Release Management

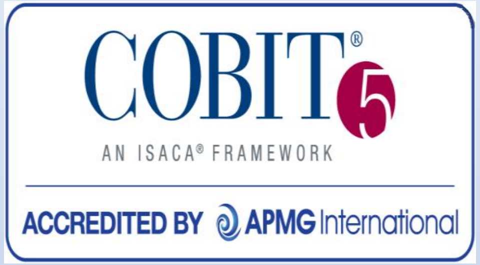
About Me

Scrum Alliance
CSM
CERTIFIED

Scrum Alliance
CSPO
CERTIFIED

Scrum Alliance
CSD
CERTIFIED

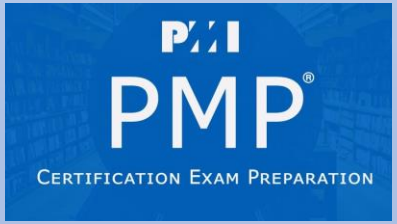
Scrum Alliance
CSP
CERTIFIED



INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR
भारतीय प्रौद्योगिकी संस्थान खड़गपुर
1951
योगः कर्मसु कौशलम्

ISO
ISO/IEC 27001
ISO/IEC 27001
CERTIFIED
PECB

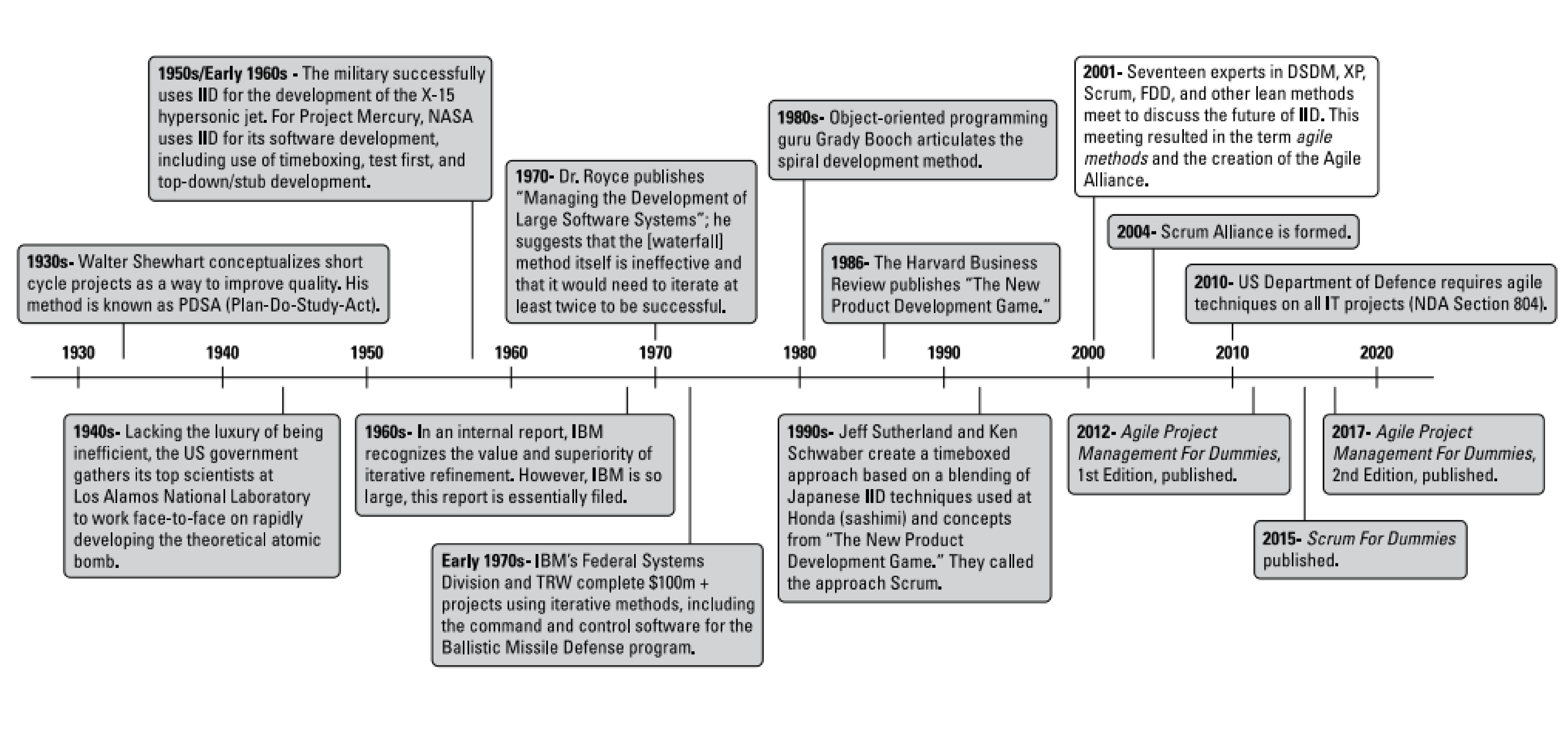
PRINCE2®
TRAINING ORGANIZATION ACCREDITED BY
PEOPLECERT ON BEHALF OF AXELOS



ITIL®
TRAINING ORGANIZATION ACCREDITED BY
PEOPLECERT ON BEHALF OF AXELOS

Mohammed Syful Islam

Agile History



1930s- Walter Shewhart conceptualizes short cycle projects as a way to improve quality. His method is known as PDSA (Plan-Do-Study-Act).

1940s- Lacking the luxury of being inefficient, the US government gathers its top scientists at Los Alamos National Laboratory to work face-to-face on rapidly developing the theoretical atomic bomb.

1950s/Early 1960s - The military successfully uses IID for the development of the X-15 hypersonic jet. For Project Mercury, NASA uses IID for its software development, including use of timeboxing, test first, and top-down/stub development.

1960s- In an internal report, IBM recognizes the value and superiority of iterative refinement. However, IBM is so large, this report is essentially filed.

1970- Dr. Royce publishes "Managing the Development of Large Software Systems"; he suggests that the [waterfall] method itself is ineffective and that it would need to iterate at least twice to be successful.

Early 1970s- IBM's Federal Systems Division and TRW complete $100m + projects using iterative methods, including the command and control software for the Ballistic Missile Defense program.

1980s- Object-oriented programming guru Grady Booch articulates the spiral development method.

1986- The Harvard Business Review publishes "The New Product Development Game."

1990s- Jeff Sutherland and Ken Schwaber create a timeboxed approach based on a blending of Japanese IID techniques used at Honda (sashimi) and concepts from "The New Product Development Game." They called the approach Scrum.

2001- Seventeen experts in DSDM, XP, Scrum, FDD, and other lean methods meet to discuss the future of IID. This meeting resulted in the term *agile methods* and the creation of the Agile Alliance.

2004- Scrum Alliance is formed.

2010- US Department of Defence requires agile techniques on all IT projects (NDA Section 804).

2012- *Agile Project Management For Dummies*, 1st Edition, published.

2015- *Scrum For Dummies* published.

2017- *Agile Project Management For Dummies*, 2nd Edition, published.

Agile Methodologies and Frameworks

1970s
Structured programming, 1969
Cap Gemini SDM, 1974 (PANDATA)

1980s
Structured systems analysis and design method (SSADM), 1980
Information Requirement Analysis/Soft systems methodology

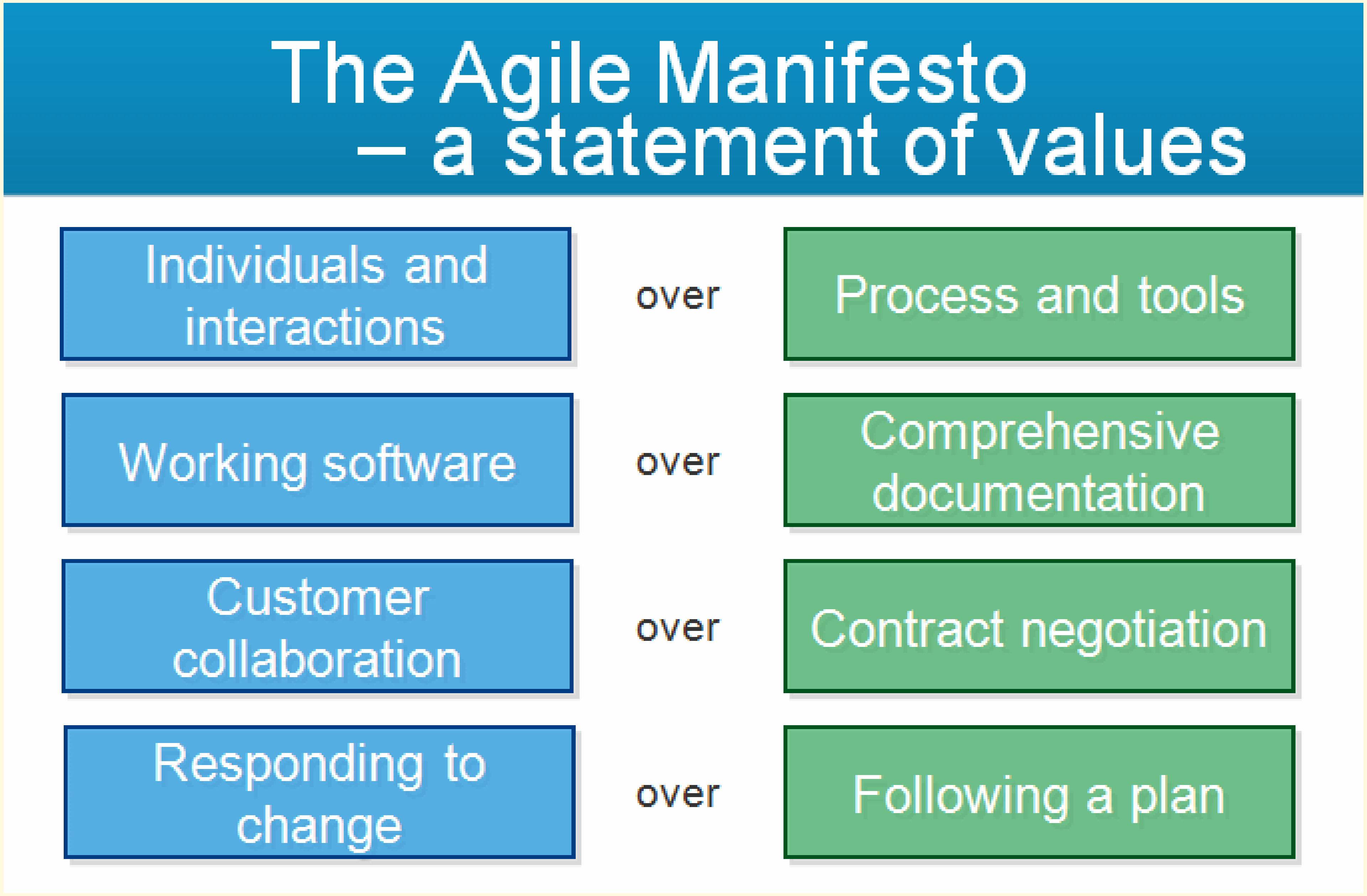
1990s
Object-oriented programming (OOP), 1960s/1990s
Rapid application development (RAD), 1991
Dynamic systems development method (DSDM), 1994
Scrum, 1995
Team software process, 1998
Rational Unified Process (RUP), 1998
Extreme programming, 1999

2000s
Agile Unified Process (AUP), 2005
Disciplined agile delivery (DAD)

2010s
Scaled Agile Framework (SAFe)
Large-Scale Scrum (LeSS)

SCRUM	XP	Battlefield Agility	Kanban
DAD	Agile Modelling	AUP	Krystal
DSDM	SAFe	Nexus	LeSS

Agile Values



The Agile Manifesto – a statement of values

Individuals and interactions over Process and tools

Working software over Comprehensive documentation

Customer collaboration over Contract negotiation

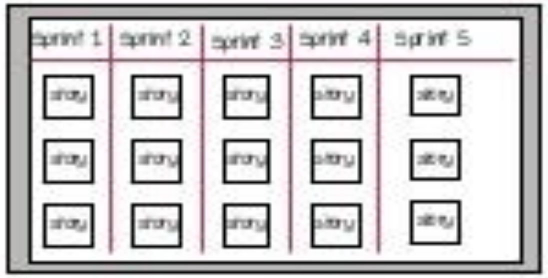
Responding to change over Following a plan

The 12 agile principles*

1 Satisfy the **customer**

2 Welcome **change**

3 Deliver **frequently**



4 Work **together**

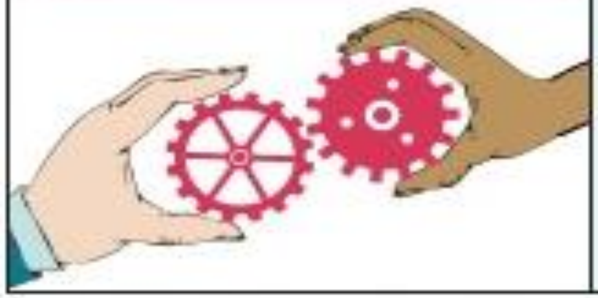
5 Trust and **support**

6 Face-to-face **conversation**

7 Working **software**

8 Sustainable **development**

9 Continuous **attention**

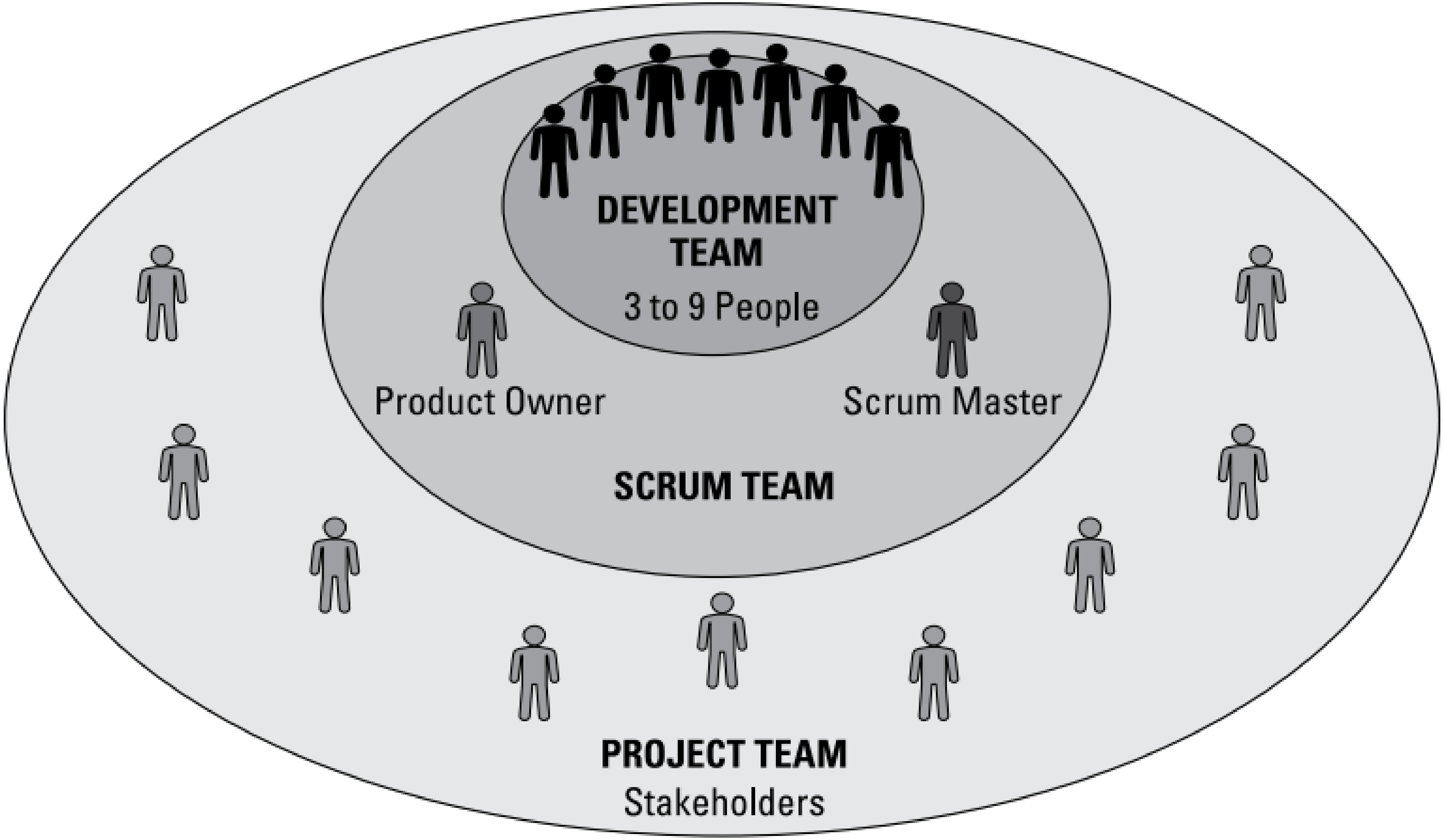


10 Maintain **simplicity**

11 Self-organizing **teams**

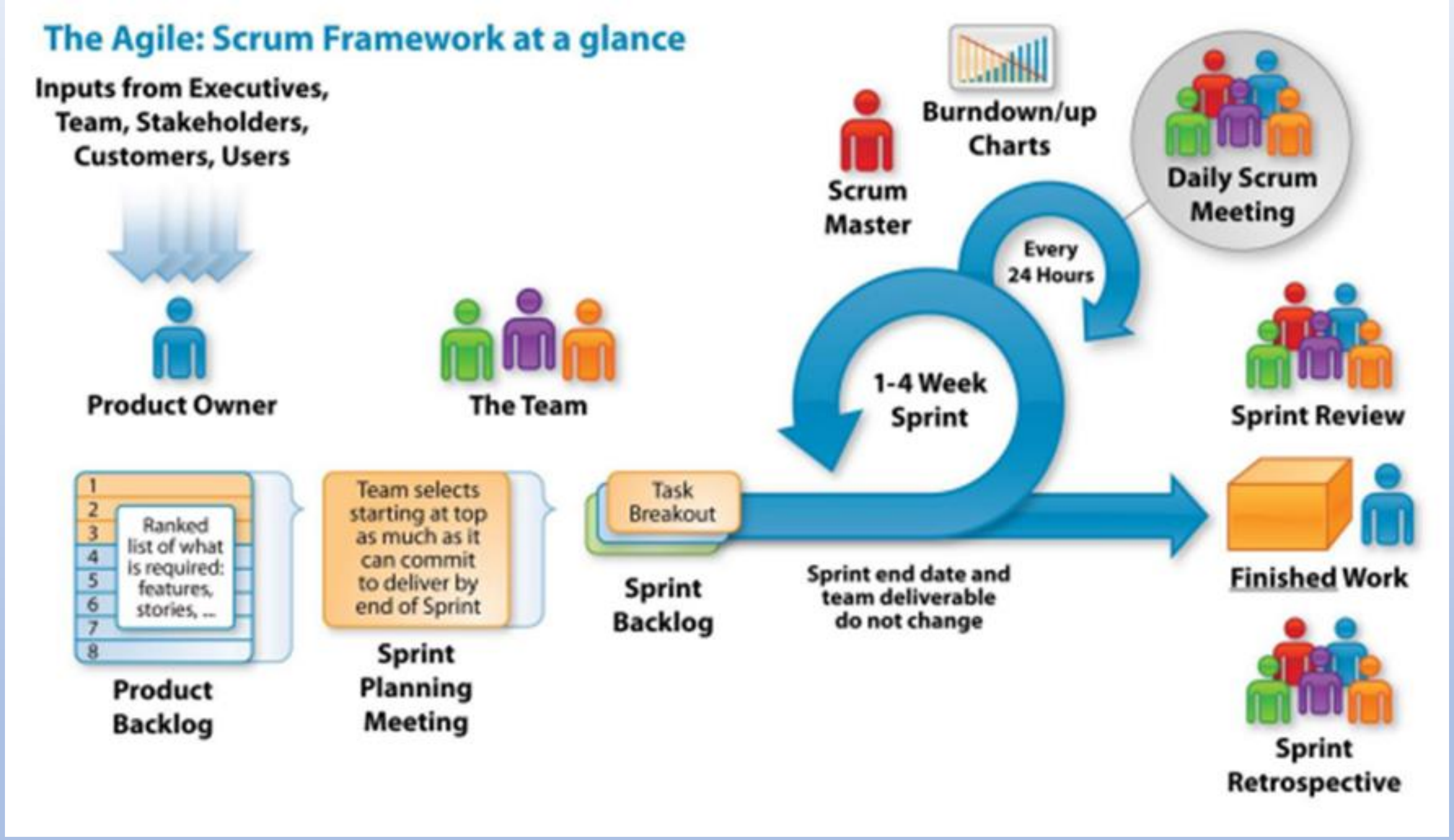
12 Reflect and **adjust**

Agile Team



Agile project team, scrum team, and development team.

Agile Process



The Agile: Scrum Framework at a glance

Inputs from Executives, Team, Stakeholders, Customers, Users

Product Owner

The Team

Scrum Master

Burndown/up Charts

Daily Scrum Meeting

Every 24 Hours

1-4 Week Sprint

1 2 3 4 5 6 7 8 Ranked list of what is required: features, stories, ...

Product Backlog

Team selects starting at top as much as it can commit to deliver by end of Sprint

Sprint Planning Meeting

Task Breakout

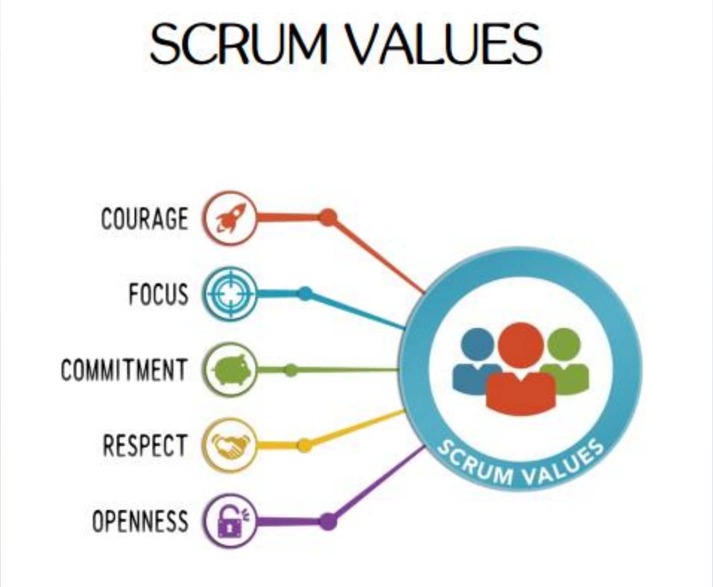
Sprint Backlog

Sprint end date and team deliverable do not change

Sprint Review

Finished Work

Sprint Retrospective



SCRUM VALUES

COURAGE

FOCUS

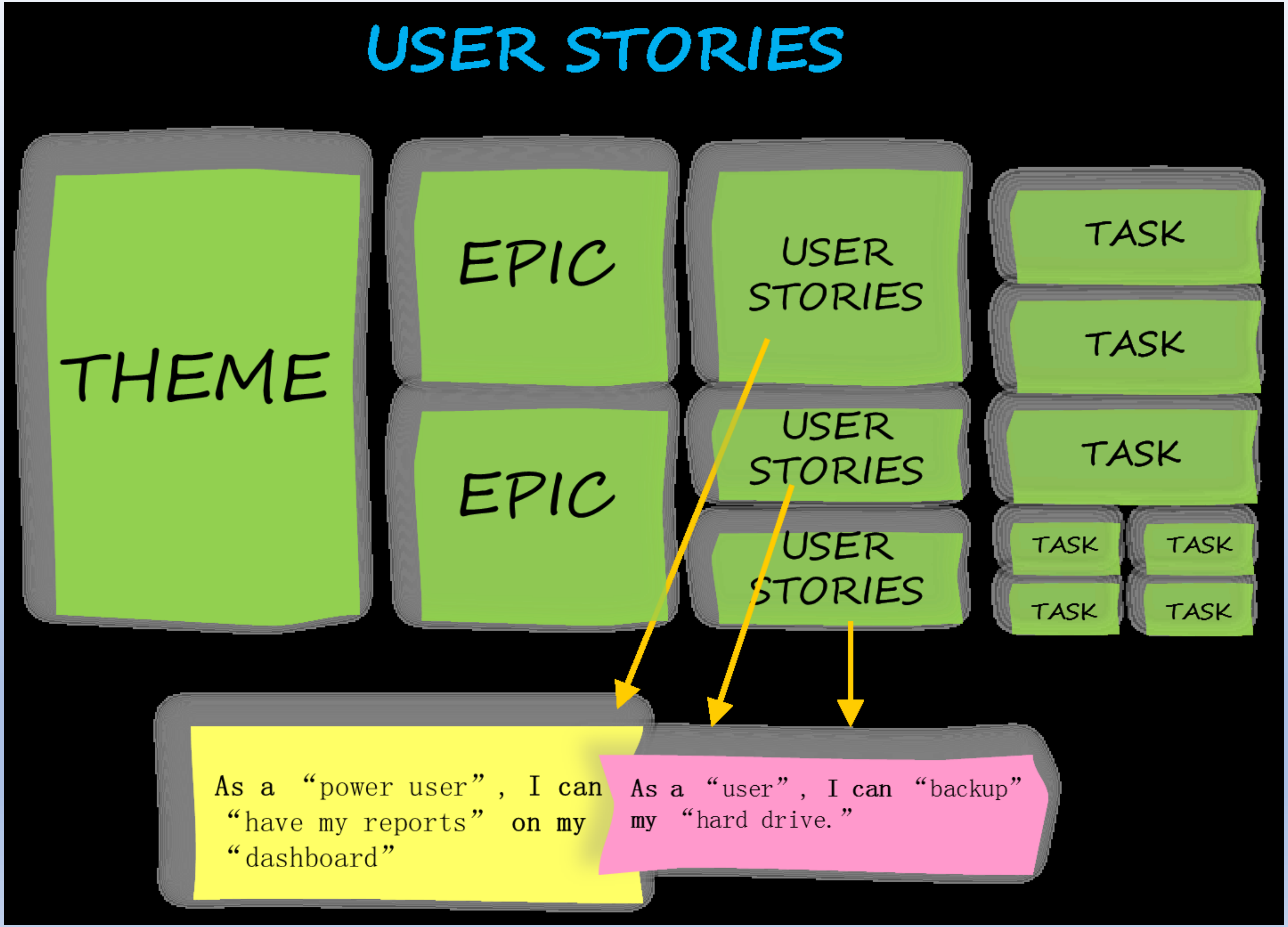
COMMITMENT

RESPECT

OPENNESS

SCRUM VALUES

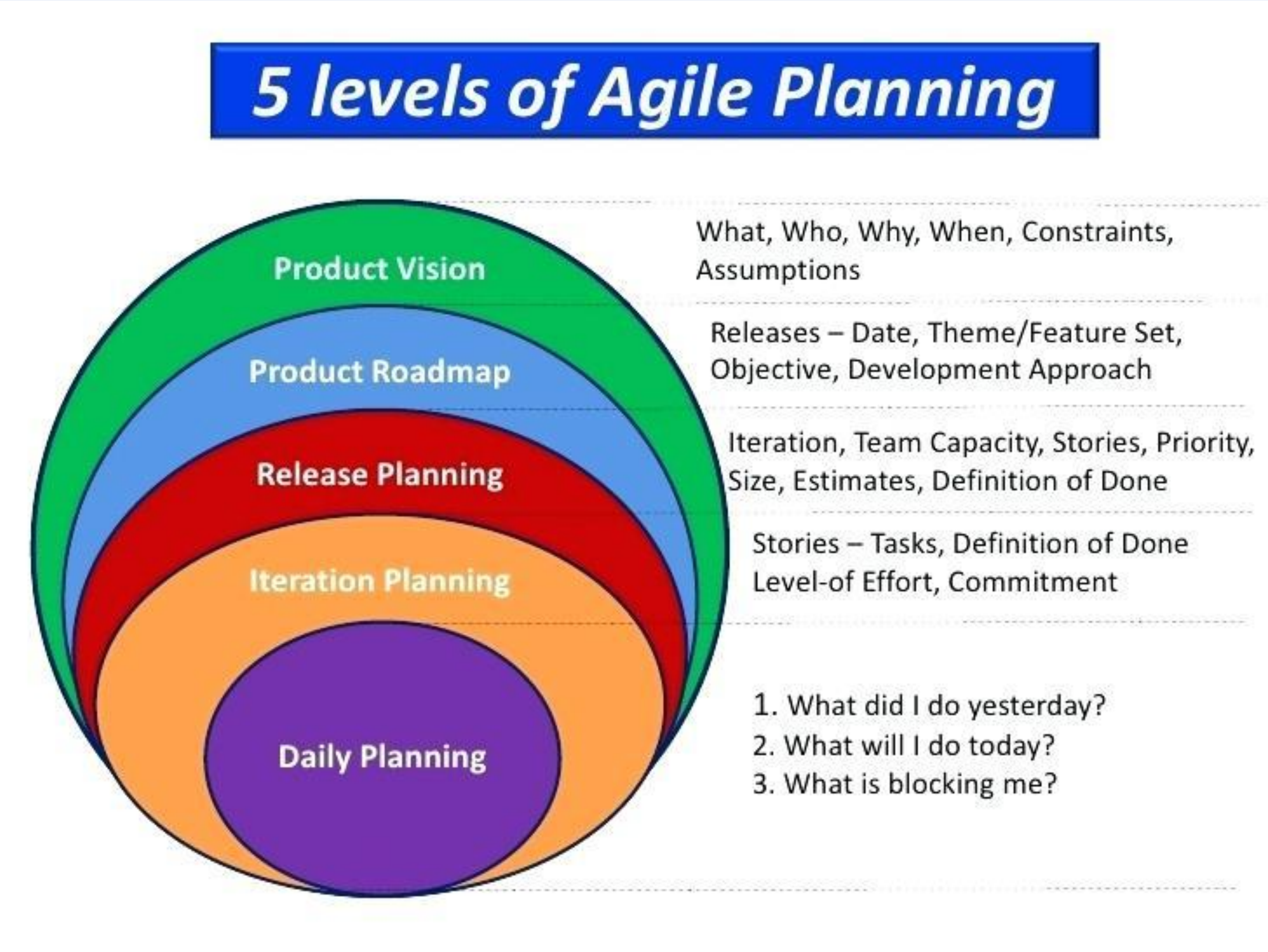
Release Management



As a "power user", I can "have my reports" on my "dashboard"

As a "user", I can "backup" my "hard drive."

Release Management



5 levels of Agile Planning

What, Who, Why, When, Constraints, Assumptions

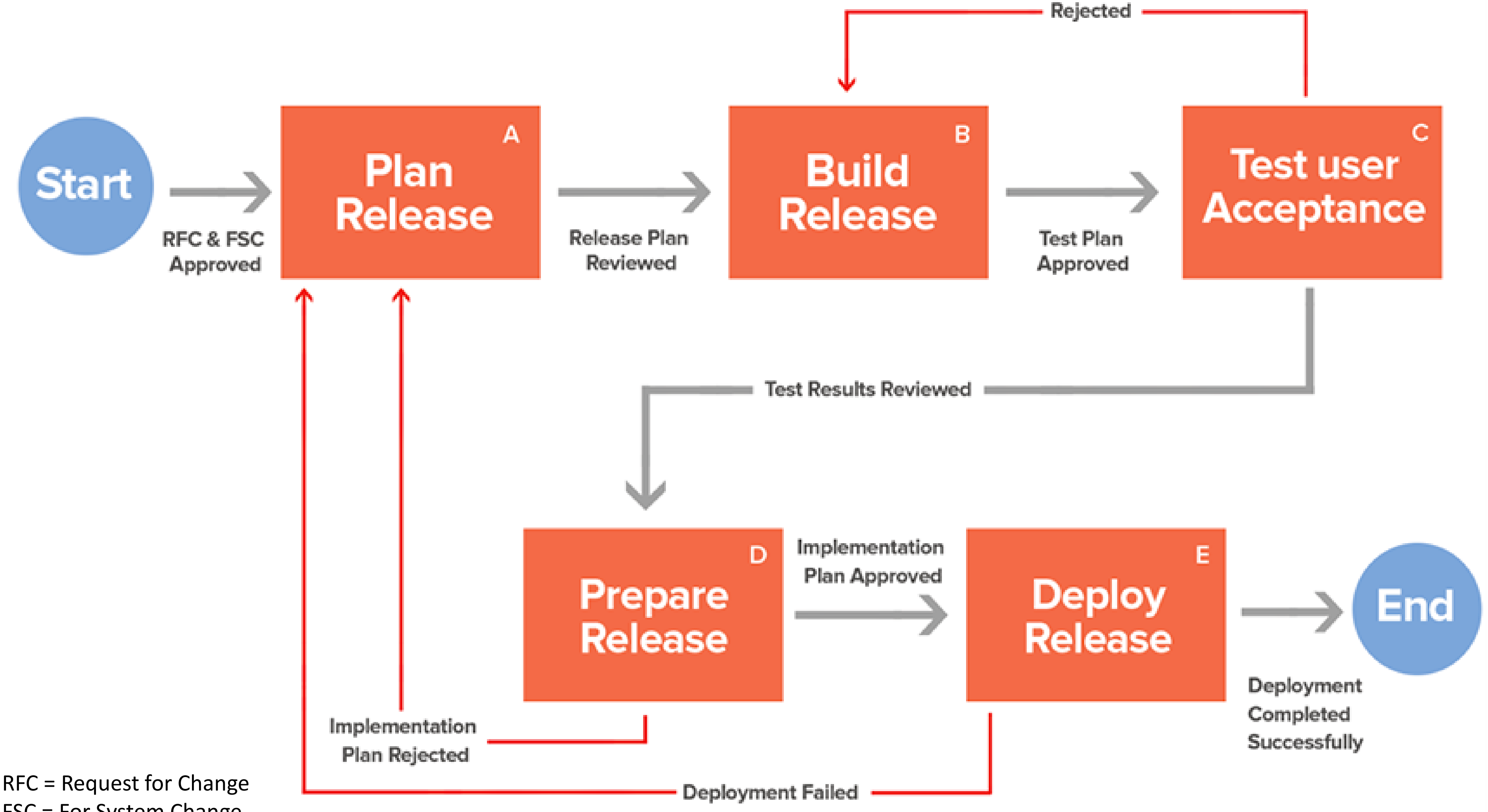
Releases – Date, Theme/Feature Set, Objective, Development Approach

Iteration, Team Capacity, Stories, Priority, Size, Estimates, Definition of Done

Stories – Tasks, Definition of Done Level-of Effort, Commitment

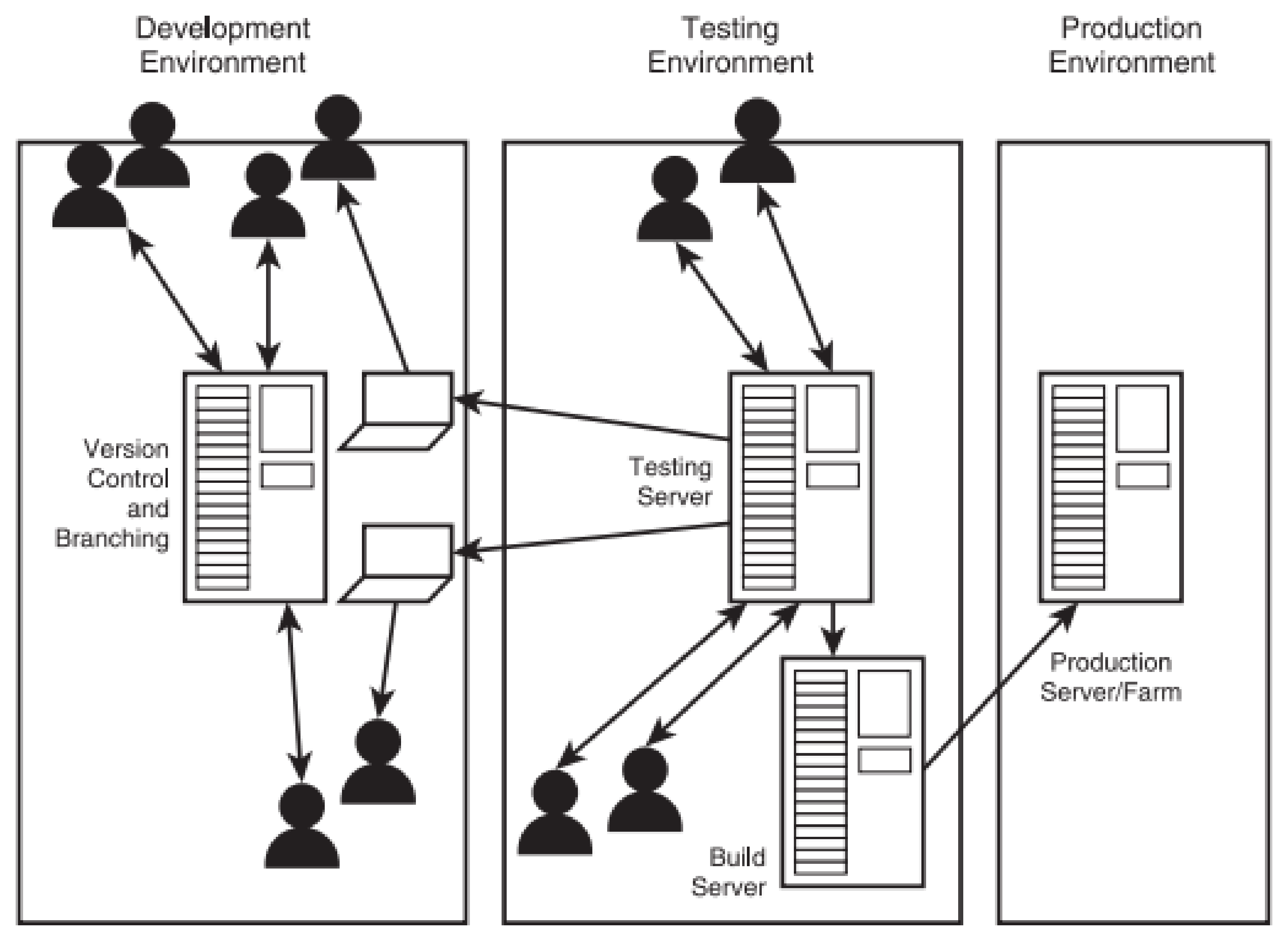
1. What did I do yesterday?
2. What will I do today?
3. What is blocking me?

Release Management



RFC = Request for Change
FSC = For System Change

Release Management



From development to testing to production.

Release Management

in

Disciplined Agile delivery (DAD)

Disciplined Agile delivery (DAD)

Disciplined agile delivery (DAD) is a process decision framework that enables simplified process decisions around incremental and iterative solution delivery. DAD builds on the many practices espoused by advocates of agile software development, including Scrum, agile modeling, lean software development, and others.

In particular, DAD has been identified as a means of moving beyond Scrum. According to **Cutter Senior Consultant Bhuvan Unhelkar**, "The DAD framework provides a carefully constructed mechanism that not only streamlines IT work, but more importantly, enables scaling.". **Paul Gorans and Philippe Kruchten** call for more discipline in implementation of agile approaches and indicate that DAD, as an example framework, is "a hybrid agile approach to enterprise IT solution delivery that provides a solid foundation from which to scale."

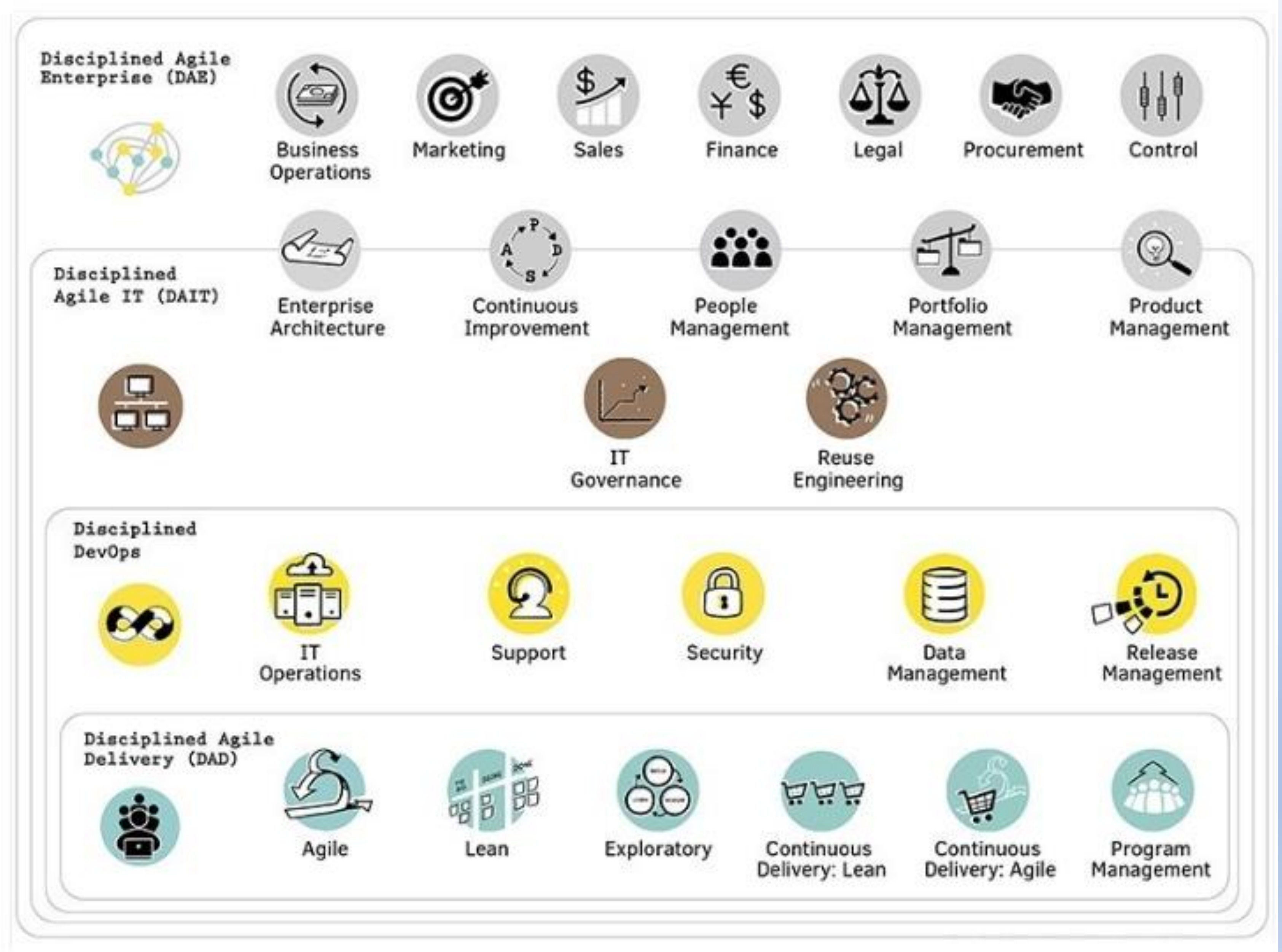
History

"DAD is a second-generation framework that strives to provide a coherent, end-to-end strategy for how agile solution delivery works in practice. DAD is a people-first, learning-oriented hybrid agile approach to IT solution delivery. It has a risk-value lifecycle, is goal-driven, is scalable, and is enterprise aware."

Scott Ambler developed the disciplined agile development process during his time as chief methodologist for **IT at IBM Rational (Summer 2006 to Summer 2012)**. It was developed to provide a more cohesive approach to agile software development; one that fills in the process gaps that are (purposely) ignored by Scrum, and one that is capable of enterprise-level scale. According to **Ambler**, "Many agile methodologies—including Scrum, XP, AM, Agile Data, Kanban, and more—focus on a subset of the activities required to deliver a solution from project initiation to delivery. Before DAD was developed, you needed to cobble together your own agile methodology to get the job done."

The DAD framework was developed as a result of observing common patterns where agility was applied at scale successfully. It reflects the experiences of IBM employees working in the field with various customer organizations, applying agile at scale internally, and from working with various business partners. "The DAD process framework recognizes not only the importance of networks of cross-functional teams, it also explicitly offers support for scaling key practices across complex working environments using techniques that link software development efforts into robust software delivery contexts“.

Disciplined Agile delivery (DAD)

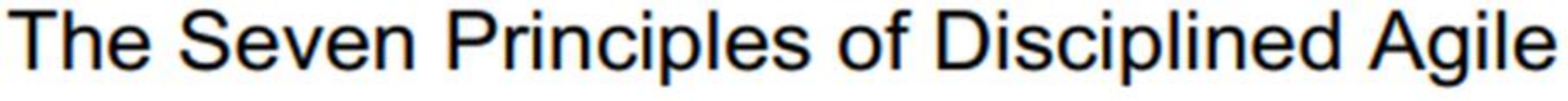


Release Management

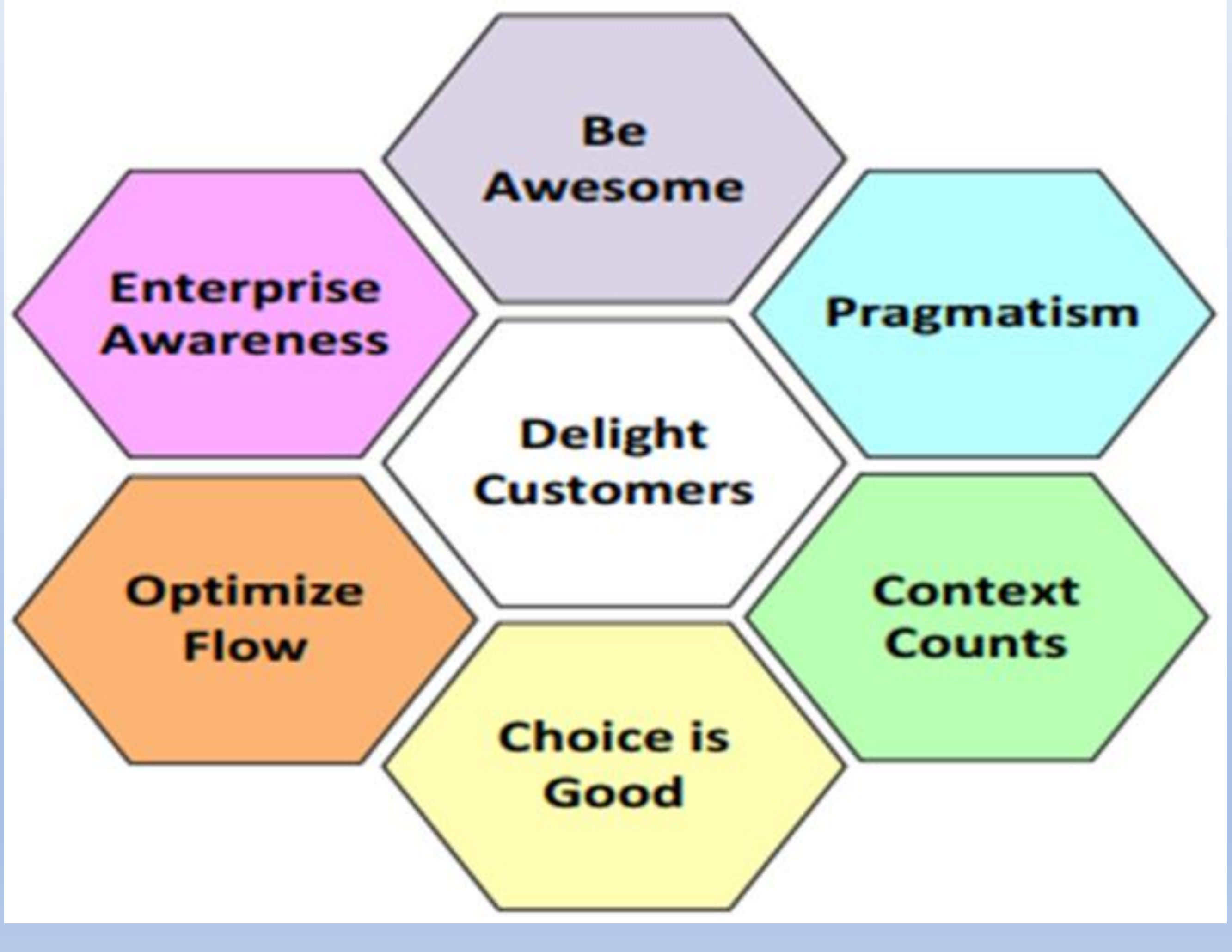
- ❑ Principles
- ❑ Scope
- ❑ Why release management ?
- ❑ Defining release management ?
- ❑ The release management mindset
- ❑ Release Management Strategies
- ❑ Parting thoughts

Release Management

Release Management



The Seven Principles of Disciplined Agile



Be Awesome

Enterprise Awareness

Pragmatism

Delight Customers

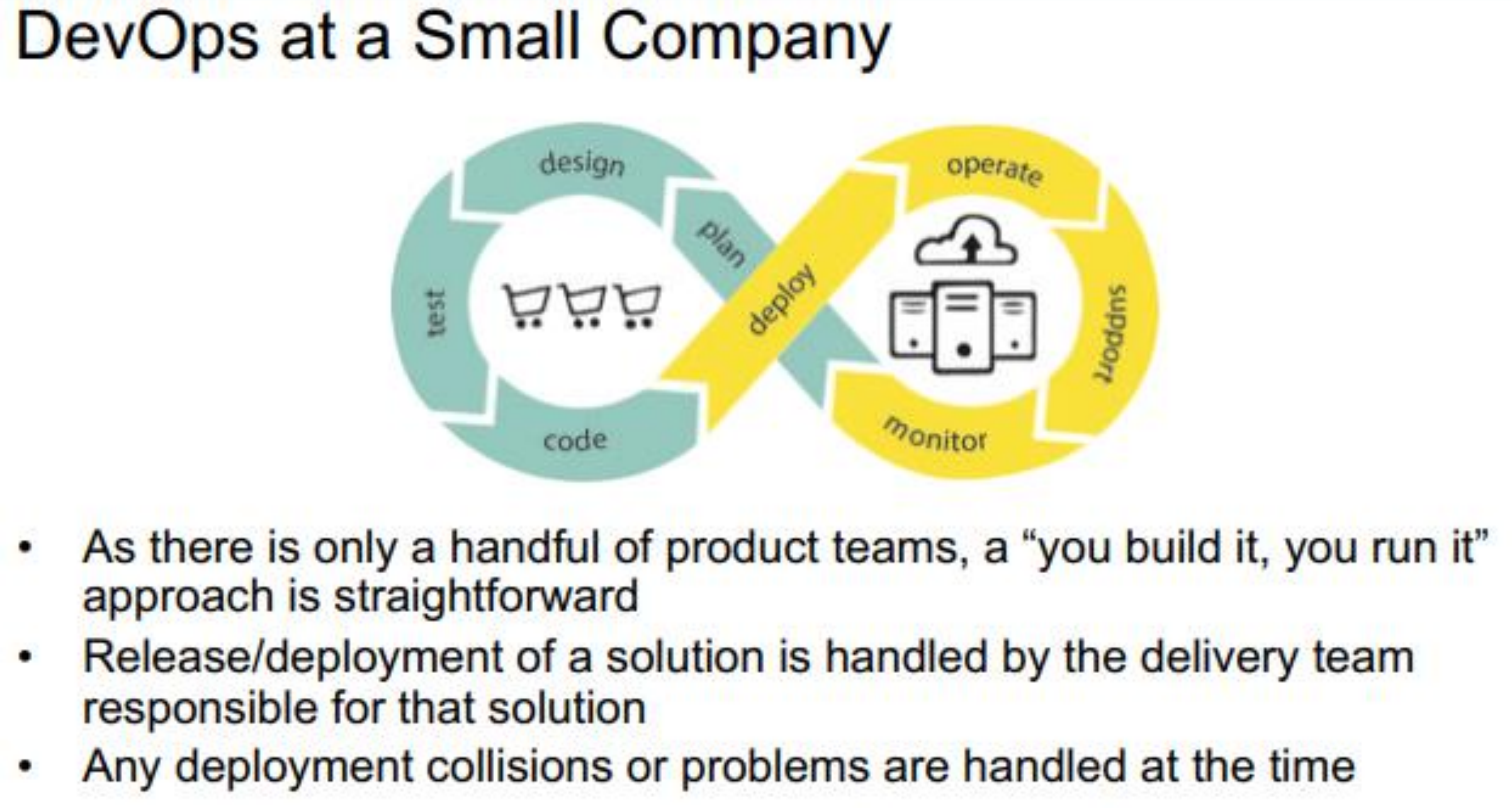
Optimize Flow

Context Counts

Choice is Good

Release Management

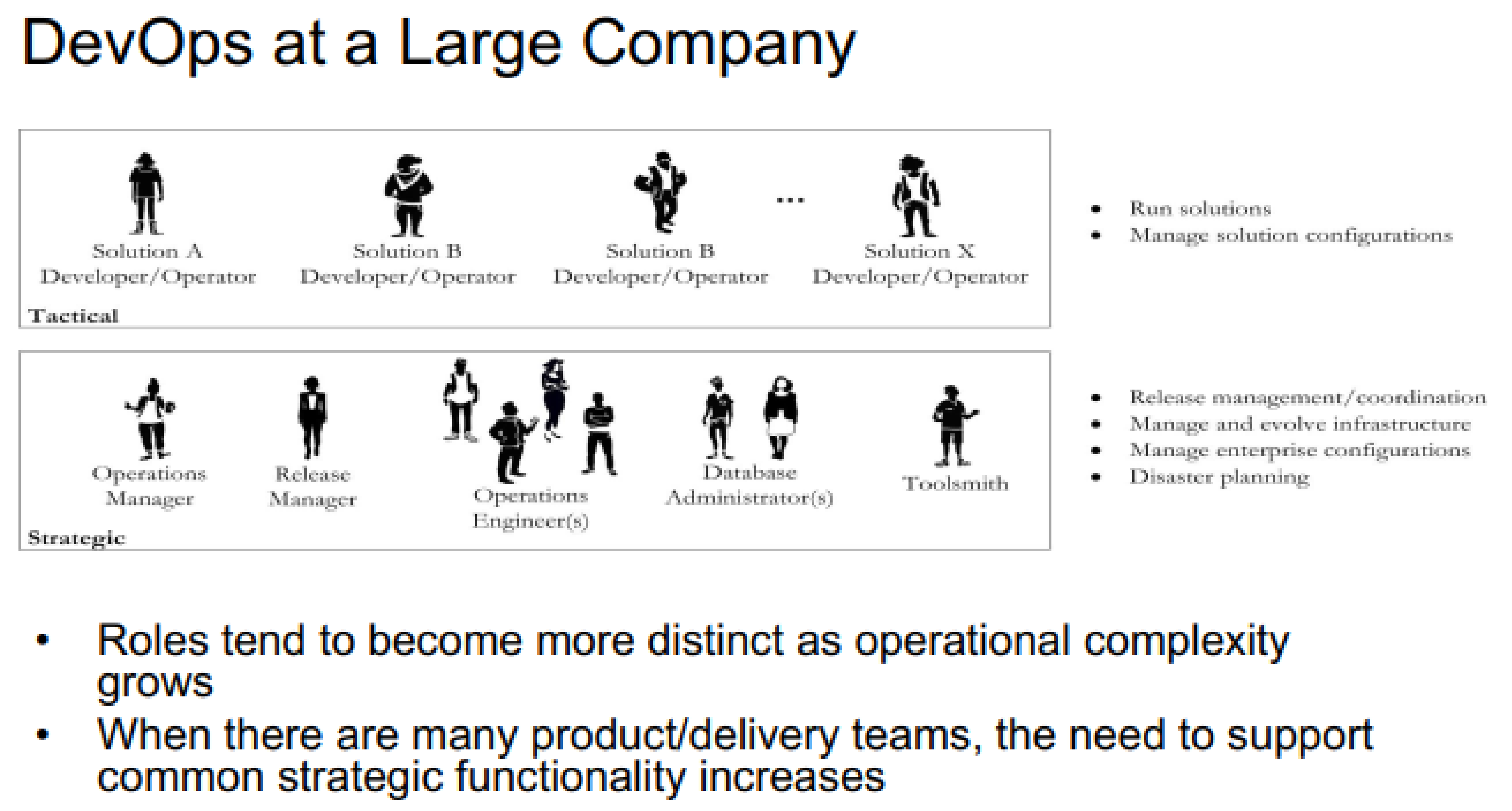
Release Management



DevOps at a Small Company

- As there is only a handful of product teams, a “you build it, you run it” approach is straightforward
- Release/deployment of a solution is handled by the delivery team responsible for that solution
- Any deployment collisions or problems are handled at the time

Release Management



DevOps at a Large Company

Tactical

Solution A Developer/Operator

Solution B Developer/Operator

Solution B Developer/Operator

...

Solution X Developer/Operator

- Run solutions
- Manage solution configurations

Strategic

Operations Manager

Release Manager

Operations Engineer(s)

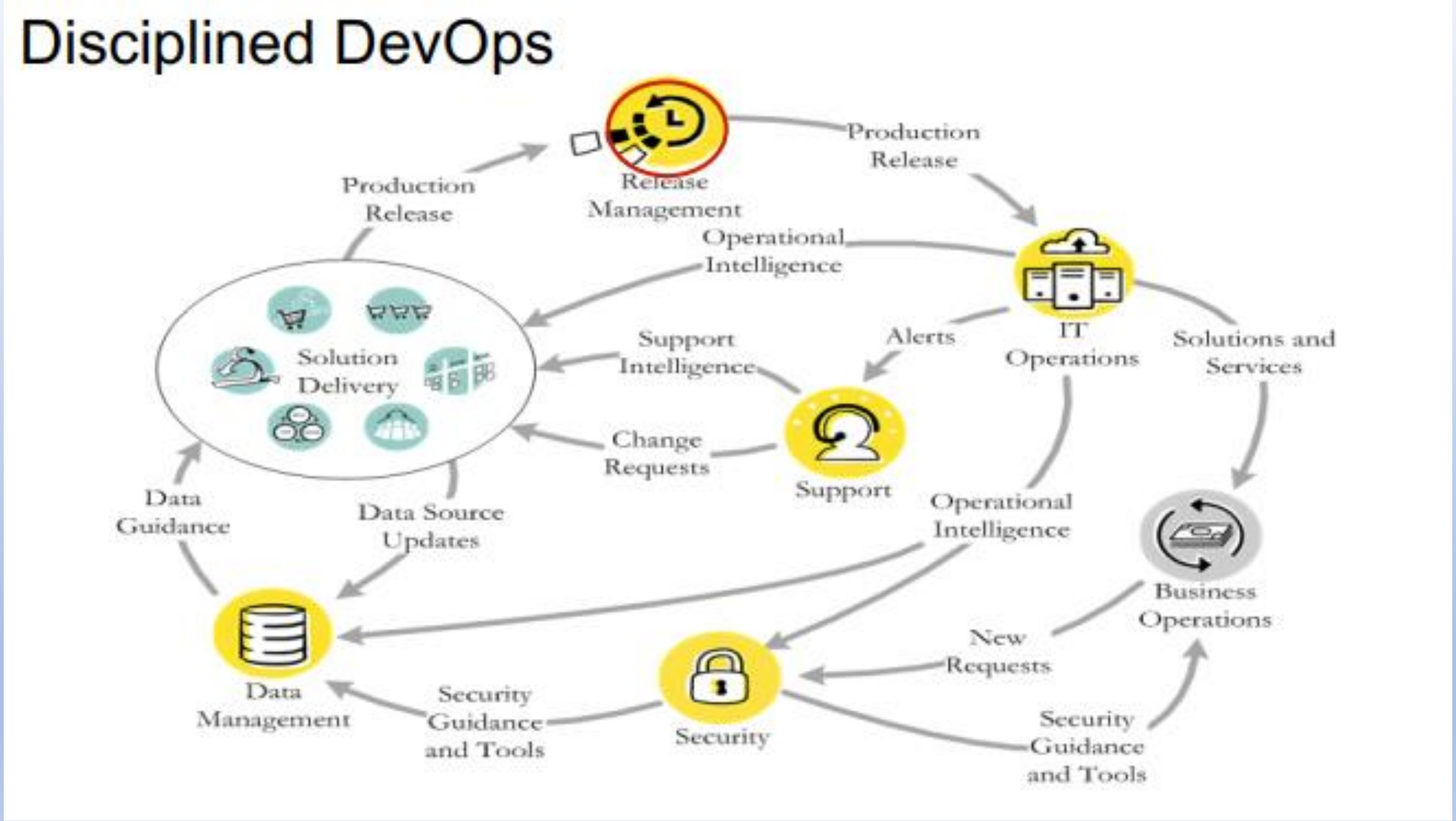
Database Administrator(s)

Toolsmith

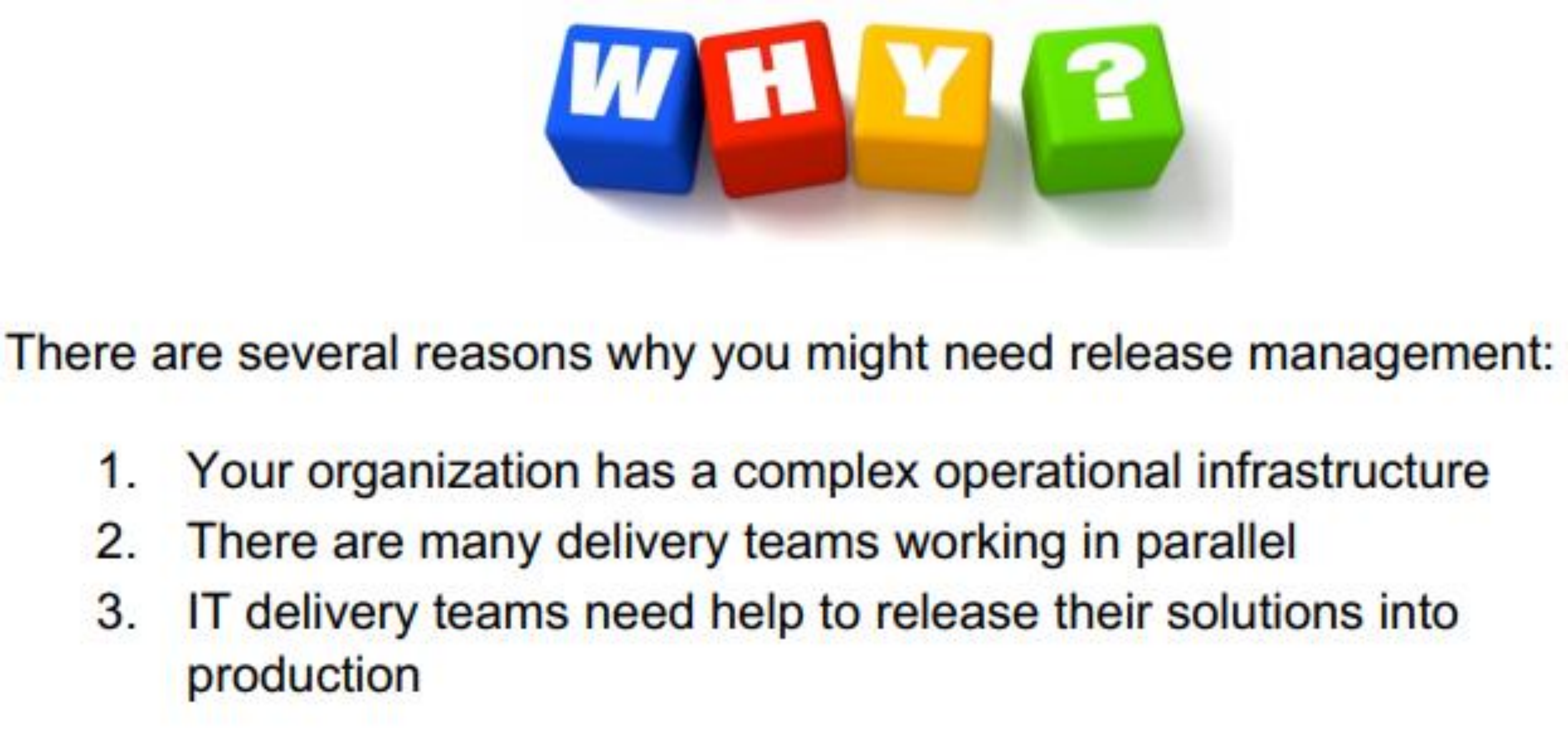
- Release management/coordination
- Manage and evolve infrastructure
- Manage enterprise configurations
- Disaster planning

- Roles tend to become more distinct as operational complexity grows
- When there are many product/delivery teams, the need to support common strategic functionality increases

Release Management



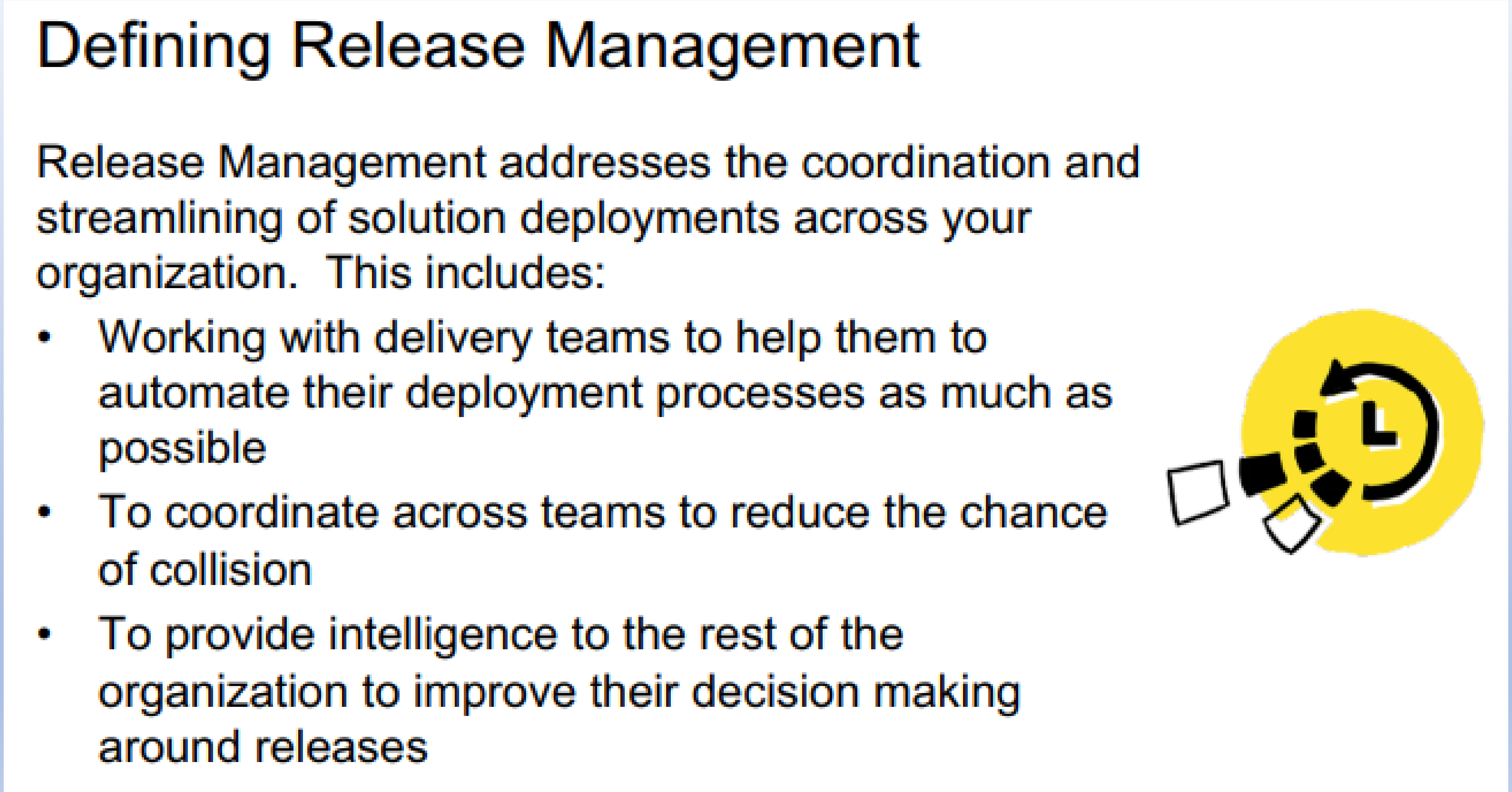
Release Management



There are several reasons why you might need release management:

1. Your organization has a complex operational infrastructure
2. There are many delivery teams working in parallel
3. IT delivery teams need help to release their solutions into production

Release Management

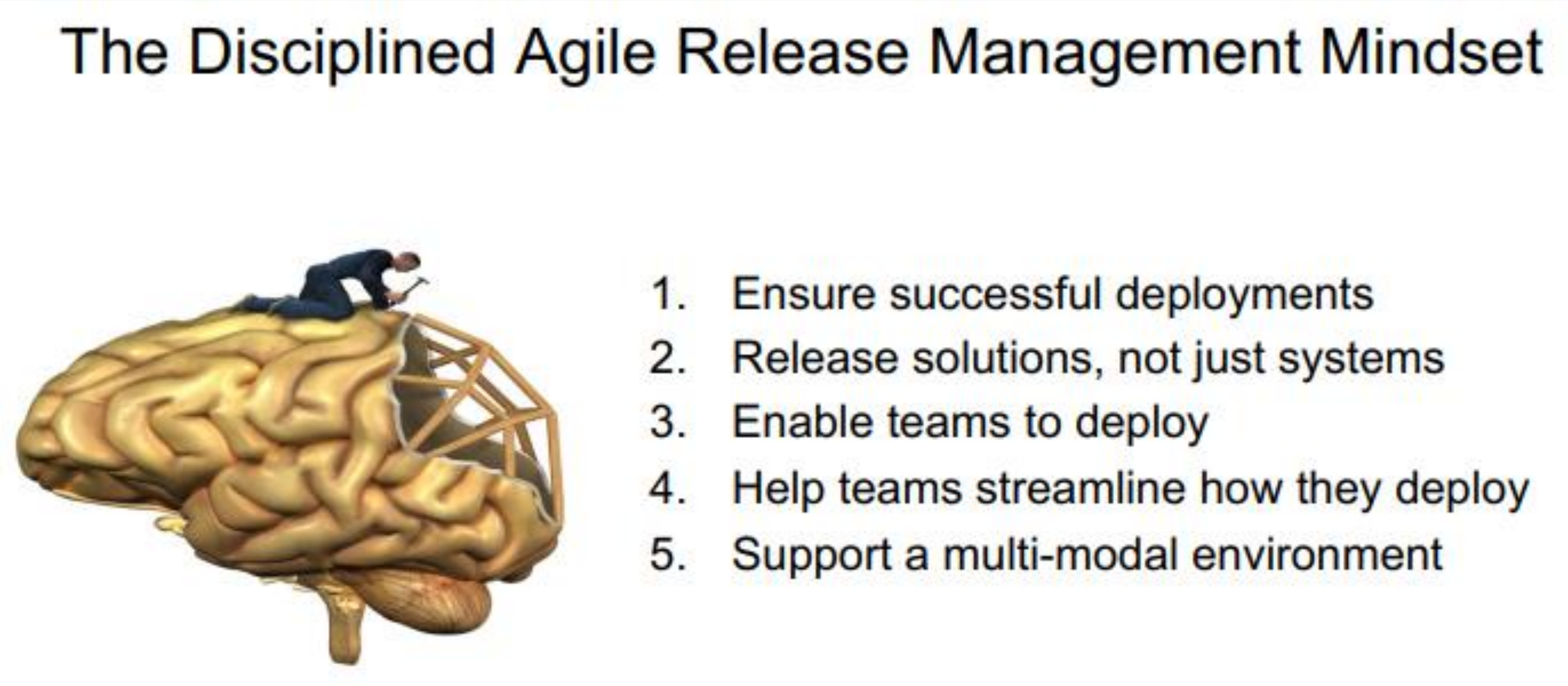


Defining Release Management

Release Management addresses the coordination and streamlining of solution deployments across your organization. This includes:

- Working with delivery teams to help them to automate their deployment processes as much as possible
- To coordinate across teams to reduce the chance of collision
- To provide intelligence to the rest of the organization to improve their decision making around releases

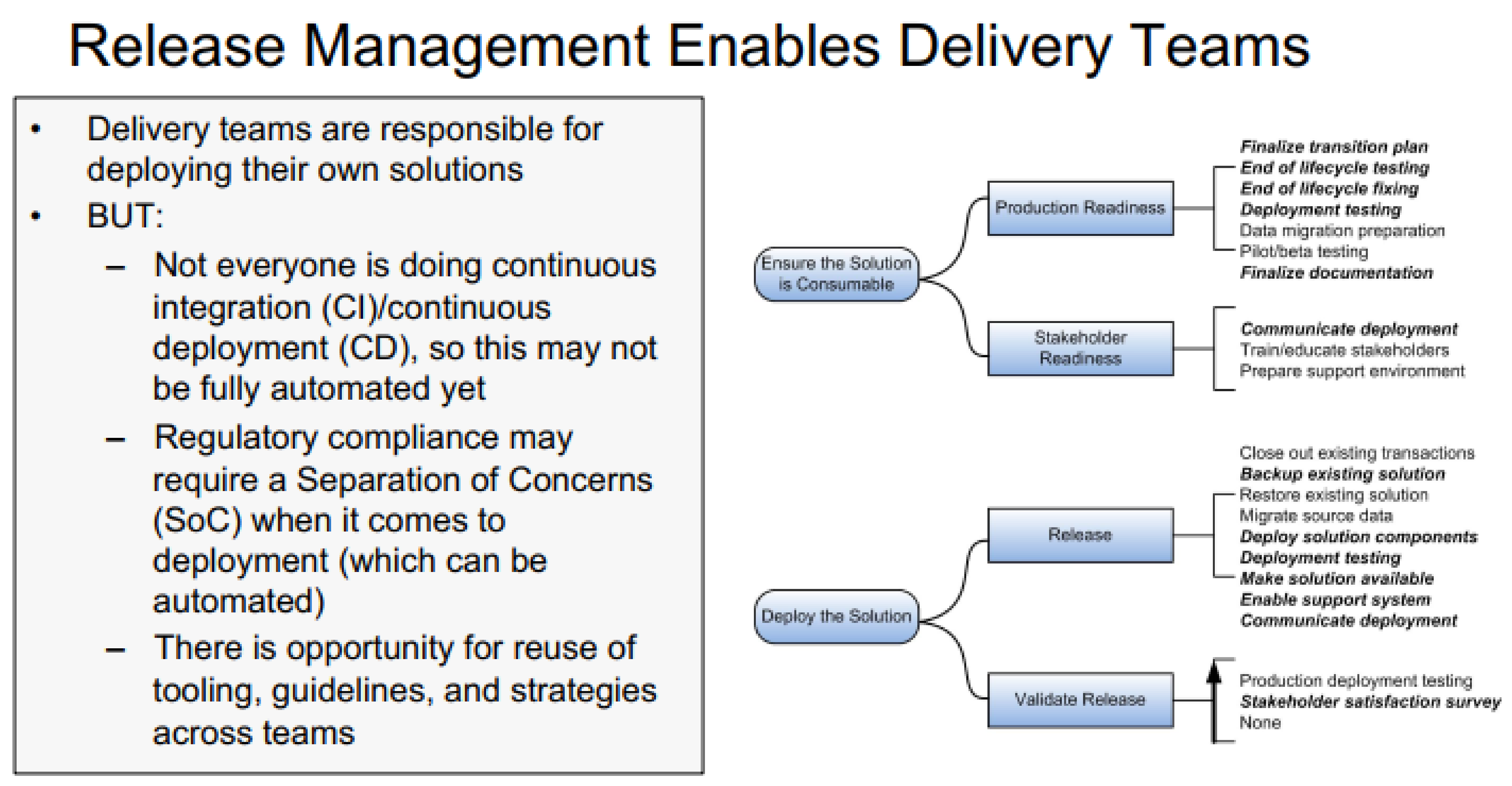
Release Management



The Disciplined Agile Release Management Mindset

1. Ensure successful deployments
2. Release solutions, not just systems
3. Enable teams to deploy
4. Help teams streamline how they deploy
5. Support a multi-modal environment

Release Management



Release Management Enables Delivery Teams

- Delivery teams are responsible for deploying their own solutions
- BUT:
  - Not everyone is doing continuous integration (CI)/continuous deployment (CD), so this may not be fully automated yet
  - Regulatory compliance may require a Separation of Concerns (SoC) when it comes to deployment (which can be automated)
  - There is opportunity for reuse of tooling, guidelines, and strategies across teams

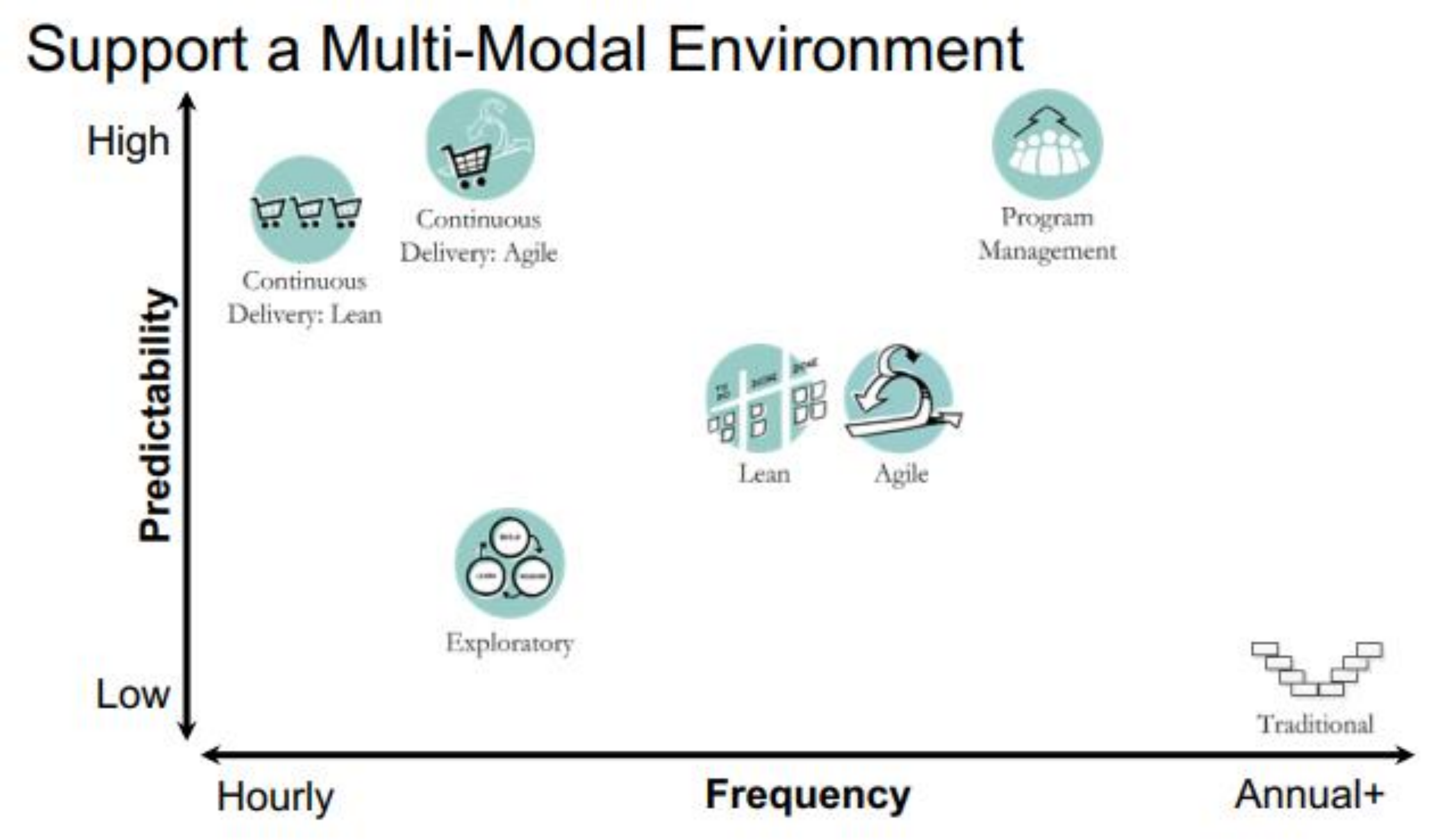
Ensure the Solution is Consumable

- Production Readiness
  - ***Finalize transition plan***
  - ***End of lifecycle testing***
  - ***End of lifecycle fixing***
  - ***Deployment testing***
  - Data migration preparation
  - Pilot/beta testing
  - ***Finalize documentation***
- Stakeholder Readiness
  - ***Communicate deployment***
  - Train/educate stakeholders
  - Prepare support environment

Deploy the Solution

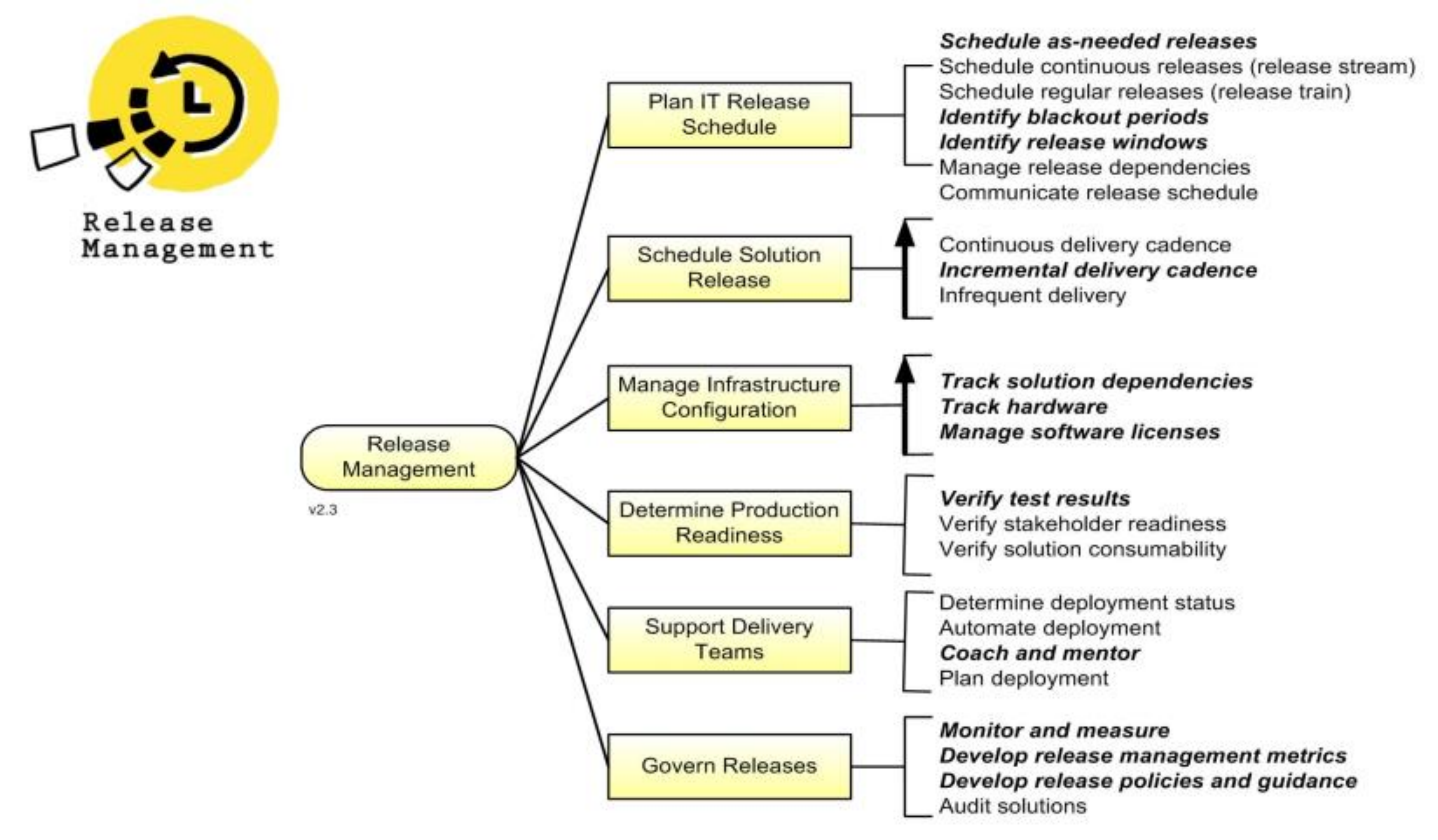
- Release
  - Close out existing transactions
  - ***Backup existing solution***
  - Restore existing solution
  - Migrate source data
  - ***Deploy solution components***
  - ***Deployment testing***
  - ***Make solution available***
  - ***Enable support system***
  - ***Communicate deployment***
- Validate Release
  - Production deployment testing
  - ***Stakeholder satisfaction survey***
  - None

Release Management



Release Management

Release Management



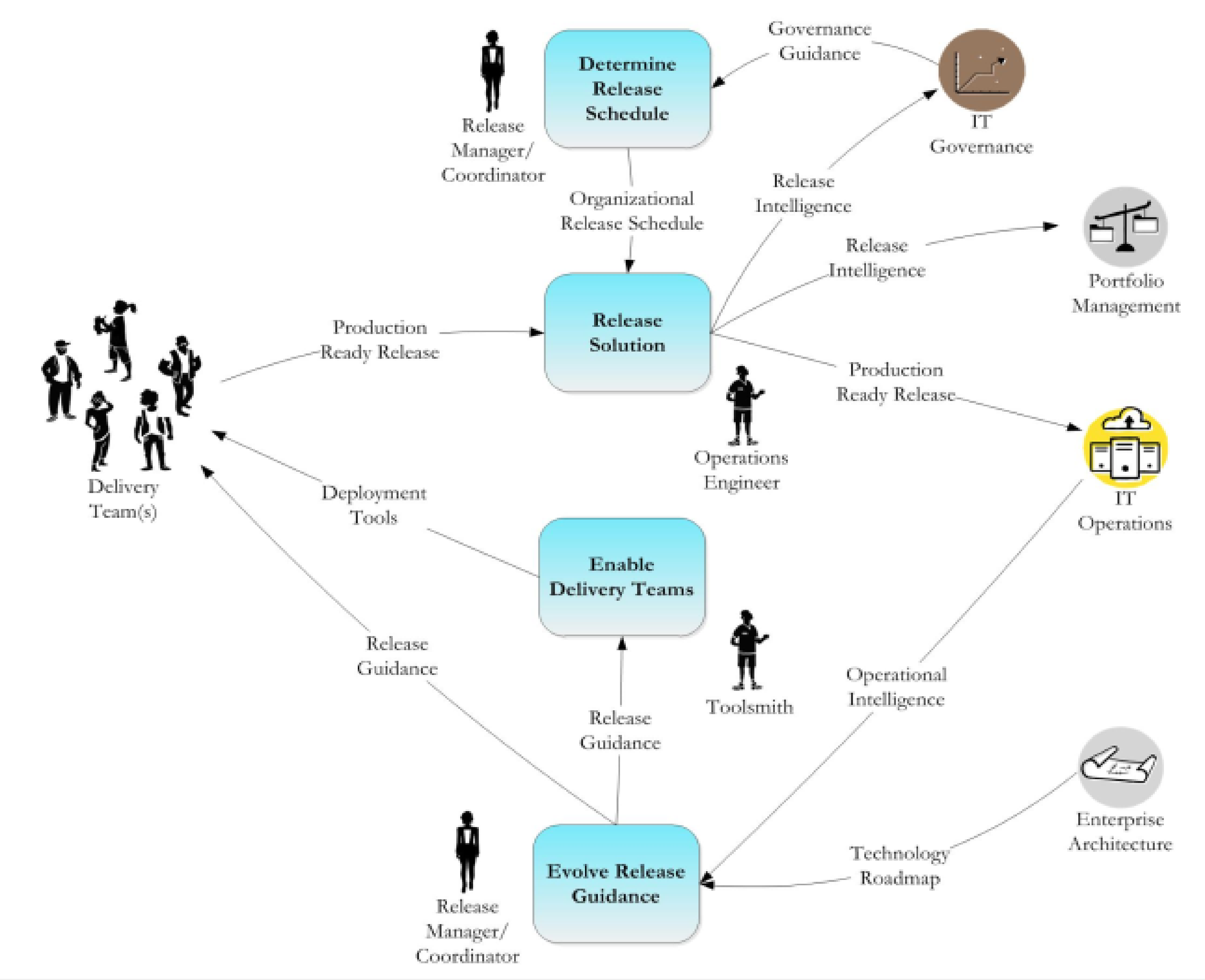
Release Management

Release Management

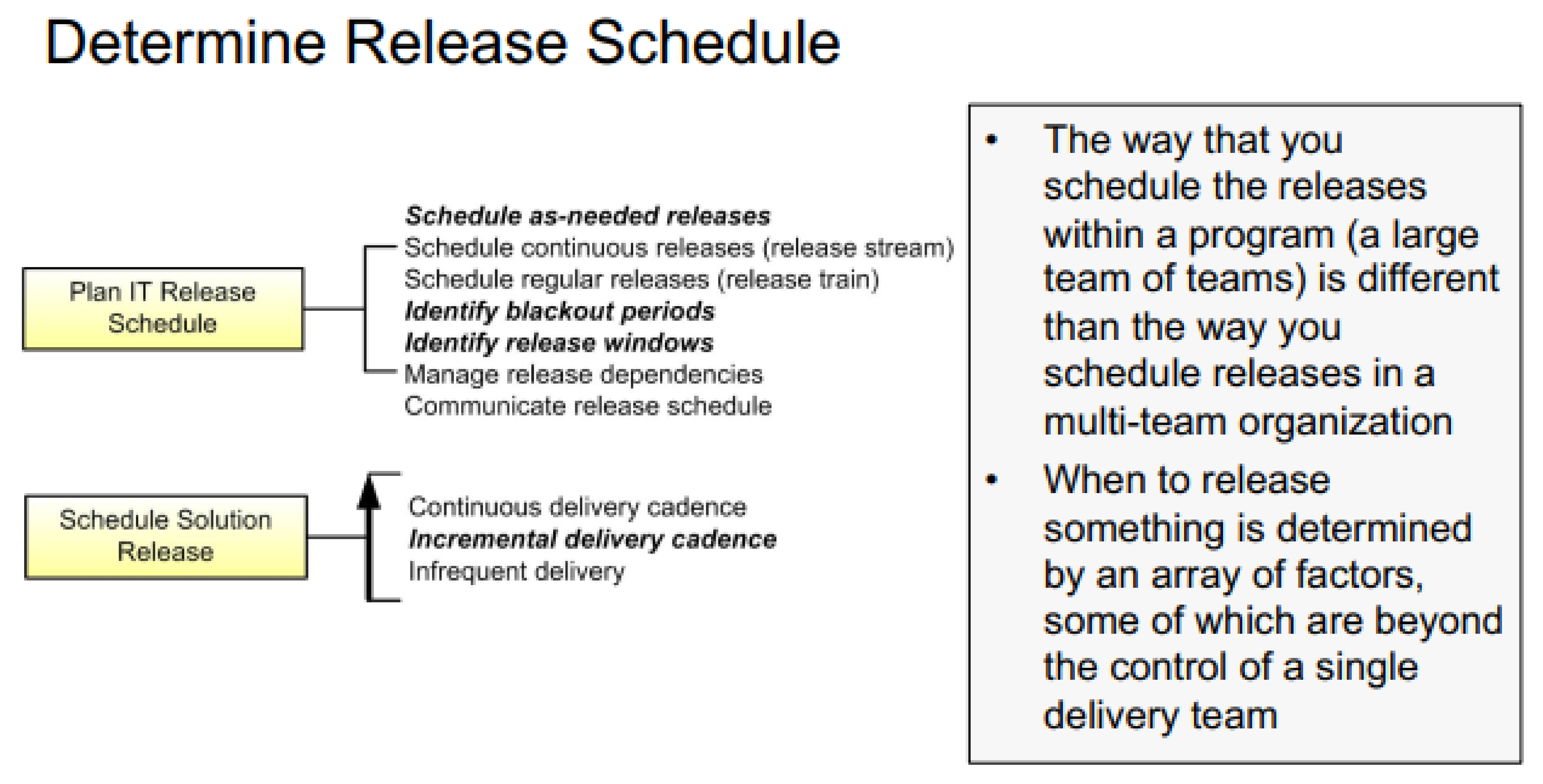
v2.3

- Plan IT Release Schedule
  - ***Schedule as-needed releases***
  - Schedule continuous releases (release stream)
  - Schedule regular releases (release train)
  - ***Identify blackout periods***
  - ***Identify release windows***
  - Manage release dependencies
  - Communicate release schedule
- Schedule Solution Release
  - Continuous delivery cadence
  - ***Incremental delivery cadence***
  - Infrequent delivery
- Manage Infrastructure Configuration
  - ***Track solution dependencies***
  - ***Track hardware***
  - ***Manage software licenses***
- Determine Production Readiness
  - ***Verify test results***
  - Verify stakeholder readiness
  - Verify solution consumability
- Support Delivery Teams
  - Determine deployment status
  - Automate deployment
  - ***Coach and mentor***
  - Plan deployment
- Govern Releases
  - ***Monitor and measure***
  - ***Develop release management metrics***
  - ***Develop release policies and guidance***
  - Audit solutions

Release Management



Release Management



Determine Release Schedule

Plan IT Release Schedule

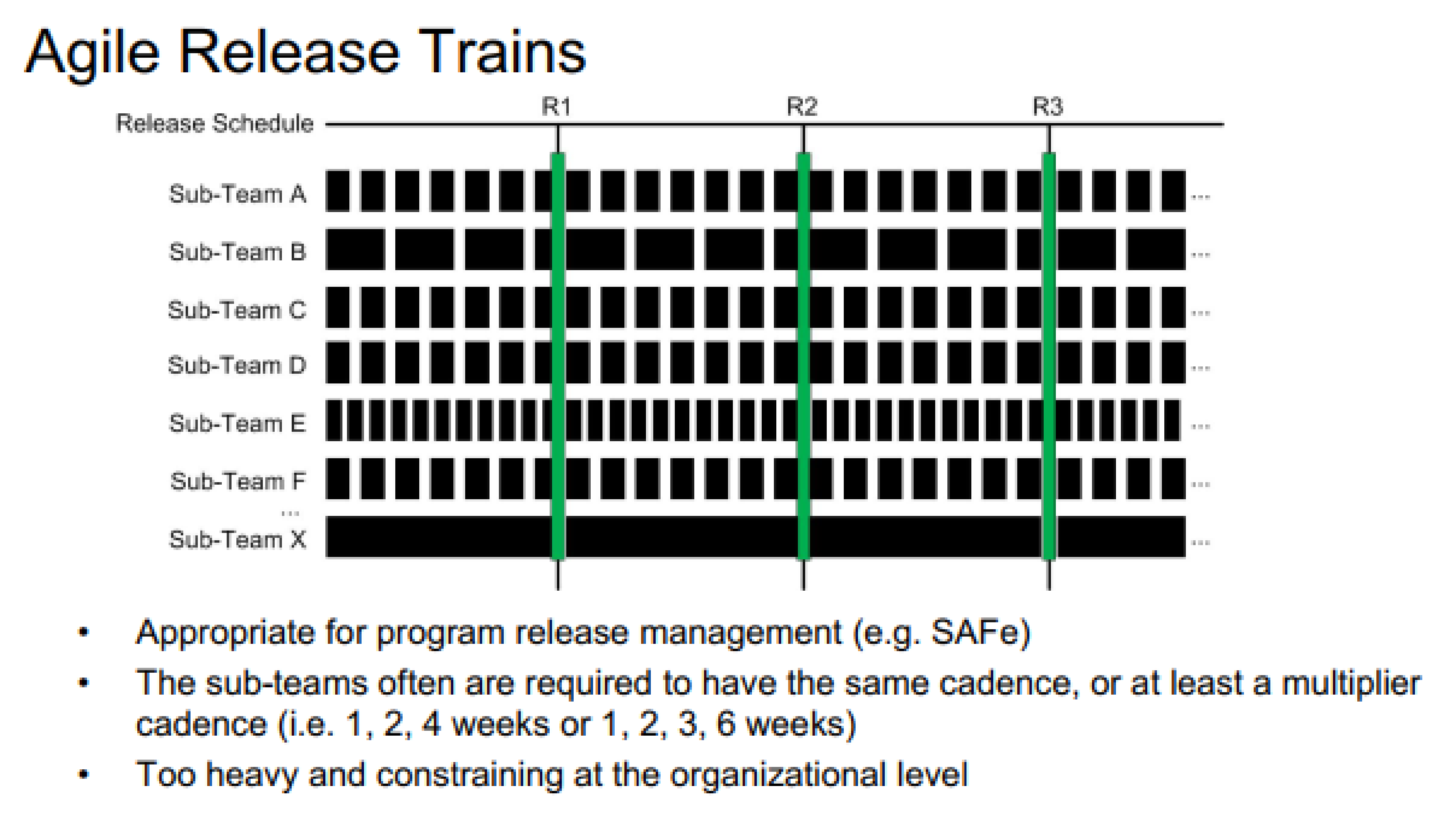
- ***Schedule as-needed releases***
- Schedule continuous releases (release stream)
- Schedule regular releases (release train)
- ***Identify blackout periods***
- ***Identify release windows***
- Manage release dependencies
- Communicate release schedule

Schedule Solution Release

- Continuous delivery cadence
- ***Incremental delivery cadence***
- Infrequent delivery

- The way that you schedule the releases within a program (a large team of teams) is different than the way you schedule releases in a multi-team organization
- When to release something is determined by an array of factors, some of which are beyond the control of a single delivery team

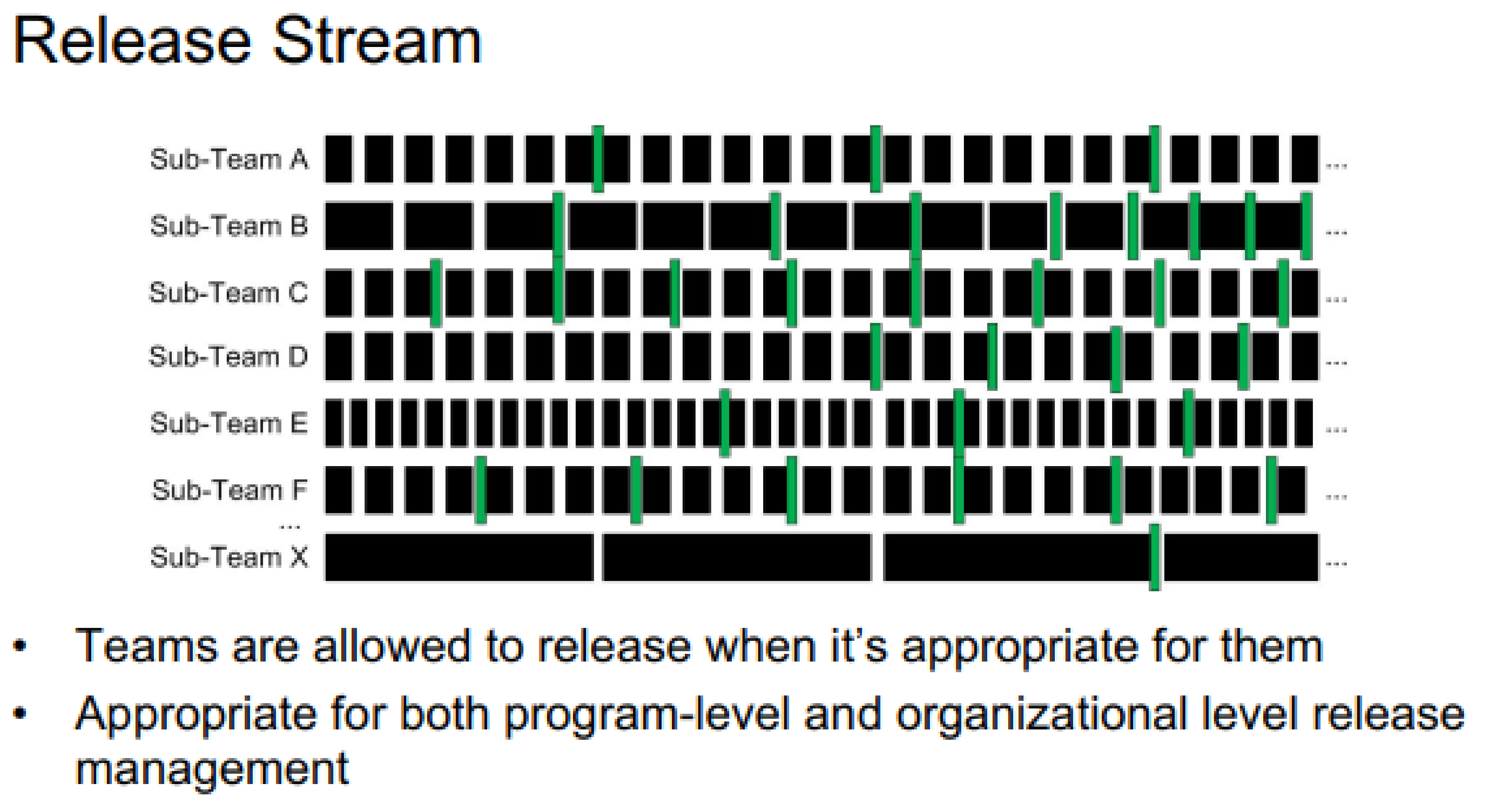
Release Management



Agile Release Trains

- Appropriate for program release management (e.g. SAFe)
- The sub-teams often are required to have the same cadence, or at least a multiplier cadence (i.e. 1, 2, 4 weeks or 1, 2, 3, 6 weeks)
- Too heavy and constraining at the organizational level

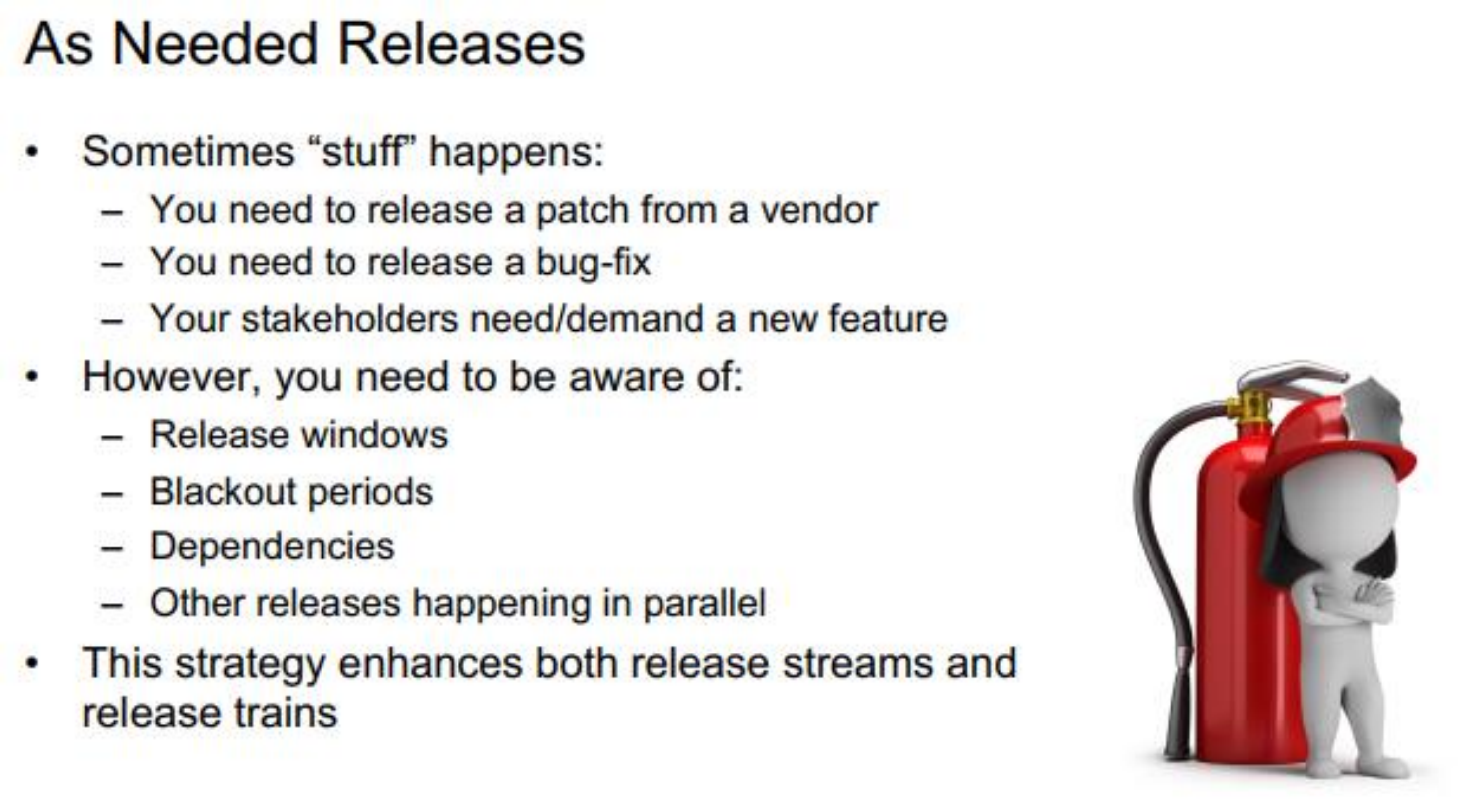
Release Management



Release Stream

- Teams are allowed to release when it's appropriate for them
- Appropriate for both program-level and organizational level release management

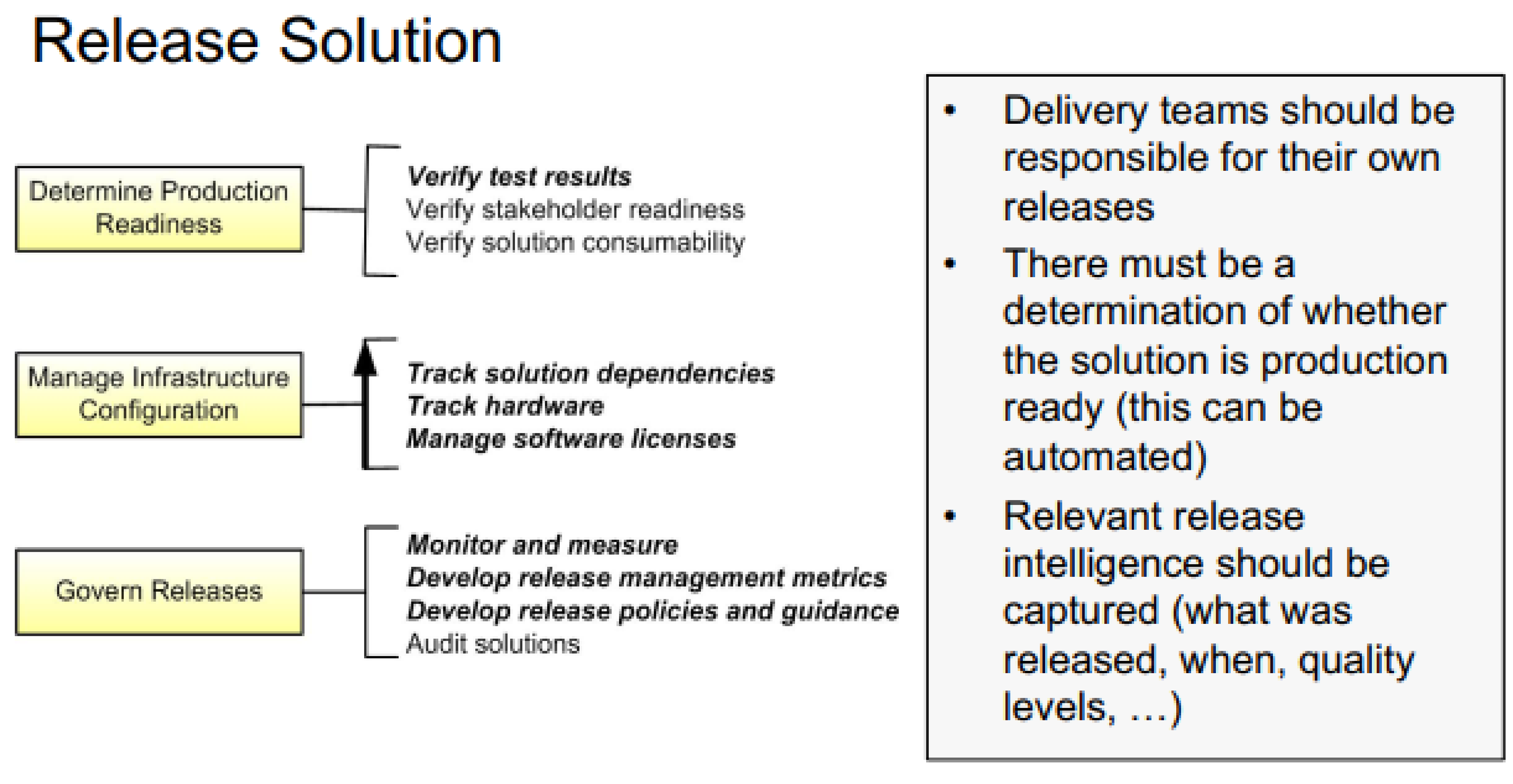
Release Management



As Needed Releases

- Sometimes “stuff” happens:
  - You need to release a patch from a vendor
  - You need to release a bug-fix
  - Your stakeholders need/demand a new feature
- However, you need to be aware of:
  - Release windows
  - Blackout periods
  - Dependencies
  - Other releases happening in parallel
- This strategy enhances both release streams and release trains

Release Management



Release Solution

Determine Production Readiness

Verify test results
Verify stakeholder readiness
Verify solution consumability

Manage Infrastructure Configuration

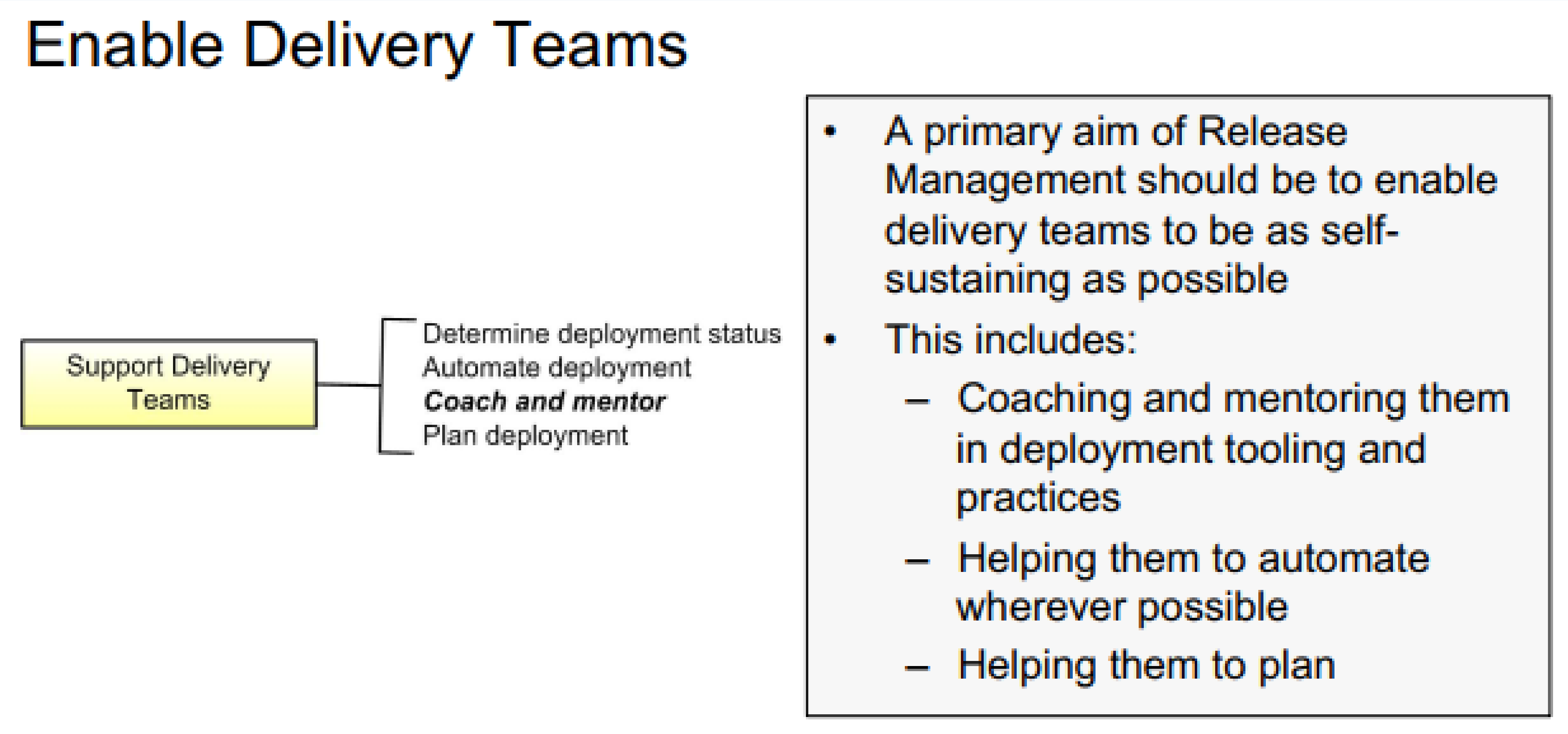
Track solution dependencies
Track hardware
Manage software licenses

Govern Releases

Monitor and measure
Develop release management metrics
Develop release policies and guidance
Audit solutions

- Delivery teams should be responsible for their own releases
- There must be a determination of whether the solution is production ready (this can be automated)
- Relevant release intelligence should be captured (what was released, when, quality levels, …)

Release Management



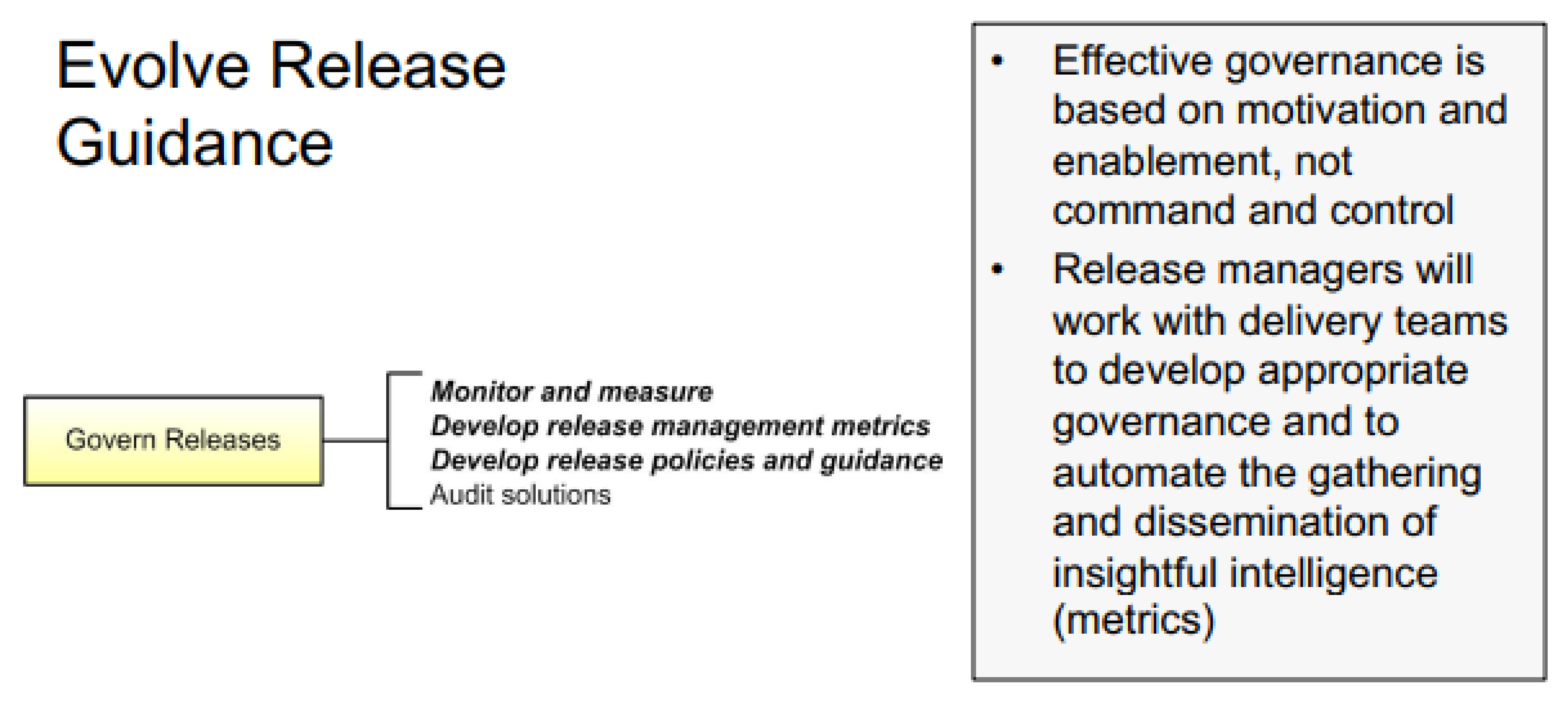
Enable Delivery Teams

Support Delivery Teams

- Determine deployment status
- Automate deployment
- ***Coach and mentor***
- Plan deployment

- A primary aim of Release Management should be to enable delivery teams to be as self-sustaining as possible
- This includes:
  - Coaching and mentoring them in deployment tooling and practices
  - Helping them to automate wherever possible
  - Helping them to plan

Release Management



Evolve Release Guidance

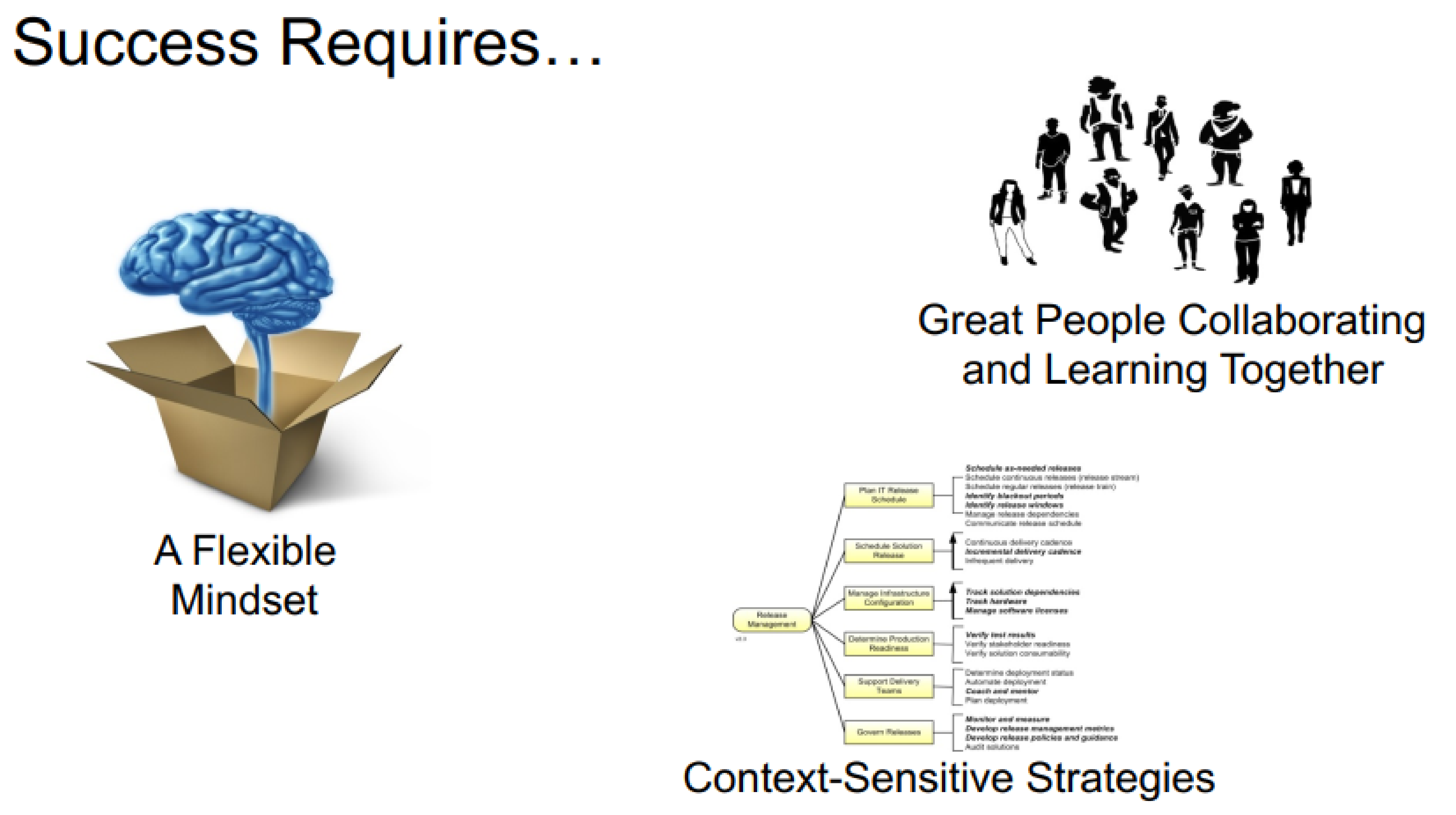
Govern Releases

- ***Monitor and measure***
- ***Develop release management metrics***
- ***Develop release policies and guidance***
- Audit solutions

- Effective governance is based on motivation and enablement, not command and control
- Release managers will work with delivery teams to develop appropriate governance and to automate the gathering and dissemination of insightful intelligence (metrics)

Release Management

Release Management



Success Requires…

A Flexible Mindset

Great People Collaborating and Learning Together

Release Management

v3.0

Plan IT Release Schedule

- ***Schedule as-needed releases***
- Schedule continuous releases (release stream)
- Schedule regular releases (release train)
- ***Identify blackout periods***
- ***Identify release windows***
- Manage release dependencies
- Communicate release schedule

Schedule Solution Release

- Continuous delivery cadence
- ***Incremental delivery cadence***
- Infrequent delivery

Manage Infrastructure Configuration

- ***Track solution dependencies***
- ***Track hardware***
- ***Manage software licenses***

Determine Production Readiness

- ***Verify test results***
- Verify stakeholder readiness
- Verify solution consumability

Support Delivery Teams

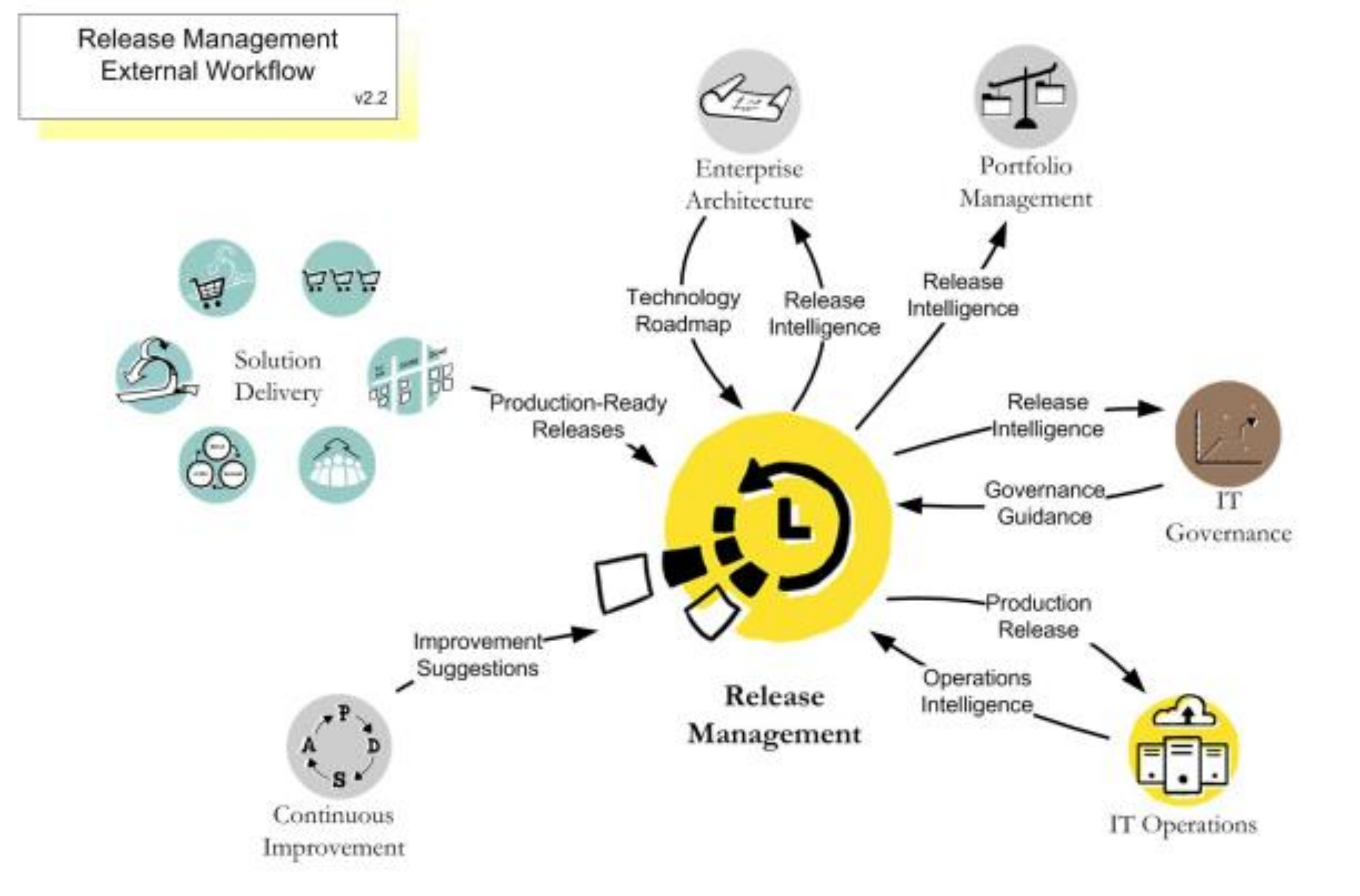
- Determine deployment status
- Automate deployment
- ***Coach and mentor***
- Plan deployment

Govern Releases

- ***Monitor and measure***
- ***Develop release management metrics***
- ***Develop release policies and guidance***
- Audit solutions

Context-Sensitive Strategies

Release Management



Release Management

Thank You!