Agile and Security.
Oil and Water?

Secret Chipmunk

Ron Parker | @scmunk

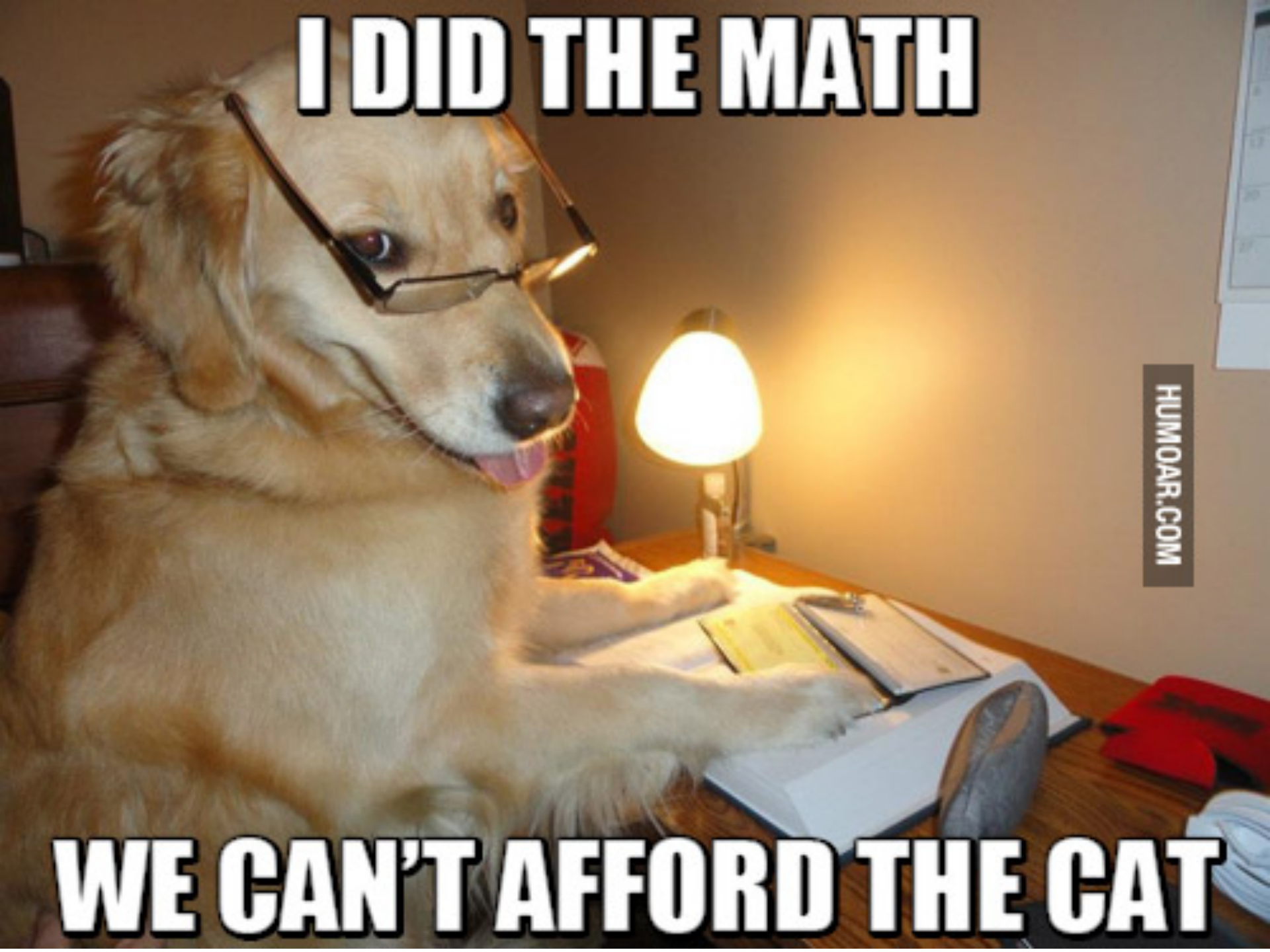
This is the condensed version of the presentation. We skip straight to the point and leave out the entertaining story of how **salad dressing** holds the key to agile security. Please read the notes as you go along.

If you would like to hear the whole story please visit YouTube and watch the video.

https://www.youtube.com/watch?v=TrUaYlAg888

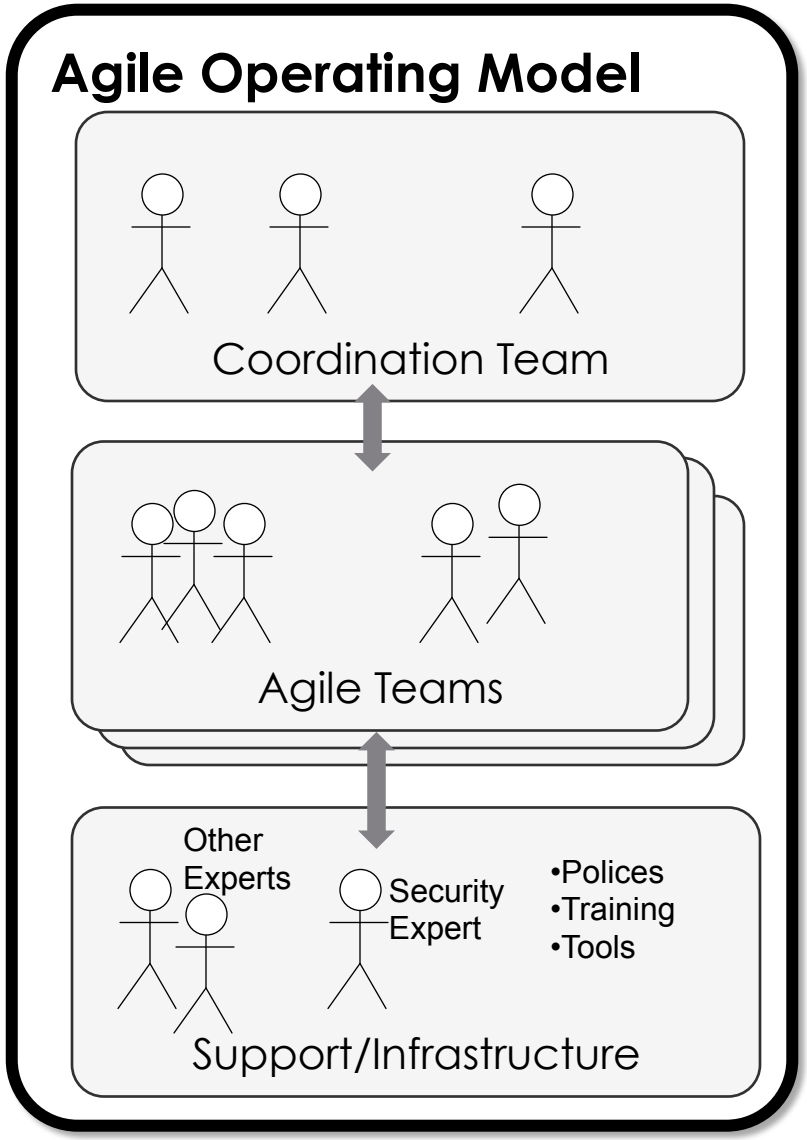
Thanks again to @IronGeek and @BsidesNash for the video production.

Some things just don't mix well.
Like Oil and Water.


I DID THE MATH
HUMOAR.COM
WE CAN'T AFFORD THE CAT

{ agile and security }

Agile Overview



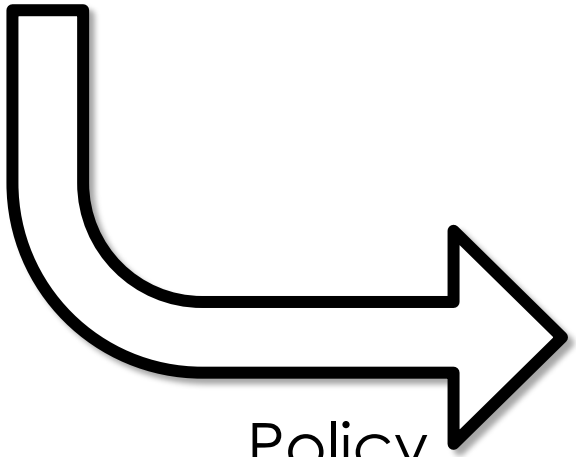
Attributes

- Accountable
- Self organizing
- Move at their own pace
- Use their own methods
- Wait on as little as possible
- Continuously improving
- Quality as a goal
- Delivery as a goal
- Customer focused

Security

High Level

- CIA+AA
- Privacy
- Compliance
- Brand



Policy
Standard
Guideline

Task Areas

- Identity management
- Access management
- Vulnerability management
- Secure coding
- Security testing
- Configuration management
- Disaster recovery
- Third party integration
- Mobile
- Cloud
- API security

Security

High Level

- CIA+AA
- Privacy
- Compliance
- Brand

Task Areas

- Identity management
- Access management
- Vulnerability management
- Secure coding
- Security testing
- Configuration management
- Disaster recovery
- Third party integration
- Mobile
- Cloud
- API security

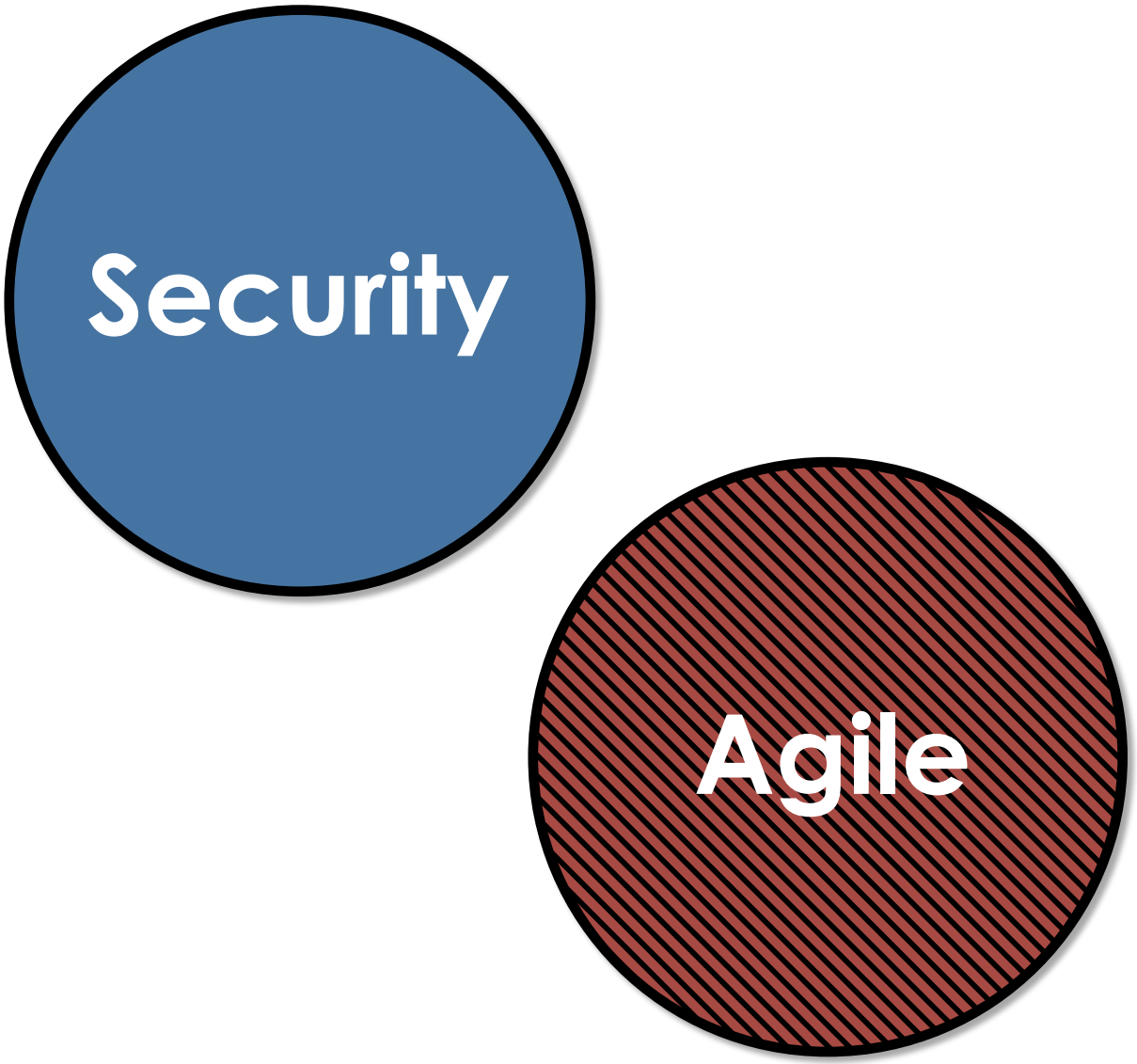
Policy
Standard
Guideline

Assurance

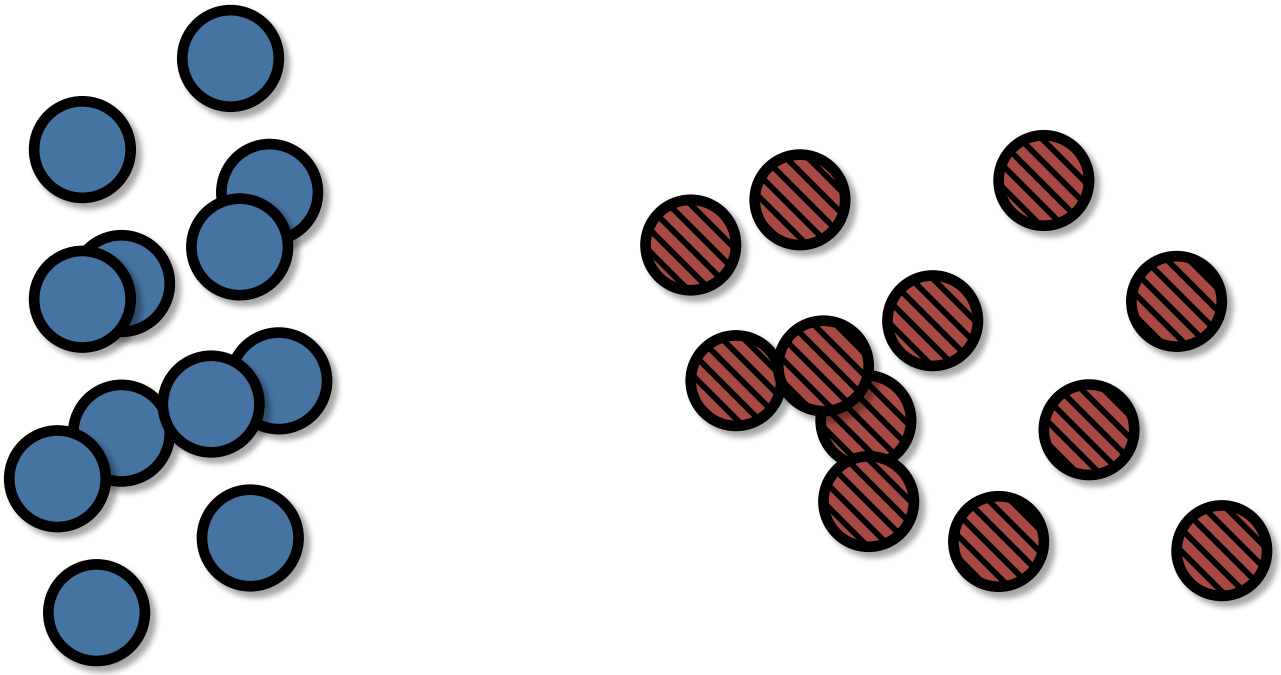
Believing and Knowing

Security needs to mix with the agile approach of the business.

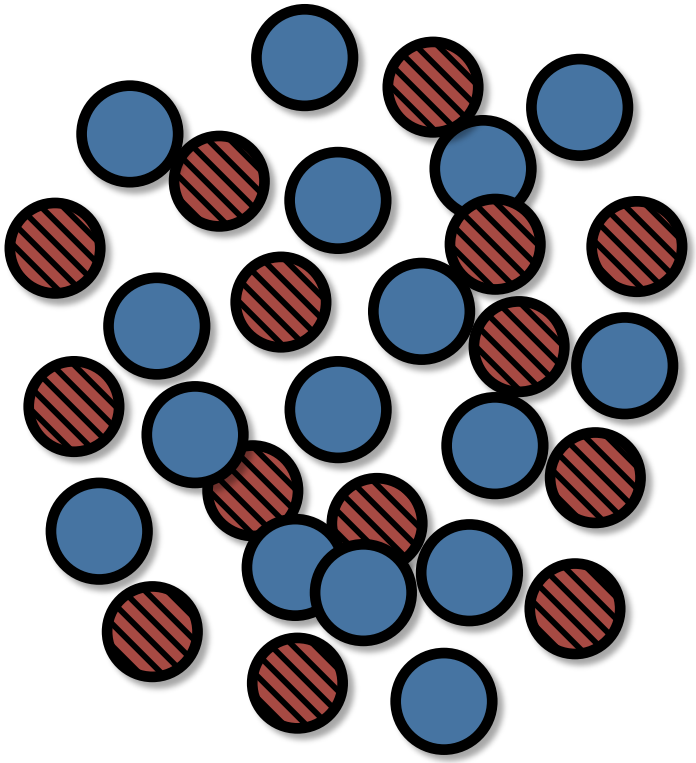
Mixing security and agile



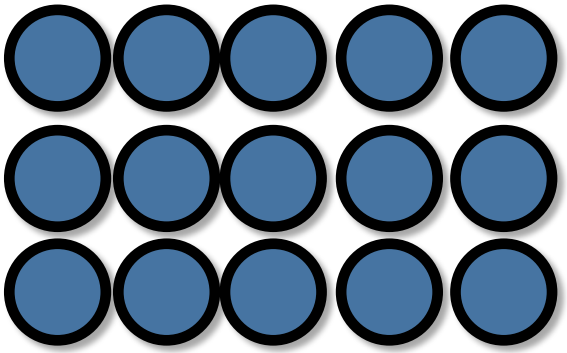
Apply effort

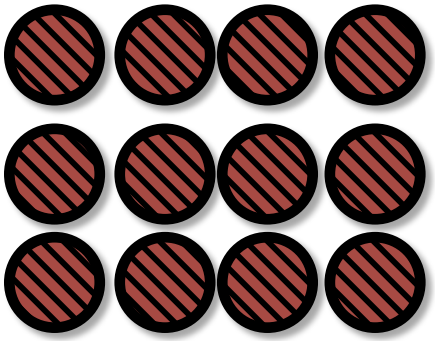


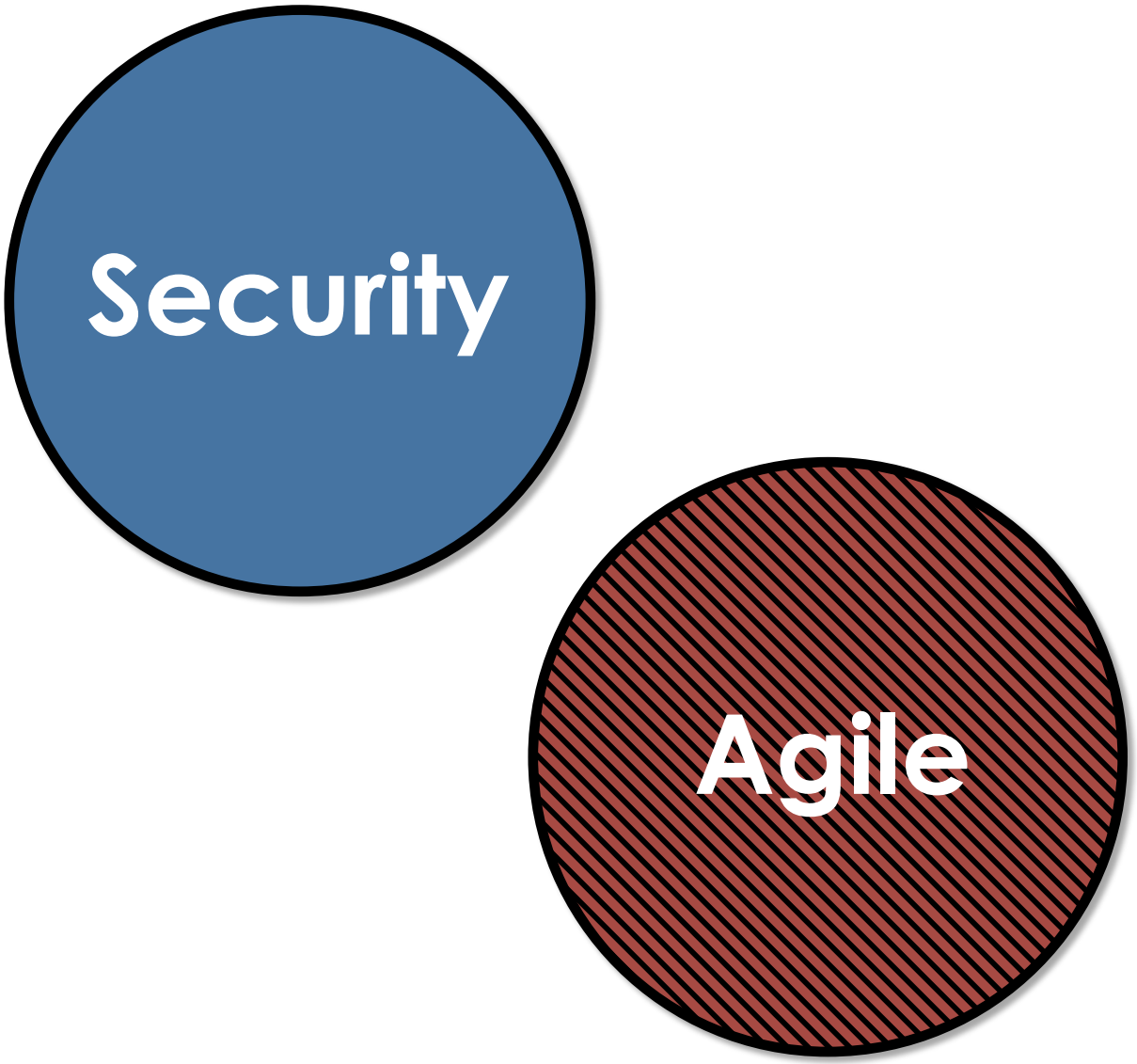
More effort



Stop and turn your back






Security
Agile

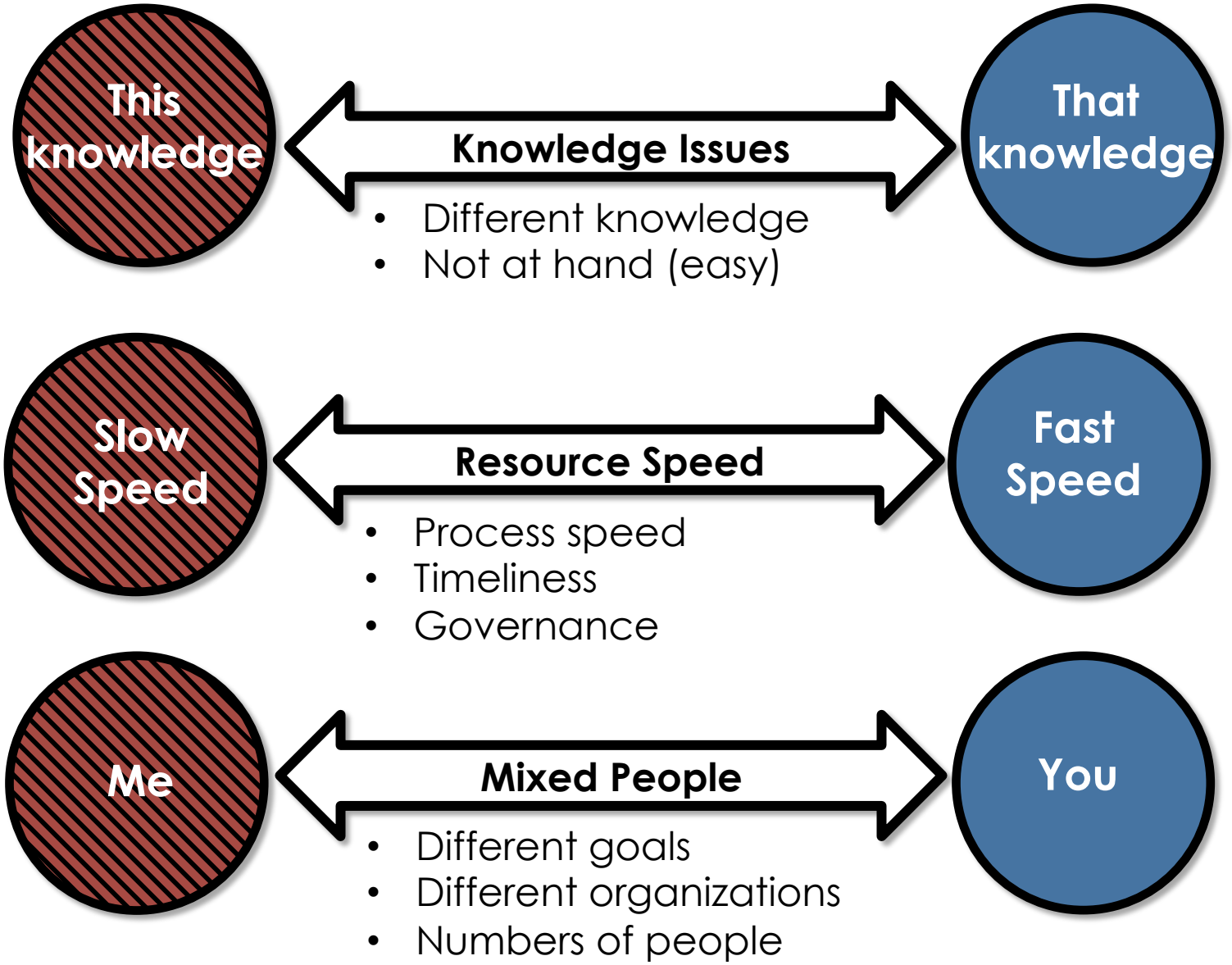

CODING COMPLETE...
38
SECURITY'S PROBLEM NOW.
memegenerator.net

Stuff that really happens

- Too many meetings with security driving them
- Development teams wait in frustration
- Insecure code gets released into the wild
- No positive or negative security testing
- Data security issues
- Data privacy issues
- Simple things are missed like the OWASP Top 10
- The same questions are repeatedly asked regarding security
- Compliance issues – customers don't like how you treat their data
- The same questions are repeatedly asked regarding security
- We the same bad things over and over

Security and agile – not a natural mix.

The problem


This knowledge
Knowledge Issues
That knowledge
• Different knowledge
• Not at hand (easy)
Slow Speed
Resource Speed
Fast Speed
• Process speed
• Timeliness
• Governance
Me
Mixed People
You
• Different goals
• Different organizations
• Numbers of people

Why do we want the mix?

We have **goals** beyond

just doing development
and
just doing basic security.

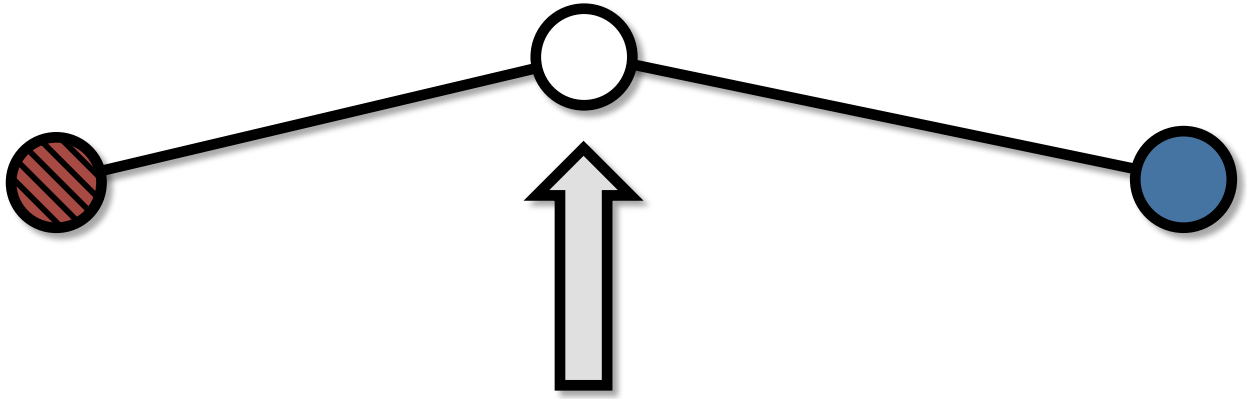
We need secure development.

What would these goals be?

ENABLE with knowledge and process

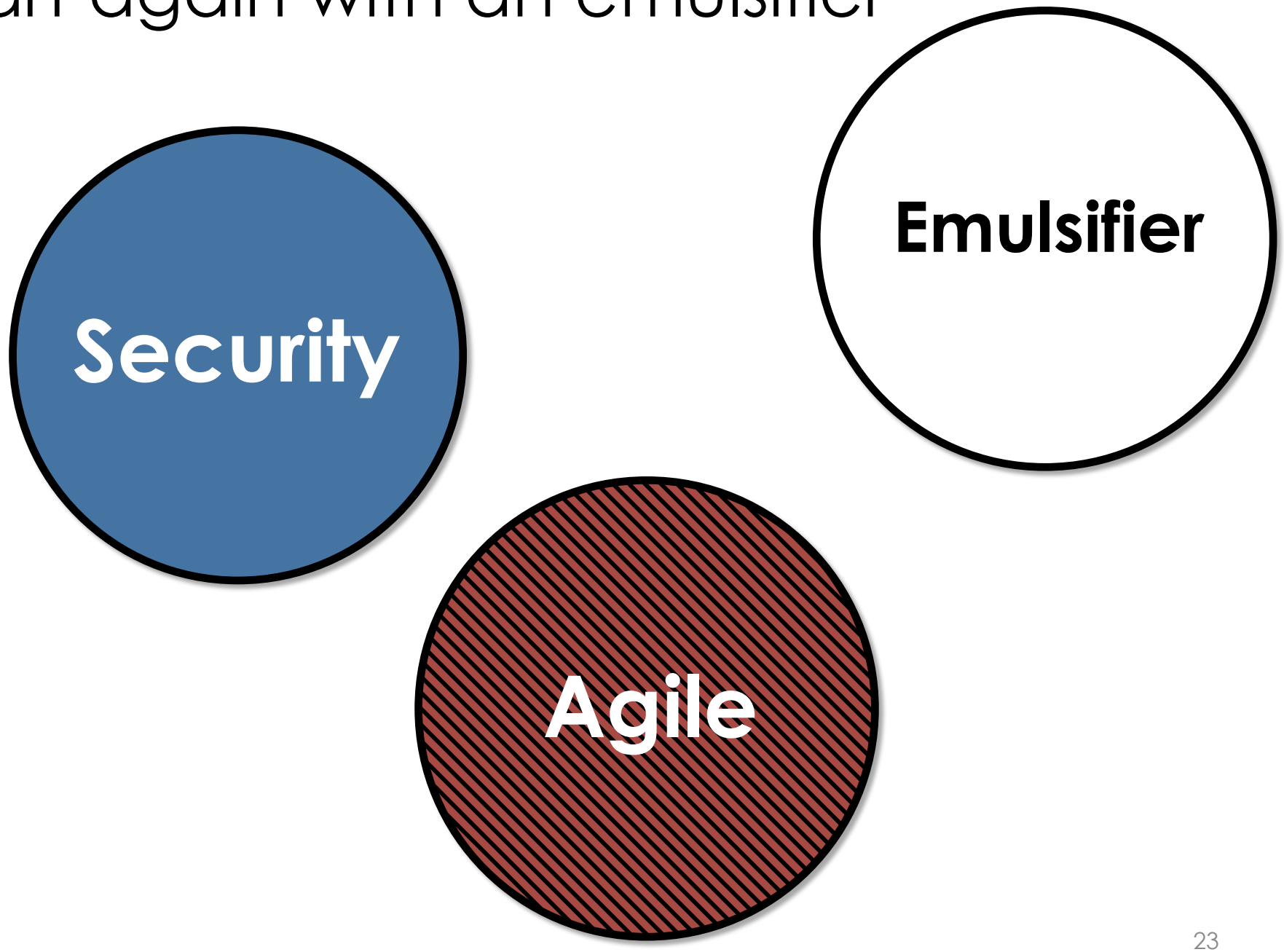
SCALE security with enabled people

ASSURE with enablement and scale

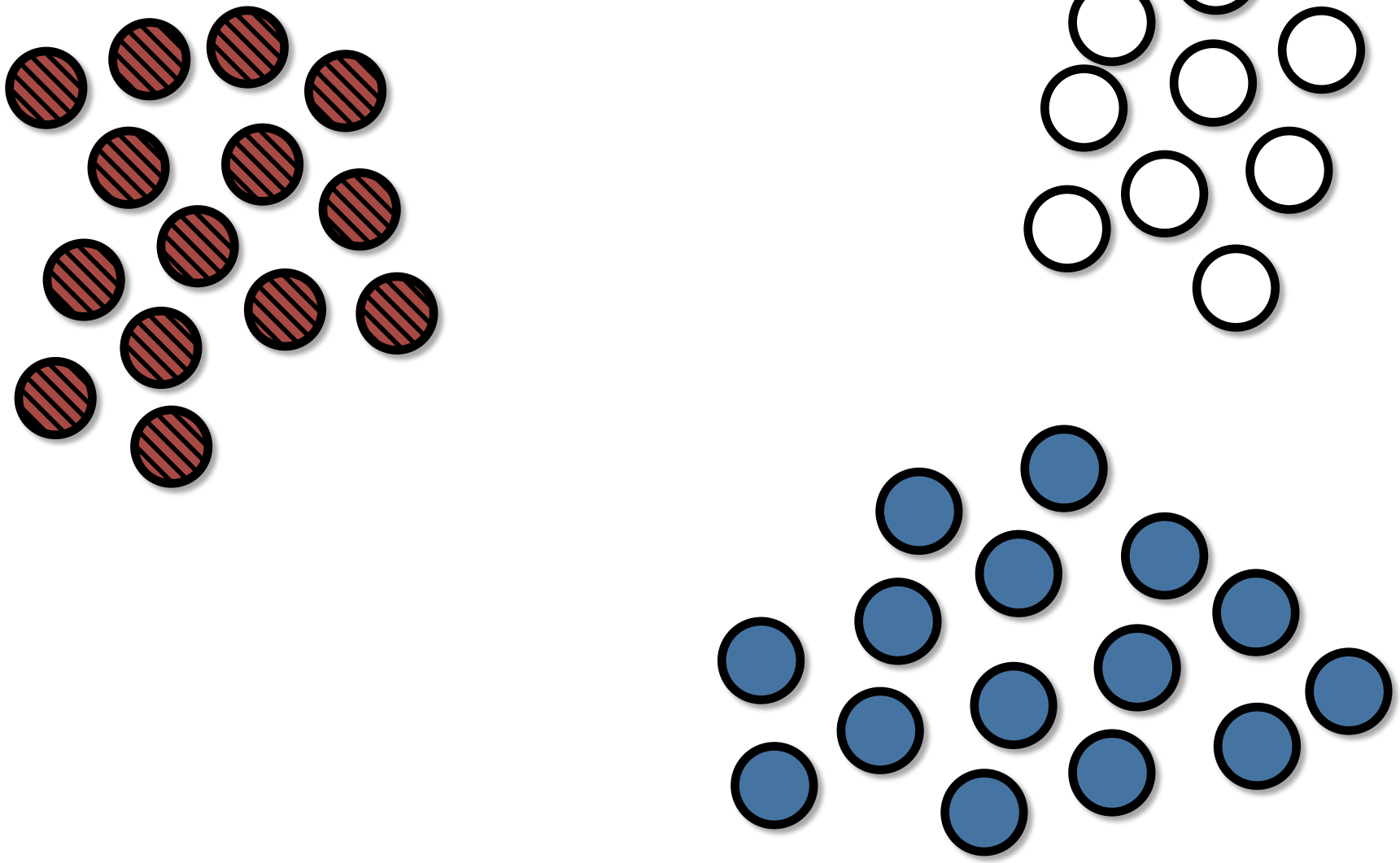


emulsifier

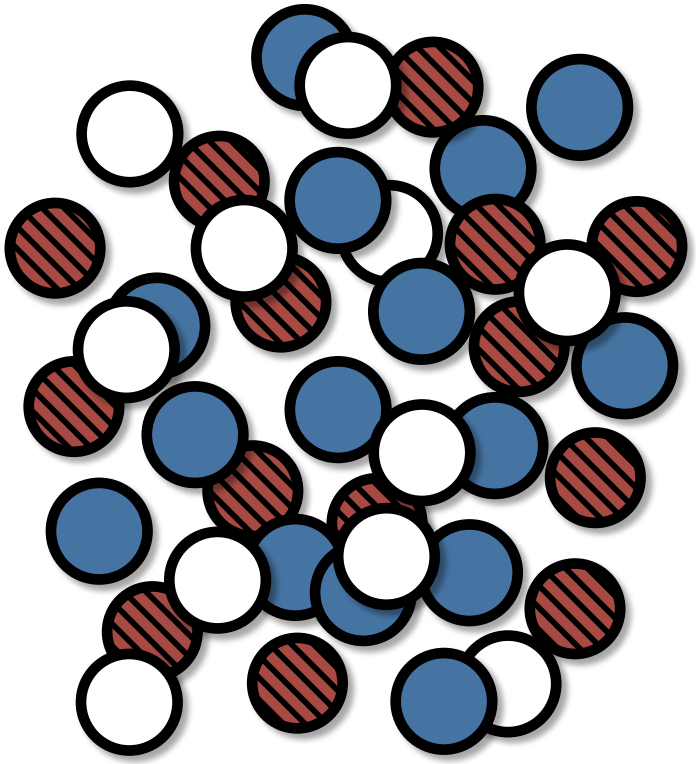
Start again with an emulsifier

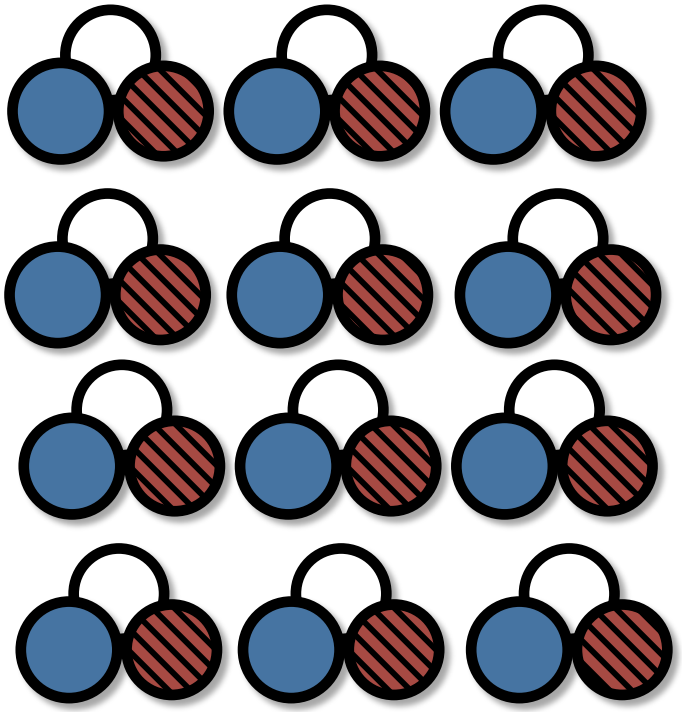


Mix



Science happens





What can bring Security and Agile together?

Security Development Lifecycle

Security Emulsifier

Wait, we have heard this before

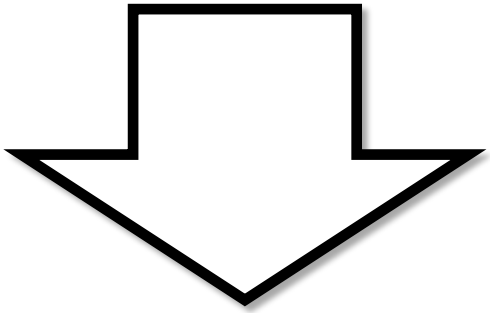
re · hash
Verb

put (old ideas or material) into a new form without significant change or improvement.

"he contented himself with occasional articles in journals, rehashing his own work“ (not that any infosec analyst would do that)

Solution

Security Development Lifecycle



Enable

- Work themselves
- Find it themselves
- Know it themselves
- At their pace

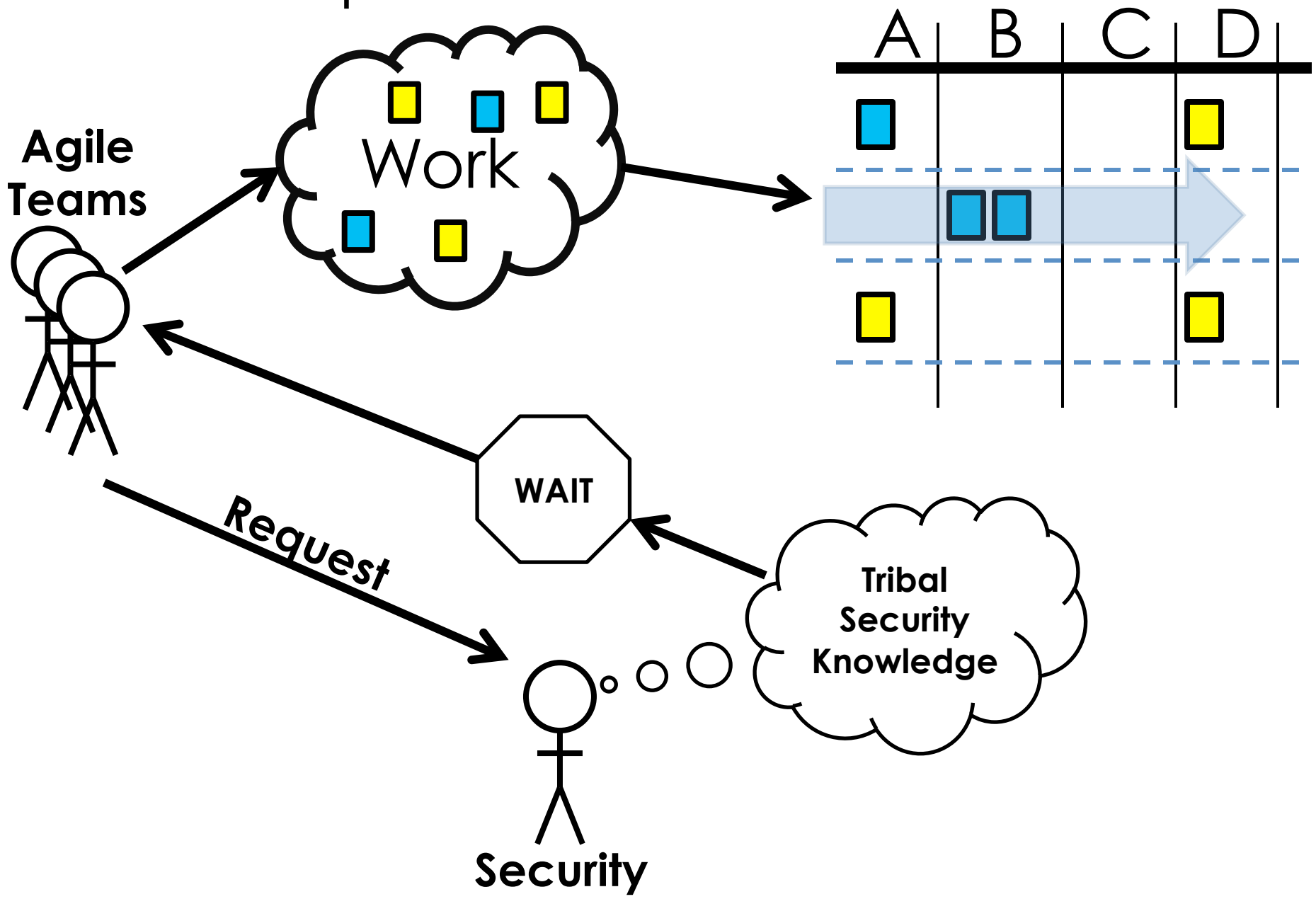
Scale

- Knowledge driven
- Process driven
- Not Security People driven

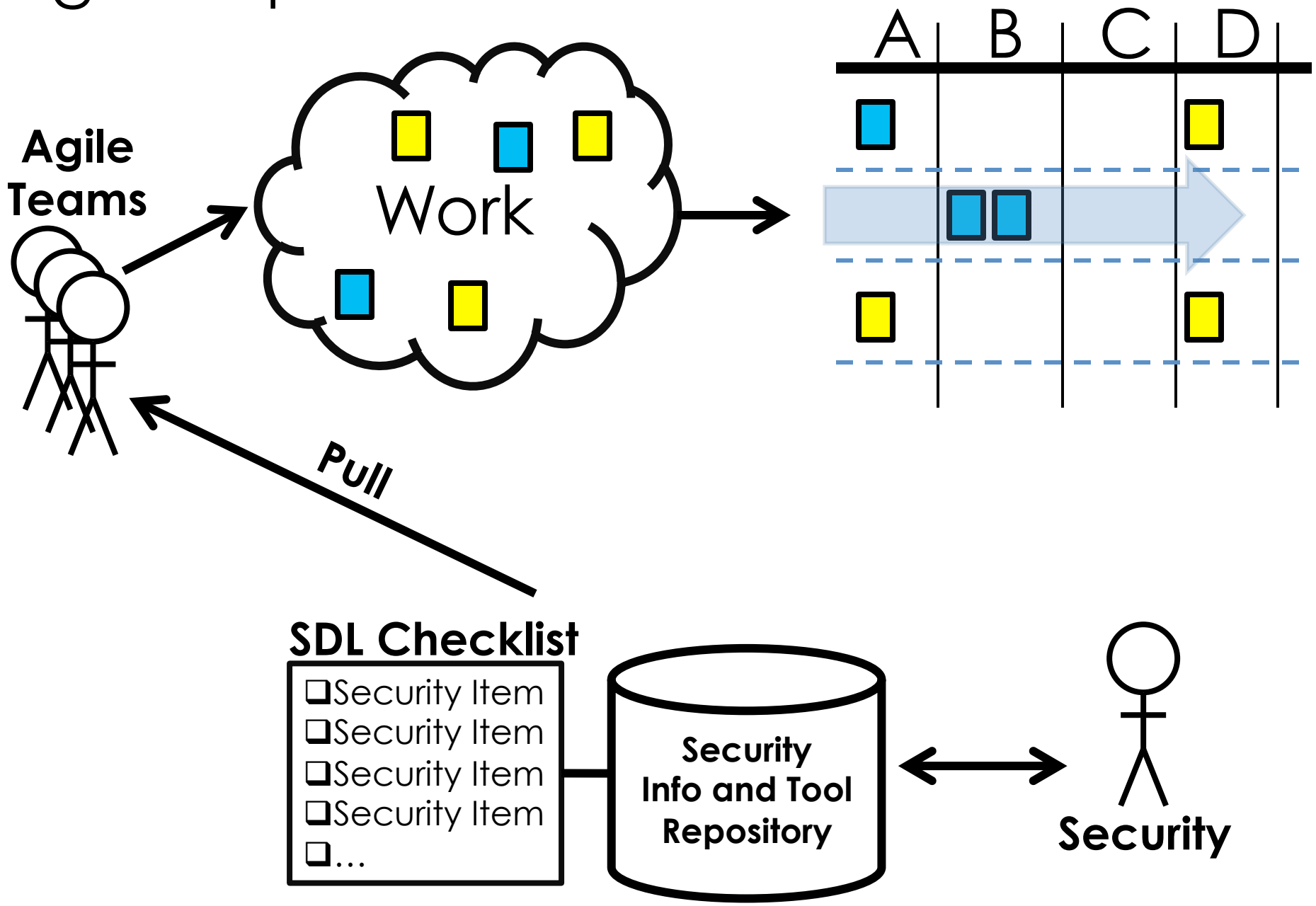
Assure

- Remember what is important
- Check during and after
- Framework for familiarity

Normal Implementation



Agile Implementation

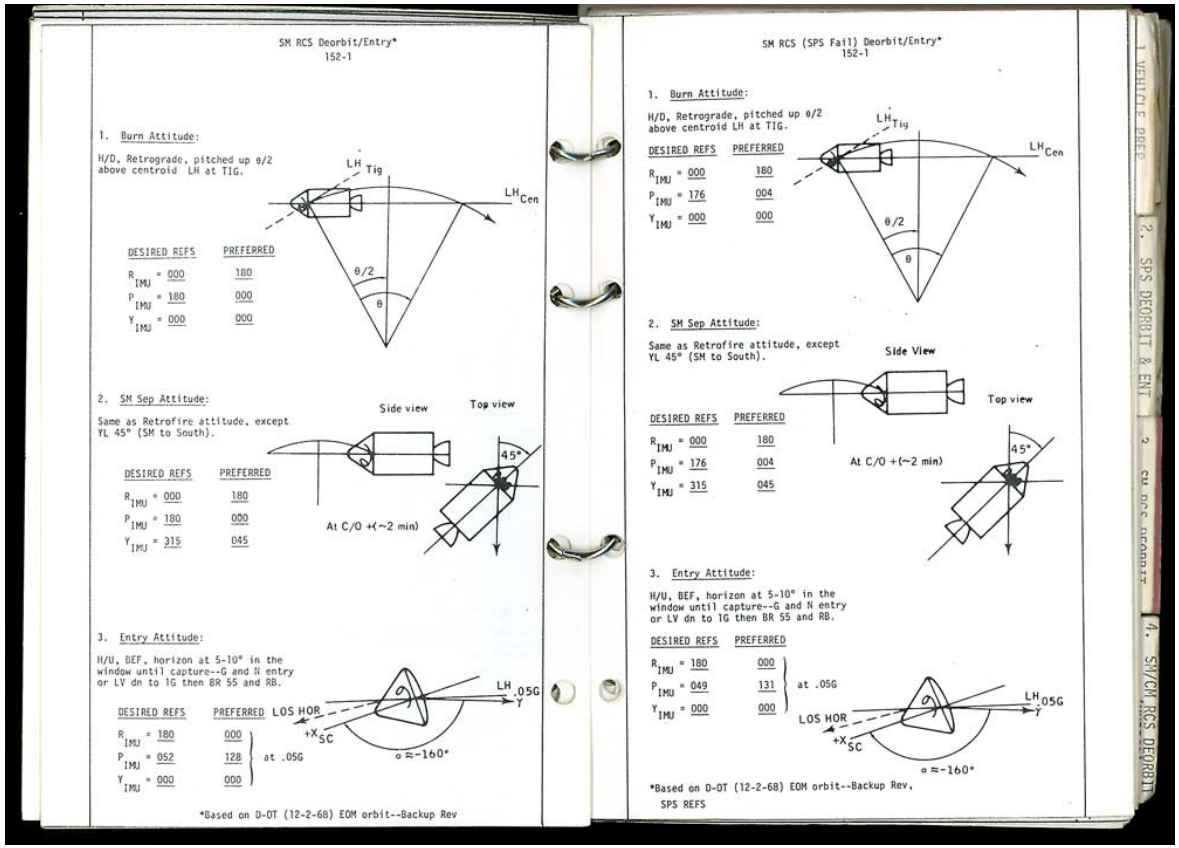


The SDL is our **Security Emulsifier**.

So how do we get one?

Timeout for Checklists

- Not checklist management
- Used in complex or high pressure situations
- Doctors, astronauts, pilots - ScrumMasters



SM RCS Deorbit/Entry*
152-1

1. Burn Attitude:

H/D, Retrograde, pitched up θ/2 above centroid LH at TIG.

DESIRED REFS	PREFERRED
R_{IMU} = 000	180
P_{IMU} = 180	000
Y_{IMU} = 000	000

2. SM Sep Attitude:

Same as Retrofire attitude, except YL 45° (SM to South).

DESIRED REFS	PREFERRED
R_{IMU} = 000	180
P_{IMU} = 180	000
Y_{IMU} = 315	045

3. Entry Attitude:

H/U, BEF, horizon at 5-10° in the window until capture--G and N entry or LV dn to 1G then BR 55 and RB.

DESIRED REFS	PREFERRED
R_{IMU} = 180	000
P_{IMU} = 052	128
Y_{IMU} = 000	000

at .05G

*Based on D-OT (12-2-68) EOM orbit--Backup Rev

SM RCS (SPS Fail) Deorbit/Entry*
152-1

1. Burn Attitude:

H/D, Retrograde, pitched up θ/2 above centroid LH at TIG.

DESIRED REFS	PREFERRED
R_{IMU} = 000	180
P_{IMU} = 176	004
Y_{IMU} = 000	000

2. SM Sep Attitude:

Same as Retrofire attitude, except YL 45° (SM to South).

DESIRED REFS	PREFERRED
R_{IMU} = 000	180
P_{IMU} = 176	004
Y_{IMU} = 315	045

3. Entry Attitude:

H/U, BEF, horizon at 5-10° in the window until capture--G and N entry or LV dn to 1G then BR 55 and RB.

DESIRED REFS	PREFERRED
R_{IMU} = 180	000
P_{IMU} = 049	131
Y_{IMU} = 000	000

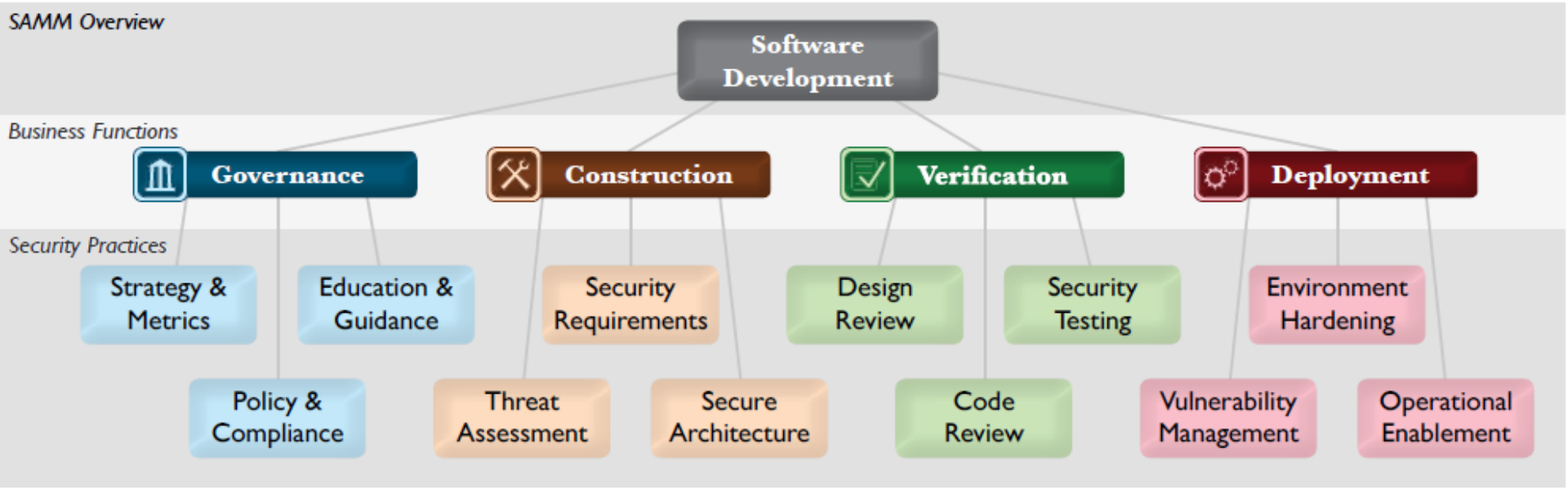
at .05G

*Based on D-OT (12-2-68) EOM orbit--Backup Rev.
SPS REFS

Acts as a reminder but also gives you sequence and may also give you alternate paths

OpenSAMM

Open Software Assurance Maturity Model



- Four major functions
- Each function has three practices
- Each practice has multiple activities

- Assess the organization's maturity
- Build a security program
- Define and measure security activities
- Built in metrics and measures

SAMM Functions

Governance

Governance is centered on the processes and activities related to how an organization manages overall software development activities. More specifically, this includes concerns that cross-cut groups involved in development as well as business processes that are established at the organization level.

...more on page 10

Construction

Construction concerns the processes and activities related to how an organization defines goals and creates software within development projects. In general, this will include product management, requirements gathering, high-level architecture specification, detailed design, and implementation.

...more on page 12

Verification

Verification is focused on the processes and activities related to how an organization checks and tests artifacts produced throughout software development. This typically includes quality assurance work such as testing, but it can also include other review and evaluation activities.

...more on page 14

Deployment

Deployment entails the processes and activities related to how an organization manages release of software that has been created. This can involve shipping products to end users, deploying products to internal or external hosts, and normal operations of software in the runtime environment.

...more on page 16

SAMM Practice

Security Requirements

The Security Requirements (SR) Practice is focused on proactively specifying the expected behavior of software with respect to security. Through addition of analysis activities at the project level, security requirements are initially gathered based on the high-level business purpose of the software. As an organization advances, more advanced techniques are used such as access control specifications to discover new security requirements that may not have been initially obvious to development.

In a sophisticated form, provision of this Practice also entails pushing the security requirements of the organization into its relationships with suppliers and then auditing projects to ensure all are adhering to expectations with regard to specification of security requirements.

Each Practice has Maturity Levels

	Security Requirements		
	SR 1	SR 2	SR 3
Objective	**Consider security explicitly during the software requirements process**	**Increase granularity of security requirements derived from business logic and known risks**	**Mandate security requirements process for all software projects and third-party dependencies**
Activities	A. Derive security requirements from business functionality B. Evaluate security and compliance guidance for requirements	A. Build an access control matrix for resources and capabilities B. Specify security requirements based on known risks	A. Build security requirements into supplier agreements B. Expand audit program for security requirements

...more on page 50

Activity – SR 1 A

Activities

A. Derive security requirements from business functionality

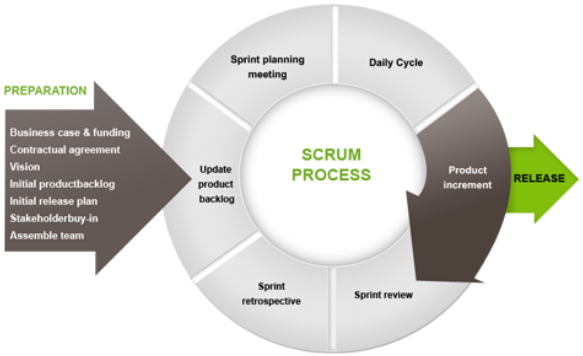
Conduct a review of functional requirements that specify the business logic and overall behavior for each software project. After gathering requirements for a project, conduct an assessment to derive relevant security requirements. Even if software is being built by a third-party, these requirements, once identified, should be included with functional requirements delivered to vendors.

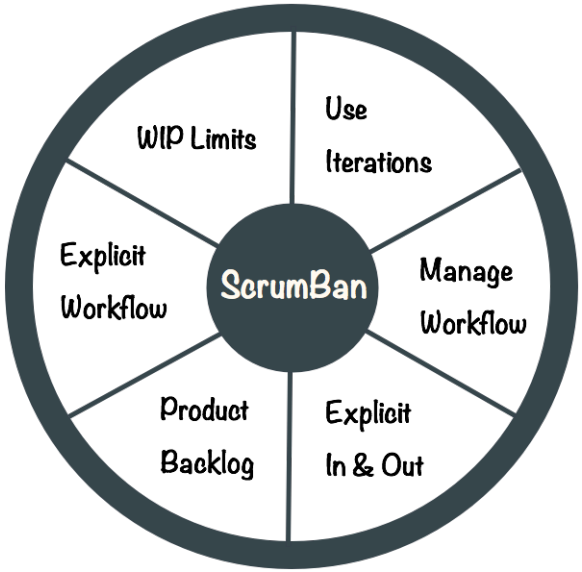
For each functional requirement, a security auditor should lead stakeholders through the process of explicitly noting any expectations with regard to security. Typically, questions to clarify for each requirement include expectations for data security, access control, transaction integrity, criticality of business function, separation of duties, uptime, etc.

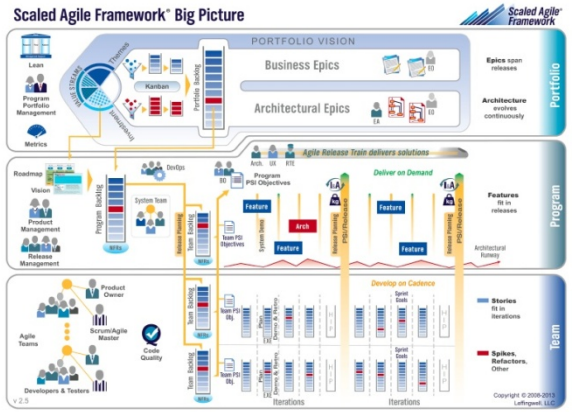
It is important to ensure that all security requirements follow the same principles for writing good requirements in general. Specifically, they should be specific, measurable, and reasonable.

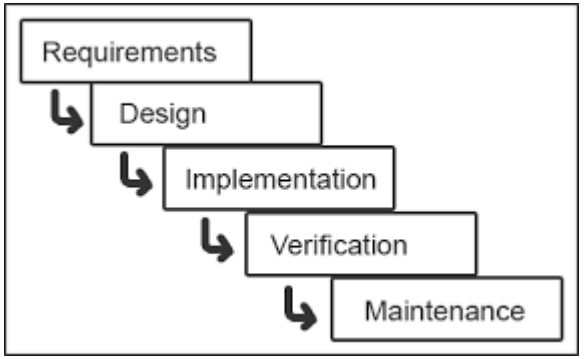
Conduct this process for all new requirements on active projects. For existing features, it is recommended to conduct the same process as a gap analysis to fuel future refactoring for security.

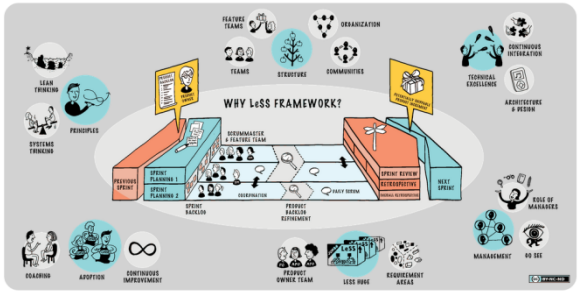
Methodologies may vary

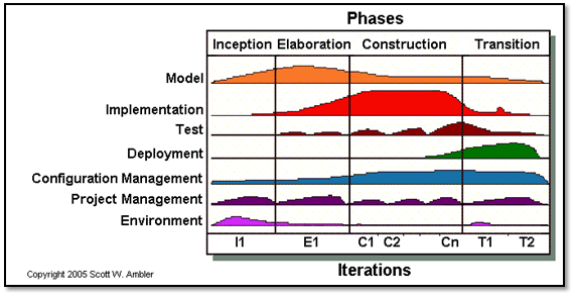




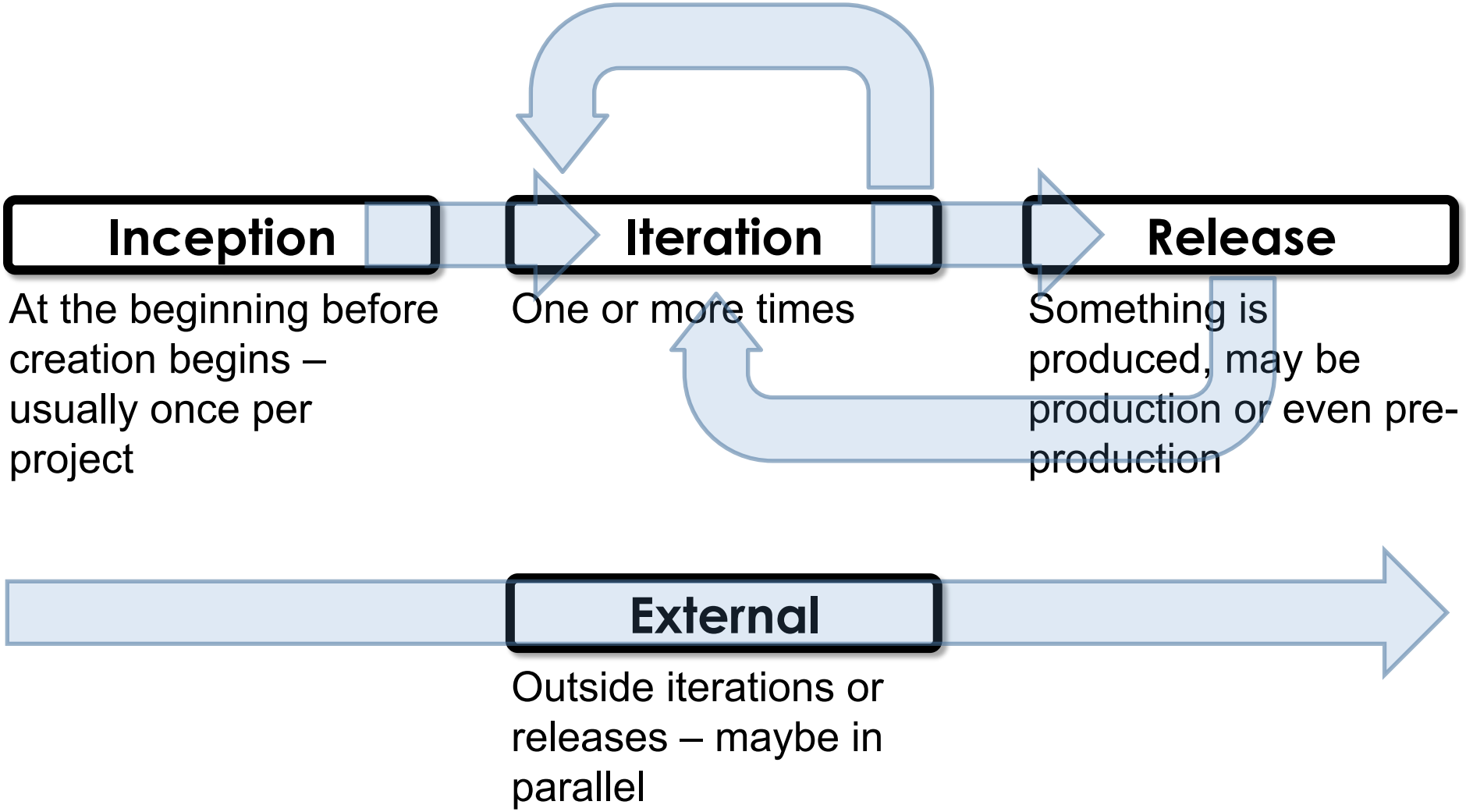








Phases



At the beginning before creation begins – usually once per project

One or more times

Something is produced, may be production or even pre-production

Outside iterations or releases – maybe in parallel

Activities

Inception

- Training
- Vuln management
- Security contacts
- Risk assessment

Iteration

- NFR for security
- Attacker profile
- Security stories
- Static code analysis
- Security architecture

Release

- Document deployment
- Document DR
- Penetration testing

External

- Vendor assessment
- 3rd-party service creation

Atypical SDL Activities

Many lifecycles are just concerned with the core security testing/coding. Real security is much larger than that.

- Role engineering
- Access Control requests
- Access Control models
- Integration of application level security
- Privacy requirements
- Customer requirements
- Disaster Recovery design and documentation
- Security Poker
- Security Stories
- Security Retrospective
- Security Spikes

Security Owner

- Is a single person with a role
- The contact for security concerns
- The SDL subject matter expert for the team
- Keeper of the List
- Is accountable for security activities and artifacts

Wiki – You Got What I Need

You Need

- Flexible but smart pages
- Knowledgebase
- Easy access and update
- Searchable
- Expandable repository

Wiki's Got

- Crowdsourcing driven
- As many pages as you need
- Document as you go
- Easy formatting
- History/version for each page
- Categories and tags
- Easy relationship linking

The Real Implementation List

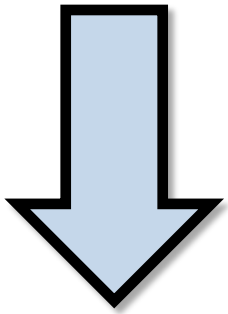
- Involve stakeholders (developers, product owners, compliance)
- Read OpenSAMM materials
- Determine your current assurance maturity level
- Determine your target level
- Determine actions needed to reach your target
- Work with your audit/controls department on what they may need
- Assign an owner for each action
- Build your basic Wiki with your primary actions
- Build a list of current tools
- Build a list of current standards and guidelines
- Begin a list of missing tools, standards and guidelines
- Use the Wiki to house knowledgebase items
- Educate your Security Owners (initial ones)
- Educate your general population
- Run a pilot project where a security member is embedded on the team
- Use feedback from pilot to streamline actions
- Metric, track number of projects using the SDL
- Assess through feedback whether the actions are necessary and correct
- Add new actions that move you towards your assurance target
- Rinse and repeat

Goals for Agile and Security

ENABLE with knowledge and process

SCALE security with enabled people

ASSURE with enablement and scale



Security Development Lifecycle

OpenSDL

The OpenSDL is an Open Security Development Lifecycle that can be used an example or as a starting point to build out a modern SDL. It is based on OWASP work and sits on top of a community driven platform.

Please take a look. The home site has a more detailed explanation.

http://www.opensdl.com

Agile and Security. Oil and Water?

Thanks!

Secret Chipmunk

Ron Parker
@scmunk
http://www.secretchipmunk.com