[CS3704] Software Engineering

Shawal Khalid
Virginia Tech
1/29/2024

Announcements

- **Homework 1due Feb 05**
  - Canvas questions (Upload file or input text)

Software Process II

Unified Process
Agile Methodologies
Research Discussion

Learning Outcomes

By the end of the course, students should be able to:

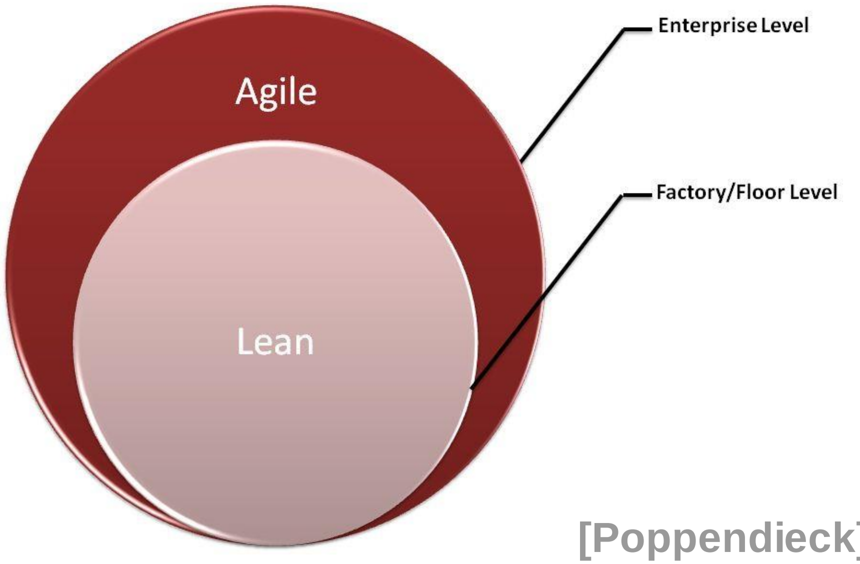
- **Understand software engineering processes, methods, and tools used in the software development life cycle (SDLC)**
- Use techniques and processes to create and analyze requirements for an application
- Use techniques and processes to design a software system
- Identify processes, methods, and tools related to phases of the SDLC
- **Explain the differences between software engineering processes**
- Discuss research questions and current topics related to software engineering
- Create and communicate about the requirements and design of a software application

Lean Software Development

- Agile-based framework focused on optimizing development time and resources

<u>7 Principles</u>

1. Eliminate Waste
2. Build Quality In
3. Create Knowledge
4. Defer Commitment
5. Deliver Fast
6. Respect People
7. Optimize the Whole



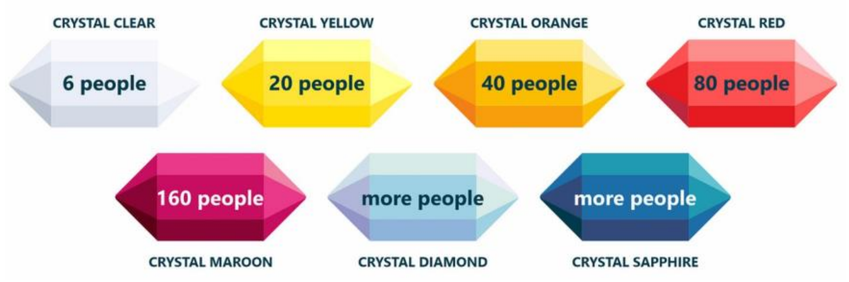
[Poppendieck]

Crystal Agile Framework

- Human-powered, adaptive, and ultra-light (little to no documentation) agile framework

<u>7 Principles</u>

1. Frequent Delivery
2. Reflective Improvement
3. Osmotic Communication
4. Personal Safety
5. Focus (on Work)
6. Easy Access to Expert Users
7. Technical Environment



[Cockburn]

Agile Pros and Cons

Pros:

– Customer satisfaction by rapid, continuous delivery of useful software

– Close, daily cooperation between business people and developers

– Better software quality and lower cost

Cons:

– People may lose sight of the big picture

– Heavy client participation is required

⚠️

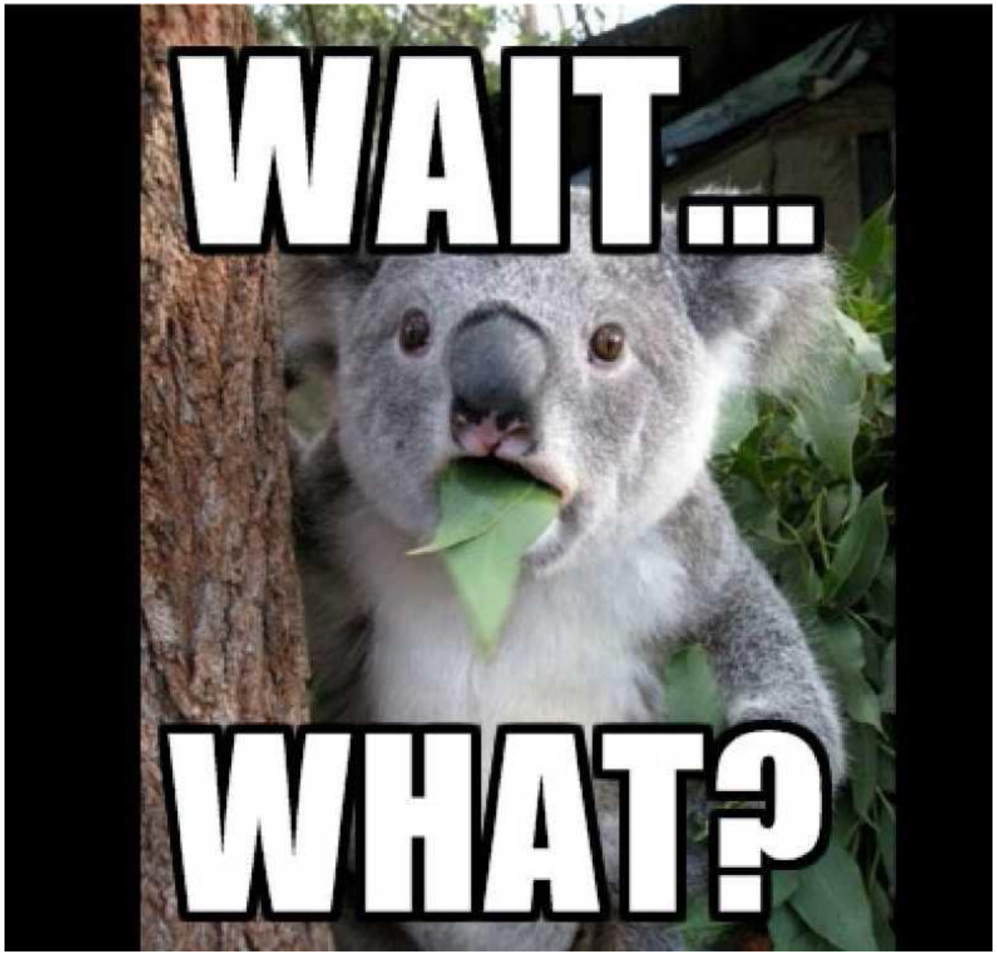
There is no perfect software engineering process…

- *Agile development is not the silver bullet!*

<u>The First Law</u>

*“No matter where you are in the system life cycle, **the system will change**, and **the desire to change it will persist throughout the life cycle.**”*

[Bersoff, 1980]

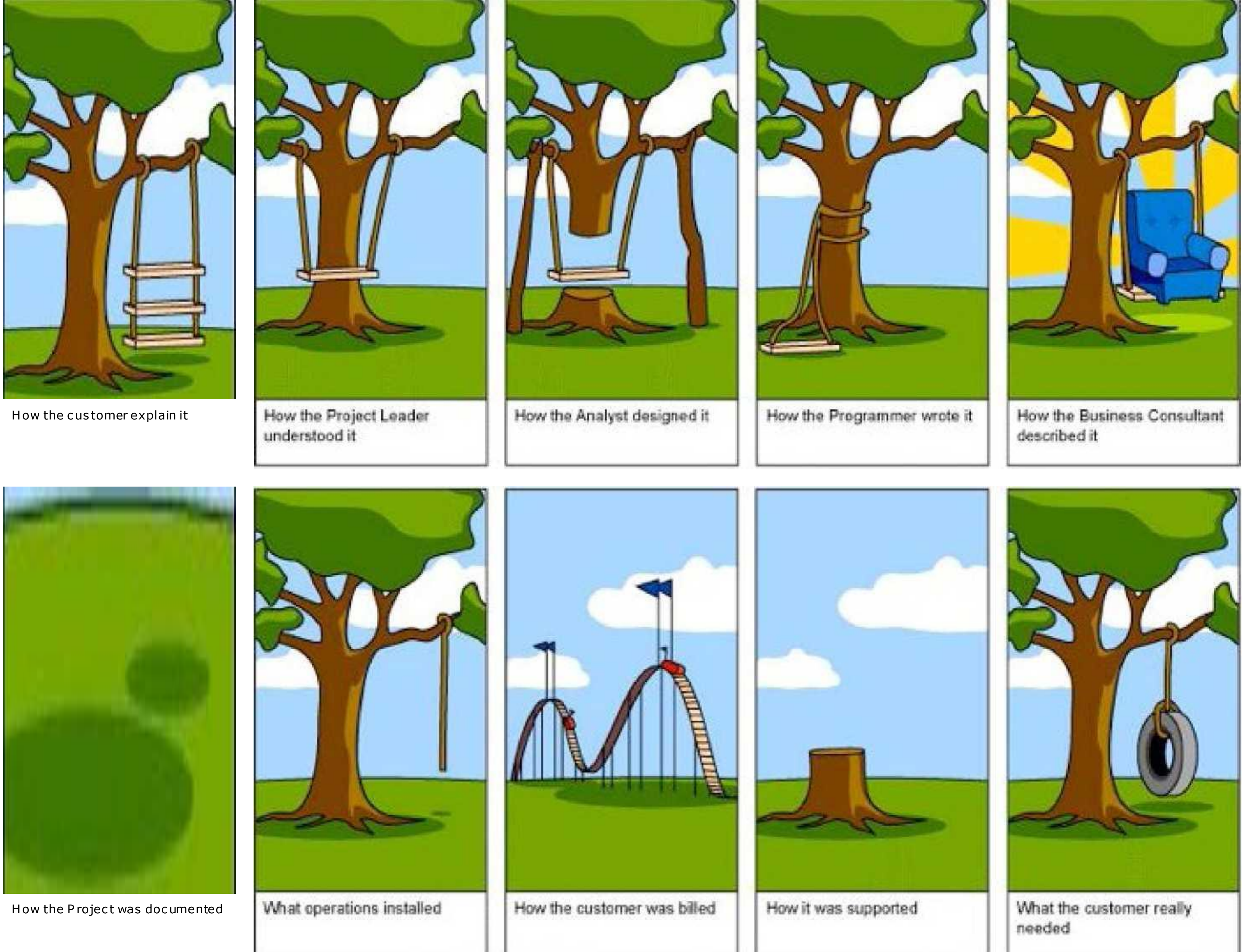


Agile Software Development

Be the customer

... just for

a minute


How the customer explain it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the Project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

3 Things ...

- We wish were true
- The **customer knows** *exactly* what they want The **developers know** exactly how to build it **Nothing** will **change** along the way

- We have to live with The **customer discovers** what he wants The **developers discover** how to build it **Many** things **change** along the way

Why adopt Agile

Industry Results*

Speed	⇧	-*__Exceeding expectations__ for both the project delivery timeline and scope*-- Small Business Executive
	⇧	25-50% reduction in Time To Market
Team Morale	⇧	*"Besides being extremely beneficial, working using Agile practices has boosted our company's team morale much better than if I were to give employees a 30% raise. "* - Director of an IT Department of a large corporation
Cost	⇩	10-30% cost reduction
Defects	⇩	20-65% reduction in number of defects

*** Source: IBM Expert interviews, ThoughtWorks experts interview, Forrester, Literature research**

Who's Adopting Agile

Accenture
AOL
Booz Allen Hamilton
BMC Software British
Telecom Business
Week Capital One
CNBC DTE Energy

Google Key
Bank Kronos
LinkDotNet
Microsoft
Nationwide
NBC Universal
Raya Corp.
Rockstar

Sapient
Siemens
Sprint
StateFarm
Nokia-Siemens
Shopzilla
Thoughtworks
Yahoo !

and many, many, many more …

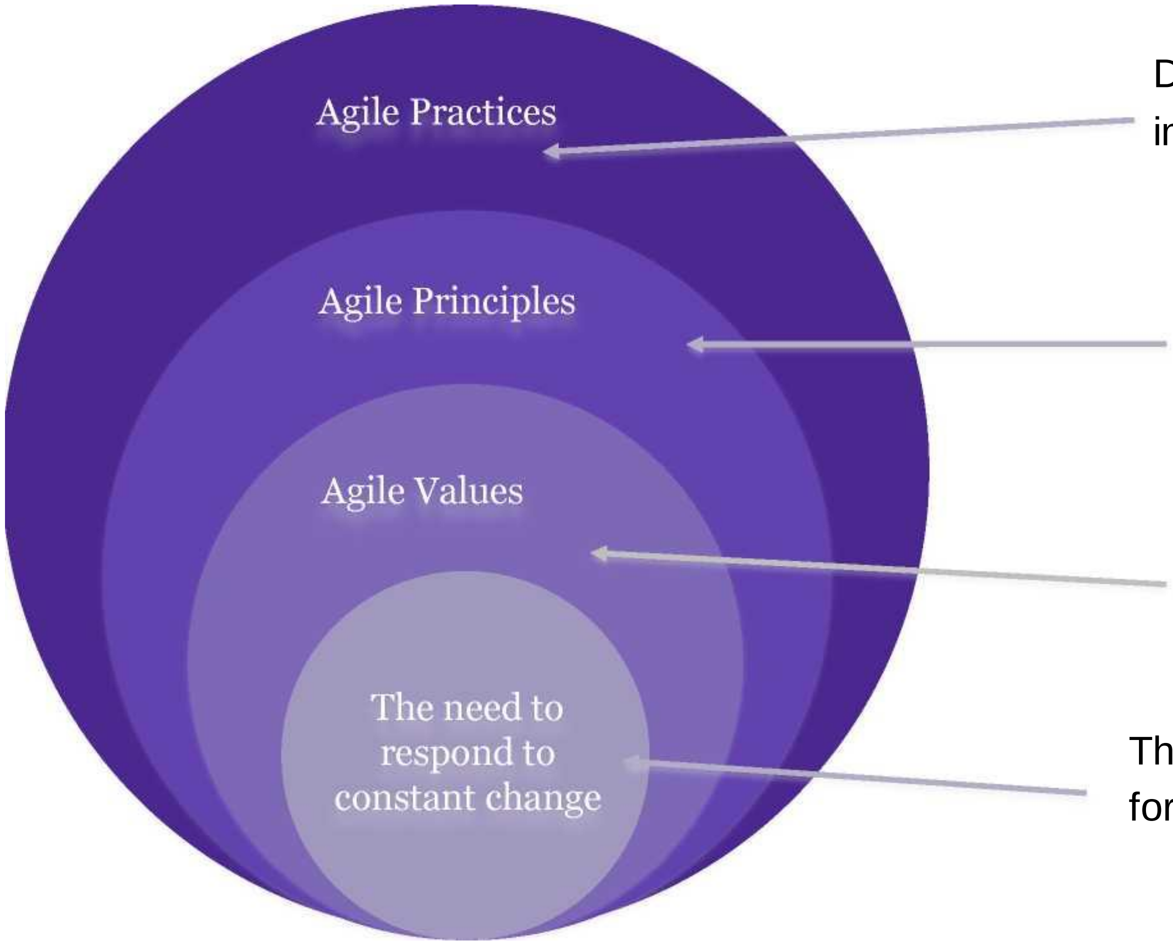
Agile is NOT …

- **NOT New**
- **NOT A silver bullet**
- **NOT Without planning, documentation, architecture, design**
- **NOT License to hack**
- **NOT An excuse for poor quality**
- **NOT Undisciplined**
- **NOT Unproven**

Agile in a word

Agile is a **mindset** *defined by* ***values*** *guided by* ***principles*** *and manifested through many different* ***practices***

Agile at a Glance



Defines in detail how this is implemented in practice

Defines a set of ways to meet the values

Defines the set of most important beliefs of what is truly important

The fundamental reason for a “new” paradigm

2001: The Agile Manifesto (Agile Values)

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

° Individuals and interactions over *processes and tools* ° Working software over *comprehensive documentation* ° Customer collaboration over *contract negotiation* ° Responding to change over *following a plan*

That is, while there is value in the items on the right, we value the items on the left more.

Agile Principles

1. Our highest priority is to **satisfy the customer** through early and continuous delivery of valuable software.
2. Welcome **changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver **working software frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people & developers must **work together daily** throughout project.
5. Build projects around **motivated individuals**. Give them the environment and support they need, and **trust** them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation**.
7. Working software is the primary **measure of progress**.
8. Agile processes promote **sustainable** development. The sponsors, developers, and users should be able to maintain a **constant pace** indefinitely.
9. Continuous attention to **technical excellence** & good design enhances agility.
10. **Simplicity**--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, & designs emerge from **self-organizing teams**.
12. At regular intervals, the **team reflects** on how to become more effective, then tunes and adjusts its behavior accordingly.

Problems with agile methods

- It can be difficult to **keep the interest of customers** who are involved in the process.
- Team members may be unsuited to the **intense involvement** that characterizes agile methods.
- **Prioritizing** changes can be difficult where there are **multiple stakeholders**.
- Maintaining simplicity requires **extra work**.
- **Contracts** may be a problem as with other approaches to iterative development.

Light-Weight “Pre-Agile” Methodologies

- **Extreme Programming** (Kent Beck, Ward Cunningham, Ron Jeffries)
- **Scrum** (Ken Schwaber and Jeff Sutherland)
- **Lean Software Development** (Mary and Tom Poppendieck)
- **Crystal Methods** (A istair Cockburn)
- **Feature Driven Development** (Jeff Deluca)
- **Dynamic Systems Development Method** (DSDM Consortium)

Scrum

- **Product Backlog - Product Owner**
- **Sprint - Sprint Backlog - Sprint Review**
- **Daily Scrum - Burndown**
- **Self Organizing - Cross Functional Team**
- **Scrum Master**

“It’ Depends On Common Sense”

Extreme Programming (XP)

- **Customer - Coach**
- **User Stories - Acceptance Tests**
- **Planning Game - Stand Up Meeting**
- **Iterations - Releases**
- **Velocity - Yesterday's Weather**
- **TDD - Refactoring - Evolutionary Design**
- **Continuous Integration**

"Embrace Change"

Agile Practices

- Self Organizing Teams
- Release Planning
- Iteration Planning
- Deliver Frequently
- Time boxing
- Client-Driven Iterations
- Retrospectives
- Team Room
- **Pair Programming**
- Automated Unit Testing
- Test Driven Development
- Continuous Integration
- Refactoring
- More ...

What is Pair Programming (PP)?

- A software development technique in which two programmers work as a pair together on one workstation.

PP Roles

- The **driver**: writes code
- The **navigator** (observer, copilot): reviews each line of code as it is typed in.

- The two programmers switch roles frequently.

Software Crisis

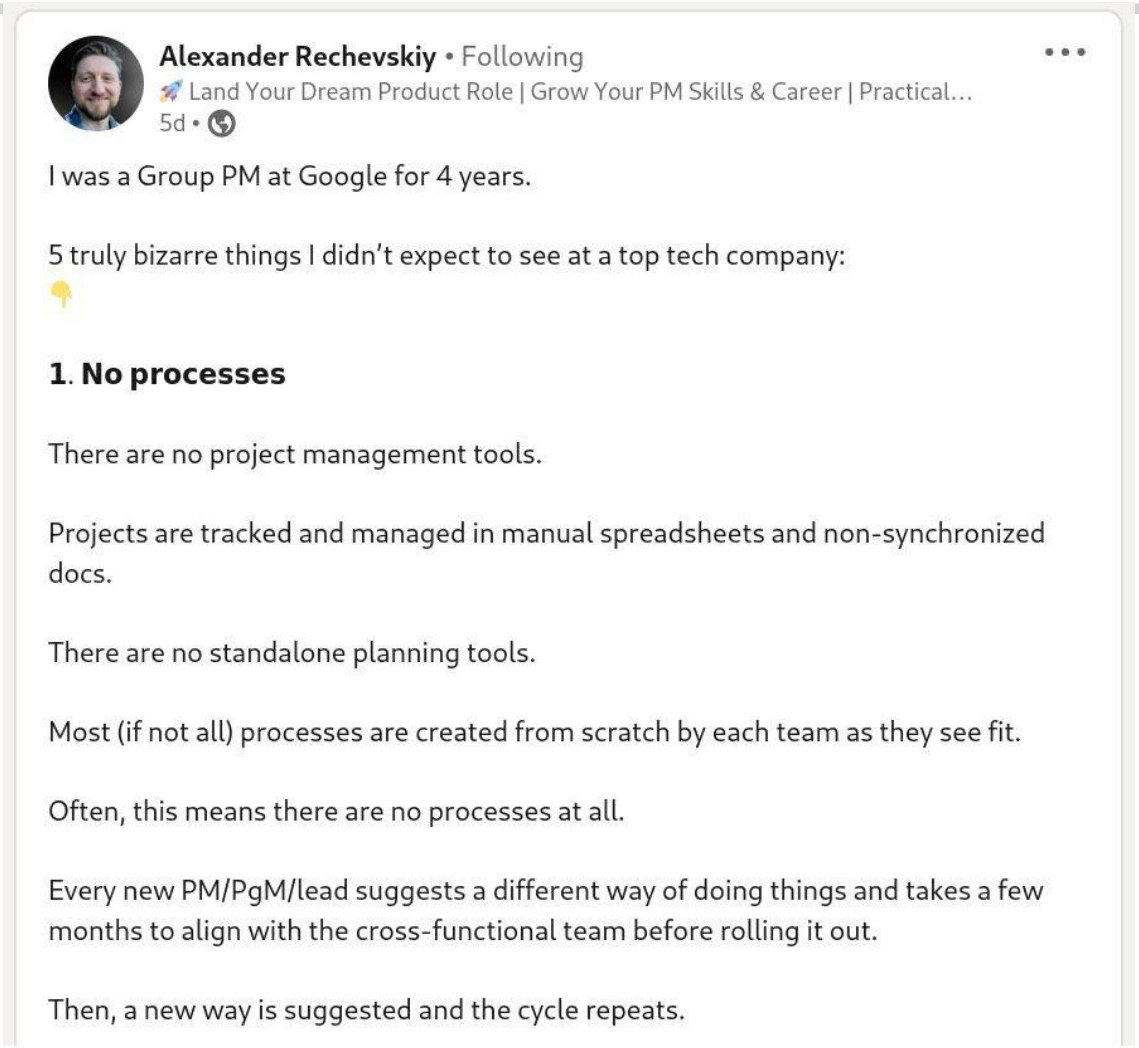
"*The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.*" [Dijkstra]

Software Crisis

- Projects running over-budget
- Projects running over-time
- Software was very inefficient
- Software was of low quality
- Software often did not meet requirements
- Projects were unmanageable and code is difficult to maintain
- Software was never delivered

Current SE Problems

- Software is too expensive and takes too long to build
    - Frequently over budget and over time!
- Low software quality
    - 3.6 billion users, $1.7 trillion caused by bugs [Tricentis, 2017]
- Software is more complex to support and maintain
- More users = more difficult to scale applications
- Failing to meet requirements
- Lack of diversity in software development teams
- Inadequate testing and security
- Ethical decisions in software engineering
- ***What else???*** **[Discuss w/ partner, and in HW1]**



Alexander Rechevskiy • Following
🚀 Land Your Dream Product Role | Grow Your PM Skills & Career | Practical...
5d •

I was a Group PM at Google for 4 years.

5 truly bizarre things I didn't expect to see at a top tech company:
👇

1. No processes

There are no project management tools.

Projects are tracked and managed in manual spreadsheets and non-synchronized docs.

There are no standalone planning tools.

Most (if not all) processes are created from scratch by each team as they see fit.

Often, this means there are no processes at all.

Every new PM/PgM/lead suggests a different way of doing things and takes a few months to align with the cross-functional team before rolling it out.

Then, a new way is suggested and the cycle repeats.

Software Engineering?

“ ‘It’s Engineering... but not as we know it’… Software Engineering - solution to the software crisis, or part of the problem? ”

[Bryant]

Next Class…

- **Project milestone coming up**

  **** Use the** `project` **Slack channel or Canvas chat to find teammates!***

References

- Standish Group. *“The CHAOS Report”*. <https://www.standishgroup.com/sample_research_files/chaos_report_1994.pdf>
- Beck, et al. *“Agile Manifesto”*. 2001 <https://agilemanifesto.org/>
- RS Pressman. *“Software engineering: a practitioner's approach”.*
- Alistair Cockburn. *“Crystal clear: A Human-powered methodology for small teams”*
- Tom and Mary Poppendieck. *“Implementing Lean Software Development”*
- Dr. Chris Brown, Dr. Seyam, Na Meng and Barbara Ryder
- Chris Parnin, Software process phase slides based on NCSU CSC216 slides by Sarah Heckman