

Secure Agile development according to SAMM

Rob van der Veer

September 26 2019



In this talk I'd like to show you the work we've done to extend SAMM with guidance on how to continuously build security in when you're 'doing Agile'.

GLOBAL APPSEC AMSTERDAM

Pleased to meet you

Rob van der Veer

Security & privacy practice lead

Software Improvement Group

Meet us in the Expo space!



- > SIG measures software quality
- > SIG advises on getting software right
- > 150 people
- > Offices in Amsterdam, Antwerp, Athens, Copenhagen, New York
- > Involved in many research collaborations and standardisation



r.vanderveer@sig.eu

[@robvanderveer](https://twitter.com/robvanderveer)

+31 6 20437187

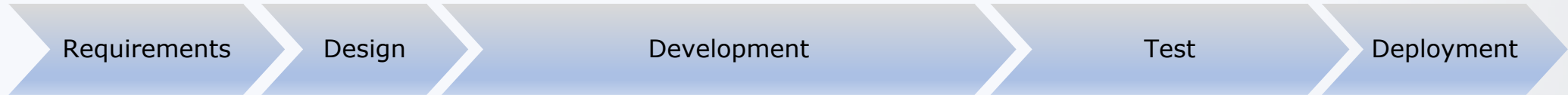
www.sig.eu/security

The (secure) SDLC



A long time ago, when I first started doing talks on software development, I showed this very generic diagram, and I would base my story on these activities. But then what happened is this:

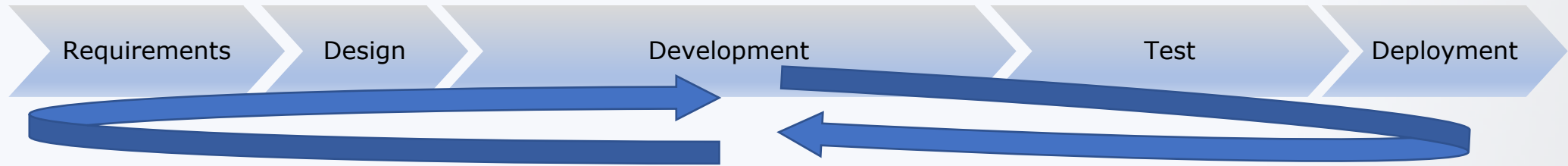
The (secure) SDLC



I increasingly got complaints and people saying that this is completely not agile, and thereby everything I say about software engineering will be incorrect and a blatant lie.

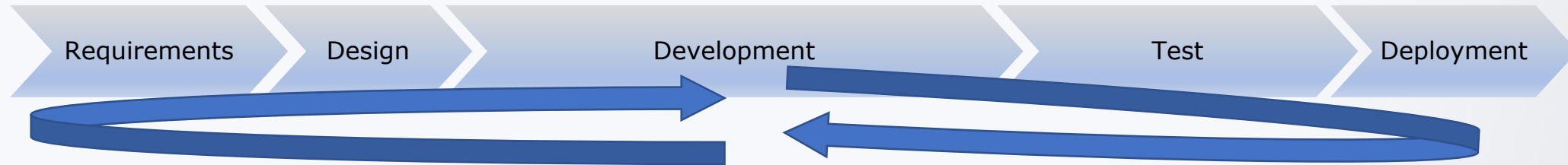
Of course the things I was saying were about the activities, and you DO have these activities in one form or another in Agile, but just to be done with it, I did this:

The (secure) SDLC



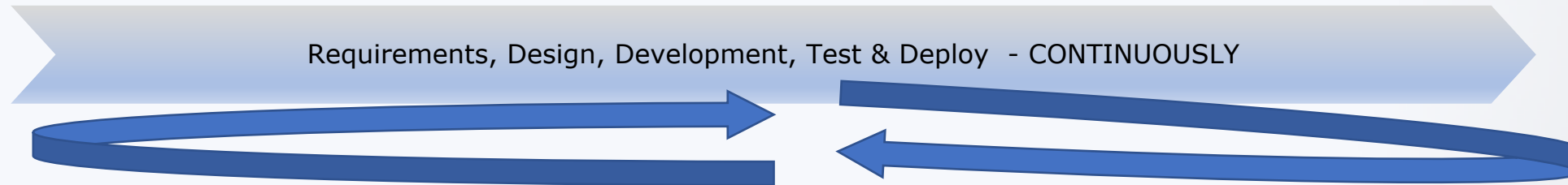
Tadaaa. Everybody happy! But no. In the last couple of years I would get more of this:

The (secure) SDLC



Apparently I was stepping on some other toes. So I decided to visualize the SDLC as what is often is nowadays: one continuous blur of activities that are intertwined:

The politically correct (secure) SDLC?



But guess what:

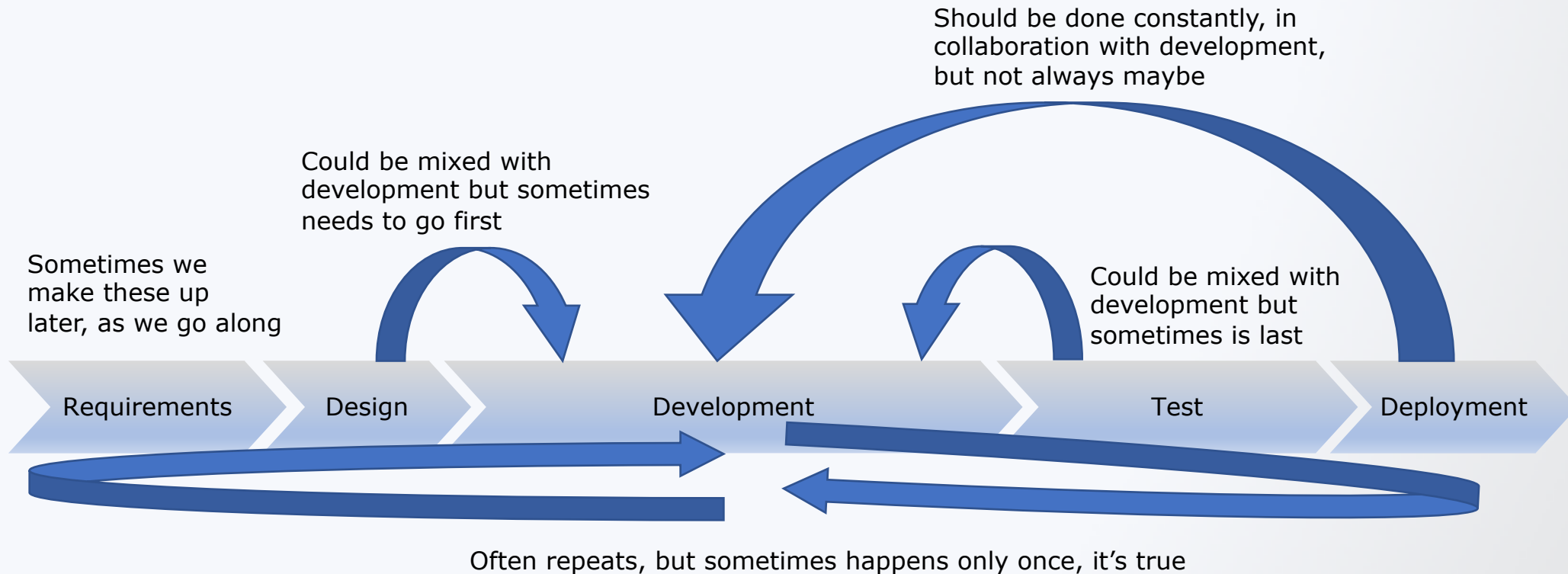
The politically correct (secure) SDLC?

Requirements, Design, Development, Test & Deploy - CONTINUOUSLY



Waterfall turns out to be not dead. So I figured I was perhaps listening to my audience too much. At that point I stopped. Otherwise it might have ended like this:

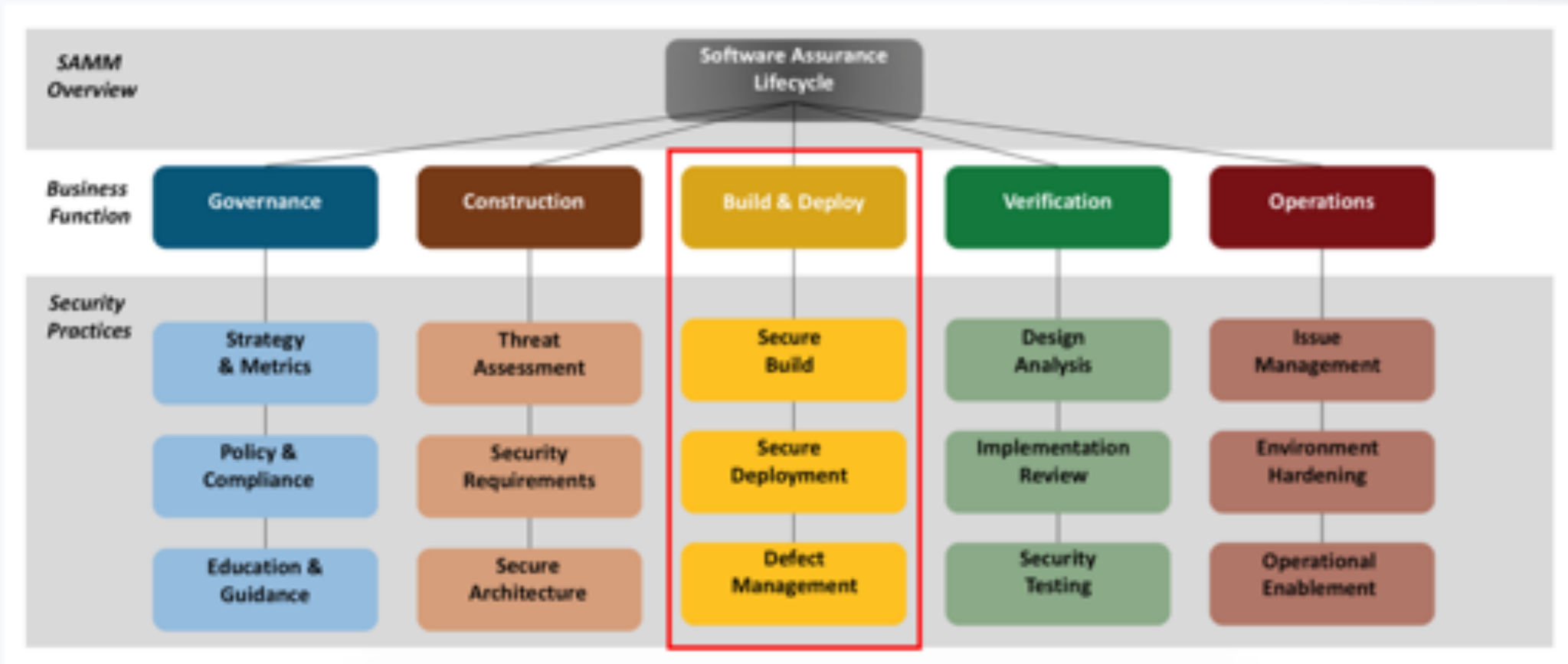
The politically correct (secure) SDLC!



My point is: by ordering activities to discuss them separately, you are not saying that these activities have to be separate phases in the process. This also applies to SAMM:

SAMM

SAMM 2.0 [owasp samm.org](https://owasp.org/www-project-samm/)



The Software Assurance Maturity Model, is OWASP's *flagship* project on how to setup and grow a secure development process. It's method-agnostic which is why Agile was not covered. So.....

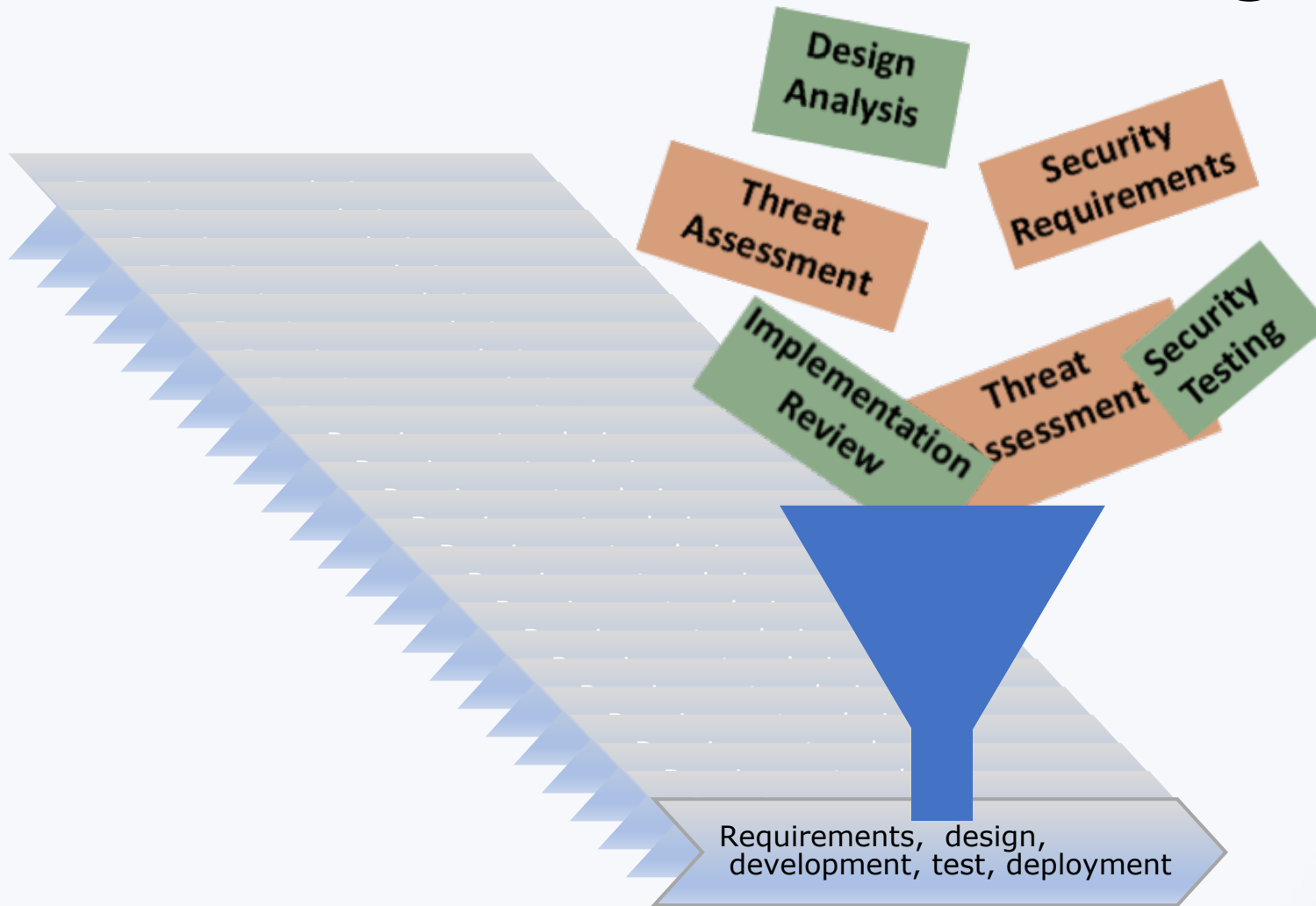
SAMM

SAMM 2.0 [owasp.org](https://owasp.org/samm)



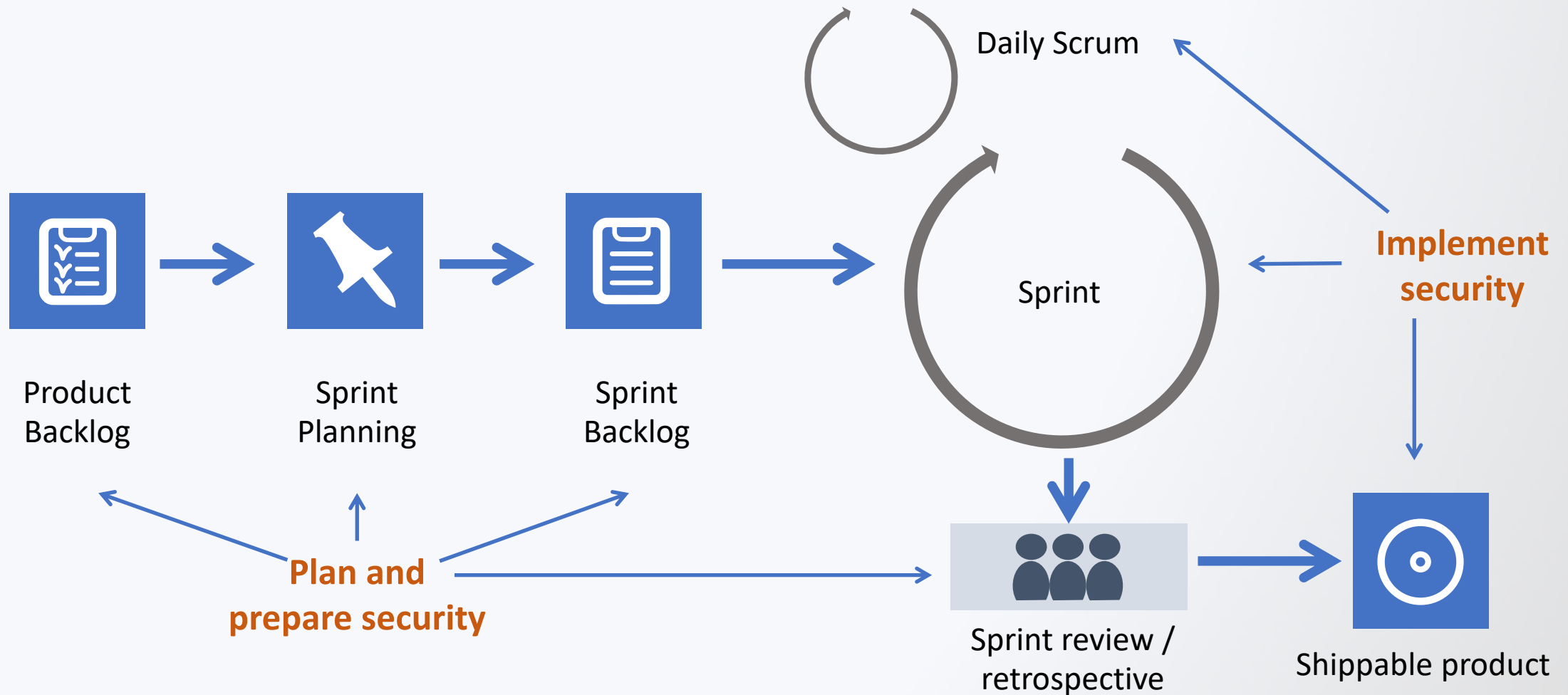
There appears to be a strong need in the industry for guidance on how to make secure software development work in an Agile environment.

How to secure the SDLC in Agile



You do many short iterations and every iteration you want to deliver software that is ready, with security built in. So how do you squeeze all the necessary activities into a sprint?

Security in Agile (SCRUM)



SCRUM, as an example of agile requires security to be built in and prepared well.

How do you do it? Let's look it up..



“Tracking and dealing with nonfunctional requirements like security and reliability is an unsolved problem in Agile development. “

So much for looking it up.

Customer quotes – agile security is a problem

“We need to show nice features every demo.
No priority for security”

“We do security in separate sprints”

“We have a wiki with 400 rules and we’ll see
results at the next pentest in 6 months”

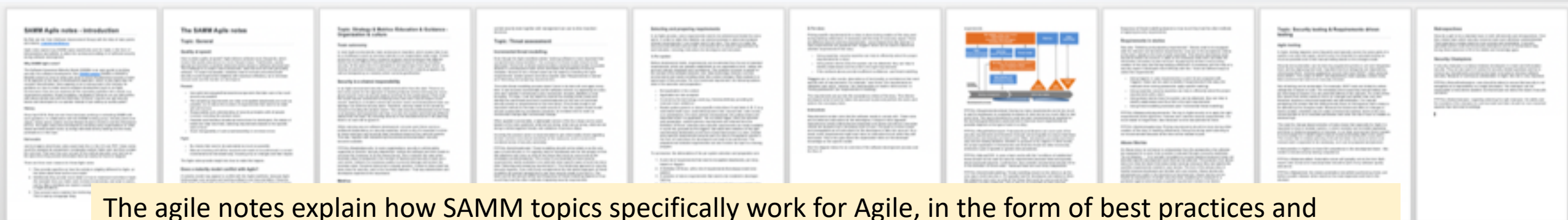
“We run this tool every sprint and voilà:
we’re secure!”

“We made John responsible for security, so now it’s no longer my problem”



SAMM Agile notes

- Secure agile development best practices and pitfalls
- Extension of SAMM 2.0
- Gathered from clients, peers and literature
- By yours truly, in collaboration with the SAMM working group, and industry peers, notably Michael Kuipers (Centric)



The agile notes explain how SAMM topics specifically work for Agile, in the form of best practices and pitfalls. Since April 2018, I have been working on this project, with industry friends, by gathering experiences, investigating organisations, and studying publications.

SAMM agile notes – main principles

Quality at speed!

- Continuous **collaboration** dev-test-ops-security and **collective ownership**
- **Automate** verification – as much as possible
- **Incrementalize**: Base all manual work on change
- **Minimize repeated expert work**
because it's scarce, it takes time and even experts make mistakes:
 - Defer it to prepared **components and frameworks**
 - Build it into prepared **tests**
 - Build it into a minimal set of prepared **instructions**
 - Make instructions **situational** : prefer hardening/hygiene over analysis

There is no time in a sprint to get the work done and then send it to security for testing. You want to shift left. This puts security professionals much more in an advisory and supportive role.

The idea is not to ditch security expertise, but to put less pressure on it and depend on it less.

Prepared instructions and tests are to make sure that developers and testers don't need babysitting.

Hygiene vs analysis

Hygiene



- Trigger: Authentication? Login form?
- Prepared requirements & tests:
 - ..
 - Reset password should take a similar time for known users and unknown users
 - **Threat:** people can find out if an individual has an account which can be private info
 - **Test:** automate logins, time and compare

Analysis



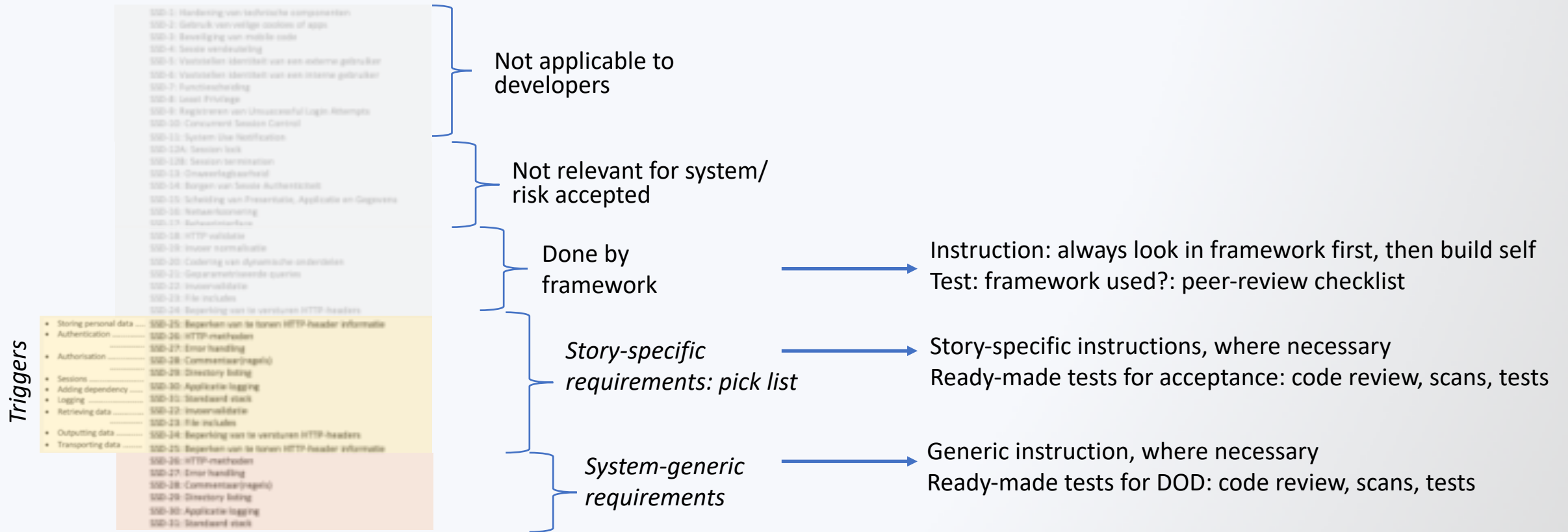
- *What are we doing?* Building a login form
- *What can go wrong?*
 - SQL injection
 - Eavesdropping of password
 - Reset password message discloses known user y/n
- *Wat do we do about it?*
 - ..

If you are having dinner you first wash your hands. Such situational instructions are there to save time in thinking about what you need to do, and they also prevent things to be forgotten.

Analysis - threat modelling in this case - is a useful approach, but only if all else fails.

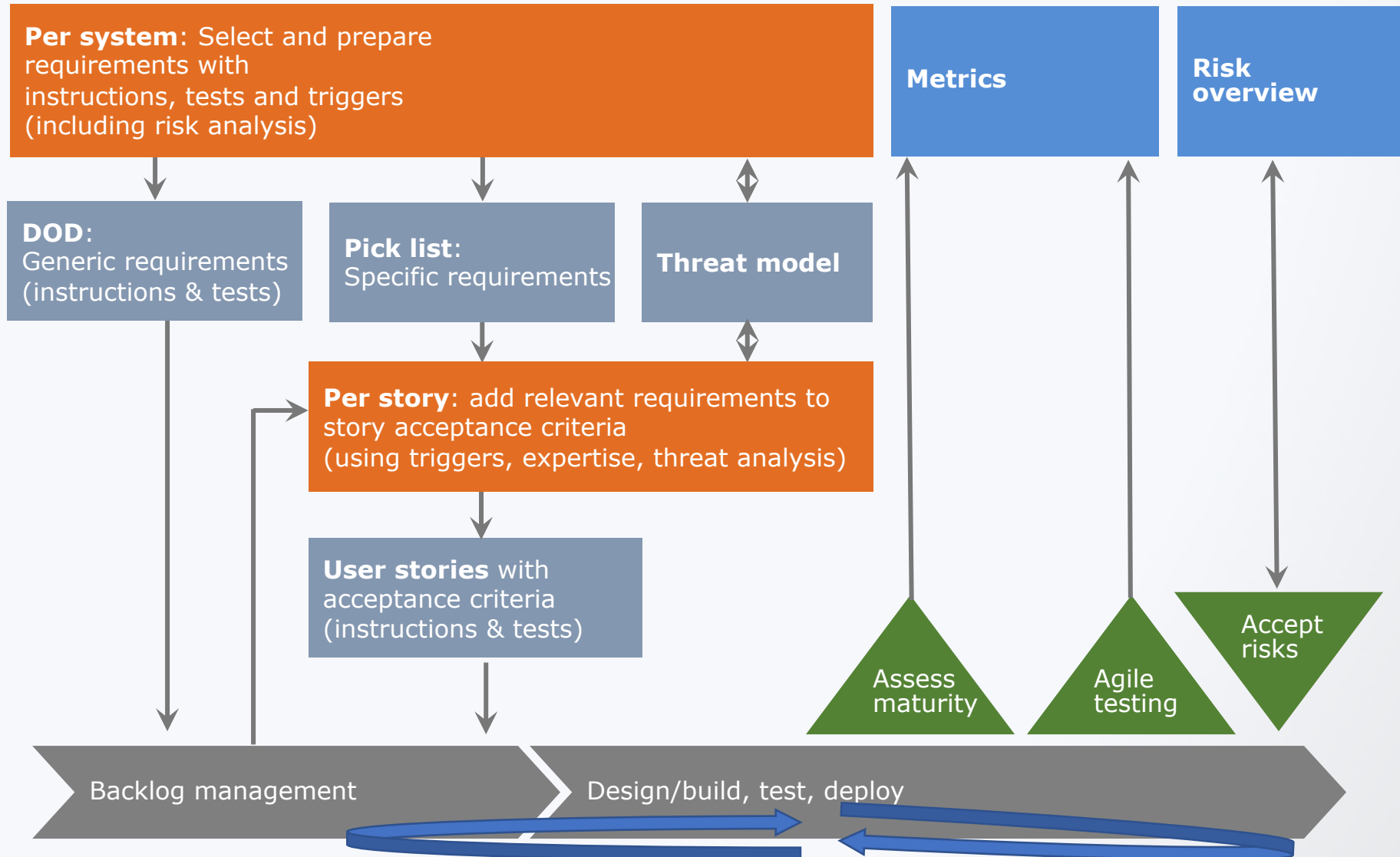
Example: requirements flow

Per system: Select and prepare requirements with instructions, tests and triggers (including risk analysis)



The key is to minimize the daily instructions that developers need to work with, and to let as many as possible depend on the type of task they are working on, for them to be effective.

Example: requirements flow



Adding the right instructions and tests to stories is done during creation of the story and during refinement – not during the plenary planning meeting because that's already tight for actual planning.

References

- <https://owaspsamm.org/head/agile-guidance/agilenotes/>

(will be mixed into SAMM topics, like other horizontal aspects)

- <https://owaspsamm.org/>
- r.vanderveer@sig.eu

We would really welcome your feedback on the Agile notes and learn from your experiences.
One final word: see if you can make your software security program a software quality program: all the mechanisms are very similar for maintainability, performance, reliability and privacy. Good luck!