Agile Software Development

DR. MICHAEL ROBBELOTH

TEACHING DEMONSTRATION

APRIL 24, 2025

ACM CS Curricular Guidelines

Agile Software Development best fits within the Software Engineering Teamwork Knowledge Unit

Part of CS Core (2+3 hours) and/or KA Core (2 hours) Topics

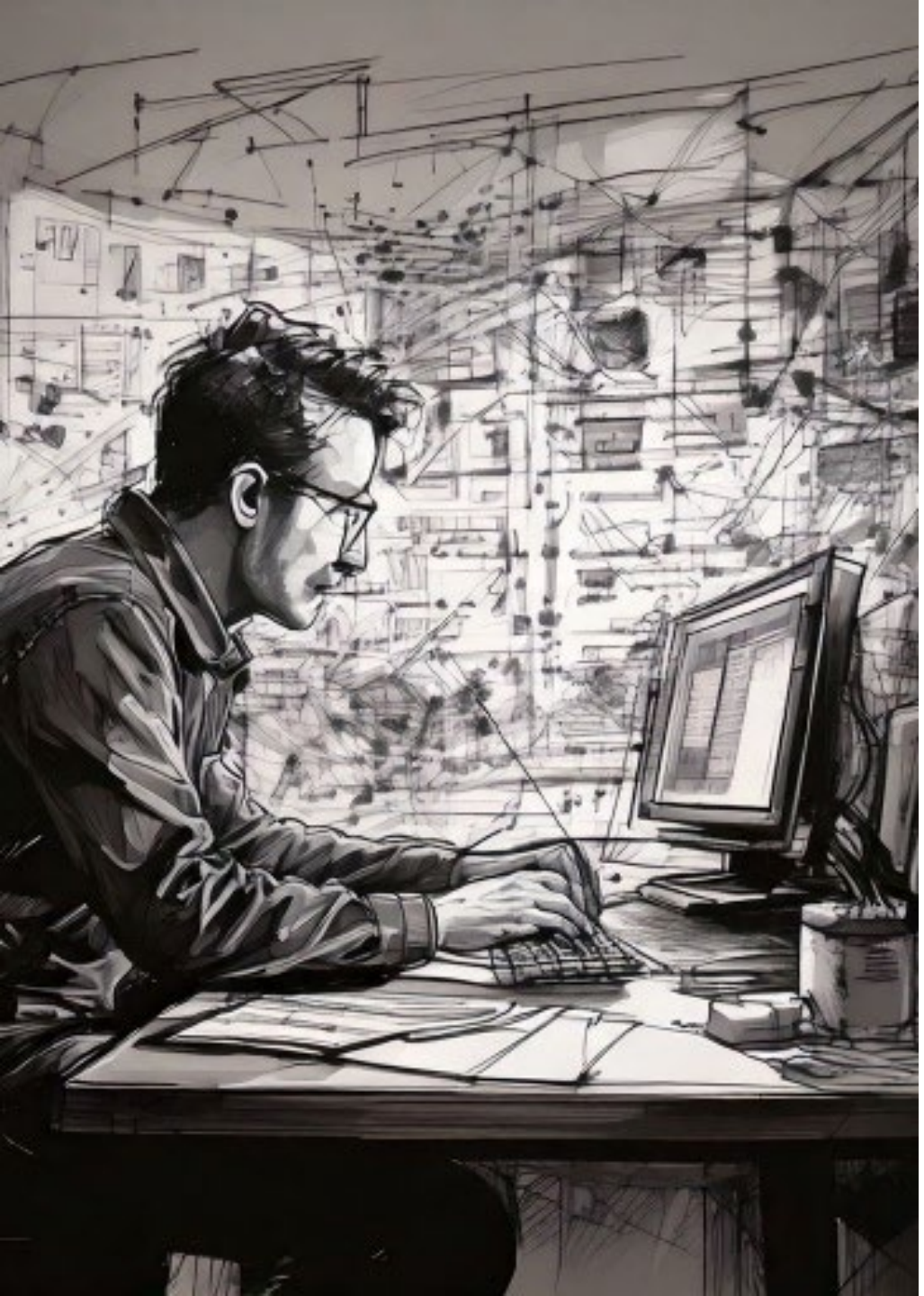
Outcome :Explain the core values and principles of the Agile Manifesto and contrast them with traditional software development methodologies (like Waterfall)

Outcome: Describe key Agile practices such as iterative and incremental development, continuous feedback, and the role of self-organizing, cross-functional teams.

Outcome: Identify potential benefits and challenges of using an Agile approach in software development projects

ABET Learning Outcomes Covered

- CS 5 Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline
- SE 3 An ability to communicate effectively with a range of audiences;
- SE 5 An ability to function effectively on a team whose members together provide leadership, create a collaborative and inclusive environment, establish goals, plan tasks, and meet objectives;
- SE 6 An ability to develop and conduct appropriate experimentation, analyze and interpret data, and use engineering judgement to draw conclusions;



Introduction

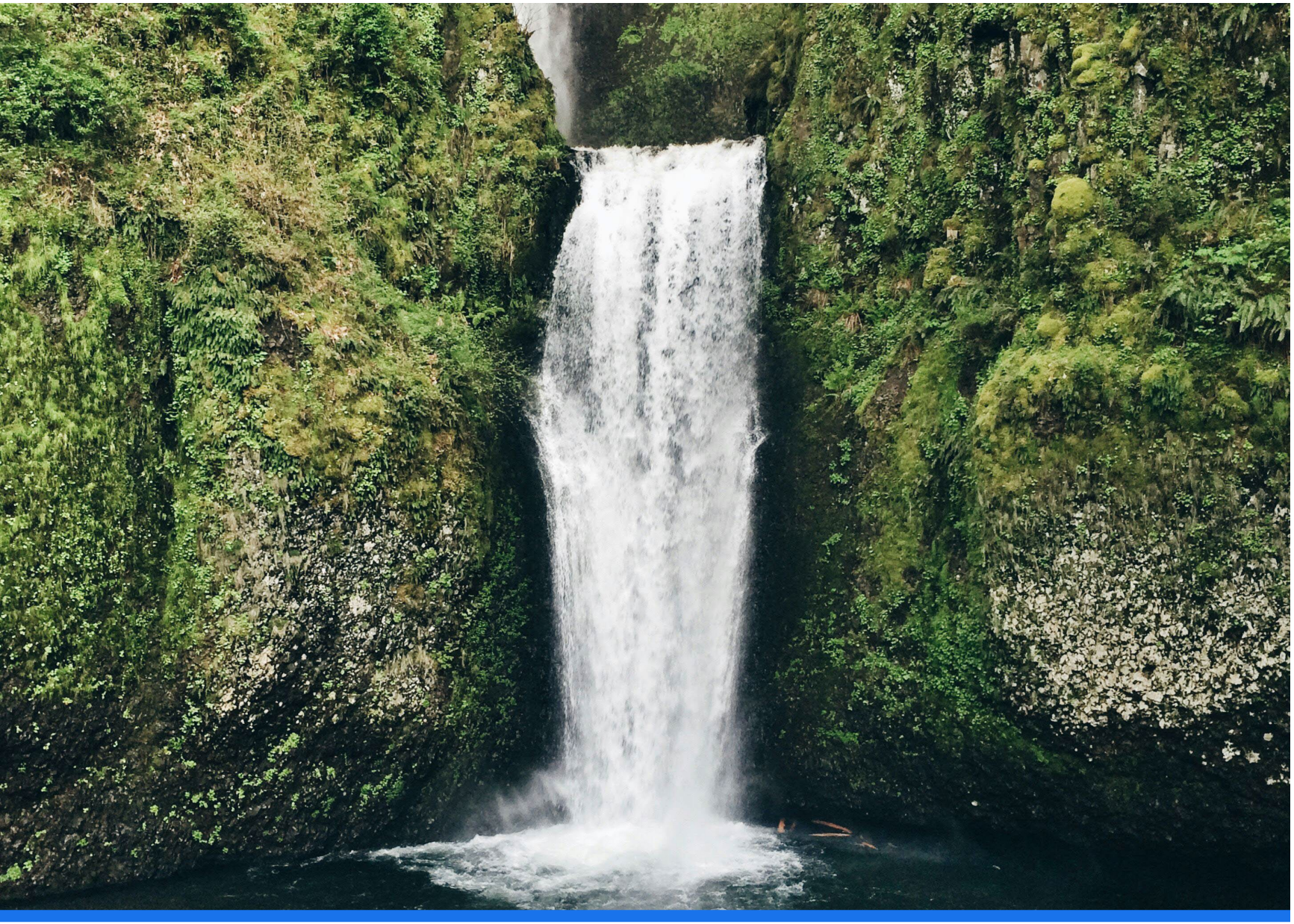
- For decades, software engineers have learned the Waterfall model.
- It is linear and plan-driven
- Each phase in sequence, with all details spelled out before coding
- Meant for large, dispersed teams in long-lived software systems
- Too much overhead in small to medium collocated groups
- For certain projects, there is a better way
- One that handles technology, user needs, and market conditions that are constantly changing
- That's what we'll explore today – the Agile model (helping developers thrive in an unpredictable environment)

Introduction

- Agile is an alternative to approach to software development
- Used quite regularly in industry today (though how well it is used is another story)
- Understanding agile will help in advancing your career

Agenda

- Setting the Stage: Traditional Methodologies vs Agile Methodologies
- The Agile Manifesto and Core Values
- Key Agile Principles and Practices
- Benefits and Challenges of Agile
- Conclusions and Q&A



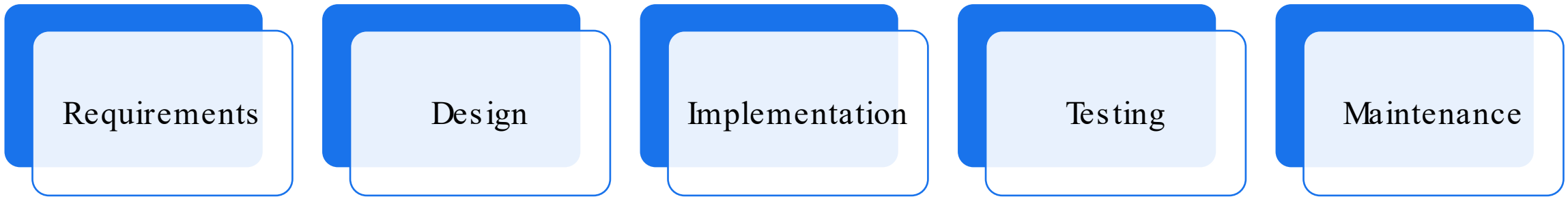
Setting the stage – Traditional Methodologies (Waterfall)

- Rigid
- Each stage must be completed before the next
- Progress is like a waterfall
- You don’t go back to a previous stage

Are you familiar with the waterfall model?

- What are the phases of traditional software development?

Setting the stage – Traditional Methodologies (Waterfall)



Setting the stage – Traditional Methodologies (Waterfall)

- Today's world is fast-paced
- Software development can be unpredictable
- Requirements might be unclear even after multiple rounds of initial interviews
- There is a clear need for rapid feedback
- Agile is the response to these limitations

The Agile Manifesto and Core Values

- Created February 2001
- 17 software practitioners meet at the Snowbird ski resort in Utah
- Individuals had used Extreme Programming (XP), Scrum, Dynamic Systems Development Model (DSDM), Waterfall, etc.
- Goal – find a better way to develop software

The Agile Manifesto and Core Values

- Frustrated by documentation-driven processes
- Stressed out by rigid software development processes
- Not satisfied with project delays and cost overruns
- Ultimately, software deliverables were not meeting customer needs due to requirements changing during development
- They wanted to restore credibility to the word methodology

Agile Manfiesto Website

The Agile Manifesto and Core Values

Individuals and interactions over processes and tools

Working software over comprehensive documentation

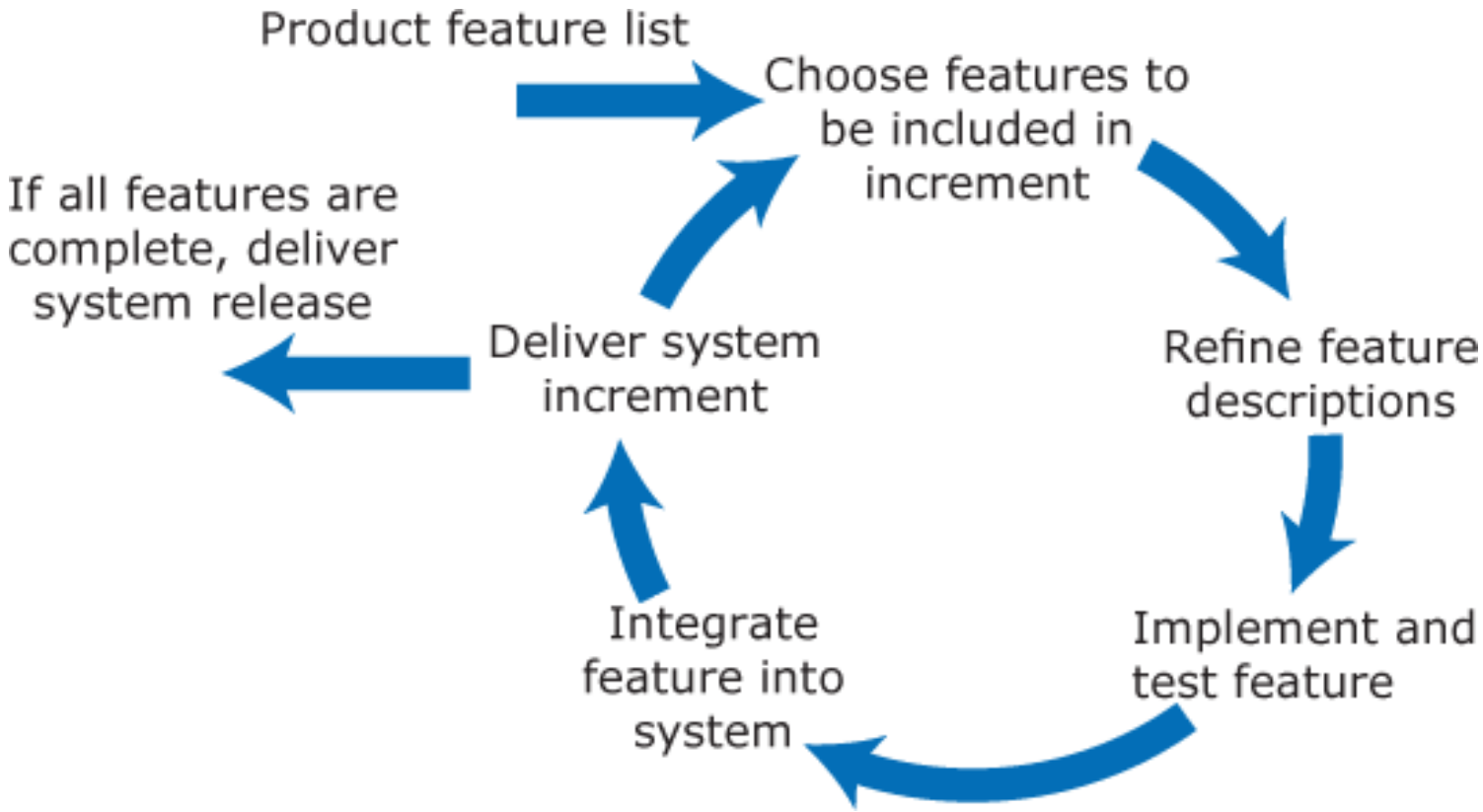
Customer collaboration over contract negotiation

Responding to change over following a plan

Key Agile Principles and Practices -- Iterative and Incremental Development

- Don't develop the entire system in one attempt
- Break the project down into small, manageable components to be worked on later
- Develop each subset of those components in short, iterative cycles (sprints)
- Sprints should be 1 – 4 weeks in duration
- Goal: deliver software incrementally
- Try to have a minimally viable product (MVP) early on

Key Agile Principles and Practices -- Iterative and Incremental Development



Key Agile Principles and Practices -- Iterative and Incremental Development

- **Why does it matter?**
- Early delivery of value
- Frequent opportunities for feedback
- Reduces the risk of building the wrong product,
- Allows the team to learn and adapt based on what they build and user feedback.

Key Agile Principles and Practice: Cross-functional, Self-organizing Teams

- Cross-functional teams are key to the successful implementation of agile
- Need frontend/backend/whole stack developers, testers (QA), designers (UI/UX), project managers, domain experts, and analysts
- No siloed teams of folks from one area (e.g., a separate requirements team, a design team, a development team, a testing team)
- Should be self-organizing – should have autonomy and responsibility to decide the best way to perform their work to complete goals for the current sprint

Key Agile Principles and Practices: Cross-functional Self-organizing teams

- Why does it matter?
- Reduce dependencies on external teams
- Streamline communication
- Empower the team to make timely, but thoughtful decisions
- Foster a sense of ownership over the project/product
- Self-organizing teams will be motivated, engaged, and innovating (in theory)

Key Agile Principles and Practices – Inspect and Adapt

- Core feedback loop in Agile
- Inspect process, product, and team dynamics
- Make adjustments to adapt
- How to do so?
- Daily standups – Short daily meetings (10-15 minutes) to describe progress and barriers and to synchronize activities
- Have iteration (sprint) reviews – Demonstrate the working product to stakeholders and gather feedback
- Have retrospectives – Team meets internally to discuss what went well in the last iteration, what could be improved, and how to make those improvements

Key Agile Principles and Practices – Inspect and Adapt

- Why does inspecting and adapting matter?
- Allows the team to learn from mistakes
- Refine their approach
- Ensure the team is headed in the right direction with the product and their collaboration

Key Agile Principles and Practices – Potentially Shippable Increment

- The goal is to have a working, tested, and integrated piece of software
- Should work reasonably well when tested by clients, end-users, or other key stakeholders
- Should be theoretically releasable (at least an MVP)
- Does not have all planned features early on; should have all must-haves and most should-haves at/near the end of the full development schedule
- Quite simply, each delivered increment should add value for clients and end-users
- In Waterfall, the client and end users would have a working product only at the end of the development process

Key Agile Principles and Practices – Potentially Shippable Increment

- Why does having potentially shippable increments matter?
- Ensures technical debt is managed
- Integration issues are found early
- Regressions are caught
- The team is always ready to deliver value (small wins at every stage, leading to success)
- Issues discovered late in the development process, like with Waterfall, are costly to fix

Key Agile Principles and Practices – Prioritization

- Scope is often flexible within iterations
- Prioritization is needed as a result
- Teams need guidance on features, requirements, stories, etc., that are the most important to the business/user to work on at a time
- Recommend a formal user requirements process and an organizational method for those requirements, like MoSCoW
- Why? So, the team is always working on the highest-value items, maximizing ROI for each iteration, and minimizing scope creep

MoSCoW Method

Student Participation Time ☺

- What are some user requirements we might have in a command-line slide extractor program? (MoSCoW method)
- We'll consider this product-focused development ☺?
- We'll sell our product into the Microsoft app store 🤣 or on our website.

Must Have

Command-line Input: The program must accept command-line arguments, specifying the PowerPoint file path and the desired slides to extract.

Slide Extraction: Must extract specific slides from a PowerPoint presentation based on user input.

Export Format: Must save the extracted slides in a commonly used format (e.g., PNG, JPEG, GIF, or PDF?).

Error Handling: Provide meaningful error messages for missing files, incorrect input formats, or unsupported file types.

Basic Documentation: Include a README explaining how to compile and use the program.

Should Have

01

1. **Batch Processing: Should** support processing multiple presentations in one run.

02

Custom Output Directory: Allow users to specify an output directory for extracted slides.

03

Supported Formats: Should handle both .ppt and .pptx formats.

04

Slide Content Filtering: Optionally extract slides containing specific keywords or text

Could Have

Metadata Extraction: Could extract metadata (e.g., slide notes, authorship, or timestamps) along with slides.

Styling Options: Could allow users to customize slide appearance (e.g., include watermarks or resize slides).

File Compression: Could provide an option to compress output files into a zip archive.

Multilingual Support: Could support command-line output in multiple languages for better accessibility.

Won't Have

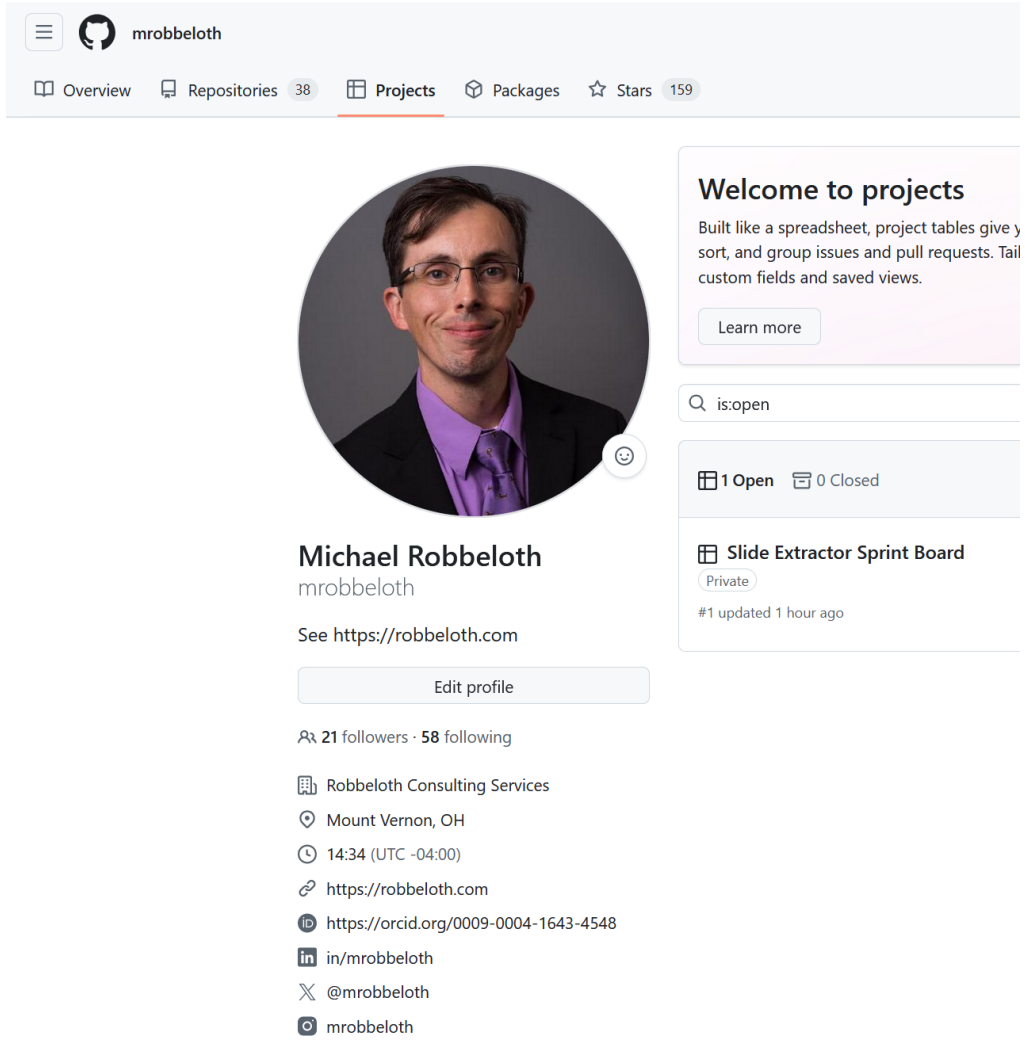
GUI Interface: This project is command-line-focused, so a graphical user interface won't be developed.

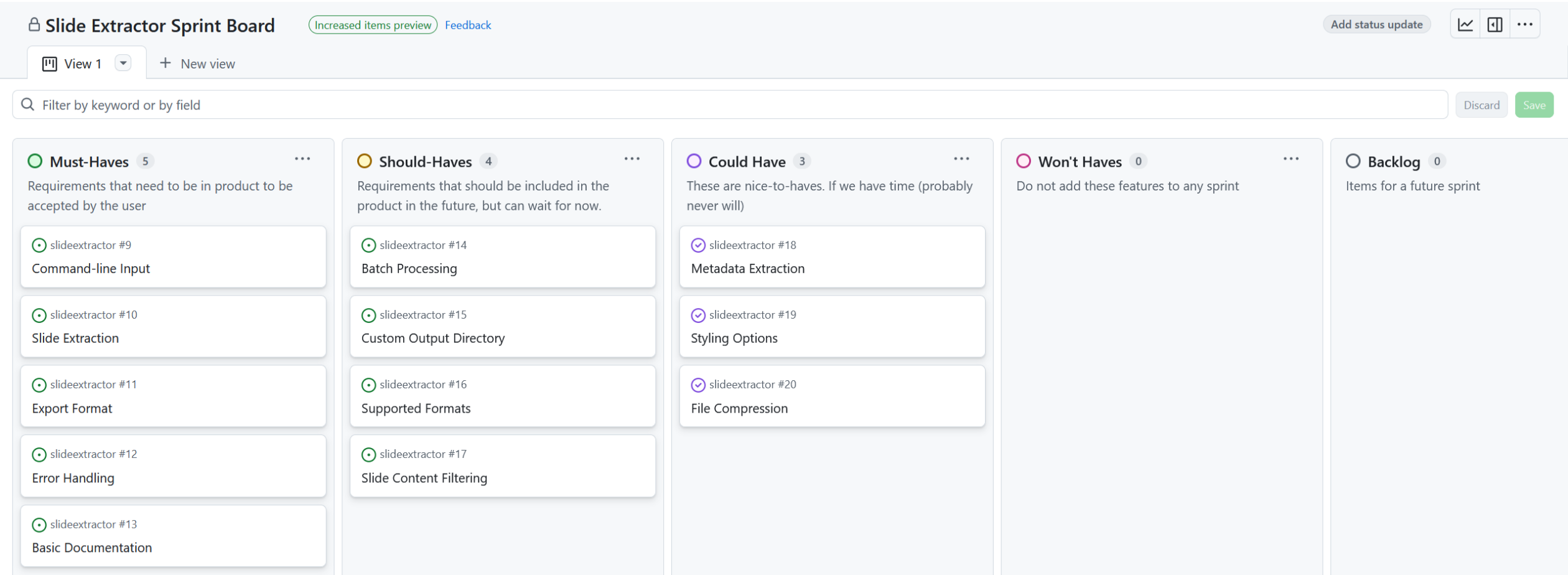
Slide Editing Features: Advanced features like editing slide content won't be included.

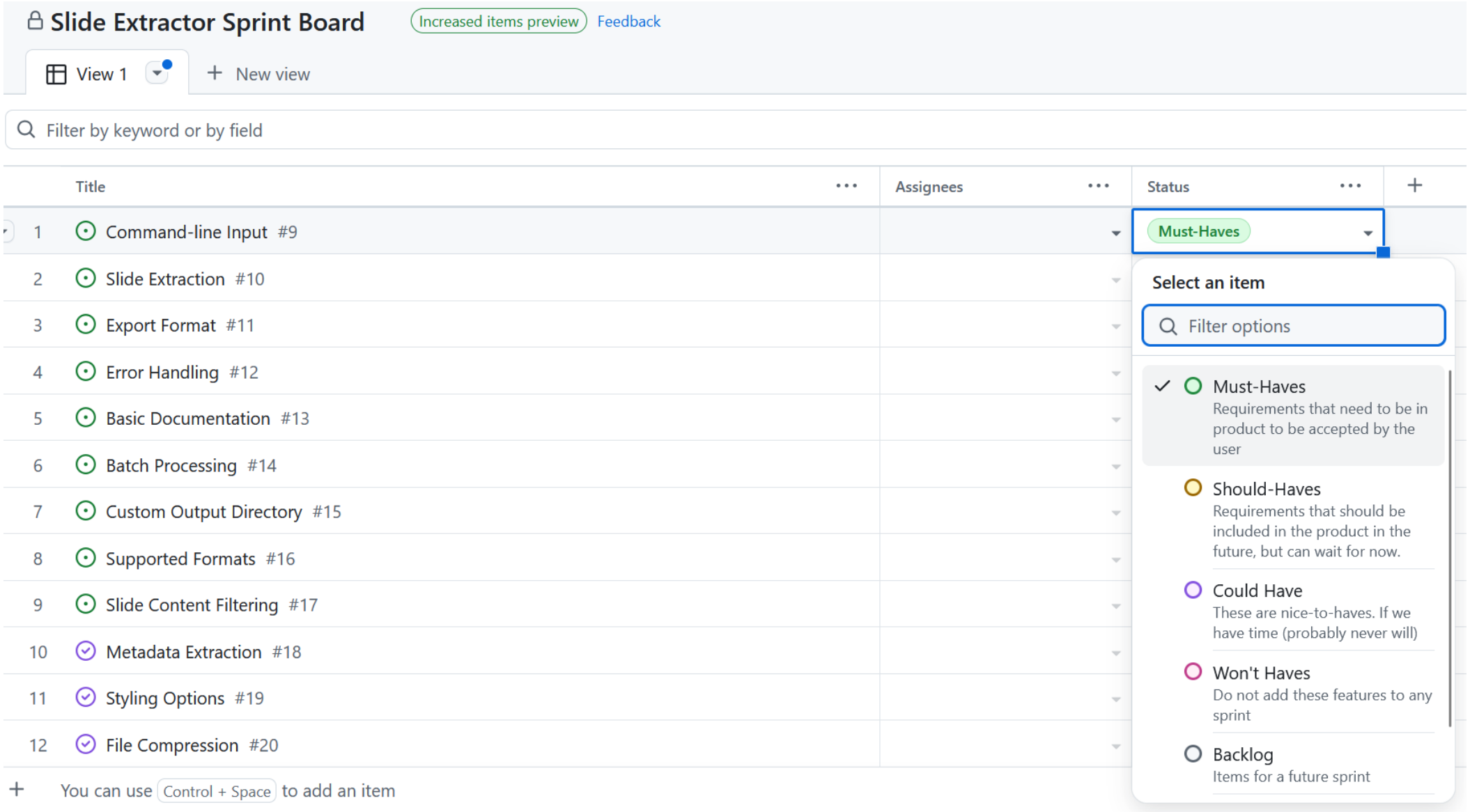
Cloud Integration: The program won't support uploading extracted slides to cloud services.

What do we do with these requirements?

- Just use the user requirements as is in a minimal documentation process
- Add user requirements to a project board with tooling of the development team's choice
- Criticism: ambiguous at this level, especially with/ different stakeholders
- Example with GitHub Projects to follow
- Could be a whiteboard using writable magnet strips instead, or an enterprise software development tool




Slide Extractor Sprint Board
Increased items preview
Feedback
Add status update
View 1
New view
Filter by keyword or by field
Discard
Save
Must-Haves 5
Requirements that need to be in product to be accepted by the user
slideextractor #9
Command-line Input
slideextractor #10
Slide Extraction
slideextractor #11
Export Format
slideextractor #12
Error Handling
slideextractor #13
Basic Documentation
Should-Haves 4
Requirements that should be included in the product in the future, but can wait for now.
slideextractor #14
Batch Processing
slideextractor #15
Custom Output Directory
slideextractor #16
Supported Formats
slideextractor #17
Slide Content Filtering
Could Have 3
These are nice-to-haves. If we have time (probably never will)
slideextractor #18
Metadata Extraction
slideextractor #19
Styling Options
slideextractor #20
File Compression
Won't Haves 0
Do not add these features to any sprint
Backlog 0
Items for a future sprint


Slide Extractor Sprint Board
Increased items preview
Feedback
View 1
New view
Filter by keyword or by field
Title
Assignees
Status
1 Command-line Input #9
Must-Haves
2 Slide Extraction #10
3 Export Format #11
4 Error Handling #12
5 Basic Documentation #13
6 Batch Processing #14
7 Custom Output Directory #15
8 Supported Formats #16
9 Slide Content Filtering #17
10 Metadata Extraction #18
11 Styling Options #19
12 File Compression #20
You can use Control + Space to add an item
Select an item
Filter options
Must-Haves
Requirements that need to be in product to be accepted by the user
Should-Haves
Requirements that should be included in the product in the future, but can wait for now.
Could Have
These are nice-to-haves. If we have time (probably never will)
Won't Haves
Do not add these features to any sprint
Backlog
Items for a future sprint

Benefits of Agile

- Increased flexibility and adaptability to change.
- Faster delivery of value to customers.
- Improved communication and collaboration.
- Higher customer satisfaction.
- Better team morale.

Challenges of Agile

- Requires a significant cultural shift.
- Can be challenging with distributed teams (though tools help).
- Requires active stakeholder involvement.
- Estimation can be tricky initially.
- There is still a potential for scope creep if not managed well.

Challenges of Agile

May not be appropriate for certain types of projects that:

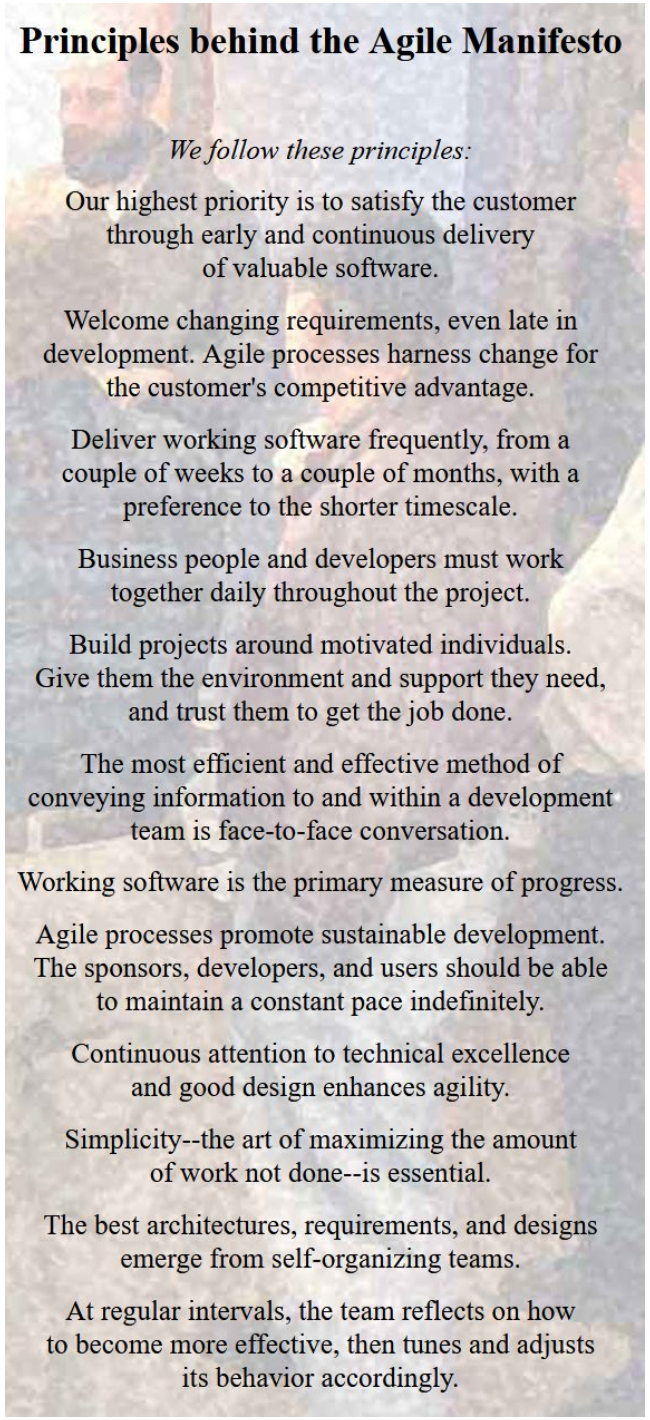
- Require formal contracts with requirements spelled out
- Those utilizing formal verification techniques
- Projects utilizing real-time, embedded systems with the potential to hurt or kill end-users or otherwise cause catastrophic damage
- Large-scale system of systems development with geographically dispersed teams (could use modified agile processes)
- In general, business may find a hybrid approach of traditional and agile methods to be appropriate

Conclusion

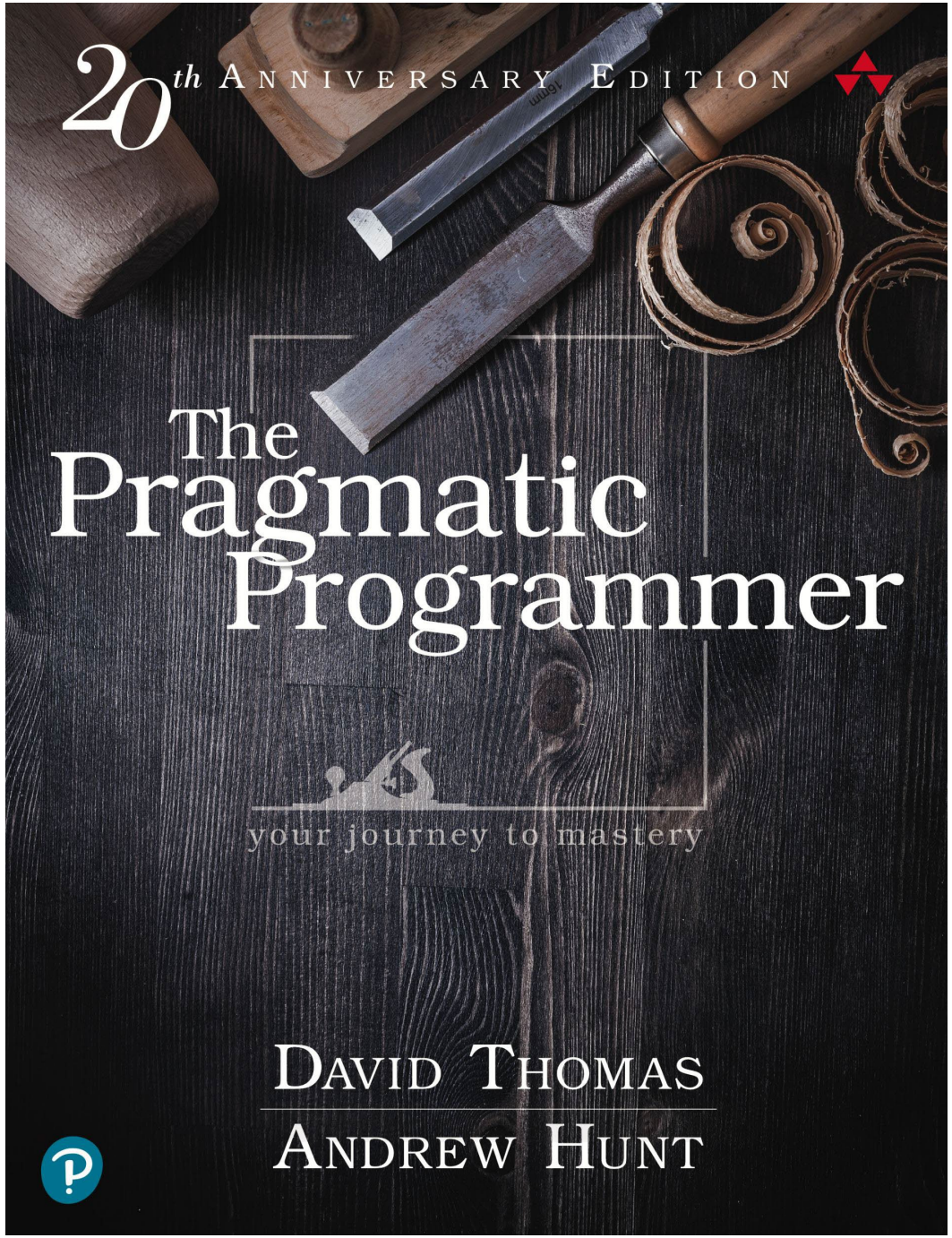
- Agile lets us move away from traditional, plan-driven methods when flexibility in development is required
- We are replacing rigid processes with ones that value individuals and interactions, working software, customer collaboration, and responding to change.
- Our goals with agile are to always deliver incrementally better software that cumulatively adds value for the end users and client
- I would encourage you to look at various Agile related methodologies like Scrum or Kanban

Questions?

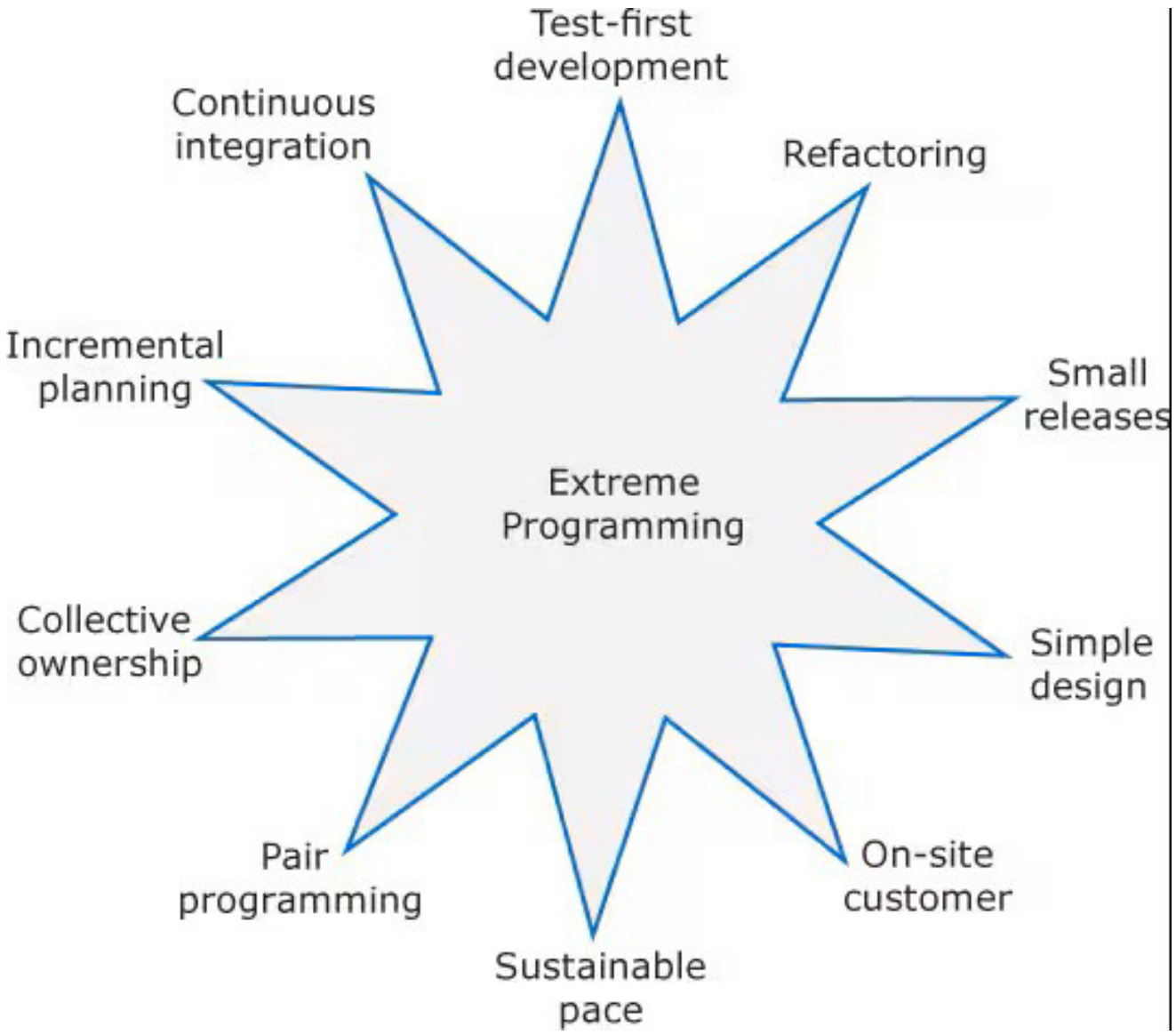
Appendix – Principles of Agile Manifesto



Appendix – Pragmatic Programmer



Appendix – Extreme Programming



Appendix – Scrum