Agile and Reliable Software

Ann Marie Neufelder

ann.neufelder@missionreadysoftware.com

321-514-4659

Problem to be solved

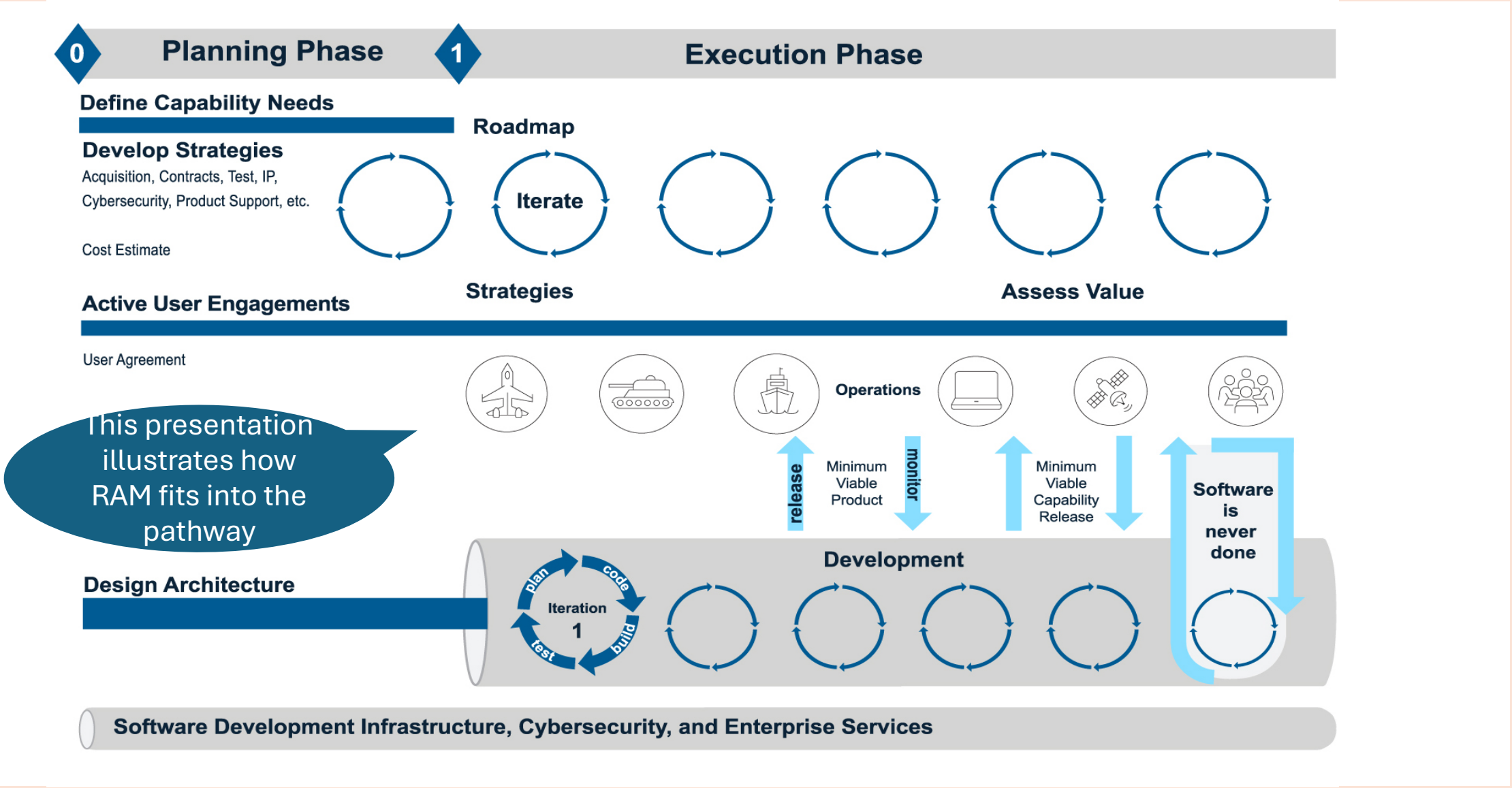
Reliable software tasks work whether the life cycle model is Waterfall, Agile or any other incremental model.

- Prediction models
- Failure tracking and trending
- Software FMEAs
- Reliable software testing

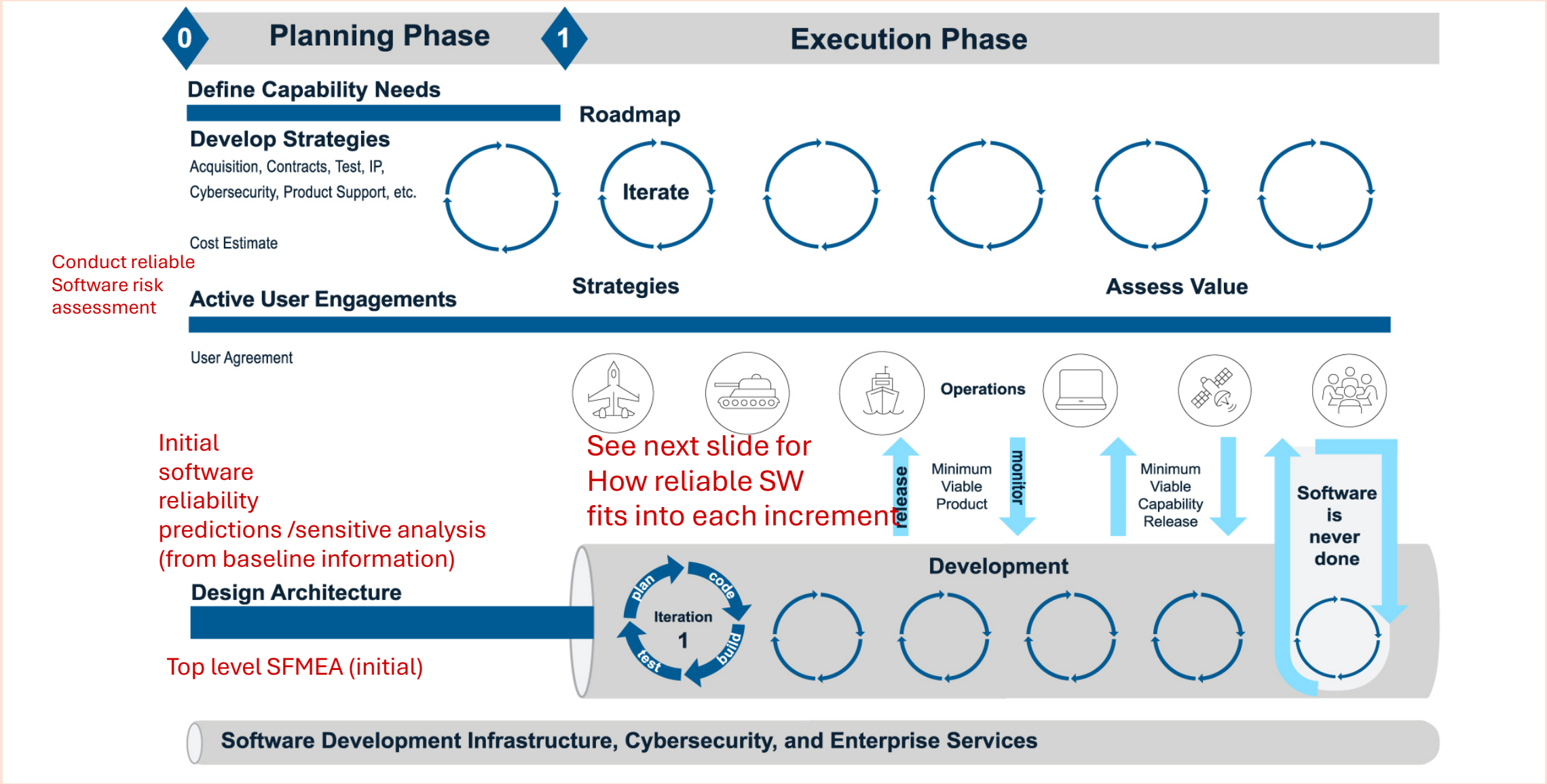
This presentation will show

- Where the reliable software tasks fit in with the DoD’s Agile framework
- How Agile development correlates to reliable software.

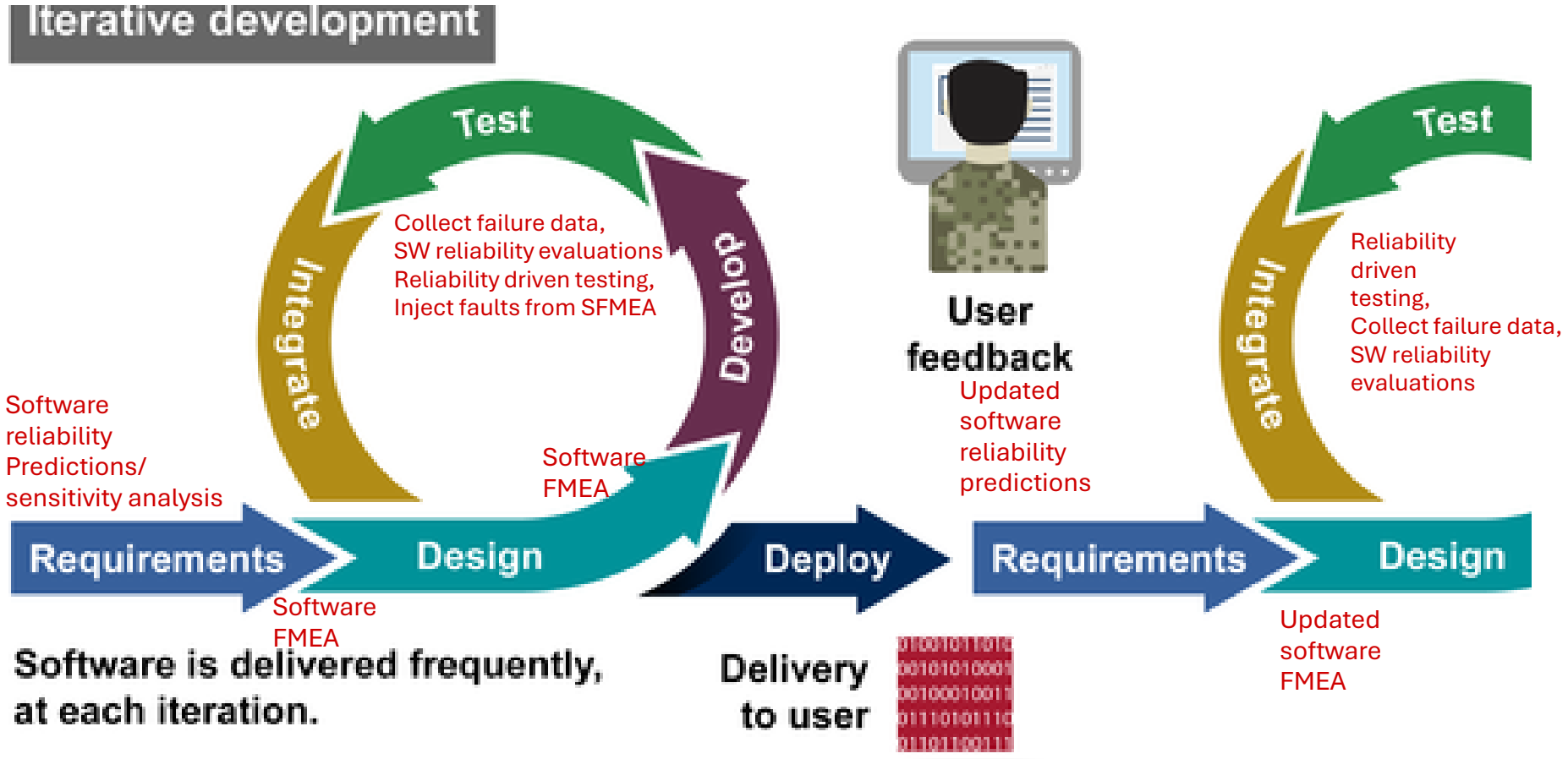
Software intensive pathway



Software intensive pathway with RAM tasks



Software intensive pathway with iterative RAM



Source: GAO analysis of Department of Defense (DOD) and industry documentation. | GAO-21-105298

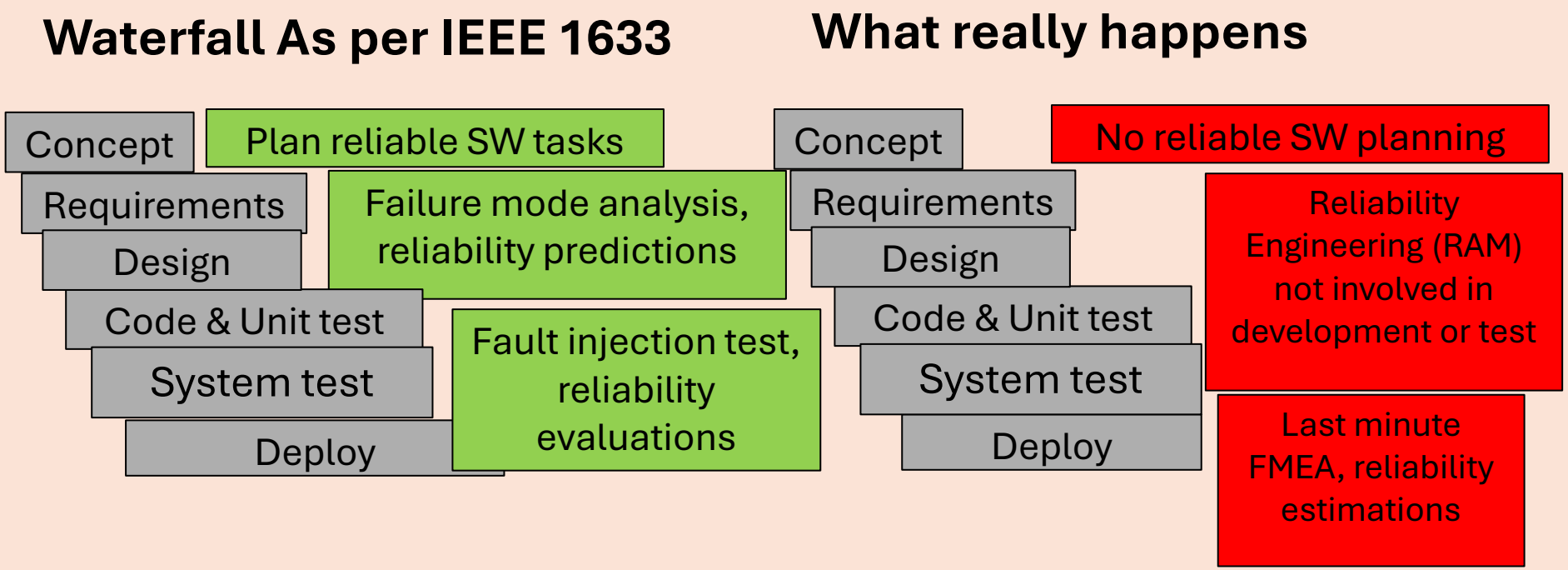
The statistics show that certain agile development factors are correlated to the success of the software intensive system[1]

Factor	Successful	Mediocre	Distressed
Average defect density in operation in terms of defects per normalized size	0.063	0.36	1.51
Cycle time in years between feature releases	0.41	1.04	3.25
Schedule granularity for each software engineer is in days or weeks	67%	53%	19%
The longest activity of any development increment is the requirements and design	60%	46%	11%
The software group executes the best life cycle model for this project	90%	59%	20%
The software group proactively involves and seeks the approval of the stakeholders	77%	68%	25%
Every software engineer understands/has been trained on the life cycle model being executed	83%	53%	17%
This is a relatively small/incremental/spiral release (< 10% of existing code changed or added)	27%	7%	0%
Stakeholders/customers are involved in deriving and reviewing the software requirements.	77%	76%	69%
The design is prototyped	69%	39%	19%
The software testers get involved early in development	79%	17%	0%
There are regular reviews between systems testers and software management	100%	60%	0%

Risks that aren’t necessarily mitigated by Agile

- Low test coverage or level of rigor in testing
- Not enough defects are fixed in each sprint - so they pile up to the next sprint
- Software engineers (SE) don’t have end user/industry knowledge
- SEs misunderstand the user stories (largely due to lack of end user experience)
- SEs skip design or aren’t good at it
- SEs don’t test software in real world environment
- SEs don’t do consistent unit testing against design or specifications
- SEs don’t consider all failure modes or scenarios
- SEs aren’t good at estimating how much work they can do in a sprint (which leads to late deliveries which is never good for reliability)
- Despite the fact that CD/CI was invented for this purpose- SEs don’t take advantage of data from each sprint so as to plan/replan the scope and effort for future sprints
- SEs sometimes tag “Agile” to anything they conveniently do or do not want to do
- SEs typically want development tasks to be “Agile” but reliability tasks to be “Waterfall”

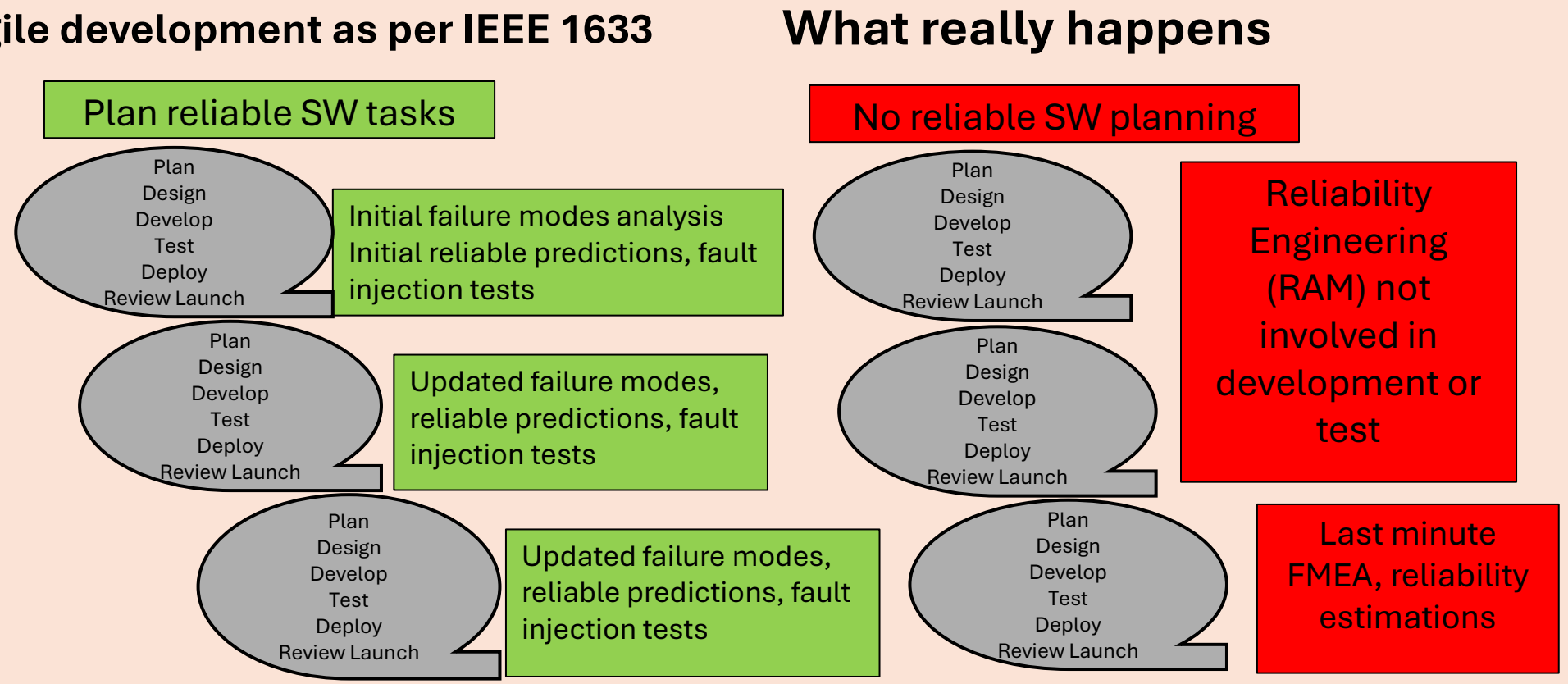
Ideal versus real world relationship between reliability and software engineering



Reliable SW tasks are supposed to be in line with development

In real world reliable SW tasks are done after project is already in trouble and no time to fix anything. Everyone is blind sided by poor reliability.

Ideal versus real world relationship between reliability and software engineering



- Reliable SW tasks are supposed to be integrated with CD/CI

- Reliable SW tasks are done after project is already in trouble and no time to fix anything. Everyone is blind sided by poor reliability.

Solutions

- Understand that software is doubling in size every 4 years
  - Pretending that it will go away hasn’t worked
- RAM adds value by
  - Understanding what the software does and what can go wrong
  - Not waiting until Materiel release to present results – they won’t get addressed
  - Developing an effective FDSC that is software-centric
  - Helping software engineering understand the FDSC and consider it in scoring
  - Focusing on failure modes that are costly to fix if found late
  - A good statement of work

References

[1] A.M. Neufelder, “The cold hard truth about reliable software edition 7”, 2024.