alienate the project teams causing their integration into the new program to be dysfunctional.

Some projects may have backlogs in different instances of the same ALM tool, but others may be using different ALM tools altogether. Merging the backlogs means you need to map items and attributes across tools, and there may not be tooling to support the automation of this migration.

Therefore, construct a separate program-level backlog, which consists of a high-level view into the individual project backlogs.

The program-level backlog is an additional backlog that has only the FRAME representation of the product solution to be built. The items in this program-level backlog are normally on the product feature (epic) level and will typically be implemented by several project-level user stories. These user stories will belong to multiple projects as shown in figure 1. All technical content (e.g. the enabling user stories along with their details and current status) reside in the individual project backlogs. Tests defined at the program level belong to the verification of overall system integration. Defects may be associated with the epic-level user stories if they represent issues with system integration. If not, defects will be raised against the appropriate project-level's backlog items.

Instead of tracking individual product backlog items, this program-level backlog enables you to manage the work at a higher level, focusing on major features and sets of related features. Each individual project will maintain its own separate backlog which contains project-level user stories and details. In order to get a picture of the overall program's status, details from each backlog will have to be distilled and translated into the status for program-level backlog items. This is by no means an automatic process,

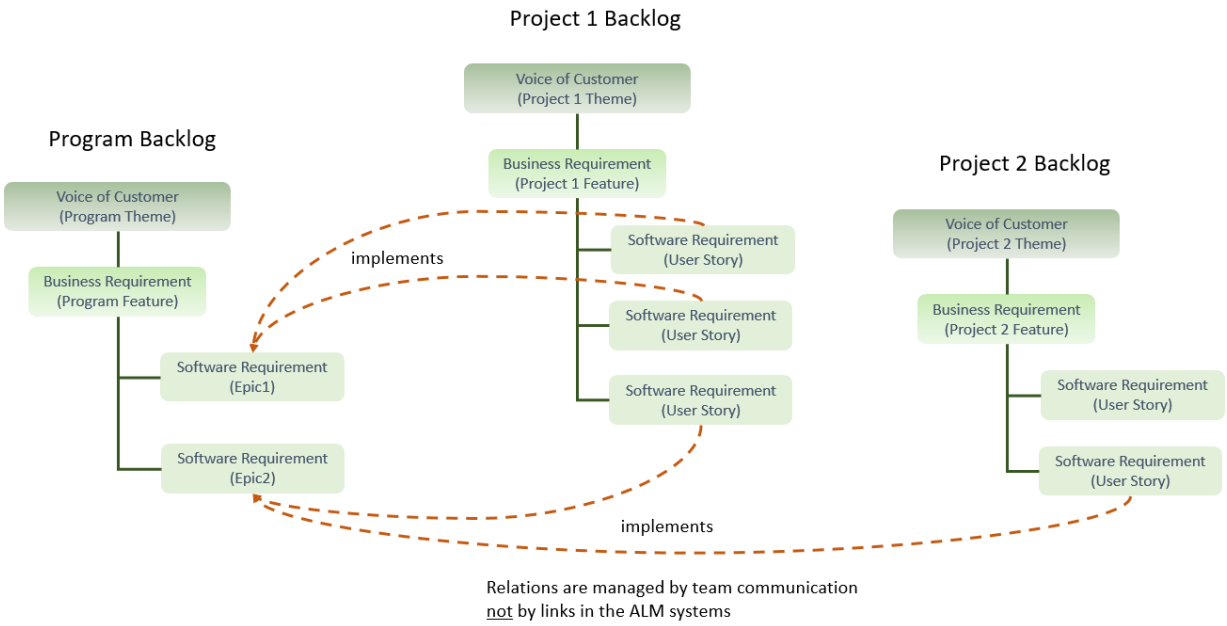

Figure 1: A high-level Program Backlog with Multiple (Unlinked) Project Backlogs

especially since the status of each enabling user story can only be found by digging into a particular project-level backlog. Depending on the sophistication of any project, this status can either be determined by asking questions of individual project backlogs and feeding ANSWERS into the current status of project-level features into program-level backlog items. Or, in the case of a project whose use of the ALM tool isn't consistent or structured into FRAMES, this might involve a lot of tweaking of information gleaned by the Product Owner by picking through details, which is then shoved into a spreadsheet and tabulated. Accurate status for an epic-level story when the enabling user stories are spread across multiple projects is even more problematic.

While not ideal, this ad hoc approach to updating the program-level backlog gets you some visibility into each project's status as well as a high-level view of the overall program's status. The major benefit with this approach is that it allows each project to keep its individual backlog format unchanged, while still enabling communication of the overall product status. It also helps support planning across the projects to work on joint features and manage dependencies in the implementations.

Because the project-level backlogs are not linked in the tooling, all dependencies and relationships are managed through inter-project communications. Consequently, these dependencies aren't explicitly represented in the program backlog. So they may be more difficult to identify and manage. Another major drawback of this approach is the difficulty of automating or ensuring the accuracy of any consolidated state of the overall product (for example, quality release criteria) drawn from separate backlogs that may reside in different tools and tool instances.

Although this pragmatic approach may seem fraught with potential problems, especially for a large program consisting of many projects, its benefits should not be overlooked. Instead of spending time trying to migrate various projects' backlogs into a common format, this approach allows you to quickly create a high-level backlog for the overall program. Since the user stories in the program backlog are at a high-level, you can focus on the big picture of what needs to be accomplished while not getting bogged down in lots of details. This high-level structure in the program backlog can also help you sort out what various projects actually are contributing to the overall product and identify overlapping work or missing work that hasn't been assigned yet to a specific project team. Over the long term, as the program grows in complexity, this simple approach may not be viable.

Caroline's story – part II

Dear Dixie, *Dec 19, 2016*

Just wanted to wish you the best for the Holidays and let you know I am still alive and kicking. Well, not literally kicking, ha-ha. Our program has been running for 9 months now and we are doing OK. Still there's lot of challenges, and I keep nagging the project managers to do a better job with their backlogs.

The program backlog has helped us a lot in the planning and verification process, but it is still really hard to know how we are performing. We manage to deploy roughly on a monthly basis now, and most deployments go well. But the last one was a disaster because we did not catch a major issue either during the project level verification or in overall system integration testing. We need

better automated testing, and need to keep it up to date as changes happen. The deployment included 2 new features. One turned out OK, but the second was influenced by some changes in the architecture that were not fully understood by the development teams. Oh, well, I did not mean to rant again. I hope we can meet in the New Year!
Your friend, Caroline

Hi Ross, Dec 24, 2016
I am looking at the latest metrics for the program from QA. As you see, we are finding more defects than we used to. It may be because of the architecture modifications recently, but I am not sure, as there was no spike in defects, just a steady degradation. Maybe after the new year we can discuss how to better align in our planning to make sure the user stories are more clear/detailed and then we might be able to improve.
Caroline

Hi Ross, Jan 10, 2017
I have been discussing with Cindy how we can better structure our requirements to improve traceability and find an easier way to generate quality metrics for the program. We think it should be possible to link items in the project backlogs and thereby know when program-level features are completed and what defects are associated. Can we discuss?
Caroline

Hi Cindy, Jan 12, 2017
Ross liked our idea, and proposed we run a workshop with the project managers to see how we can better integrate the backlogs. Next week?
Caroline

(Text to Tommy) Jan 22, 2017
Please buy some good wine for tonight. I am picking up dinner. I know you are tired of my program problems, but you are the one who told me to stay… we had a workshop today and I am so mad! Most of the project managers were OK, but they are just so unmotivated. Don't they understand that their projects are not successful unless the program is successful too? Anyway, it went OK in the beginning, but then Ivan just did not want to collaborate. His project never delivers! I think he just does not want better visibility and improved planning because it will be more apparent that his team is not delivering. Right now I am contemplating various ways to remove him from earth ☺…

Hi Dixie, Jan 23, 2017
I need to see you! Lunch?? Please!

Dixie, Jan 24, 2017
I cannot believe you are going to Australia for 4 months!! It sounds fantastic! So don't worry about my problems, I will figure it out. Ping me when you are back!
Safe trip, Caroline

Ross, Feb 28, 2017
We discussed last year having Martin come back for some coaching. When can we do this? I think it would be good to have some outside help look at how we are planning features. QA keeps finding a lot of defects (broken logic in the functionality) during system integration testing. This points to the program-level features not being implemented as they should by the projects. This is really slowing us down.
Caroline

Hi Ross, Feb 28, 2017
I do understand that we have budget limitations, but we really need to do something to correct the current situation. Frank (the new Product Owner) is asking me for a lot of data on how we are doing, and when features will be ready, and it is costing us a lot of time and effort to find the answers for him.

I think some help from Martin can really speed up our delivery – if he can convince all the project managers to play along. Some are already linking their user stories to the program backlog, and this helps us in automating the metrics. But we still have a couple of projects that need to migrate to the shared tooling, and especially one project is not playing along.
Caroline

Hi Ross, Mar 10, 2017
Thanks, then I will set something up with Martin!

Hi Frank, Mar 10, 2017
Thanks for talking with Ross the other day – he finally agreed to get us some help!

Hi Martin, Mar 11, 2017
We are back! Will you be able to spend some time with us in April/May to help us with backlog management? We have started to link project-level items to the program backlog as you suggested, but we have a couple of problem projects that are not there yet. One of the project managers is just very new and quite confused but the coaching will help there – I am not worried. The problem child is Ivan's project. I think he is pissed off that the program manager job went to Ross, and he is doing everything he can to make us not succeed. It must be hurting his career too, but he seems not to care as long as he can disrupt our ability to deliver. And he is so close with the program architect who keeps supporting him. Please keep this to yourself; I just want you to be aware of the situation. Can you please send us your availability and a financial proposal?
Caroline

Hi Frank, Apr 15, 2017
Glad you like Martin, yes he is very knowledgeable.

(Text to Tommy) May 3, 2017
I promise not to be late again tonight!! You are an angel to stick with me!

Dixie, you are back!! Jun 1, 2017
How was your trip? We must meet soon and you can tell me all about it!

Man, did I miss your support, it felt really scary not to have you here. So we have made big changes to the program. With help from Martin, even Ivan had to cave in and we now have all projects linking their user stories to the program backlog. But it has cost a lot of time and effort, and some fighting that left some scars. Martin has been very helpful; this would not have worked without his consulting. Still, I am not sure how much longer I am willing to work on this. I am nearly at the end of my rope. Tommy has been a great support, but he is getting really tired of my rants about the program, and I have to be careful not to bother him about it all the time.

On a happier note, I am going to the Agile conference this year!! So we will meet there? Have you read the works of Johanna Rothman, or attended her sessions on program management? I have a feeling we could do much better managing the program and the program backlog and metrics as well. I really want to attend her sessions this year and see if she can provide some more insights.
Caroline

Hi Dixie, Jun 4, 2017
I cannot believe you know Johanna! Do you really think I can ask her for advice? I often see people talking to speakers after sessions, but I have never had the nerve to do so. Maybe you can introduce us?
Yours, Caroline

At this point in our story, each project in the Bluebird Program has linked their individual backlog items to the appropriate program-level backlog items. This allows for a more accurate reflection of each project's status and how it contributes to the program. It has required a lot of convincing to get every project manager on board with the idea as well as a commitment to using common ALM tooling.

Pattern: Linked Product Backlogs

Your software development program consists of multiple projects that work together to create parts of a larger product. All projects are using the same ALM tooling, but each project maintains its own project backlog that contributes to the overall program. The software process details vary from project to project. Stakeholder activities like planning (roadmaps, business priorities) and delivery happen at the program level, meaning that there is a common plan that all projects share and collaborate on. Product features often require contributions from several projects, meaning that there is close communication across project boundaries.

How do you organize and manage the program-level backlog for a program consisting of a set of closely aligned projects when each project has its own mature and extensive backlog and these backlogs are structured differently?

Individual project backlogs have been built over time and they are structured to suit the needs of each project. They probably existed before the program was created, and were structured with no consideration of collaborating outside the project team. Because of their diversity, this means that moving to a single, unified backlog format will require a lot of restructuring and changes for each project. This will take time away from development and slow every project down for a while.

The project backlogs are typically aligned with the working process of each project team, and have been optimized over time to the project's style of collaboration. Any major changes to the backlog structure and contents will be disruptive.

The program wants some level of traceability between the user stories completed by the projects and the features (end-to-end operational workflows) that are enabled on the program level, but with each project operating with a separate backlog this is hard. The program also wants to automate metrics, and for this to work properly they need to know what work on the project level contribute to implementing requirements in the program-level backlog.

Therefore, create a program-level backlog and link project backlog items to the program level backlog for traceability.

This approach lets the individual projects have their individual backlogs structured to support their own way of working while still being able to automate ANSWERS on product completeness and outstanding issues. The typical backlog items to link from the individual backlogs would be user stories to the respective features, and test cases to test suites/test plans in the program-level backlog. This will provide you with a clearer picture on how the work in the project teams contribute to the overall product.

Technical items will likely also exist at the program-level. For example, a product-wide requirement to support a common API for micro-service monitoring might be a feature story in the program backlog which would be linked to work items in separate project backlogs. Or necessary refactoring to use a new time series based logging, which will cut across many services, most definitely would cause work items in each of the project backlogs, all linked to a refactoring item in the program-level backlog.

Linking items across backlogs will be easier if all the backlogs reside in the same collection in an ALM tool. But there are ways to link even if this is not the case. Some ALM tools support links to external items, although this will give a rather crude solution where gathering metrics will be harder (not supported in a single ALM tool instance so requiring manual work or writing some code to deal with merging the information), But there are products that support the integration of multiple ALM tools and allows for these items to be synchronized across the tool/tool instances (for example Tasktop's Integration Factory [19].

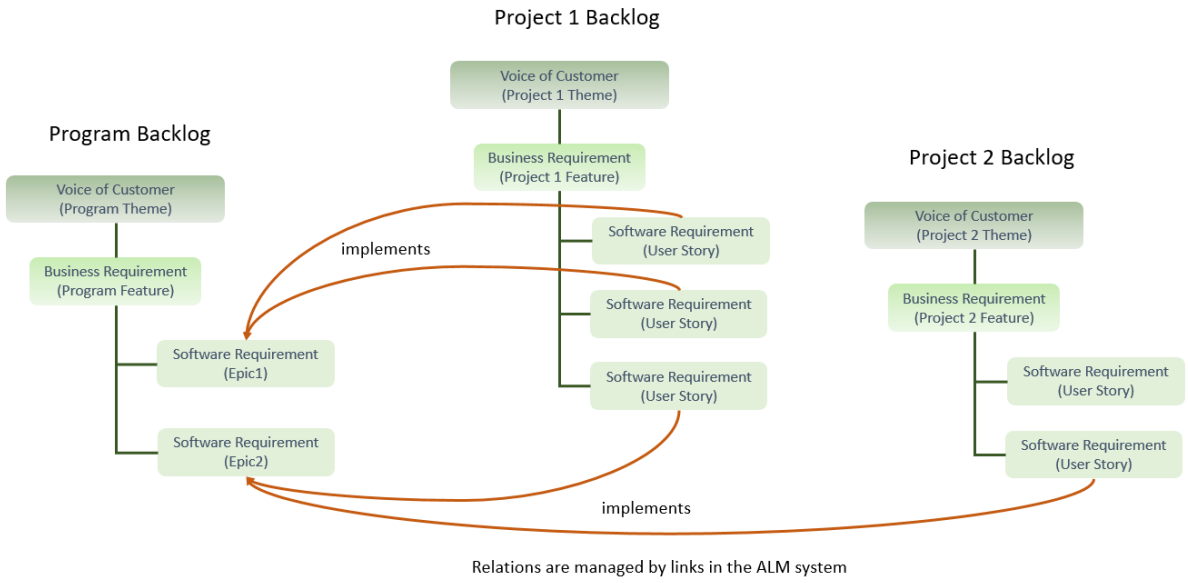

Figure 2: Appropriate Project Backlog items are linked to the Program Backlog

Creating links between project-level and program-level backlog items may be a challenging task in the beginning. You may need to migrate some projects' backlogs to a newer version of the tool or even convert them to use the common tooling. However, the real challenge is more likely to be getting projects to learn how to better collaborate. But as your overall program organization goes through a few iterations of planning, implementing, and deploying features together, the program team will gain experience that helps them decide how to break down items in the program backlog to project-level user stories, and how to deal with inter-dependencies between projects. Additionally, projects may choose to temporarily assign individuals to program-level feature teams to help deal with the dependencies and ensure completion of items at the program level. For example, this program-feature team may need to go beyond implementing and verifying any user story working at the project level to take on responsibility for the program-level feature being verified and successfully deployed.

It should be noted that there are possible ways to link items between different tools, as well as other synchronization solutions that are available in the market. These synchronization approaches come with added complexity but may be a plausible alternative if it is too difficult to migrate to the same tool and directly use the tool's linking capabilities.

Caroline's story – part III

(Text to Tommy) *Jul 19, 2017*
Hi there, just to let you know I have arrived safely at my hotel. The Agile conference starts tomorrow – I am so excited! Love you!

Hi Johanna, *Jul 20, 2017*
I really enjoyed your presentation. Thanks so much for answering my questions. I really like your ideas for considering options: come up with at least three options. We've already tried multiple ways to get us useful data to help us with planning future iterations and coordinating the work of dependent project teams. We started with independent project backlogs. Then we tried linking individual project backlogs to our program level backlog. Yet, we still have the problem that some teams have trouble accurately tracking their progress. What are your thoughts about moving to a single backlog for our entire program?
Sincerely, Caroline

Hi, *Jul 21, 2017*
I like the idea of a single roadmap, which shows all the feature sets as you think they might get delivered. Program managers need to see the program as a whole. And, if you have interconnected deliverables, you might need to see the feature sets as they evolve. Will the platform payments be done in time for the new banking experience? That kind of thing. I said I would send you a link. Here it is.
https://www.jrothman.com/mpd/2017/09/alternatives-for-agile-and-lean-roadmapping-part-7-summary/

Sometimes, several feature teams work on a single feature set. In that case, yes, absolutely create one backlog for the several teams. I've seen this work quite well for a small program of up to six teams. I'm not sure how well that idea works for more teams. The six teams sat very close to each other (not in one room, but in two team rooms down the hall from each other. They felt squeezed, but they were able to easily talk with each other. Each room had its

own PO. The PO's job was to keep talking with the teams, and accept stories (or explain why the story didn't work) every day. The product manager met with the POs every week as a Product Value Team. The product manager met with clients so he had the most recent client info. Every couple of weeks, he brought one of the POs to meet with local clients. Yes, the POs got out of the office.

Let me circle back to your question. Yes, I like a single, unified view of the program's possibilities and commitments. I think of a backlog as the "what we are sure we will do sooner, rather than later." To me, the backlog is a relative commitment. We are pretty sure we will do this work over the next few weeks, maybe month. I think of the roadmap as the list of possibilities and when we might start them. The roadmap helps people understand what we might do in the future.

People working on programs need to see both, because our decisions today affect our ability to consider later possibilities. The roadmap allows us to decide later (not sure if "as late as possible" is the right wording, but later works for me) so we don't paint ourselves into a corner. The backlog says, "Do this now, and think about this next bit as you proceed." That way, if we think we need architectural spikes or other exploration, we can put that work in the backlog. We aren't surprised when we get to that part of the program.

I don't know if that helps your thinking. Let me know if I confused you.
Johanna

Hi Ross, Jul 22, 2017
I got to talk with Johanna Rothman yesterday after her presentation. She is the renowned expert on running agile programs that I told you about, remember? I learned so much from her talk, and then she listened to my problems and sent us personal advice and pointers to where we can read more. I have lots of notes and ideas to share when I am back. This can really help us!
Caroline

(Text to Tommy) Jul 24, 2017
On my way home with lots of new ideas! But I also got time to think about what you said – we need to have more time together. I know work has taken too much of my energy lately. See you tonight!

Dear Dixie, Aug 5, 2017
First off, it was great to spend the time with you at the Agile conference. The insights about managing programs and the need for better metrics that we can act on is keeping me busy and excited!

Well, that was the god news. It is harder than I thought to engage my team in taking up new ideas. I guess they are tired of all the changes we have already made. But we have made progress. Some projects are now using the same structure as the program backlog. Actually, a couple of teams have even ditched their own backlogs now and only use the program-level one. For these, I can easily automate the metrics (incorporating some of the new ideas I got at the Agile conference). But Ivan, with the help of Jane, is actively opposing every improvement I propose. Sometimes I think they want us all to fail just so they can get it their way. But if we fail, then Ivan's project will also be in jeopardy – unless there is something I do not know that they do. Ross as usual is not engaging, or maybe he has other problems to deal with. I never really know with him.

And I need to think about what I want to do, personally. I feel that I am losing Tommy – we are drifting apart. I spend way too much time and energy at work, and my personal life is going nowhere. Maybe I should seriously think about another job??
Yours, Caroline

Hi Cindy, Aug 9, 2017
Thanks for your support in the workshop yesterday. It really helps others to take the lead from the Bluebird project as it is the most mature of them all. Hopefully we can convince all projects to migrate to the program backlog as you did.
Caroline

Ross, Aug 9, 2017
The workshop yesterday was another example of the many problems that hinder us working as a unified team across the program. I think the idea of trying to form feature teams as well as having a more flexible backlog structure will bring us a lot of value, and these are proven practices in the software industry. But we need all teams onboard, to make this work. Could you please speak with Ivan and convince him to try this??

Frankly, I am running out of steam and I really need all project managers to work with me and not against me. I need your support on this. And not only that, I need you to engage more with us to all to make this happen.
Thanks, Caroline

(Text to Tommy) Aug 12, 2017
I am seriously thinking I want to say yes!! Today was frustrating – I have had it! Your idea sounds like heaven.

Dear Dixie, Aug 13, 2017
I am tired to the bone, and although we are making slow progress I am not sure I am willing to go on like this. Half of the projects are now using the program backlog only to manage their work and we have automated the metrics and most of our planning and monitoring processes are working fine. But we still have the holdouts. Ivan. And Jane. And they do not even say hello when we meet in the corridor anymore.

Tommy wants to travel to Australia for 3 months. And he wants me to come too. Financially we are cool, but I would have to leave work. But I feel it is either go on this trip or no more Tommy. And I've always wanted to go Australia. What to do?!?
Hugs, Caroline

Dixie, Aug 14, 2017
I did it. I left. I am going to Australia!! Caroline

Hi Ross, Sep 30, 2017
Nice to hear from you! We are doing well and enjoying our adventure. Yes, I can recommend a good BA. Her name is Dixie,

and here is her contact info (…). Sorry to hear you are having some issues, but I am sure she can help you out. She runs her own company, so you would need to contract with her.
Caroline

Hi Dixie, Oct 10, 2017
So the roles are reversed now – I am the one on the outside. Just make sure you stick to the condition that you get to decide how the backlog is managed. And if they will not listen, well I am sure there are other customers waiting.
Caroline

Hi again Dixie, Nov 11, 2017
I can't believe it – so they are all using the one backlog now? Amazing. Well, I suspect it will take a few features being delivered before all the process quirks are ironed out. But that is great news! So Ross has been asked to transform to a DevOps process and deliver more frequently? This will be even more interesting!
Well, I am here and not there with you. But no regrets. I'm so happy. Today we go pick blueberries on the O'Leary farm.
Best of luck and I will let you know as soon as we are back.
Your mate, Caroline

The program has evolved to use a single program backlog. Individual program-level backlogs have been merged into one common backlog. This took some effort and the project teams had to agree upon consistent ways to manage their backlog items. At the same time, the program continues to look for ways to improve their process and allow teams using the common backlog to find unobtrusive ways to adjust backlog items in support of their unique ways of working.

Pattern: Linked Product Backlogs

You are part of a software development program with multiple projects that all contribute to producing a solution. The overall program is either in its initiation phase, or there are other reasons why individual project backlogs are small or non-existent or unimportant to preserve. Perhaps requirements are captured in documents, and an ALM tool is not yet in use to manage the backlog electronically. Or perhaps the use of such tool is in an early phase with limited contents at this point in time. Alternatively, you may have a number of existing project backlogs that are structured in a very similar way and so are easy to adjust to a common unified model. The reason that the development effort is organized into multiple projects is mainly due to the overall size of the effort. Each project is responsible for a functional area of the solution, but none of these projects produce software that will be delivered as separate products. The projects apply the same development process, and this process is applied for overall program as well. For example, the program applies a scaled Scrum process and each project applies Scrum at the project level.

How do you organize and manage the backlog for a program consisting of closely aligned projects when the projects have no current backlogs or backlogs that are small and structurally similar?

For the program, it is important to have a full view of the product features to be delivered, as well as their current state as they move through implementation and acceptance. This includes a view of defects and tests and other work items in the backlog that all contribute to the finished product.

At the same time, project teams typically want to easily find the relevant items for their specific project without having to deal with work items that belong to other projects, or get confused and distracted by a large amount of content that obscures their project-level view of the work.

All users need to easily navigate the backlog to find the information they need.

Project backlogs should be structured to support the specific development process applied by the project. For instance, if applying Scrum they want to easily see the user stories assigned to a sprint. The same is true at a program level.

No two projects are run exactly the same, even if they are part of a program and ostensibly use the same development process. There will always be some variation as individual project teams have different preferences and work styles. Enforcing 100% alignment is futile and will only alienate people to the process. Allowing project-level solutions that do not interfere with the overall program's approach enables the program to try out potential improvements. For example, individual projects may have different ways to deal with technical backlog items ("software plumbing" that must be done but do not provide explicit functionality), items capturing quality requirements, and other items not representing user functionality if these are unique to that specific project.

Extracting information for dashboards and reports requires the backlog to be structured to support this, and for the contents to be kept up to date so the extracted information conveys an accurate representation of the projects and the program. This also requires that core definitions are shared across all projects within a program, for instance the "definition of done" for a user story.

Therefore, define a single, unified backlog shared by all the projects within the program, but allow for projects to apply attributes, tags, and filters that provide them their own specialized project level view of the contents.

The FRAME and VIEWS are defined for the program, and all work items exist within the overall scope of the program. We recommend that program-level features be broken down to a level where they can be fulfilled by a single project. So to deliver a new feature, when the program team is analyzing and creating the associated user stories, they need to take care to split the functionality into parts that are granular enough to support this. This way the project sprints can run without being derailed by cross-project dependencies. Should the organization use a flexible model rather than the more static project model, you would assemble a team or

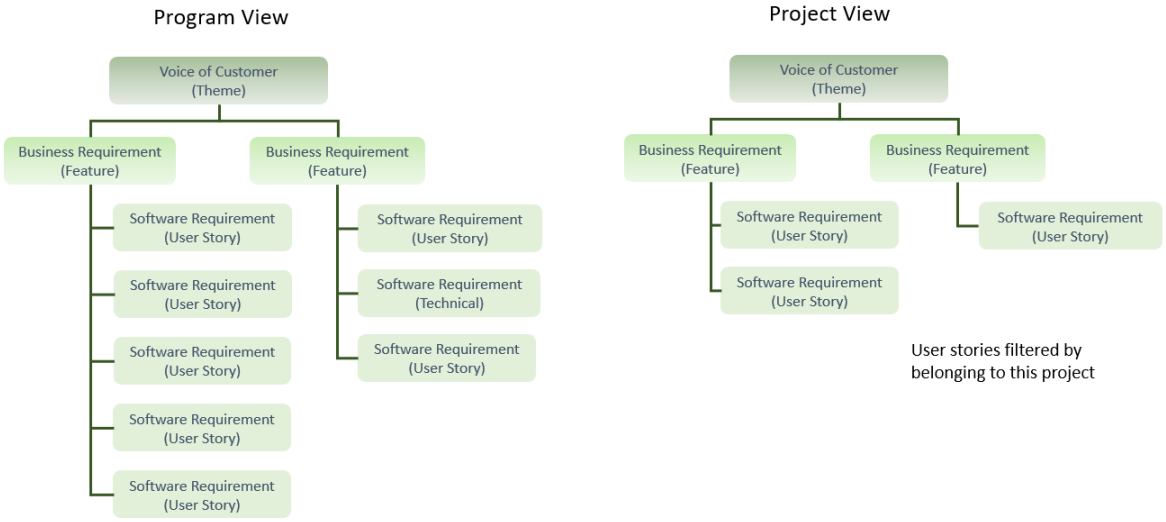

Figure 3: Appropriate Project Backlog items are linked to the Program Backlog

a squad [20] around each user story that takes it through to deployment. Structuring the backlog is influenced by the way the team and organization works. You want your backlog to support the desired development process and not be an obstacle.

Typically the program backlog structure has a set of goals, features, and user stories representing the user functionality. Goals and features are most likely shared between all teams, and there is no need to filter them at the project level. The items that you will most likely want to view and work with specifically on a project level are the user stories and the defects. So for these, you can use attributes (for example the "area path on Microsoft VSTS) to indicate either the project or team directly, or the domain that a particular team is responsible for. A complete program view and a filtered project view showing only the items that are the responsibility of a project are shown in figure 3.

Test plans and test cases are other work items that you may want to view at both a project and the program level. Most likely you will create test plans for projects as well as for the program, and the same test cases may be used by both.

A unified program-level backlog should cause less work and have fewer items to keep up to date than either PRAGMATIC PROGRAM BACKLOGS or a LINKED PROGRAM BACKLOGS approach. This is because overall structures like the FRAME and CONNECTIONS are defined once and reports are shared and can easily be created automatically. A UNIFIED PRODUCT BACKLOG requires all project teams to adhere to agreed backlog management practices. Even though all team members work on the same backlog, the potential for contents to deviate in style and granularity increases, as does the risk for the backlog structure to deteriorate over time. To mitigate these problems, we recommend that a single person (or small team if the program is very large) oversee the backlog and be the owner of the FRAME and the CONNECTIONS.

The business analyst role may be a good fit for this task, performing periodic MAINTENANCE. This role also typically leads the effort to groom the backlog contents and is familiar with the overall contents and *RULES*. The Product Owner will also be involved in this work, but for large products will normally not have time to perform all the detailed work but instead concentrate on defining the overall structure.

With a shared backlog, however, there is also a higher risk of changes disrupting many projects if the structure change or contents change. There is also less freedom for any project team to follow their unique style of working with the backlog, although small project level adaptations that do not interfere with the overall approach are possible and even encouraged. It is better to allow some variation than to alienate project teams by being too rigid with the backlog structure.

By having a unified backlog, it is easier to focus the whole program on the same goals (i.e. current work-in-progress selected from the shared backlog) and follow the progress of work items towards delivery. Defect tracking and reporting is also more consistent and reports are likely to be more accurate.

6 Discussion

When multiple projects exist in the context of an overall program, there is a need to coordinate and plan the effort of individual project teams. You need to be able to follow the progress and health of the overall product as well as plan and maintain a consistent roadmap for delivery that you share with all stakeholders. Although we document our backlog management patterns in the scope of a program/project structure, similar issues exist for any development effort where multiple teams work together to generate a product. We feel that other kinds of teams can benefit from understanding

and applying our program-level backlog patterns. These teams do not need to be working on different projects, but can be feature teams, squads, or whatever an organization uses to structure their coordinated efforts.

Since the product backlog, whether implemented as a single backlog or a combination of backlogs, serves as the system of record for the work done by and planned by the overall product development organization, a fundamental question is:

How do you organize and manage the product backlog for a multi-team organization to best serve involved teams as well as the overall program in planning, implementing and monitoring the emerging solution?

We ask this question in the context of product development of large systems that support complex operational processes, have hundreds of detailed requirements and possible safety and security concerns, and that may need to support external audits or prove that sufficient testing was done before deployment.

We started on our journey with this tantalizing question: When should you have one combined product backlog and when is it better to have multiple backlogs? Hopefully, nuanced answers to this question emerged throughout Caroline's story, as well as in details of the three patterns including forces, context, and consequences. Our patterns reflect three distinct ways to organize a product backlog for a program: a single backlog shared by all the projects, UNIFIED PRODUCT BACKLOG; a program-level backlog with links to individual project backlogs, LINKED PRODUCT BACKLOGS; and a program-level backlog with individual project backlogs that are not physically linked in the tool, PRAGMATIC PROGRAM BACKLOGS. Possibly there is also a fourth solution – that of only having project level backlogs and no shared backlog on the product level at all. But this is not a pattern that we have written because we have not seen it successfully applied for large, complex programs.

Table 1 summarizes the three patterns with benefits and resulting issues:

Table 1: Comparing Product Backlog Patterns for Program

Name	Description	Benefits	Issues
Pragmatic Program Backlogs	How do you manage a program-level view of the work of individual projects with highly different backlog implementations, so that you can better coordinate their work? Therefore, construct a separate program-level backlog which consists of a high-level view into the individual project backlogs.	✓Existing project backlogs can remain as they are ✓The program level backlog enables planning and reporting on the program level	– Disconnected backlogs increase the risk of badly aligned detailed requirements and gaps in functionality – Reporting (insights) on the state of the overall solution is more complex
Linked Program Backlogs	How do you organize and manage the program-level backlog for a program consisting of a set of closely aligned projects when each project has its own mature and extensive backlog and these backlogs are structured differently? Therefore, create a program-level backlog and link project backlog items to the program level backlog for traceability.	✓Existing project backlogs can remain with some changes ✓Linking work items reduce risk of missing functionality to fulfill a feature	– Some refactoring of project backlogs may be needed to make it clear which user story should be linked to a requirement in the parent backlog – Reporting should be easier that for independent backlogs but may still be somewhat challenging
Unified Program Backlog	How do you organize and manage the backlog for a program consisting of closely aligned projects when the projects have no current backlogs or backlogs that are small and structurally similar? Therefore, define a single, unified backlog shared by all the projects within the program, but allow for projects to apply attributes, tags, and filters that provide them their own specialized project level view of the contents.	✓Planning within the program will be easier with a shared backlog structure ✓Reporting will be vastly simplified compared to having individual backlogs	– Significant rework may be needed to merge the current backlogs into one – Maintaining the backlog contents may need dedicated resources to ensure it follows a clear strategy

One thing we should note is that not all experts agree on approach to managing program backlogs. From our experience, we have found that there is not a single solution that is optimal for every program. Which solution to choose depends on the context of the program and the situation of the individual projects, as well as the individuals involved and their forms of communication. The key to choosing a particular approach is to understand the trade-offs and forces involved, and to be able to select and adjust a solution to the actual situation at hand. And while one approach may work in the short term, it may be that as a program evolves, so does its backlog approach. In the following sections we extract out the essentials to help with this decision.

In Caroline's story, her program first chose to let each of the projects keep their own backlog and then manage the program planning with a separate program backlog. As the projects got better at joint planning and execution, the program then evolved to linking work items from the project-level backlogs to the parent program backlog. Eventually they decided they were ready for a single backlog. This storyline helped us illustrate the differences between the three solutions and the considerations and trade-offs that are involved in the decisions, but we feel it is a realistic path for a program that starts off by joining a number of more or less mature projects. For a program that is up and fully running from its inception, planning and monitoring the product development progress would be easier starting with a single backlog that includes all work items for the entire program.

When evaluating the approach for managing a program backlog, the decision tree in figure 4 may be helpful. Note that we have not created this with a path from one solution to another (like in the story about Caroline), but rather suggest to start by re-asking the first question if you are finding that your program's context is changing.

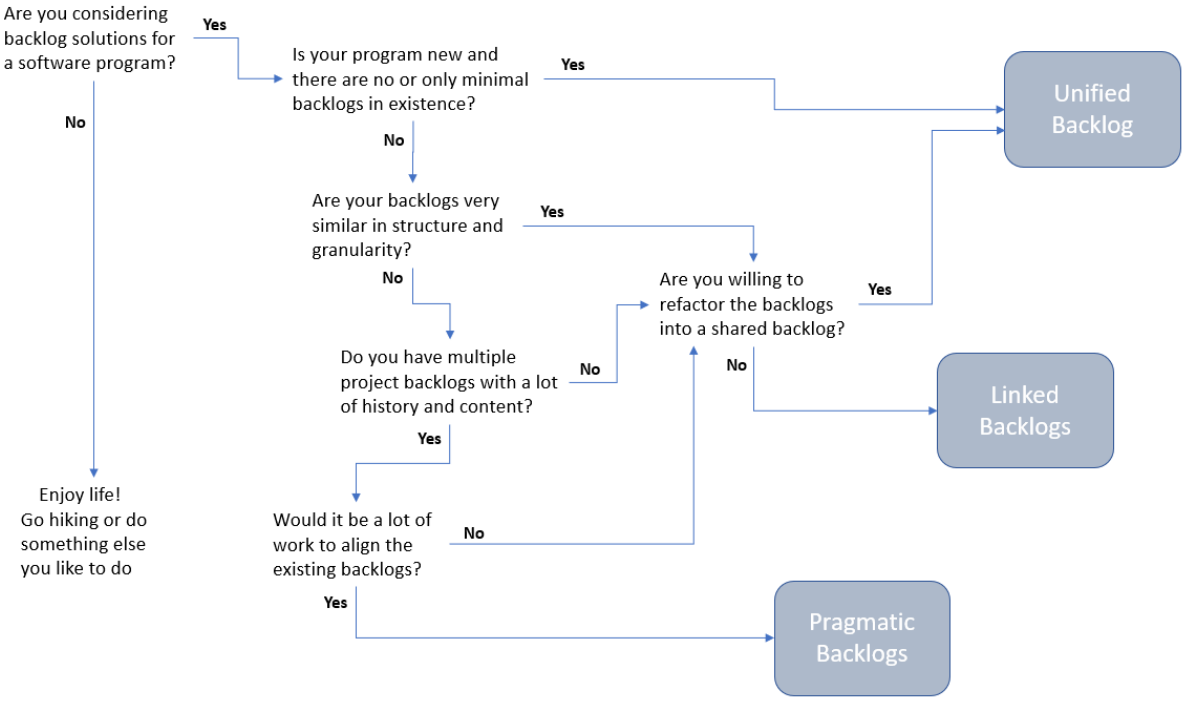

Figure 4: Decision Tree for Selecting among the Program Product Backlog Patterns

7 Summary

This paper is part of a larger work giving guidance to software development teams in creating and managing product backlogs. Earlier papers have covered patterns and pattern sequences on the project level, while this paper focused on program-level backlogs. In our previous writing we have experimented with different ways of illustrating the patterns with a story. In this paper we tried a more personal approach, following the story through the eyes of the program Business Analyst, Caroline.

A future paper to be workshopped at PLoP 2018 covers the remaining patterns we have identified but not yet documented (*REMODEL*, *RULES*, *DEFINITIONS*).

Once we finish this collection of individual patterns we are then considering to work all the backlog pattern papers into book form.

ACKNOWLEDGEMENTS

We are very grateful to our shepherd, Allan Kelly who asked us many deep and challenging questions. Allan forced us to deal with the concepts and solutions more thoroughly and made the paper much better. We would also like to thank our EuroPLoP 2018 workshop participants Michael Weiss, Michael Kopf, Frank Frey, Andreas Fiesser, and Klaus Marquardt for their valuable feedback – we have done a number of changes that should improve the clarity of the writing and better explain the concepts.

To gain a better understanding of software requirements and the processes around requirements engineering we have consumed a lot of literature, and we especially appreciate Karl Wiegers' writings on Software Requirements [10], Jeff Patton's work on Story Mapping [21], the Scrum Guide [11] by Ken Schwaber and Jeff Sutherland, and Ellen Gottesdiener's and Mary Gorman's workshops and books [5, 6, 7]. As we embarked on covering the program-level product backlog, we explored the various Scrum of Scrum models and as well as the program management book [13] and blogs [22] from Johanna Rothman that were extremely helpful to us. Thanks again, Johanna, for agreeing to be part of Caroline's story, providing your expert opinion and making us think deeply about having at least three options.

Appendix A The Backlog Patterns Collection

The Magic Backlog patterns collection are patterns that give practical help on building a good quality product backlog for a software project or program. The need to structure and manage the backlog and the associated development workflows with some degree of formality increases with project size. Note that the specific context of the Magic Backlog collection of patterns is backlogs for products of significant scope and complexity, with at least a three-year time frame for gradually delivering the full system. These projects have little choice but to use professional tooling and dedicate time and resources in backlog management.

In two earlier papers [2, 3] we have documented twelve of the patterns that are part of the Magic Backlog collection. These patterns are fundamental in that they provide the needed basis for backlog creation and management. As we explore additional contexts our patterns collection is gradually growing more complete. One area we wanted to get deeper into is the additional demands on the backlog in the case of programs rather than projects, which is the focus of this paper.

When developing a set of patterns, you realize that there are patterns in other pattern collections that relate to/work with/expand/overlap with the patterns in your own collection. This is great, because it is all part of our body of knowledge, and finding related patterns often strengthens the trust in your own patterns being useful in a shared context. But you are also faced with some challenges:

- The time and effort in finding related patterns and understanding their interaction with your own. Although there are some preliminary pattern catalogs, most patterns are found by researching the contents of papers, books, and online resources. This is very time consuming and hard to achieve for pattern authors who are not academics.
- The trade-off between incorporating patterns from other authors into your writings versus writing your own version of those patterns. Patterns from other authors may differ in style and context so although they are relevant they may not fit well into your collection. They may need some tweaking to be fully applicable for the type of problem you are addressing. But rewriting a large number of patterns that are already well documented is a bit of a waste and will make your paper/writing bloated.

Table 2 presents the patterns currently identified as part of the Magic Backlog Patterns collection in short form. For the full documentation of the first twelve patterns please refer to our two earlier papers [2, 3].

Table 2: Backlog Patterns Overview

Pattern name	Description
FRAME	How do you organize the main structure of the backlog to best provide the benefits of a quality backlog to a variety of users? Choose a backlog structure that represents a functional breakdown of your system. Create a hierarchical structure and link items in this structure in a way that best represents the product to the backlog users. A functional structure is a model that most likely aligns the understanding for most roles on the development team.
VIEWS	How can the backlog provide representations of a product that is intuitive to a variety of user roles? Create additional backlog structures to reflect alternate views of the product, for instance an architectural view and a quality view. Lower level backlog items can be linked both to items in the functional product structure (the Frame) and to items in the alternate structures. As an example, a User Story can be linked both to a main Feature (in the Frame) and to a Subsystem (in the architectural view).

Pattern name	Description
PEOPLE	How can you represent the various aspects of your system's users in a backlog? Create backlog items for personas to cover the dimensions of user profiles, and associate the personas with the appropriate functional backlog items. Their descriptions are then readily available for any team member with access to the backlog. Either tag a user story with the name of the persona, or link the persona backlog item to the functional item.
TALES	How can you improve the understanding of how users interact with the system and the impact on dependencies between individual user stories? Include narratives that give a free-form representation of product usage in your backlog. Most likely your narrative will span multiple user stories, and the natural level to link it in is to the feature level. The actual text for the narrative is captured in a document which is then uploaded as an attachment to the narrative backlog item.
USAGE MODELS	How can you improve the understanding of how individual user stories contribute to a business transaction or user goal? Enrich your backlog with models that provide a structured representation of product usage. Each usage model represents a business transaction or a use of the system as a whole to accomplish a complex task. The purpose of the model is to improve your understanding of how the system is used and provide a tool to prioritize, plan, and verify your product deliveries. Possible models are Use Cases and Business Process Models.
PLACEHOLDERS	How can you represent partly unknown functionality in your backlog? Create temporary backlog items as placeholders to be exchanged for detailed items later, when they have been elaborated. When the detailed items are created, you will want to replace your placeholder backlog item with the new detailed items. If you instead keep the placeholder item and link these details to it, you will increase the levels in your backlog thereby making querying and backlog maintenance that much harder.
PLANS	How are the backlog items associated with your plans for delivery? Associate the detailed requirements slotted for the next delivery to an entity representing this delivery. Tools normally associate backlog items with iterations and releases by using a planning-related attribute on backlog items. Backlog contents can then be filtered based on the values of this attribute to produce lists of items for a specific release.
CONNECTIONS	How can you explore the diverse contents of your ALM system? Create connections from other item types to the appropriate requirements backlog items. You want to establish these connections systematically following a defined model, normally linking tests to requirements, defects to both requirements and to the tests that detect and/or verify the defect resolution, and change sets to the requirements they implement or defects that they resolve.
ANSWERS	How can your team gain insights about the product from the backlog? Create shared queries and reports that can be reused by your team. The primary focus when extracting information from the backlog should be on the direct development team needs, and not stakeholders. The goal is for the core team to always know where they are and be able to prioritize their efforts on the most pressing work.
PIPELINE	How can you ensure that you always have some backlog items with sufficient maturity to enter the development process? Design a process that creates a steady stream of prepared backlog items. The process works as a pipeline that steadily refills the backlog with items with enough detail to be meaningful to the developers.

Pattern name	Description
FUNNEL	How and when do you introduce new product ideas into your backlog? Keep a list of future product ideas to explore that is separate from your Product Backlog. When an idea has been accepted into the product scope and has matured enough to be represented by epics level items, then introduce these into your Backlog. Expect that a good portion of product ideas will never be fully developed. Some may be discarded early after limited investigation either because they cannot be supported by a business case, because they are too costly to develop, or because they just do not fit into the portfolio.
MAINTENANCE	How do you keep your backlog as a reasonably accurate representation of the planned and implemented product? Regularly and consistently maintain the backlog contents. Maintaining the backlog is more than adding details and updating statuses. New contents need to be added as new requirements are elicited. Business priority changes will adjust the user story sequence/iteration planning. A maturing understanding of the product may require refactoring of the structure for the *Frame* and the alternate *Views*. Objects and attributes that the team uses for its planning and metrics need to be updated as the items go through the *Funnel* and the *Pipeline* and then through implementation/verification, making sure that structure and attribute changes caused by new material is consistently applied across the full set of contents.
REMODEL	*Refactor the backlog to refocus the contents*
RULES	*Who can do what and how in the backlog*
DEFINITIONS	*Sharing definitions to align contents and states of work items*
UNIFIED PRODUCT BACKLOG	How do you organize and manage the backlog for a program consisting of closely aligned projects when the projects have no current backlogs or backlogs that are small and structurally similar? Define a single, unified backlog shared by all the projects within the program, but allow for projects to apply attributes, tags, and filters that provide them a specialized project level view of the contents. Typically the program backlog structure has a set of goals, features, and user stories representing the user functionality. Goals and features are most likely shared between all teams, and there is no need to filter them at the project level. The items that you will want to view and work with specifically on a project level are the user stories and the defects.
LINKED PRODUCT BACKLOGS	How do you organize and manage the program-level backlog for a program consisting of a set of closely aligned projects when each project has its own mature and extensive backlog and these backlogs are structured differently? Create a program-level backlog where you keep individual project backlogs, and link project backlog items to the program level backlog for traceability. This approach lets the individual projects have their individual backlogs structured to support their own way of working while still being able to automate ANSWERS on product completeness and outstanding issues. The typical backlog items to link would be user stories and test cases in the individual backlogs to the respective features and test suites/test plans in the product level backlog.
PRAGMATIC PRODUCT BACKLOGS	How do you manage a program-level view of the work of individual projects with highly different backlog implementations, so that you can better coordinate their work? Construct an additional backlog that has only the FRAME representation of the product solution to be built. The user stories in this program-level backlog are normally on the epic level and will typically be implemented by several project level user stories. These user stories will belong to multiple projects. Instead of tracking individual product backlog items, this program-level backlog enables you to manage the work at a higher level, focusing on major features and sets of related features.

REFERENCES

[1] Lise Hvatum. 2014. "*Requirements Elicitation using Business Process Modeling*" in Proceedings of the 21st Conference on Pattern Languages of Programming (PLoP 2014), ACM Digital Library, New York, NY.

[2] Lis Hvatum and Rebecca Wirfs-Brock 2015. "*Patterns to Build the Magic Backlog*" in Proceedings of the 20th European Conference on Pattern Languages of Programming (EuroPLoP 2015). ACM Digital Library, New York, NY.

[3] Rebecca Wirfs-Brock and Lise Hvatum. 2016. "*More Patterns for the Magic Backlog"* in Proceedings of the 23rd Conference on Pattern Languages of Programming (PLoP 2016), ACM Digital Library, New York, NY.

[4] Ian F. Alexander and Ljerka Beus-Dukic. 2006. *Discovering Requirements: How to Specify Products and Services*. Wiley, West Sussex, England.

[5] Ellen Gottesdiener. 2002. *Requirements by Collaboration*. Addison-Wesley.

[6] Ellen Gottesdiener. 2009. *The Software Requirements Memory Jogger*. GOAL/QPC.

[7] Ellen Gottesdiener and Mary Gorman. 2012. *Discover to Deliver: Agile Product Planning and Analysis*. EBG Consulting.

[8] Rosemary Hasslenhopp and Kathleen B. Hass. 2008. *Unearthing Business Requirements: Elicitation Tools and Techniques*. Management Concepts.

[9] Karl Wiegers. 2009. *Software Requirements* 2nd Edition. Microsoft Press.

[10] Karl Wiegers. 2006. *More about Software Requirements*. Microsoft Press.

[11] Ken Schwaber and Jeff Sutherland. 2013. *The Scrum Guide*. http://www.scrumguides.org/, captured October 31, 2018.

[12] Scholastic. 2015. *The Magic School Bus*. https://www.scholastic.com/magicschoolbus/books/index.htm, captured October 31, 2018

[13] Johanna Rothman. 2016. *Agile and Lean Program Management: Scaling Collaboration Across the Organization*. Practical Ink.

[14] Kurt Bittner, Patricia Kong, and Dave West. 2017. *The Nexus Framework for Scaling Scrum: Continuously Delivering an Integrated Product with Multiple Scrum Teams***.** Addison-Wesley.

[15] Craig Larman and Bas Vodde. 2008. *Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum.* Addison-Wesley

[16] Craig Larman and Bas Vodde. 2016. *Large-Scale Scrum: More with LeSS.* Addison-Wesley.

[17] Dean Leffingwell. 2018. *SAFe 4.5 Reference Guide: Scaled Agile Framework for Lean Enterprises (2nd Edition)***.** Addison-Wesley.

[18] Tuckman, B.W. 1965, "Developmental Sequence in Small Groups", https://web.archive.org/web/20151129012409/http://openvce.net/sites/default/files/Tuckman1965DevelopmentalSequence.pdf

[19] Tasktop, https://www.tasktop.com/, captured October 31, 2018.

[20] Henrik Kniberg and Anders Ivarsson. 2012. *Scaling Agile @ Spotify* retrieved October 31, 2018 from https://blog.crisp.se/wp-content/uploads/2012/11/SpotifyScaling.pdf

[21] Jeff Patton. 2014. *User Story Mapping: Discover the Whole Story, Build the Right Product*. O'Reilly.

[22] Johanna Rothman, blog Managing Product Development, https://www.jrothman.com/blog/mpd/, captured October 31, 2018.