Who Will Read My Patterns?

On Designing a Patterns Book for Target Readers

REBECCA WIRFS-BROCK, Wirfs-Brock Associates
LISE HVATUM

As patterns authors, we spend a lot of time writing. Usually our work is driven by what we want to share of experiences, working solutions, and ideas we are excited about. But how often do we think about the readers other than initially stating our intended audience? Who is going to consume our writing, and are we really putting in the effort to reach these readers in the best way possible?

This paper started because we are two authors who plan to make a book from a set of related patterns papers, and in doing so we really want to reach our target audience and provide a practical and useful piece of work. Our approach had two distinct investigation activities. One was to do an informal analysis of books about software processes and methods to gain a better understanding of characteristics and book design practices for this kind of book. The second was to use a combination of interviews and surveys to gather feedback on what readers are looking for when accessing information about software engineering. To better understand our potential readers and how to design our writing for their consumption, we created personas where we incorporated feedback from the reader survey into their personalities and preferences. Finally, we combined our information sources to create guidance for our future work of turning a set of patterns papers into a cohesive whole.

Categories and Subject Descriptors: •**General and reference~Document types~General literature**

General Terms: Patterns, Requirements

Additional Key Words and Phrases: Software book design

ACM Reference Format:

Wirfs-Brock, R. and Hvatum, L. 2019. Who Will Read My Patterns? On Designing a Patterns Book for Target Readers. 26th Conference on Pattern Languages of Programming (PLoP), PLoP 2019, Oct 7-10 2019, 21 pages.

1. INTRODUCTION

Since 2015, we have worked on a collection of patterns for creating and managing a product backlog for software development. A product backlog is a set of work items that constitute the work needed to build a software product, e.g. items representing features, bugs, tests, etc. Our work is focused on larger software projects that have backlogs of a size and complexity that require digital tooling to manage them. At this point, we have documented the patterns that we have found so far, and we are starting to integrate the patterns papers presented at EuroPLoP and PLoP [Hva2015, Hva2017, Wir2016, Hva2018] into a single, cohesive work.

In our papers we deliberately experimented with different techniques to illustrate the application of the individual patterns through examples. In one paper we used e-mails and text messages to and from a main character. In other papers, we used storytelling or a set of interconnected examples. Although these experiments seemed to work in individual papers (based on feedback from our pattern readers at PLoP and EuroPLoP conferences) we do not really have any good measures to tell us what worked best. We also do not know how these techniques will work out for a consolidated group of patterns since the work will be so much longer than a typical patterns paper.

Looking back over the five years working on these patterns, the whole process around the pattern papers was a massive learning process on our part, with critical feedback from shepherds and workshop participants. Not in the least, was learning how to effectively put our ideas into writing and to blend our experiences. Although we believe the publication of papers significantly helped us to clarify our concepts and make the work more accessible to readers, so far we have not really put ourselves into the shoes of our intended reading audience.

It is clear to us that we want and need to spend time and effort on designing our final product. In doing so, we want to drive the structure and style of our work based on the needs of our readers rather than from our desires and aspirations as writers.

2. APPROACH

Starting from our motivation as explained in the introduction, we decided to follow a method adapted from the user experience (UX) community using personas and stories. Personas are fictional characters created to represent users (in our case to represent readers) and explore their needs and goals, as well as their personalities and behaviors. Stories or scenarios are then created to illustrate use of the product by the defined personas. This builds a better understanding of the use of the product. It also allows the designer to get some distance from the product itself and gain a broader perspective (e.g. the persona is using the product and the designer is "observing" them). UX techniques are increasingly used to design software products, and since our book is a product in the software domain this was a natural way of thinking for us.

In summary, the flow of events we pursued (and will continue to pursue) is as follows: data collection and analysis (research), creating persona descriptions, developing scenarios or stories, followed by developing design prototypes. A part of this flow should be approvals of those design prototypes (e.g. creating a hypothesis with a design and testing this out by sharing our writing with select user representatives). This latter activity is part of our future work and outside the scope of this paper.

We set out to do informal research in two areas: reviewing books in the same domain as the one we plan to write, and getting feedback from colleagues and friends who are readers and authors of such books. We stress the word informal, as we are both practitioners and not academics. Our approach follows no scientific method. Our fundamental goal is to gain a better understanding of how books connect with and bring value to readers, and then use this to guide our own work.

The purpose of the book review was to analyze what we think is a representative collection of books to extract ideas from which can help us with our own design. This may be compared to apprentices studying the masters. To get feedback from people we created a survey with rather broad questions, and sent this out to a number of contacts. We ended up with around twenty responses that we then tried to summarize and compare to gain further insights. We now had ample materials to work from, and could compare the readers' preferences with how various authors had designed their books.

We then used the input from our real readers to adjust the behavior and preferences of the personas created to represent our potential readers, and wrote stories of these personas reading our (so far imaginary) book and using what they read to do their job. Finally, we created a first outline of our book and agreed on the most important principles of book design that we will apply for our own writing, hopefully creating a book that real readers will find a decent piece of work.

To keep the paper shorter, and help readability, there are two appendices to this paper. The first is a description of the books we used for our research. The second is a more complete list of the design principles that we gleaned from our research. See, we are applying our design thinking to this paper!

3. WHAT WE LEARNED FROM THE BOOKS

The books that are most relevant to our work fall into the category of software processes and practices. Our selection is a mix of books that we perceive to be fundamental books in our industry, as well as recent writings. We also included business/organizational books often referred to by software professionals because they describe practices applicable to software companies and organizations. We have some writings that are not full-fledged books but articles. The selection of books was influenced by what we have access to, but since we have both been piling up books for years we do not think that our final list was too limited. Several books mentioned when gathering feedback from readers were books we already had selected for our review.

Let us first introduce the books, before discussing design elements and solutions as the outcome of our research. Note that a lengthier introduction of the books can be found in Appendix A. We grouped the books into the following categories: fundamental books, books on specific processes, books for specific software professions, books on DevOps (representing a hot topic), patterns books, and business books.

Figure 1 Some of the selected books

- Fundamental books (classics): The Mythical Man Month, Software Requirements, The Deadline: A Novel about Project Management, Retrospectives
- Books on specific processes and methods: Extreme Programming Explained, The Scrum Guide, The Nexus Framework, Large-Scale Scrum, Writing Effective Use Cases
- Books for specific professions: Code Complete, The Pragmatic Programmer, Refactoring, Software Systems Architecture, Agile Testing, How to Break Software, Lessons Learned in Software Testing
- DevOps books (representing current hot topics): The Phoenix Project, The DevOps Handbook, Site Reliability Engineering, The Site Reliability Workbook, Accelerate
- Business books: Agile and Lean Program Management, The Goal, The Lean Startup
- Patterns books: Design Patterns, the Patterns of Software Architecture (POSA) Series, Analysis Patterns, Domain-Driven Design, Organizational Patterns, Fearless Change, Business Patterns for Software Developers

By no means do we argue that this is THE list of books to read (if you want that list there are several web pages that recommend such lists. It is simply the list of books that we ended up processing. We are also not looking at classifying individual books as "good" or "bad." Every book had something to teach us and gave us ideas of book design practices. And each book had something that could have been done differently and arguably better, although that judgment is based on individual readers' opinions. Finally, I, Rebecca, didn't argue for including my own books in this list, although they certainly influence the lens through which I examine other books.

Technical books are certainly different from novels, and while (we think) that writers of fictional work are in the profession because they somehow enjoy writing, authors of technical books are more driven by the desire to teach and share knowledge, possibly with a pinch of desire for peer recognition added to the mix. The wish to share does not necessarily make one into a good author, and authors of technical books normally write one or a few books while spending most of their time working in their profession. This may be one reason some books can be a hard read even if the content is very valuable to the reader. This brings us to the problem of trying to classify what makes a good software process book. Is it the readability or the contents? Is it its timeliness or its ability to survive over time? We are not trying to reach a conclusion to these questions in this paper, but rather to look at elements and practices in the design of books on software processes and practices that can help guide us as we write a book that hopefully is a good read to a decent audience and that stands up over time.

We need to clarify one point—when we describe books as belonging to the software processes and practices domain we mean this in a broad way to differentiate between books about mindset, workflows and practices, and team collaboration from software books about specific technologies, programming languages, or detailed coding practices. Looking at our selection of books, this should be evident, but we still wanted to state this to avoid any misunderstanding.

3.1 Book Styles

The first thing we realized is that there are different styles of books. What style to choose is driven by the purpose of the book, as well as by the communication style of its writers. We believe that being both conscious about what style to select and aware of the benefits and challenges is necessary when finding a book style that fits with the authors' goals and abilities. In table 1 we show examples of books for each style.

STORY-TELLING

In these books, the goals of a methodology gradually become clear through stories, either as examples or introductions or authors personal experiences, or in the ultimate way as full-blown novels. This style may make the advice of what to do more obscure—the reader needs to extract these essences out of the story and find out how these ideas might work in their own context. But this style allows for a more intuitive and personal understanding of the whys, and it leaves more to the imagination of the reader, which is more fun and challenging than books that are very prescriptive. To be successful with this style, especially if creating a book that is a full-blown novel, the authors must be able to write prose well.

MINDSET/PHILOSOPHICAL

These writings (whether the whole book or the introductory part of a book) are focusing on the way of thinking rather than specific activities. The challenge here is to get the reader to understand a value system and goals, and doing so in a way where the reader can see how this thinking can apply in their own organization and to their specific context. You really cannot tell people how to think. Reading the text must gradually build up the reader's own perceptions, as well as internalizing the authors' values. Some readers may be able to attend workshops or talks by the authors, and even engage in a dialogue where misconceptions can be weeded out. But most readers will only have the book as their guide.

PRACTICAL GUIDES

Most of the software process books fall within this category, describing a number of activities in enough detail that the readers can practice them. In many cases, these books become tools that the reader repeatedly refers to when performing the practices. Some guides have a mindset introduction part, but most of the book turns into a tool for the reader with guidance on what to do, and when. They can be somewhat weaker on the why's behind the practices. Sometimes the practical guide is a companion to other works that deal with the mindset.

REFERENCES

Some books really are collections of a specific type of content, as for example some of the patterns books. They differ from the practical guides in that there is no story involved in the sequencing of their contents, nor goal of building up a methodology through following the practices.

EMPIRICAL

Contents of empirically grounded books are built on a certain amount of research based on studying teams, organizations, or systems to extract out essential information. Rather than being the loosely founded ideas or opinions of the authors, they are based on empirical studies. Books can have elements of empirical contents, for instance including one or several case studies.

Table 1 Book Styles

BOOK TYPE	BOOK EXAMPLES
Story-telling (or elements of)	*The Deadline* *The Phoenix project* *Fearless Change*
Mindset/philosophical	*The Mythical Man Month* *Extreme Programming Explained* *Agile Testing* *Business Patterns for Software Developers*
Practical Guides	*Retrospectives* *Writing Effective Use Cases* *How to Break Software* *Fearless Change*

References	*Design Patterns* *POSA* *Org Patterns* *Software Systems Architecture*
Empirical	*Accelerate* *Org Patterns*

3.2 Book Design Topics

The next insight we gained was that there are a number of topics where the authors need to make conscious decisions when creating the design of a book. The style of book will impact these decisions. Let us explain the design topics:

OVERALL STRUCTURE

When writing a theater play, there is a certain recipe to follow: introducing the characters, setting the context, building up the suspension in the first act, and resolving it in the last, creating the finale. The design of technical books may be less evident and there is a lot of variation. However, good technical books typically have a structure that helps the reader navigate their contents. They often provide both an overview of the communicated knowledge as well as the details.

A common practice that works well is to start with an overview, which introduces the concepts and major ideas. If this is written as a standalone section, readers who are not ready to dig into the details can still appreciate the book, and possibly recommend it to others and/or put it aside for further study later. This method is often used in Practical Guides, to first create an understanding of the overall domain or methodology before being followed with the more direct advice or detailed techniques. The same structure can be applied within each chapter: starting with an overview explaining the more detailed concepts followed by the details of their application.

Both a book's contents as organized into chapters, and the order of those chapters have a big impact on readability. Although not a novel, there is still a story to be told or a methodology to be unfolded, and the reader needs to be exposed to the concepts and ideas in a logical sequence. Typically, this proceeds in one of two ways: from the basics to more advanced topics, or, from the start to the end of a development process timeline.[1] Books written by multiple authors, and not least where authors write individual chapters are especially challenging, with a higher risk of repetition and even potential conflicts or inconsistences in terminology and practices.

NAVIGATION

Navigation is related to structure in that a book with a good flow of content is also easier to navigate. But help with navigation can also be accomplished by explicitly explaining the structure of the book in the introduction[2], or by pointing the reader to particular sections targeting special roles or interests. The essential navigation tool is the table of contents (TOC). For the TOC to work well, it is important to give descriptive names to the chapters so that it is clear what they are about. Cool or catchy titles may increase the entertainment value, but at the cost of navigational support.

LENGTH

After years of reading, and being avid readers at that, we observe that the length of the book is a key element of its design. Too short, and a book is more like a booklet and may not gain the full respect as a book would. But we both have read too many books that are far longer than they needed to be. Some carefully designed repetition to help readers remember the essence (like a summary at the end of each chapter) is good. But for many books, readability would improve if the authors removed some repetition, and in some cases, removed sections that are not relevant to the main ideas of the book.[3]

[1] Although this ordering from basic or fundamental topics to more advanced concepts may cause the reader to skip important concepts that are stuck in the back of a long book. Eric Evans has remarked that the most important part of his Domain-Driven Design book was the second half and unfortunately, because of this book's length, not many readers got that far.

[2] Assuming that the readers read the introduction, of course.

[3] During the writing process of my second book [WM], I, Rebecca, felt I really had to include sections on documenting designs and handling exceptions. While these were great topics, they were not core to responsibility-driven design. Good chapters, but not essential ones.

USE OF ILLUSTRATIONS

Illustrations add to the understanding and they break up the monotony of the text. They need to be relevant, and they need to be accompanied by explanatory text. They also need to be intuitive enough not to require in-depth study to make sense. Finally, they need to work in color as well as in monochrome.

Illustrations in a book are part of creating a book's identity. It works really well when illustrations follow the same style throughout.

"HOW TO" EXAMPLES

When writing about software methodology rather than software programming or design, examples are probably best done as small stories (e.g. the examples are not code snippets). The story must be long enough to provide the reader enough information to understand how to use a practice without taking up too much space or including too many superfluous details. It also can't appear contrived or artificial. The story must connect with the core readers using a simple enough context so that it does not require implicit knowledge to follow.

One practice is to use a running example throughout the book. This makes it easier for the reader to follow (e.g. not changing the context for each example). But speaking from experience, this is much more difficult for the writer who has to creatively think how to weave the topics she wants to cover into a coherent story. Technical books that are written as novels have accomplished this, as the ideas are explained through one extensive example. However, the downside of this one example approach is that not all the advice may be so easily woven into a single storyline, especially if there are competing or alternate sequences of practices that could be followed.

CHOICE OF AUTHOR'S VOICE

Authors who share personal stories tend to increase credibility – it builds trust as the reader learn about the author and her personal experiences. However, some readers may not like a book that is too personal (they want distance from the author and desire that the content speaks for itself). So choosing what voice to use must be done with care. And if there are several authors, how best can you blend their voices? A related issue is if whether the author should address the reader directly, or keep the form more generic. Either way, the book needs to apply the same approach consistently throughout.

STYLE OF NOTES AND REFERENCES

Some academic works include references to such an extent that it affects readability. When writing a book for practitioners it may be better to keep references more low-key and not have too many footnotes.[4]

PATTERNS

Since our planned book has patterns, we need to think specifically about the above topics in the context of a patterns book. Unless we create a patterns book that is more of a reference type book, it is a challenge to get the appropriate balance between the flow and the collection of patterns, and to deal with the broader context and specific details without being repetitive within the individual patterns.

Another issue is how to include examples in a natural way. The practice of "3 known uses" in every pattern is not the most user-friendly solution. A running example and a good discussion of the application of the patterns may be better.

A patterns book, in contrast to a conference paper, has the benefit of dealing with a collection of patterns as a larger body of work. That enables the authors to set the stage by explaining the shared context outside the individual patterns. It also allows the authors to introduce typical pattern sequences and to tell stories that weave the collection together into the whole that "patterns people " long for.

There is no one solution for addressing all these topics in a book design. The appropriate way to deal with a particular topic depends on the context of the book, the needs of its intended readers, as well as the preference of the authors. Many books are likely written without an explicit decision about each design topic. Be we firmly believe that to make an effort into understanding and deciding on how to deal with these topics will benefit our final work.

[4] In the first book I co-authored [WWW], I, Rebecca, intentionally did not include any references. While object technology was relatively new at this point, we did have other sources for our inspiration. In hindsight, we should have pointed our readers to them, perhaps in an appendix. In my second book, for each chapter we decided (if there were any for the topic in that chapter), to include an optional section, Further Reading, which followed the Chapter Summary.

4. WHAT WE LEARNED FROM THE READERS

We started off considering what we like about technical books, being consumers as well as authors (Rebecca is an experienced book author, Lise is an aspiring book author). By interviewing each other about books we have read, and discussing what we appreciated with the these books and what we would have changed, we came up with a set of questions to send to people we know in the software community. The questions and a summary of the feedback are shown in table 2.

The primary readers of our patterns are roles that work extensively on managing requirements using digital tooling (e.g. ALM systems). Because we felt that our thinking around writing technical books was not particularly specific to this topic, we also decided to reach out to people that we know are reading books regularly and have a broader reference frame. Targeting only people in a Product Owner or Business Analyst role would limit the people we could ask for feedback and the books that we would analyze for design ideas without good reason. We ended up with 19 responses from a variety of roles and affiliations.

Going through the various answers we created a table extracting the essence of the responses. This is shown in table 2 below. Quotes from the feedback are used in Appendix B to support a number of book design principles. The feedback also helped in creating the personas and story in the following chapters.

Table 2 Summarizing the Responses

QUESTION	EXTRACTED COMMON THREADS
What makes you choose a software process book?	Recommendation from colleague or friend (someone trusted) Meeting or hearing the author talk at a conference or online webinar (either speaker is an author or the speaker recommends a book) Knowing the author Popularity Required for work
Is there a book that you would recommend to your colleagues and if so why?	Recommended books are typically classics or very new, they are mostly very known, and connected to the role of the person recommending (QA, architect, etc.) Books can serve as a way to build common understanding and vocabulary and so are recommended to colleagues as a way to share ideas, and given to clients as a way to promote concepts.
Do you have a personal favorite of this type of book, and if so what do you like about it?	Books that are: - Conceptual - Helpful (constructive) - Straightforward - Books as novels
Do you have a software process book that you read more than 2 years ago that still has an impact on your work?	Most have personal favorites that they return to, for example Fowler's Analysis Patterns, or The Mythical Man-month, or Extreme Programming.
If you think of a book that you did not particularly like or find useful, can you explain why?	Books that bring nothing new and have too much fluff Books written for the wrong reasons (marketing of company, or joining the hype bandwagon without proper experience in the topic) Hard reads (technical, good content but hard to access) Too much detail, not enough abstraction
If you have written books yourself, do you have any advice to share?	Use stories. Write less. Publish more. Create several small books rather than one big one. Use LeanPub to allow the book to grow. Write most of the book before you go to a publisher.
How much of your technical knowledge gathering is from reading books versus information from other sources?	Varies from 5% to 70%, so very individual, but many readers consume a larger amount of books and use this medium for the deeper insights. Other sources are conferences and online resources.

What medium do you prefer for gathering knowledge about software process/practices/tools (books, articles, blogs, other online sources)?	Many different ways to gain insights, but books are still important to most. Conferences or online resources may bring initial ideas, but the fundamentals in a well-written book last longer. Online resources are a strong contender for some (or straight out preferred). Audio books are a good alternative to some as you can listen while doing other things. About e-books, readers seem to either prefer or really dislike them.

As we reached out to colleagues and friends for feedback, we got many answers that resonated with our own opinions. These were good contributions that helped us organize our thinking and confirm our initial ideas. But we are individuals with different experiences and personalities, and from different organizational cultures. Some of the responses we received were eye openers that brought new insights and ways of thinking about aspects of books and mediums of learning in general. Below are some new insights that we extracted from the feedback:

People recommend books that are meaningful to them, books that they really care about, so the recommendation is a gift and a way of community building, of sharing ideas through a respected medium (a popular book, an acclaimed author).

> "The ability to improve stuff and help people enjoy life at work - that's probably my #1 attribute of great books"

> "I may hear a talk by the speaker, or a book is recommended to me by somebody whose opinion I respect."

The enjoyment of reading stories rather than dry facts was a common thread in the feedback. In general, people were most positive about books that engage and excite, that communicate with them and where the ideas resonate with what they are already thinking and doing but then providing additional insights and/or helpful practices. The book needs to fit onto the value system of the reader.

> "I love this book because it made me think"

It was clear that it is common to have a small set of books (a "tool box") that one returns to for ideas and practices, maybe learning something new or deeper, or freshening up on how to do something.

> "I continue to go back to "Writing Effective Use Cases" and "User Stories Applied" for defining functionality"

5. CREATING PERSONAS

Instead of doing user-centered design of a software product, we are doing reader-centered design of a book for software professionals. When designing for users (readers), a persona is a way of representing user roles or user goals in a way that both make the user more personal to deal with, and that can represent more than just the role itself. A persona can be a user of a certain type with a defined experience level and preference, and this helps the designer think about all aspects of users.

USER ROLES

Our product is a book about how to use Application Lifecycle Management (ALM) tooling to manage product backlogs for systems and development organizations that are beyond what can be managed by simpler means like post-its on a wall. Our readers can be anyone in the product development organization, but we specifically target the roles that deal the most with requirements: *Product Owner, Business Analyst, Senior software developer/architect.*

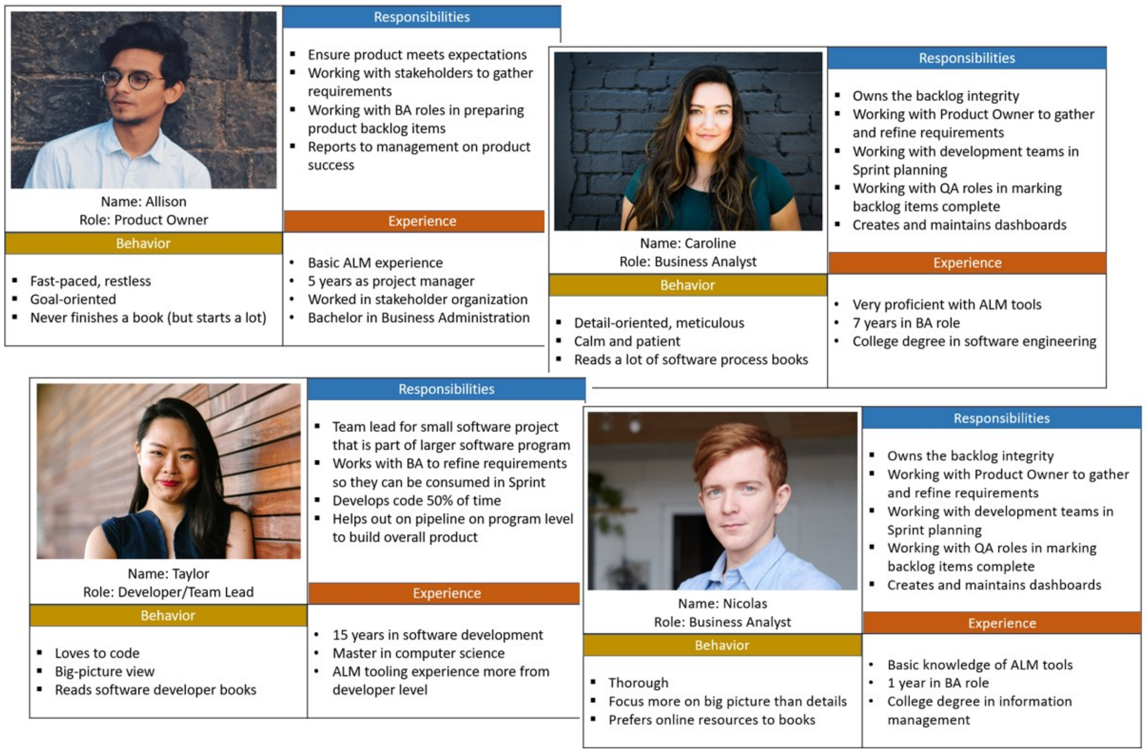
EXPERIENCE

To cover more variations of use, we want our users to have different levels of experience: *Not familiar with digital product backlog tools, used product backlog tools before but did not create the structures, expert user of product backlog tools (admin level privileges).*

PERSONALITY

We used the reader feedback to define reader personality types. The reader's personality is important to include, because two individuals with the same role and experience level may still approach the learning process very differently. We selected to cover: *the impatient reader who only reads to get the general understanding and skims through most of a book, the immersed reader who will study the whole book, and the reader who prefers online media but will read books when necessary*.

We found all these reader personality types among the people who responded to the survey. Bringing together the three dimensions of user characteristics we created four personas as shown in figure 2.



Name: Allison
Role: Product Owner

Responsibilities
- Ensure product meets expectations
- Working with stakeholders to gather requirements
- Working with BA roles in preparing product backlog items
- Reports to management on product success

Behavior
- Fast-paced, restless
- Goal-oriented
- Never finishes a book (but starts a lot)

Experience
- Basic ALM experience
- 5 years as project manager
- Worked in stakeholder organization
- Bachelor in Business Administration

Name: Caroline
Role: Business Analyst

Responsibilities
- Owns the backlog integrity
- Working with Product Owner to gather and refine requirements
- Working with development teams in Sprint planning
- Working with QA roles in marking backlog items complete
- Creates and maintains dashboards

Behavior
- Detail-oriented, meticulous
- Calm and patient
- Reads a lot of software process books

Experience
- Very proficient with ALM tools
- 7 years in BA role
- College degree in software engineering

Name: Taylor
Role: Developer/Team Lead

Responsibilities
- Team lead for small software project that is part of larger software program
- Works with BA to refine requirements so they can be consumed in Sprint
- Develops code 50% of time
- Helps out on pipeline on program level to build overall product

Behavior
- Loves to code
- Big-picture view
- Reads software developer books

Experience
- 15 years in software development
- Master in computer science
- ALM tooling experience more from developer level

Name: Nicolas
Role: Business Analyst

Responsibilities
- Owns the backlog integrity
- Working with Product Owner to gather and refine requirements
- Working with development teams in Sprint planning
- Working with QA roles in marking backlog items complete
- Creates and maintains dashboards

Behavior
- Thorough
- Focus more on big picture than details
- Prefers online resources to books

Experience
- Basic knowledge of ALM tools
- 1 year in BA role
- College degree in information management

Figure 2 Persona Cards

6. CREATING A STORY

On a beautiful summer evening, Allison is meeting with friends and colleagues at a restaurant in Boston. They are all attending a conference, and have met up to share some food and drinks after a long day. As the evening turns towards night, Allison gets into an interesting conversation with Anna who also works as a product owner for a large software company. When Allison explains about his frustrations with managing requirements – he is the product owner for a software program with a complex stakeholder organization and with development teams distributed in several countries and time zones – Anna tells him that she recently read a book that had some good advice on using digital tooling to deal with large and complex sets of requirements. She said the book had very practical advice and had helped her team find a way forward where they had built a product backlog that really supported their development process as well as their process of developing the requirements and bringing visibility to all parties involved. The name of the book was The Magic Backlog. "Why magic?" asked Allison. "Read it and see for yourself," Anna replied and laughed.

A couple of days later, Allison was at the gate waiting for his plane back to Phoenix when he thought about what Anna had told him. He went online and quickly purchased an electronic copy of The Magic Backlog. Well seated in the airplane, he opened his downloaded copy and started to read. Allison's style of reading is somewhat impatient and some would say erratic. He looked at the recommendations and recognized some names of people recommending the book. So far, so good. Then he flipped over the foreword and introduction and started reading on chapter 2 that gave an overview of the contents and was focusing on the overall approach. He then looked at a few chapters, studied the illustrations, and read a couple of patterns (well, he did not really read them but kind of skimmed through). He then looked at the table of contents. By that time, the drinks were being served, and he turned off his tablet, enjoyed a glass of wine and fell asleep.

The next day in the office, he popped his head into the office of Caroline. "Hi there, how are things going?" he asked, not really expecting more than a busy "fine, fine…". But today he caught Caroline in a bad moment. She was trying to produce a status update on the latest features and was wading through e-mails and Slack channels and individual team backlogs to try to piece together the report. After a few minutes of frustrated explanations of what she is trying to do, she finally asked Allison "So how was the conference, anything useful for us?" "You know what," answered Allison, "someone recommended me a book and I took a look at it on the plane on my way home. I think it might be interesting to you? I will send you the link to it."

It took Caroline a couple of days to even have time to open the link, but when she did she decided to buy a hard copy of the book. Being a real bookworm, Caroline preferred to read books the "old way," so she waited a couple more days for the paper copy to arrive. That weekend was rainy and cold, so she made some tea and curled up on the couch to read the book. As she started reading, she quickly realized that there was advice in this book that addressed her problems managing and reporting on requirements for the software program that she was assigned to. It was large and complex, just like examples in the book. She really liked the way the book was organized in giving her ideas about how to get started and what practices to do early, and which were more advanced.

On Monday morning, she went straight to the office of Taylor, her good friend and the project manager for one of the projects of the software program. "Taylor" she said and handed her The Magic Backlog, "if you want us to remain friends, please read this book and then work with me to make my life less miserable. I am dying trying to handle all the requirements and their statuses through development for the overall program. I think we could use some advice from this book and make some big improvements." Taylor, being a good friend, takes the book and promises to read it. And when she does, she realizes that Caroline is right. The book is very practical, and although it has a lot of practices, it is very clear on how to start. The running examples with illustrations are really helping to understand the implementations. She also appreciates the way the patterns deal with tradeoffs and possible negative consequences – it gives the feeling that this is real stuff and not fluff. So she decides to try with the fundamental practices, and she and Caroline get together to plan for workshops with her team to start structuring their backlog following the advice from the book.

As Caroline and Taylor are working through improvements, Allison is also getting involved. They eventually persuade the program manager to adopt the basic backlog practices for the overall program. This leads to a better planning of features across the program, with better visibility of progress. As things are falling in place, Caroline uses the examples from the book to tune their implementation as she gains new insights. Over time, Caroline returns to the book to look at the more advanced practices. She gets to a point where she no longer implements the patterns strictly following the book. She has a good grasp of the concepts and can see how the implementations can be done in the best way within her organization. In the beginning she appreciated the practicality and examples; now more and more she appreciates the discussions and the ways of thinking that are encouraged by the book.

Word of the improvements for this program reaches the company's Software Director, and she visits Caroline to discuss. This results in the internal training program being updated with a training using patterns from The Magic Backlog.

Caroline and Taylor are also asked to help a struggling project in the organization to establish a better backlog and better workflows for the team. This project has a Business analyst named Nicolas, who has only worked as a business analyst for a year. Caroline decides to use an informal approach so not to seem like a "know-it-all" expert and alienate the troubled team. She walks over to the team area and pops into Nicolas' small office. "Hi there," she says. "I am Caroline, the BA for the GFP program over in building C. I was asked to share some of our experiences with you. Do you have time to talk?" At first, Nicolas is a bit protective, but after a while he gets comfortable due to Caroline's down to earth style. Her practical questions make him realize that she has a lot of experience.

As Nicolas is opening up, Caroline goes into listening mode. Nicolas tells her that he is not so familiar with the use of ALM tooling, and being new to the project he is struggling to understand how the team is working. He is supposed to provide dashboards and weekly reports on the status of the requirements, but finds it hard to extract anything meaningful out of the backlog items. The team is often upset with the reports being wrong, but they are not good at keeping the backlog items up to date. Caroline soon realizes that she cannot just work with Nicolas, but needs help from Taylor and Alison to work with the whole team, and that this will take time. But to start, she will focus on mentoring Nicolas. She invites him over to her area to show him how her program is managing the backlog, as an example. “Just know that we found a way of working that suits our team,” she says. “It took a lot of learning and failing before we got there. But maybe it can help you avoid some of the mistakes we did. I also would recommend you to read a book that helped me a lot, and still does.”

Caroline can see from Nicolas’ reaction that he very interested in seeing how her team works, but he is not so keen on having to read a book. “Do you really think that a book will help?” he asks. “You know,” replies Caroline, “there are other resources you can access first, and then you can read the book if you want later. The authors have a very good web page that give you access to webinars, talks, and articles that is a different way of gaining this understanding. Maybe start listening to the talk they did at the Agile conference last year. It will give you a good introduction. It was very practical and focused on the basics of backlogs using digital tooling. They also have a very good webinar on dashboards, but you know if you do not have your contents in a good shape then you cannot trust your dashboards to show the right information.”

Nicolas thanks her for her help, and they agree on a time for him to come over to Caroline’s area to see how she is working. Two days later a cheerful Nicolas shows up in Caroline’s office. “Thanks,” he says, “thanks for showing me that web page! I have listened to several of the webinars, and this stuff is easy to share with my teammates too. Taylor came over to talk with my project manager, and we have agreed to sit with you and Taylor to define a roadmap for how to get our project moving forward with a better backlog. My project manager bought the Magic Backlog book, and I have been reading a bit in it. But I love the online resources. Now show me how you super-smart people are doing this!”

❖❖❖❖❖❖❖❖

In this story we illustrated the following goals of a book design:

- Fast grasp of what the book is about
- Providing practical advice
- Includes deeper meanings
- Ability to become a reference, a tool (not a onetime read)
- Support for several roles on several levels

We should point out that our personas were first created before we ran the reader survey. They represent the readers we envision for our work. Their profiles are not drawn from the survey responders but as we got their feedback and learned what people want in a reading experience, we started weaving these preferences into our persona behavior and into their stories.

Our feedback showed us that not everyone used books as their primary media, so we included this in the story. There is a discussion of books versus other media and how they can come together as a whole in the discussion chapter later in the paper. Our story was not about the particular advice/patterns that would be in the book, as this would require the reader of this paper to be familiar with our patterns collection.

7. DESIGNING A BOOK FOR OUR PERSONAS

Based on our feedback, readers read technical books because they need knowledge to solve a problem, or they want to learn more about a topic, or in some cases need to learn more about the author. So if the reader has a purpose, then we as authors need to understand that and help the reader quickly understand what we cover in the book with the goal of matching the purpose of the reader with the contents and intention of our work. We want to be clear about our target audience and why we think they can benefit from reading the book. The reader invests time in reading and we want to provide return on that investment. Even more, we want the readers to be excited about the insights they gain from reading it, so that they will recommend the book to their friends and colleagues. This way we can reach a broader audience and get return on our investment as authors by knowing that we have shared our experience and our practices with our community and hopefully helped people having the same need for guidance that we had when we started out using ALM tools for product backlogs.

⇨ Defined target audience: Software development teams, in particular the roles of Business Analyst, Product Owner, and Development Lead
⇨ Purpose of book: Provide advice for product backlog management using ALM tools
⇨ Approach: Use a combination of storytelling and patterns/pattern sequences, focus on building understanding (why) and providing practical advice (how)

Which particular book a person selects is driven by recommendations by colleagues and friends, e.g. by word of mouth from someone you trust and who knows you and your work. Knowing the author or getting exposed to the author as a conference speaker also drives the choice. Dealing with how our book can become highly recommended is a tough challenge for us to plan for! But we can at least select a book title that makes it very clear what the book is about (we want the readers that want our book). Maybe a short title with a longer subtitle? Then, a short and crisp introduction that quickly takes the reader into the material. This part is super important – it needs to build trust quickly that we have the knowledge, that we present it plainly and simply, with no fuss, but enough detail that the advice can be followed. The table of contents is a way to tell the reader what the book contains, so each chapter title must be carefully selected.

- ⇨ Title: Magic Backlog – a guide to building and grooming amazing product backlogs
- ⇨ TOC: The table of contents is designed as a navigation tool.
- ⇨ Outline: The first part of the book can be seen as a stand-alone part that spells out the main concepts and who the book is for. Subsequent chapters move from the basic practices of structuring and navigation to providing insights from the backlog, backlog building and maintenance, to backlogs for large projects and programs. This way, readers can stop when they feel they have enough information. And readers can come back to more advanced practices after first implementing the basic ones.
- ⇨ Chapters start with a story that illustrates the use of the patterns in that section.

Before online resources were common, a book had to be complete in providing all the knowledge it could about the topics it covered. Today, a book can be a part of an overall ecosystem of knowledge. That implies that the authors must decide what goes in the book, and what is better suited to be covered by other media. That said, the book also needs to stand on its own and be complete enough to not depend on additional resources for its core goals. The length of the book can be kept shorter by using other media for some of the details. Two areas of content that are particular importance to us as we are writing a patterns book: the full patterns and how they relate to implementation examples.

- ⇨ Length: reduce length of book by using other media for full patterns
- ⇨ Length: reduce length of book by using other media for detailed implementation examples
- ⇨ Design the book as part of an ecosystem that can grow over time

8. DESIGN OF THE BOOK IN RELATION TO OUR PERSONAS

Allison, Caroline, Taylor, and Nicolas are roles that match our target audience. We believe that the design of our book addresses their needs in the following ways:

Allison is a busy product owner for a complex product being developed by distributed teams. To Allison, the title intrigues him. More important, the book is recommended by colleagues and his initial first-impression is that the book is practical and useful. His attention is limited. He does not read books cover to cover (in fact, reading a book that is attractively laid out and easy to grasp an overview before diving in and then skimming is what will hold his attention. He likes that individual chapters of the book mostly stand on their own and do not rely upon successive chapters. Informative illustrations and graphics are also important.

Caroline, an analyst, is mired in day-to-day problems at work. When she reads a book, she likes to absorb it and reflect on how practices it describes can make her life (and that of her co-workers) better right away. The book connects with her because it contains practical advice that relate to her current problems at work. She can use what practices are of immediate use and refer to more advanced topics as she needs to. The book isn't an all-or-nothing book. Being an avid reader, Caroline also appreciates the story-telling aspect of the book. Not every technical practice book needs to be a dry read or full of bulleted items and checklists!

Taylor is a busy project manager who supports process improvements, especially if they are practical and grounded in real-world experience she can relate to. She likes the straightforward, pragmatic advice in this book: based on your situation, here's where you could start, and here's a reasonable way to proceed. Taylor appreciates that not every business situation warrants the same approach and recognizes the need to adapt any practices to the current situation at hand (and that the business context is likely to evolve over time). Taylor appreciates that the book doesn't gloss over subtleties or realities that are often entirely ignored or gloss over in other process books.

Nicolas, a relatively new analyst, looks for practical advice. His first inclination is to learn new practices by observing how others more experienced than he is, actually do their work. He likes hearing stories about how they solved problems or better yet, avoided them. He enjoys watching videos and short presentations about topics he is interested in. If a presentation is too long he does not hesitate to fast forward through it or stop watching. Sometimes he plays videos at double speed, just so he can get through them more quickly. He likes controlling how he consumes information and finds the length and structure of a book daunting. He likes reading short blog posts and tips, but again, prefers browsing until he finds something that strikes him as interesting or potentially useful. He will dip in and read portions of a book, once he is more comfortable with the topic, but his preferred kind of book is an online one where he can clip important parts and drop them into his note keeper app.

9. DISCUSSION/CONCLUSION

There are many valuable things we learned in the process of writing this paper. We asked for feedback from potential readers as well as authors. They freely shared what books had an impact on them and why. While we can't plan to write a timeless best seller, we now know that we need to more fully understand our intended readers and think about creative ways to write our book incrementally and test it out on potential readers.

We also came to appreciate that a well-designed book should have a central place in an information ecosystem that surrounds it. But books these days don't stand in isolation. Readers and learners these days expect support for different modes of learning and often prefer to consume online media, videos, presentations, and podcasts. If we want our Magic Backlog patterns to have a broader impact, we need to not only write a book that appeals to our targeted audience, but also create multi-media and online material that supports it.

Finally, as we were crafting our personas and filling in some details of their stories, our suspicion grew that yes, indeed, we likely have a few gaps in our existing patterns. For example, our patterns are focused around backlog changes as a project moves through the lifecycle and the business needs are changing. There is another kind of change that we didn't directly address that we also feel is important to address: change that is driven by the need or desire to improve the current way of working. This also led us to realize that our intended readers could also benefit from some specific advice on how to introduce such changes (e.g. those particular to changing existing processes and restructuring and managing a backlog) without appearing threatening to the organization or being disruptive to ongoing work[5] . This will need to be addressed as we work on the book and the ecosystem around it, and most likely we will detect more areas to cover as we work our way through this.

10. ACKNOWLEDGEMENTS

As we are completing this paper, we need to express our gratitude to all the people who were involved along the way – our friends and colleagues who were willing to share their opinions about books, and those who engaged in further discussions, the authors of all the books we used for our research that have contributed to this amazing body of knowledge that we all benefit from, and the people who influenced the creation of the personas that help us in our design process. A very warm thank you goes to the author of Fearless Change and our shepherd for PLoP 2019, Mary Lynn Manns – you put your finger very nicely and ruthlessly on the weakest points of our work, you caused us long discussions and major rewrites, and with your coaching we could take this from a very unfinished work to something that we hope can be useful to our audience. We also appreciate the insightful feedback we got at the PLoP 2019 conference workshop that helped us further improve the paper – thank you to all the workshop participants.

[5] Yes, we were inspired by Fearless Change patterns [MH], but suspect there are unique patterns for introducing backlog management changes into organizations (depending on the role of the change agent and the nature of the project/product teams).

REFERENCES

[Beck] Beck, K. Extreme Programming Explained: Embrace Change. Addison-Wesley, 1999.

[Bitt] Bittner, K. The Nexus Framework for Scaling Scrum: Continuously Delivering an Integrated Product with Multiple Scrum Teams. Addison-Wesley, 2017.

[BM] Beyer, B. and Murphy, N. The Site Reliability Workbook: Practical Ways to Implement SRE. O'Reilly Media, 2018.

[Broo] Brooks, F. The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition (2nd Edition), Addison-Wesley, 1995.

[Busc] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M. Pattern-Oriented Software Architecture: A System of Patterns.Wiley, 1996.

[CH] Coplien, J. and Harrison, N. Organizational Patterns of Agile Software Development. Prentice Hall, 2004.

[Coc2001] Cockburn, A. Writing Effective Use Cases. Addison-¬Wesley, 2001.

[CBP] Caner, K., Bach J., and Pettichord, B. Lessons Learned in Software Testing: A Context-Driven Approach. Wiley, 2001.

[CG] Crispin, L. and Gregory, J. Agile Testing: A Practical Guide for Testers and Agile Teams, Addison-Wesley, 2009.

[DeMa] DeMarco, T. The Deadline: A Novel about Project Management. Dorset House, 1997.

[FHK] Forsgren, N., Humble, J. and Kim, G. Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations. IT Revolution Press, 2018.

[Fowl96] Fowler, M. Analysis Patterns: Reusable Object Models, Addison-Wesley, 1996.

[Fowl99] Fowler, M. et al. Refactoring: Improving the Design of Existing Code 1st Edition. Addison-Wesley, 1999.

[GHJV] Gamma, E., Helm, R., Johnson, R., Vlissides, J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.

[GC] Goldratt, E. and Cox, J. The Goal: A Process of Ongoing Improvement. North River Press, 2014.

[HT] Hunt, A. and Thomas, D. The Pragmatic Programmer: From Journeyman to Master. Addison-Wesley, 1999.

[Hva2015] Hvatum, L. and Wirfs-Brock, R. 2015. Patterns to Build the Magic Backlog. 20th European Conference on Pattern Languages of Programming (EuroPLoP), EuroPLoP 2015, July 8-12 2015, 36 pages.

[Hva2017] Hvatum, L. and Wirfs-Brock, R. 2017. Pattern Stories and Sequences for the Backlog: Expanding the Magic Backlog Patterns. 24th Conference on Pattern Languages of Programming (PLoP). PLoP 2017,October 23-25 2017, 26 pages.

[Hva2018] Hvatum, L. and Wirfs-Brock, R. 2018. Program Backlog Patterns: Applying the Magic Backlog Patterns. 23rd European Conference on Pattern Languages of Programming (EuroPLoP). EuroPLoP 2018, July 4-8 2018, 22 pages.

[KBS] Kim, G., Behr, K., and Spafford, G. The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win, 5th Anniversary edition, IT Revolution Press, 2018.

[KDWH] Kim, G., Debois, P., Willis, J., and Humble, J. The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. IT Revolution Press, 2016

[Kell] Kelly, A. Business Patterns for Software Developers. Wiley, 2012.

[Kert] Kerth, N. Project Retrospectives: A Handbook for Team Reviews. Dorset House, 2001.

[LB] Larman, C. and Bodde, V. Large-Scale Scrum: More with LeSS. Addison-Wesley, 2016.

[MBCP] Murphy, N., Beyer, B., Jones, C., and Petoff, J. Site Reliability Engineering: How Google Runs Production Systems. O'Reilly Media, 2016.

[McCo] McConnell, S. Code Complete: A Practical Handbook of Software Construction, Second Edition. Microsoft Press, 2004.

[MR] Manns, M. and Rising, L. Fearless Change: Patterns for Introducing New Ideas. Addison-Wesley, 2005

[Ries] Ries, E. The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. Currency, 2011.

[Roth] Rothman, J. Agile and Lean Program Management: Scaling Collaboration Across the Organization. Practical Ink, 2016

[RW] Rozanski, N. and Woods, E. Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives. Addison-Wesley 2005.

[SS], Schwaber, K. and Sutherland, J. The Scrum™ Guide The Definitive Guide to Scrum: The Rules of the Game. 2017.

[Wieg] Wiegers, K. Software Requirements 2nd Edition. Microsoft Press, 2009.

[Wir2016] Wirfs-Brock, R. and Hvatum, L. 2016. More Patterns for the Magic Backlog. 23rd Conference on Pattern Languages of Programming (PLoP), PLoP 2016, Oct 24-26 2016, 18 pages

[Whit] Whittaker, J. How to Break Software: A Practical Guide to Testing. Pearson, 2002.

[WM] Wirfs-Brock, R. and McKean, A. Object Design: Roles, Responsibilities and Collaborations. Addison-Wesley, 2002.

[WWW] Wirfs-Brock, R., Wilkerson, B., and Wiener, L. Designing Object-Oriented Software. Prentice-Hall, 1990.

APPENDIX A: THE BOOKS

This appendix presents an overview of the books that we selected to analyze, each with a short description. We grouped the books into the following categories: fundamental books, books on specific processes, books for specific professions within the software industry, books on DevOps (representing a hot topic), patterns books, and business books.

Fundamental books

These books are classics in the software domain, books that many software professionals know of and look upon as a fundamental part of a proper software education that include some understanding of the history of software development.

– **The Mythical Man Month** by Fred Brooks is a collection of essays on software engineering and was first published in 1975, with a second edition in 1982 and a third edition in 1995. The lessons learned and the insights provided in these essays have mostly stood the test of time and most of this book is as valid today as it was when first published. The topics are about software development processes on a conceptual level that do not depend on a particular software process, and about human nature.

– **Software Requirements** by Karl Wiegers and Joy Beatty provides a fundamental understanding of software requirements and their characteristics. The third edition was published in 2013, while the first edition dates back to 1999. The tables with quality characteristics for requirements and requirements collections are just as valid and valuable today as they were in 1999.

– **The Deadline**: **A Novel about Project Management** by Tom DeMarco tells the story of a software manager who is downsized but then is kidnapped to an imaginary country and gets the opportunity to test out project management principles in a large scale experiment. By choosing the novel format, the author makes this an entertaining read, and since this is a type of storytelling it is easier to remember the advice given on a number of topics. Some of the process thinking has evolved since the book was published in 2011, without that making the book less of a recommended read.

– **Retrospectives** by Norm Kerth is the first book about doing retrospectives, and although some of the thinking is a bit out of touch with today's business reality (like an off-site 3 day event), and it is missing the idea of heartbeat retrospectives, it is still the primary retrospective book that lays the foundation of the practice not least from the perspective of the philosophy of reflection and trust.

Books on specific processes and methods

Every software methodology has its core documentation, and here we have selected a few representative items.

– **Extreme Programming Explained** by Kent Beck first published in 1999 is the fundamental XP guide describing engineering practices like unit testing, continuous refactoring, and continuous integration that have had a profound influence on all agile methodologies.

– **The Scrum Guide** by Ken Schwaber and Jeff Sutherland is according to the authors the definitive guide to Scrum, and they make it freely available on its own web site independent of any commercial interests. This writing is a short and very specific description of how Scrum works that should be a must read to any member of a Scrum team. First published in 2010, it would also classify as a fundamental work.

– **The Nexus Framework** by Kurt Bittner, Patricia Kong and Dave West tackles the scaling of Scrum to handle multiple Scrum teams working from one product backlog (e.g. program level methodology). It is a short book that follows the crisp and "less is more" style of the Scrum Guide.

– **Large-Scale Scrum** by Craig Larman and Bas Vodde is their version of a multi-team Scrum methodology. This book provides more insights in the goals and "why's" than the short Nexus documentation.

– **Writing Effective Use Cases** by Alistair Cockburn was THE book about use cases, and for those that feel that the limitation of user stories is not quite enough to really understand users interaction with their product this is still a very useful book.

Books for specific professions (architects, developers, and testers)

The processes and methods in these books are mostly on the level of the individual developer.

- **Code Complete** by Steve McConnell is a highly recognized guide to developers. The second edition was published in 2004, (TBD)
- **The Pragmatic Programmer** by Andrew Hunt and David Thomas was first published in 1999, and it has tips and practices for software engineering (not a consistent methodology). A second edition is coming this year.
- **Refactoring** by Martin Fowler addresses ways to deal with legacy code.
- **Agile Testing** by Lisa Crispin and Janet Gregory is an essential guide for how the role of testing is integrated in agile teams.
- **How to Break Software** by James Whittaker has a number of testing techniques to attack software products in a systematic way. It is short and very practical, and written in a fun way following the philosophy of the exploratory testing community seeing testing as an intellectual activity.
- **Lessons Learned in Software Testing** by Cem Kaner, James Bach, and Bret Pettichord is another book from authors that are thought leaders within exploratory testing. It contains practical testing techniques as well as the ways of thinking as a tester.
- **Software Systems Architecture** by Nick Rozanski and Eoin Woods is a book that provides a framework for designing and documenting relevant views of architecture.

DevOps books

Since DevOps is the buzzword these days and we have read some of the books on the topic we include this to represent current writings.

- **The Phoenix Project** by Gene Kim, Kevin Behr, and George Spafford is written as a novel following the story of an IT manager who is given 90 days to turn around his company's work processes to improve product delivery (e.g. DevOps).
- **The DevOps Handbook** by Gene Kim, Jez Humble, Patrick Debois, and John Willis is the companion to the Phoenix Project that covers the workflows and methodology that emerges in the novel in a systematic fashion.
- **Site Reliability Engineering** edited by Betsy Beyer et. al. is a collection of essays and articles by multiple authors that describe the DevOps approach within the Google organization.
- **The Site Reliability Workbook** edited by Betsy Beyer et. al. is describing how the Google SRE is done in practice within Google and other organizations that are implementing SRE based on the Google model.
- **Accelerate** by Nicole Forsgren, Jez Humble, and Gene Kim is based on their research when doing the State of DevOps reports from 2014 to 2017. It provides an empirical study of the effectiveness of DevOps practices.

Patterns books

- **Design Patterns** by Eric Gamma, Richard Helm, Ralph Johnson, and John Vlissides is the fundamental classic of software design patterns containing 23 documented patterns.
- **The POSA Series** by Frank Buschmann et. al. are six books that focus on pattern-oriented software architecture.
- **Analysis Patterns** by Martin Fowler focuses on patterns for business processes and the architecture of domain specific systems.
- **Domain-Driven Design** by Eric Evans focuses on patterns for identifying the core business process and developing a common language among business experts and developers who create software that supports the business.
- **Organizational Patterns of Agile Software Development** by Jim Coplien and Neil Harrison provides patterns for software process based on an extensive study of high-performing teams.
- **Fearless Change** by Mary Lynn Manns and Linda Rising is a patterns book but also a business book, presenting patterns to introducing change in an organization from a grassroot level.

- **Business Patterns for Software Developers** by Allan Kelly is also a patterns book within a business context. It is aimed at roles that are considering starting businesses in the software industry whether doing consulting or selling software products.

Business books

There are business books that are written by software professionals and/or targeting software organizations, and also general business books that have gained popularity in the software community:

- **Agile and Lean Program Management** by Johanna Rothman is a thorough Guide for creating and managing software programs (e.g. multiple projects working together to deliver a product).
- **The Goal** by Eli Goldratt is now available in its fourth edition. First published in 1984, the novel follows a manufacturing manager demanded to turn around productivity at the factory site he is managing. The Phoenix project is written following the same recipe but focusing on software product deliver instead of the delivery of good. In both books the manager is helped by a role working as a hands-off coach.
- **The Lean Startup** by Eric Ries is inspired by lean manufacturing ideas and is focused on entrepreneurial management based on short product development cycles and rapid validation by customers.

APPENDIX B: DESIGN PRINCIPLES FOR SOFTWARE PROCESS BOOKS

This is the complete list of principles for designing software process books that we found through the combination of reviewing books and analyzing feedback from readers and authors. The ones that we applied in the chapter on designing our book are marked with an asterisk (*).

They are not listed in a particular order, but we tried to group them to have a natural flow. We also added some thoughts around how to apply the principle, added quotes from the feedback where applicable, and provide examples of books that apply the principle.

BRING SOMETHING TO THE TABLE

There should be a reason to write a book, more than the desire of the writer to be an author. If your idea of a book does not bring any new knowledge or perspective, or at least provides a systematic and useful and well-written combination of information that is already documented elsewhere, you should ask yourself if you really should spend time writing a book.

> "For me to continue to read a software process book, it has to be providing me new information. It needs to be challenging the way I think and/or do things."

> "it gives me insights I have failed to collect myself"

> "I get my technical knowledge from reference manuals and online examples. The books give me more of a deeper understanding or "wisdom"."

Books that provide value on known topics: *Writing effective use cases, Lessons Learned in Software Testing*

FOCUS ON GOALS NOT ON TASKS

This is not only good advice when writing user manuals, but also when dealing with software processes and practices. Good pattern names typically follow this advice, it is not so much about specifically what to do but on what you want to achieve (e.g. focus on why and how to generate a desired result). Failing applications of agile processes are falling in the trap of focusing on tasks; e.g. doing described practices but failing to understand and achieve the purpose.

> "Books need to be to the point and implementable […] If I have enough backing as to why principles or ideas work, I can also sell them more easily in my organization"

> "I read books that solve a specific problem"

> "My recommendations for this is because it's a "why" book. It explains why we want to certain practices. Not how to do them, but why to do them."

Books that focus on the goals: *Business Patterns, The Pragmatic Programmer*

DON'T MAKE ME READ THE WHOLE BOOK*

Some books have an introductory part that gives an overview and general understanding of the contents. Following chapters then go deeper into the various concepts, and there is a guide in the overview section that clarifies what chapters are covering what content. This means that readers who mainly want a general understanding can get away with reading maybe the first 40-50 pages. Readers who are familiar with, or particularly interested in, parts of the material can choose what chapters to read. In some cases each chapter is also structured in an overview section and then going deeper.

> "[…] in general I'm not a big fan of reading books. I don't really have the patience for it"

> "It is possible to get lots of benefit from that book without reading all of it"

Books with a first part introducing the concepts: *Domain-Driven Design, Agile Testing, Software Systems Architecture*

PROVIDE A CONTENT MAP*

Helping the reader understand the structure and flow of the book will help them access the contents more successfully. A reader may be specially interested in a subset of the total contents, for example not yet ready for more advanced practices but wanting to find basic practices to try. That reader may return later for more challenging concepts. Any book that serves as a reference for people needs to support reader navigation. A book may become a reference for an individual even if the author did not plan for this, so this principle should always be on the authors mind. Headings and table of contents (TOC) are important navigational tools. Special care should be placed on the chapter titles and sub-titles as they appear in the TOC creating a content map. Title terminology should be from the users domain if possible, so that concepts do not have to be explained before the TOC can be understood, and focus on the users goals. A flat structure is usually easier to navigate. If topics are self-contained, and follow a natural flow of introduction and learning, the reader will not have to move back and forth in the book.

> "The text on the title mentioned a few technical practices I follow [...] I knew I had to read it."

> "As much as I was interested in the big picture, I was also looking for details […] without spending too much time reading it."

Books that are easy to navigate: *How to Break Software, Accelerate*

STORIES ARE GOOD*

The telling of stories is fundamental in the human history as a way of sharing ideas and knowledge. It is hard to remember bullet points on a slide, or a list in a book, but we remember stories that speak to us. Stories that feel real, with events and characters that the reader can relate to, also help to build trust between the reader and the author. Stories that are well composed are an easier read than technical text. That is also a challenge – not everyone is good at creating or finding stories. Writing a novel to provide software process insights is a tall task, but storytelling can also be done by smaller stories woven together into a whole, with the story and the technical contents alternating through the book. Some authors include story snippets, either as lead-ins to chapters or short breaks in the action to reinforce a concept or practice. Some of these stories may be gleaned from experiences other than the authors.[6]

> "I loved the way it described how DevOps works, because it nicely applies storytelling"

Ultimate storytelling in books: *The Deadline, The Goal, The Phoenix Project (novels)*
Books with stories: *Agile Testing, Fearless Change, Large-Scale Scrum*

ILLUSTRATIONS THAT MAKE SENSE

Illustrations are important for several reasons. They give a different way to document and so complement the text. Use them as alternatives to text. Lighten up the reading. Must make sense and fit I the flow. Place then where the illustration is discussed and always do discuss the illustration. Make sure it is easy to understand and that it complements the text.

> "[to choose a book] Style: text+ pictures+ illustrations, easy to read/fluid"

> "provide visual cues […] which help readers understand and remember the content. I lean strongly on books which leverage visual models as a communication tool to support software processes and practices"

Books with informative illustrations: *Large-Scale Scrum, Project Retrospectives*

CONSISTENT VOICE AND VOCABULARY

Books written by multiple authors often suffer not only the changes in style from one author to the other, but also in being inconsistent in terminology and the application of practices. The role of the editor is really important in this situation. Especially some of the later (hot topic) books suffer from groups of authors writing individual chapters. This is great in bringing in experience and viewpoints from multiple sources, but it has a challenge because of the variances between companies and organizations in terminology and implementation of practices.

> "use [the book] to establish a common vocabulary and shared mindset with colleagues"

---

[6] Another pattern for story-as-inspiration, that I, Rebecca used when writing *Object Design*, was to lead in each chapter with a short couple of paragraphs introducing a short interlude about design, or design thinking from a totally different context (e.g. what painters or writers or Christopher Alexander said about a somewhat-related topic). This was then followed by a couple of sentence lead-in to the upcoming chapter. This was hard to do and took a lot of effort to find the right story and then to write about how it related to the chapter. But it was absolutely essential to writing the chapter (and I found I couldn't really start writing the chapter until I had identified this story).

Books by multiple authors that are consistent: *Fearless Change, Organizational Patterns of Agile Software Development*

LIMIT REPETITION

This should be an evident thing to remove when a draft reaches an editor, but we have encountered several technical books of 300 pages or more that would have done better staying below 200 pages. Note that some repetition can be done by design, as it is known that it can help the reader remember. But then it should be done by chapter summaries and not by what seems to be verbose sections with little content.

> "Entire content of the book can be told in two articles. Instead the book has 400 pages with lot of redundancy."

Books that avoid repetition: *Software Requirements, Agile Testing*

SHORTER IS BETTER

Length is related to the principle of limiting repetition, but goes further in not just trying to avoid repetition but in keeping sentences short and specific, and mercilessly cutting out content that is not required to assist the reader. The Scrum Guide is a good example where you get the feeling that every word and every sentence is carefully selected and created, and there is no fluff or part that is not deliberately included.

> "Books with fluff to increase page count or no coherent narrative."

Books that are short and concise: *The Scrum Guide, The Nexus Framework*

SIMPLER IS BETTER

Deal with complex contents by careful structuring and by introducing concepts gradually, using examples and stories to help the understanding. Stating ideas in simpler ways takes time and editing skill, but when undertaking the significant workload of creating a book it is sad if that effort is wasted because the readers are not getting the insights. Do not complicate to show off. This may be more frequent in academic writings, but sometimes one can get the impression that the author wants to show off their cleverness to their readers.

> "[I do not like] books with fluff to increase page count"

Books that are straightforward and easy to read: *Large-Scale Scrum, Business Patterns for Software Developers*

HAVE A PERSONAL VOICE

As an author one should not be afraid of making the writing personal while still showing respect for the reader. Maybe the right voice is professionally personal – not including stories involving family and friends but rather focusing on personal experience as a software professional, and always considering if telling a story is working towards the goal of making the contents more accessible to the user. A personal style tends to be less dry and more entertaining, and it lends credibility to the contents.

> "I really like the books of authors I personally know (and like). I sort of feel like the author is talking to me."

Books with a personal approach: *Project Retrospectives, Fearless Change*

SHARE THE EXCITEMENT

This one is difficult. Technical stuff is by nature rather boring, so how to not bring your reader to sleep? In most cases the book turns out to be more attractive to read if there is a genuine feeling that the author cares deeply about the contents. This can be done by sharing some personally experienced situation, or by sidebars with additional insights. This also breaks up the monotony of the text.

When the author is excited about the topic, the book typically avoids harping on negative issues. It is better to focus on what is working and how to get to the desired outcome than to spend paragraph upon paragraph about bad practices. It is often more helpful to know what to do than to know what to avoid, and an author that keeps on and on about negative topics will quickly lose the reader. On the other hand, what might go wrong (and how to remedy it) can be useful. One way to rephrase this kind of information is to describe signals or signs that things are working well.

> "I have a particular liking to books with stories and jokes. These books can really keep me engaged. If the entire book is dry and technical, it can be hard to keep my attention"

Books that feel engaged: *Agile Testing, The Lean Startup*

AVOID MARKETING

Books from authors should avoid marketing hype or overselling the benefits of practices. Readers are not happy to invest time in what turns out to be a sales pitch or promoting a consulting business, or when the book seems to be something it is not just to boost sales.

> "… like they stuck the word patterns in the title, but there was little in the book that was recognizable to the patterns community as a pattern. There may have been good stuff in there, but I never got over what felt like false advertising to me"

> "Books written for the wrong reasons (marketing of company, or joining the hype bandwagon without proper experience in the topic)"

> "Actually, there is a genre of books which I don't like which seem to be the "Hay this is great, if you do it my way it's great, company X did this and ..., company Y did this and... " but they never actually tell you what they did, just what the result was"

Books that avoid the marketing feel: *The Nexus Framework, The DevOps Handbook*

DO NOT TRY TO SEEM ALMIGHTY

Readers appreciate a certain level of humility in the author. Yes, they want to feel confident that the author knows her stuff. But they trust more when the person provides valuable insights but still is aware that there is more to learn and that every situation brings unique challenges. People seem to really dislike reading software process books that declare that their practices are the only viable ones and are not open for interpretation and discussion.

> "Or if the author does not have humility to recognize that there is a lot he does not know, I will often be disappointed."

Books from knowledgeable authors with a humble feel: *Retrospectives, Software Requirements*