Release management and testing

WHY THIS WAY AND NOT ANOTHER

Disclamer

- Only some slides are finished. The rest are unfinished artefacts provided as is.

The ultimate goal

AND THE MAIN CHALLENGE

The ultimate goal – transform infrastructure practices

The goal is for distant future, but it will not come before we are made redundant if we move too slow

Recognize that we cannot longer afford over-engineering practices inherited from times when systems were planned for scale-up applications and timeframes of 5-7 years:

- Feature-rich systems from leading vendors won tenders
- The assurance (and responsibility) for the whole solution is in on vendor

Spend wisely, become agile - build organizational and technical resources and capabilities allowing quickly adapt to new or changed demands:

- System with just enough features for immediate applications is to win tender; new features are added from the same or different vendor not earlier than a new business application comes
- It will only work if the assurance becomes a fundamental group capability

The challenge

Faced with the choice between changing one's mind and proving that there is no need to do so, almost everybody gets busy on the proof.

John Kenneth Galbraith, American economist

Agenda

- What type of lifecycle management is needed initially?
- Key ideas
- Initial scope of the process suite
- Simplified process flow
- Process details
- Roles and responsibilities
- Organizational aspects
- Summary

Easy start

WHAT TYPE OF LIFECYCLE MANAGEMENT WE NEED INITIALLY?

Triggers for verification and certification

- Hotfixes, service packs
- Feature / functionality update or release e.g. for integration with external system
- Management model (e.g. heat stack) update / upgrade
- NFV Infrastructure
  - HW upgrade (e.g. CPU, NIC, disc drives);
  - Driver, firmware update
  - Host OS, libvirt, agent update
- VIM
  - OpenStack / SDN controller update or upgrade;
  - Compute / network / storage agents update
- VNF
  - VNF update or upgrade;
  - New VNF
  - VNF manager update or upgrade
- Customizations pushed by vendors

What* might cause an issue?

Release size	Example	Components that might cause an issue
Small	An update of single component like Network Interface Card firmware or driver update from previous version to next minor version	Network interface card itself and components that depend on it: Host OS, Guest OS, vSwitch, etc.
Medium	A set of updates for a single domain like the whole infrastructure network domain firmware / software release pack	Rack aggregation layer switches; NFVI-PoP aggregation layer switches; Network Gateways Layer routers/switch-routers; Firewalls of Network Firewalls Layer; Load balancers; NFVI-to-PE interconnection devices; vSwitches; vFirewalls; vLoad Balancers; SDN controllers
Large	A set of major updates for the whole NFV Infrastructure and VIM	Compute domain (firmware for motherboard, disc controller, network adapter, IPMI module; host OS, guest OS, drivers in host and guest OS), storage domain (firmware, software, including storage, failover managers), network domain (the list of components is above), hardware management systems (for compute, storage, network management), OpenStack (Compute (Nova), Networking (Neutron), Block Storage (Cinder), Identity (Keystone), Image (Glance), Object Storage, Dashboard (Horizon), Orchestration (Heat), Workflow (Mistral), Telemetry (Ceilometer), Bare Metal (Ironic), Messaging (Zaqar), DNS (Designate), Search (Searchlight), vendor-specific component, etc.), SDN controller (for example, for OpenDaylight-based: user interface, ALTO, Authentication and authorization, Armoury, BGP, CAPWAP, DIDM, GBP, L2switch, L3VPN, LACP, LISP, Messaging4Transport, NEMO, NetIDE, NIC, ODL-SDNi, OpenFlow plugin, OpFlex, PCEP, Service Function Chaining, SNMP, Unified secure channel, Virtual Tenant Network, NETCONF module, YANG-PUSH); environment, relevant procedures and tools, etc.

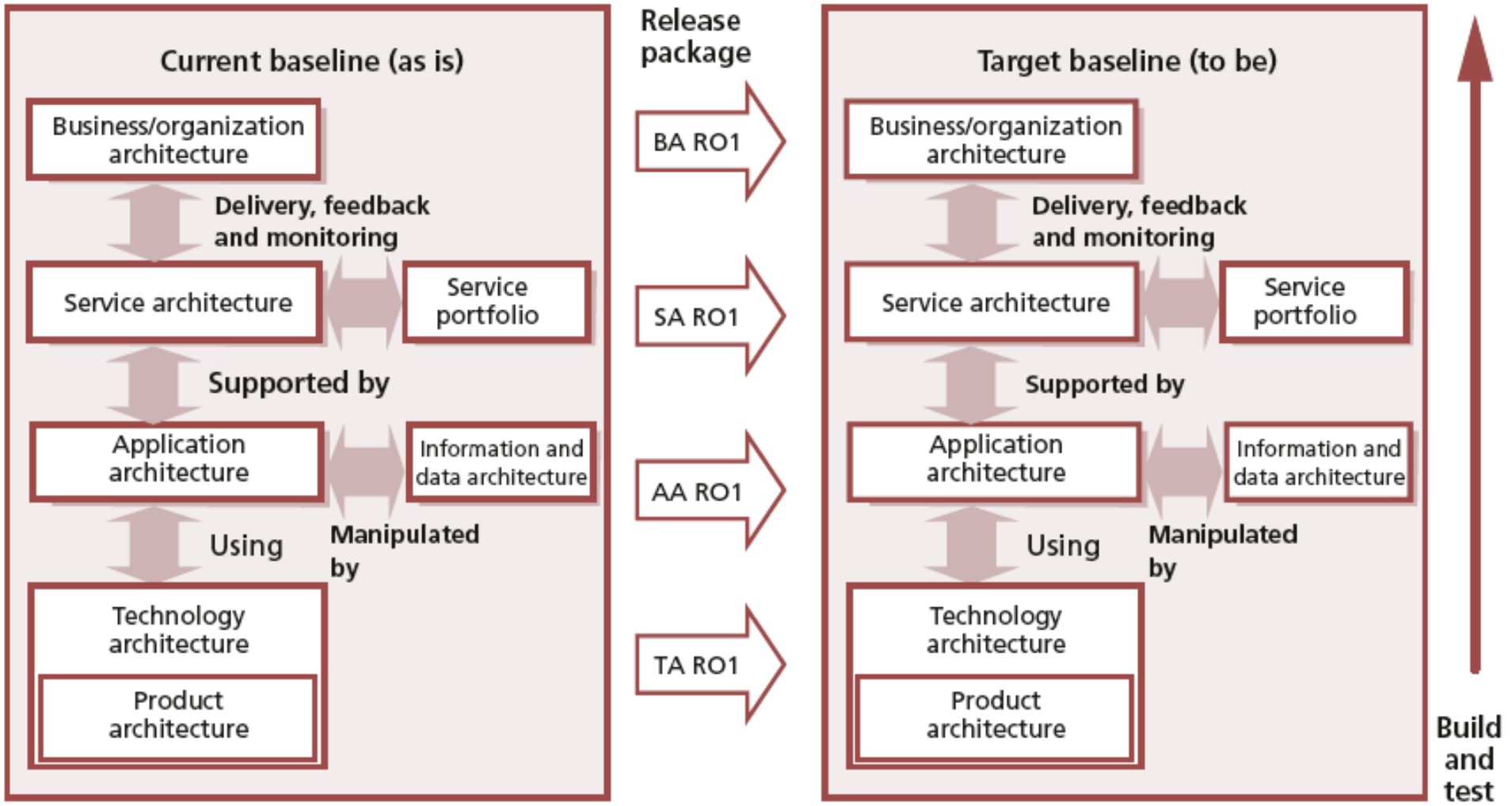
*Only NFV Infrastructure and VIM aspects listed, but components of other systems might fail as well: VNF, VNFM NFVO, OSS/BSS

Minor vs. major updates

Minor releases do not require baseline changes, but the major ones do

	Minor release (routine improvement)	**Major release (remarkable shift)**
Examples	• Routine equipment swap for a very similar alternative • Bug fixes, security updates, patches • Minor software releases like upgrade from version 2.1.1 to 2.1.2	• Key network component (for example, MSC) replacement with a new-generation alternative • Major software releases like upgrade from version 1 to version 2
Key difference	No influence on performance, functionality, service architecture, relevant processes, etc.	Major change in performance, functionality, service architecture, relevant processes, etc.
Frequency	(At least) once a quarter / half a year for majority of the components.	Rare – usually once in several years
Recurrence	Relatively similar repeatable actions	Relatively unique temporary endeavors
Governance	Through (process) standardization (and automation)	Through program / project management

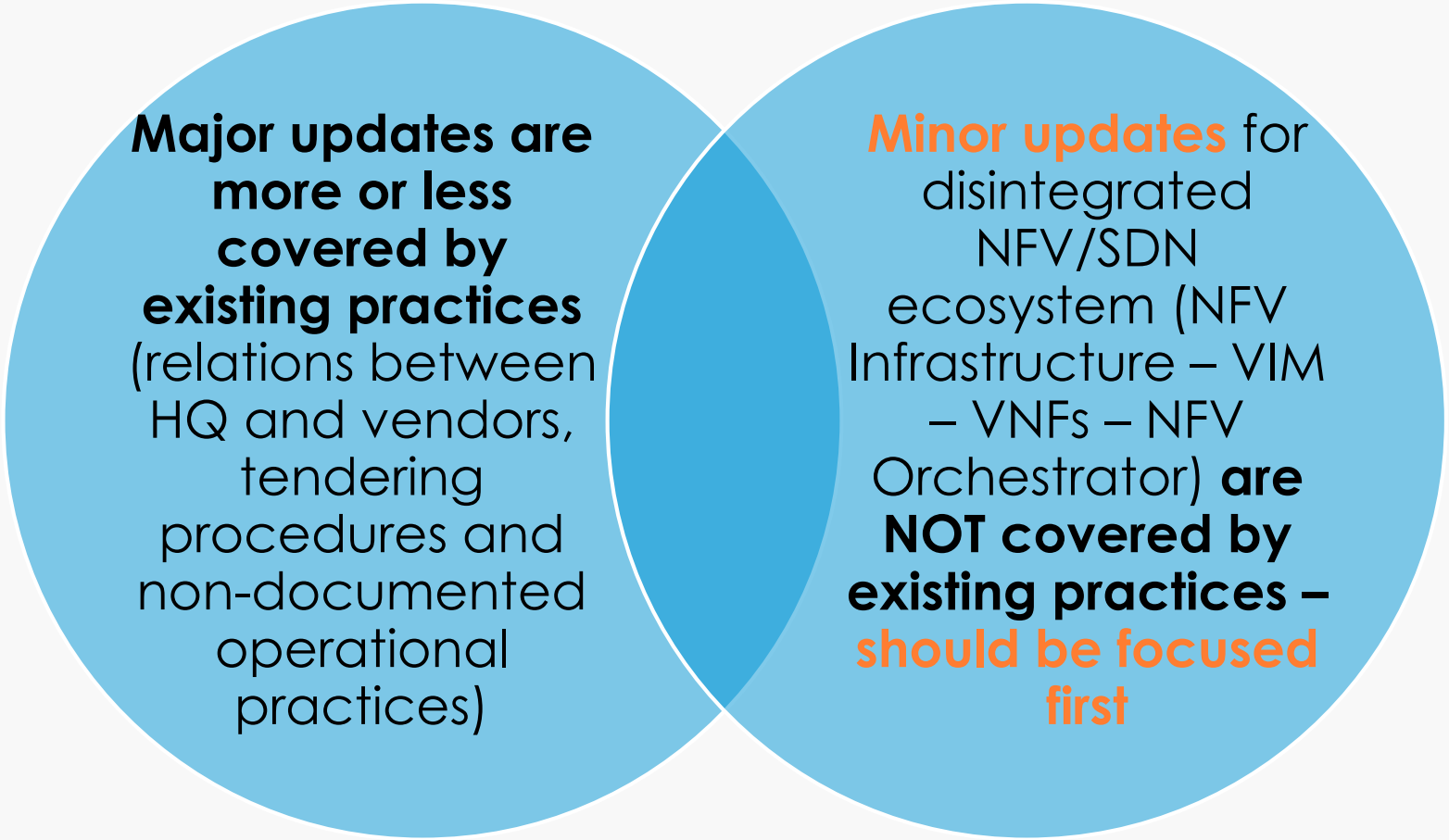
Major release = major changes



Source - ITIL® Service Transition 2011 edition

Initial scope

For release management and testing



Frequency of minor updates

A rough estimation just to get some baseline

~ **70 components** for NFV Infrastructure and VIM listed on above (these are not all components)

assume **20%** have critical updates each quarter

=

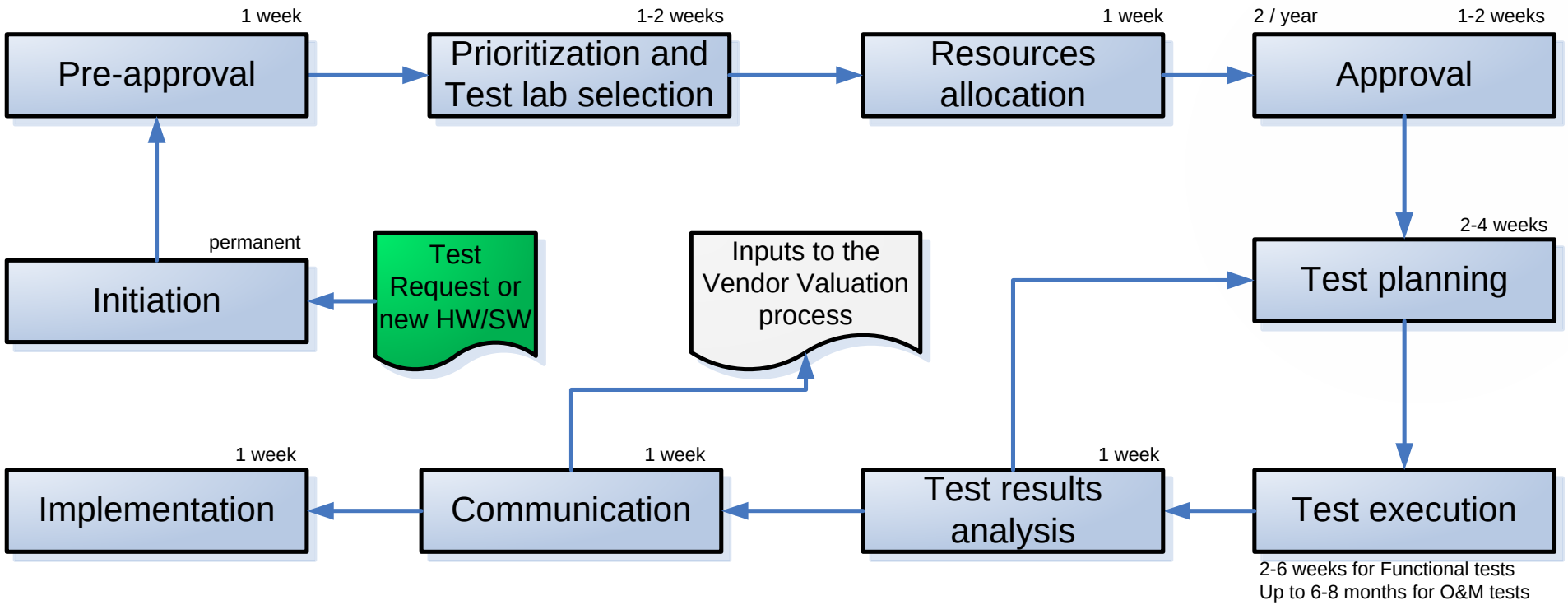
updates frequency:

- ~ 14 components to update each quarter
- ~4-5 components to update each month
- **~ 1 component to update each week**

Note: above-mentioned is only for NFVI Infrastructure and VIM, but there might be updates for other systems (VNF, VNFM, NFVO, OSS/BSS) – they might also trigger verification and certification activities

Minor update processing time

Process example - some IT equipment testing process



Duration: 11-19 weeks (3-5 months) for functional, **9-13 months for O&M tests**

How can we make it?

NFVI + VIM update at least every week, but 3-19 months for each certification / verification activity

Finalize certification / verification process for each update in a week

(or much less if we take into account VNF, VNFM, NFVO, OSS/BSS updates)

In current practices it is NOT possible

Thus

There should be a significant change in release management & testing practices

Allowing to **DO CERTIFICATION / VERIFICATION** for minor updates much faster **(IN 2-4 WEEKS)**

And

We should process updates **IN SEVERAL PARALLEL STREAMS**

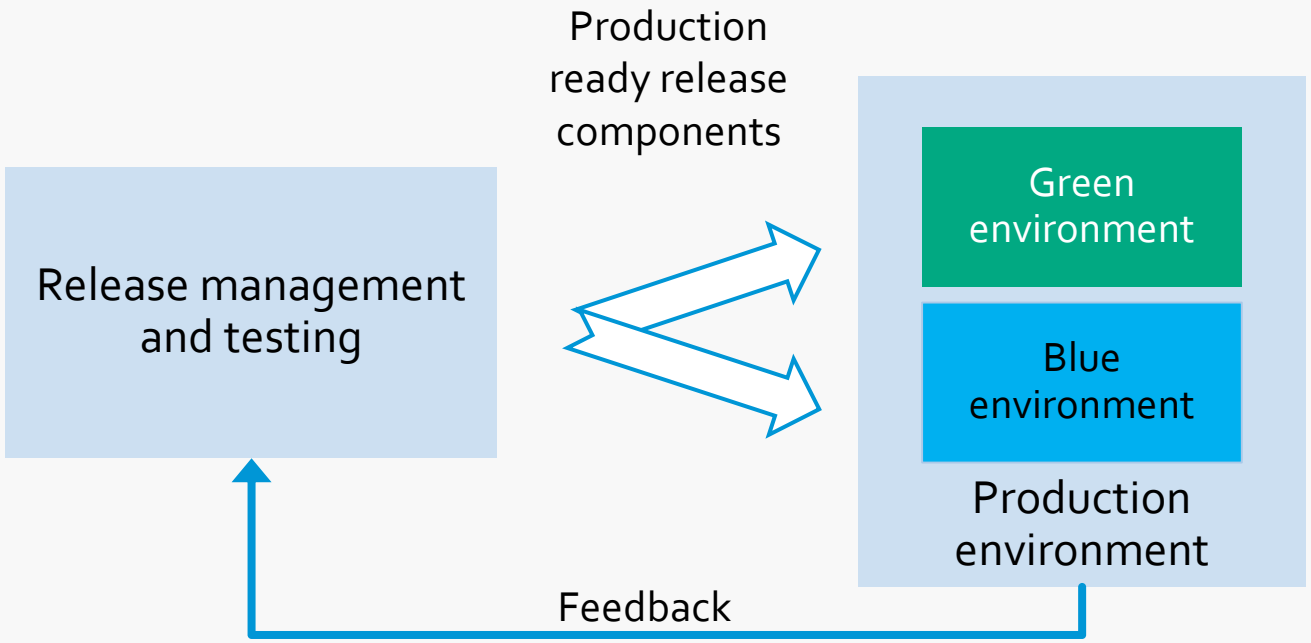
(10-20 streams and more)

Key ideas

TO HAVE IN MIND WHILE DESIGNING LIFECYCLE MANAGEMENT PRACTICES

Proposal #1: divide huge challenge into manageable chunks

Minor update cycle: plan -> build -> test -> scale up / down -> gather feedback -> repeat. Several parallel cycles



Proposal #2: automate processes

It is not viable to do manually several parallel verification / certification cycles in several days

If we approach NFV / SDN manually,

It is the same as to push a car manually

NFV and SDN are automation methodologies

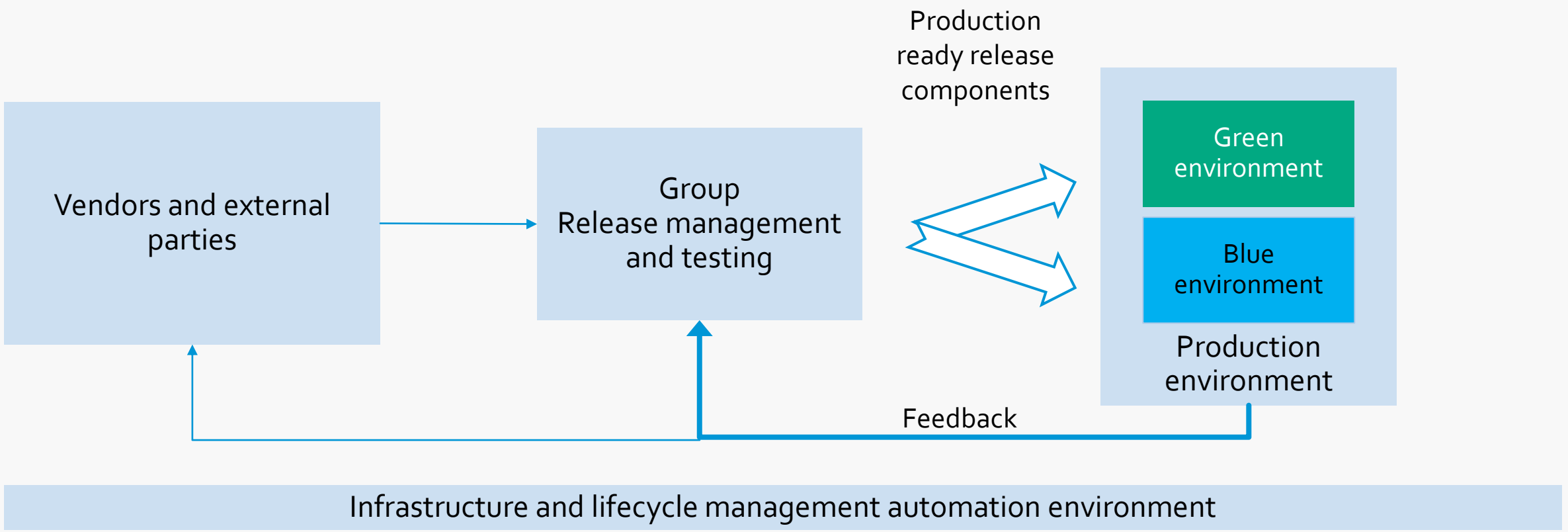
NFV/SDN-ready solutions and supporting environments are designed to be automated

Infrastructure and lifecycle management automation environment

Picture is from http://www.liveinternet.ru/users/prilukov/post114365653/

Proposal #3: integrate vendors

Group is to architect and orchestrate, vendors are to do labor-intensive tasks

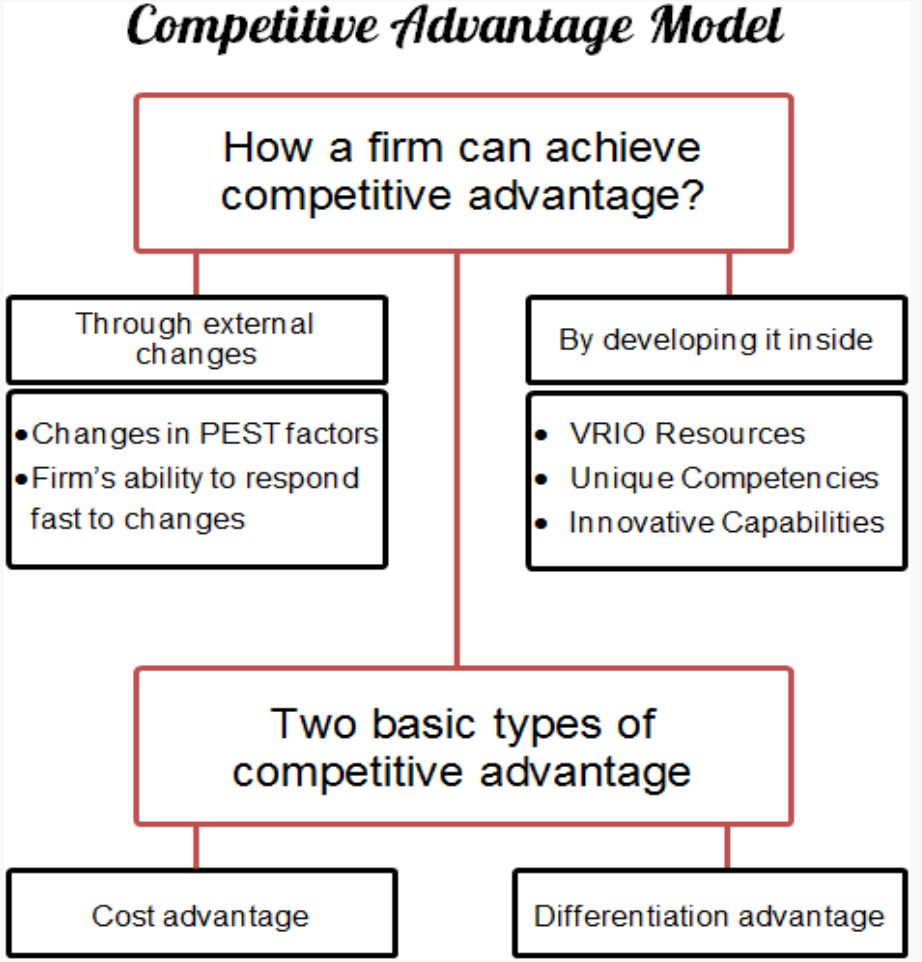


Proposal #4: retain control in Group

Outsource responsibility = outsource competitive advantage

- **Vendor's involvement** and outsourcing **shouldn't lead to loss of control** (what equals to loss of group's competitive advantage and later the business itself)*.
- It is suggested to **develop internally in group's key resources and capabilities in release management**, including automation environment. It will lead to competitive advantage (over the direct competitors) if we make it:
- With **less cost** (there should be a strategy how to do it – topic for another discussion)
- **Different**ly (using industry innovations smarter)

* If we outsource everything (NFVI + VIM + lifecycle management environment + relevant services), how are we different from the competitors who do the same?

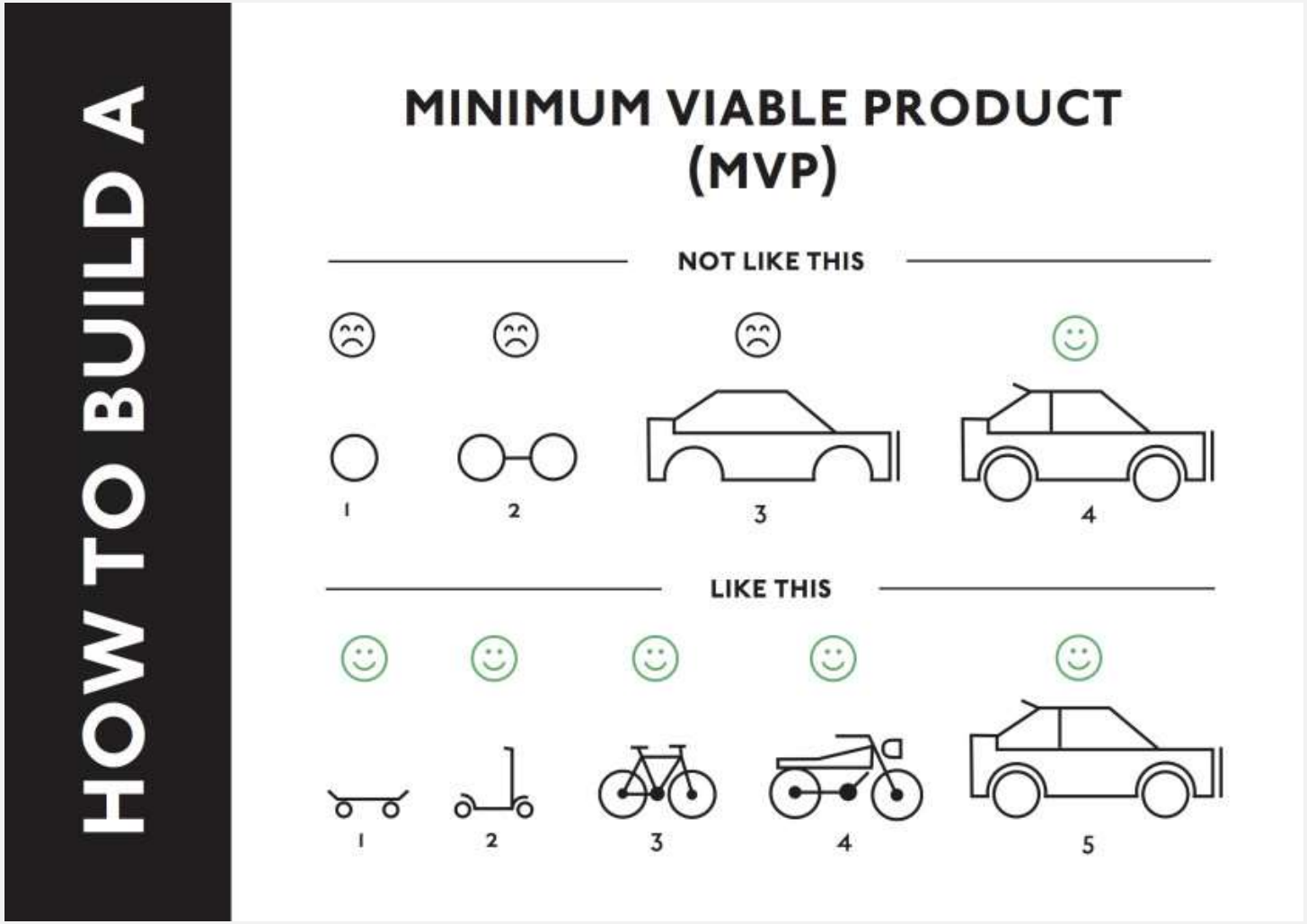


Picture is from https://www.strategicmanagementinsight.com/topics/competitive-advantage.html

Process suite: initial scope

LIFECYCLE MANAGEMENT PROCESS SUITE

“Whole car right away” or “skateboard first”?



Source - http://mw2016.museumsandtheweb.com/paper/the-agile-museum/

Big bang or phased approach

Whole "car" right away or "skateboard" first

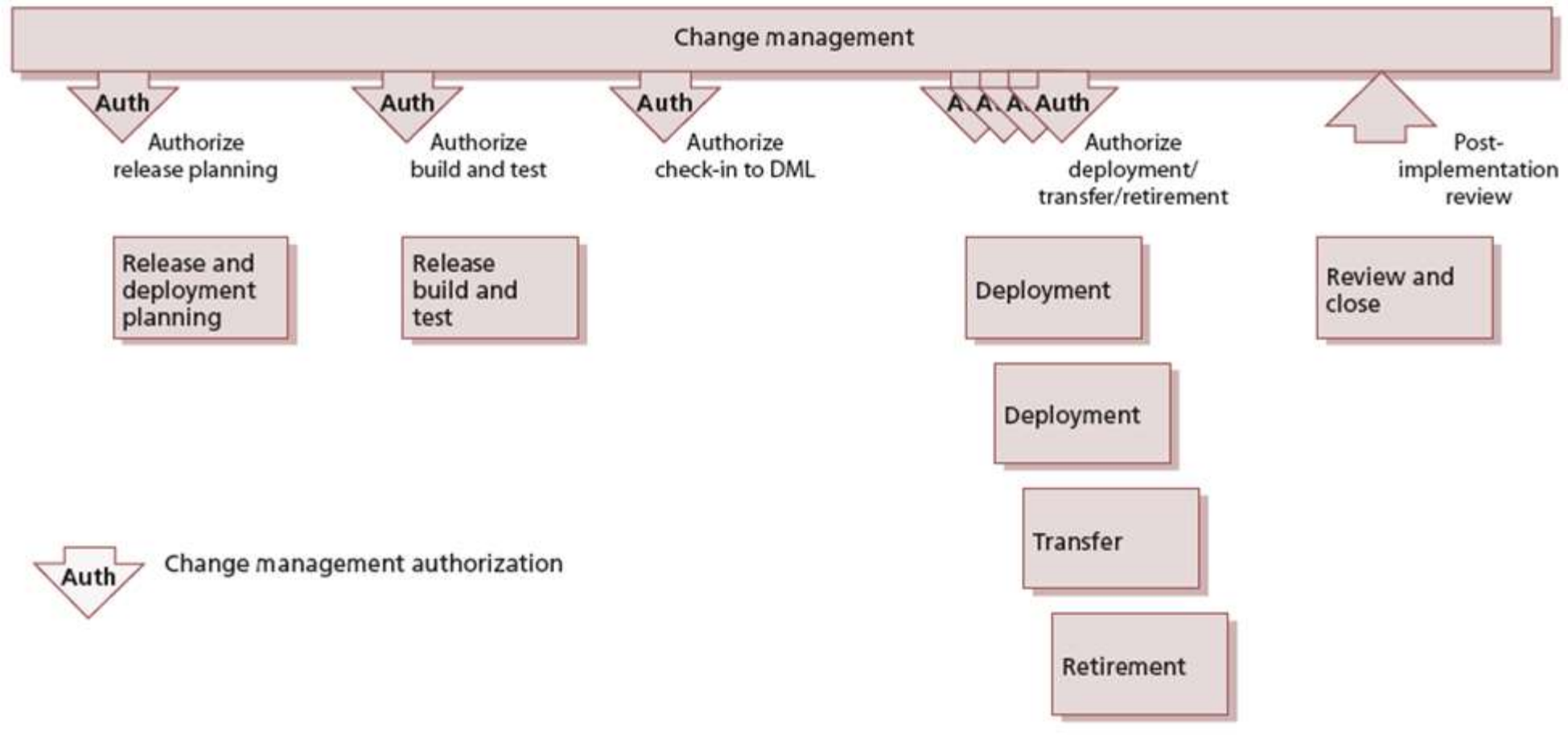
	Big bang (a "car" right away) approach	Phased (a "skateboard" first) approach
Description	• Complete change / release/ test management process suite with complex organizational transformation • Feature-rich process automation environment	• Only core processes (release planning, building, testing, service validation) with minimal change in how organization works • Basic environment for test automation to ensure that release components will work in production
Main concerns	• Very difficult to implement • Extremely expensive • Inevitable vendor lock • Difficult to disintegrate • Difficult to innovate • Unclear how to beat competitors	• Less attention (thus support) of top management • Not many accustomed to take any responsibility • Resistance to even minor changes
Main benefits	• Not much change in how organization works • Responsibility might be outsourced	• Easier to implement • Best fit for desintegration approach • Easy to innovate • Inexpensive

Release management & testing: process flow simplified

INITIAL SCOPE

Inputs: authorized changes

Changes (major and minor) processed by change management and change evaluation (external processes)

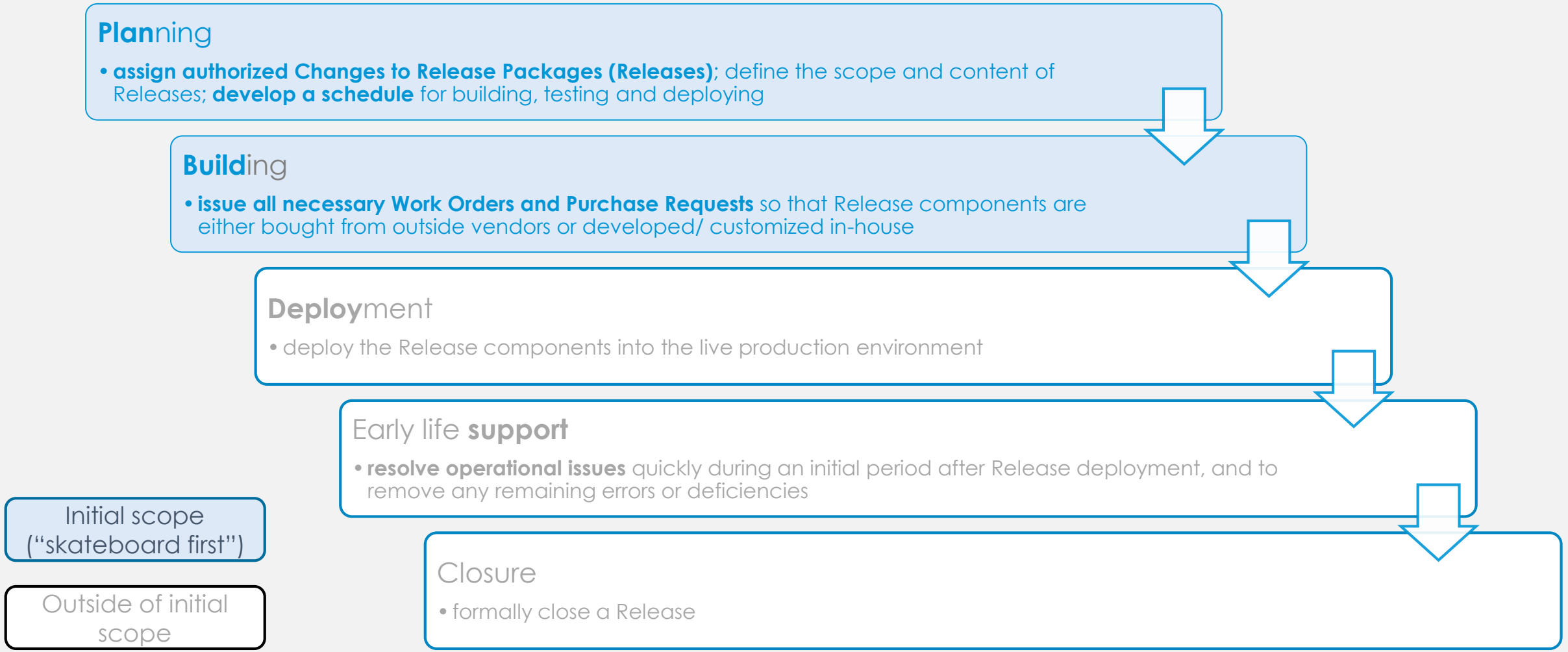


Note: change management and evaluation processes are outside of initial scope (based on currently established practices)

Source – ITIL Service Transition 2011 edition

Release & deployment management: sub-processes

ITIL-based descriptions



* Text is from http://wiki.en.it-processmaps.com/index.php/Release_and_Deployment_Management

Service validation and testing: sub-processes

ITIL-based descriptions

Test model definition

- defines the testing concept and specific test cases

Release **component acquisition**

- acquire the components of a Release and submit them to an initial assessment

Release **test**ing

- test all Release Components and all tools and mechanisms required for deployment, migration and back out

Service acceptance testing

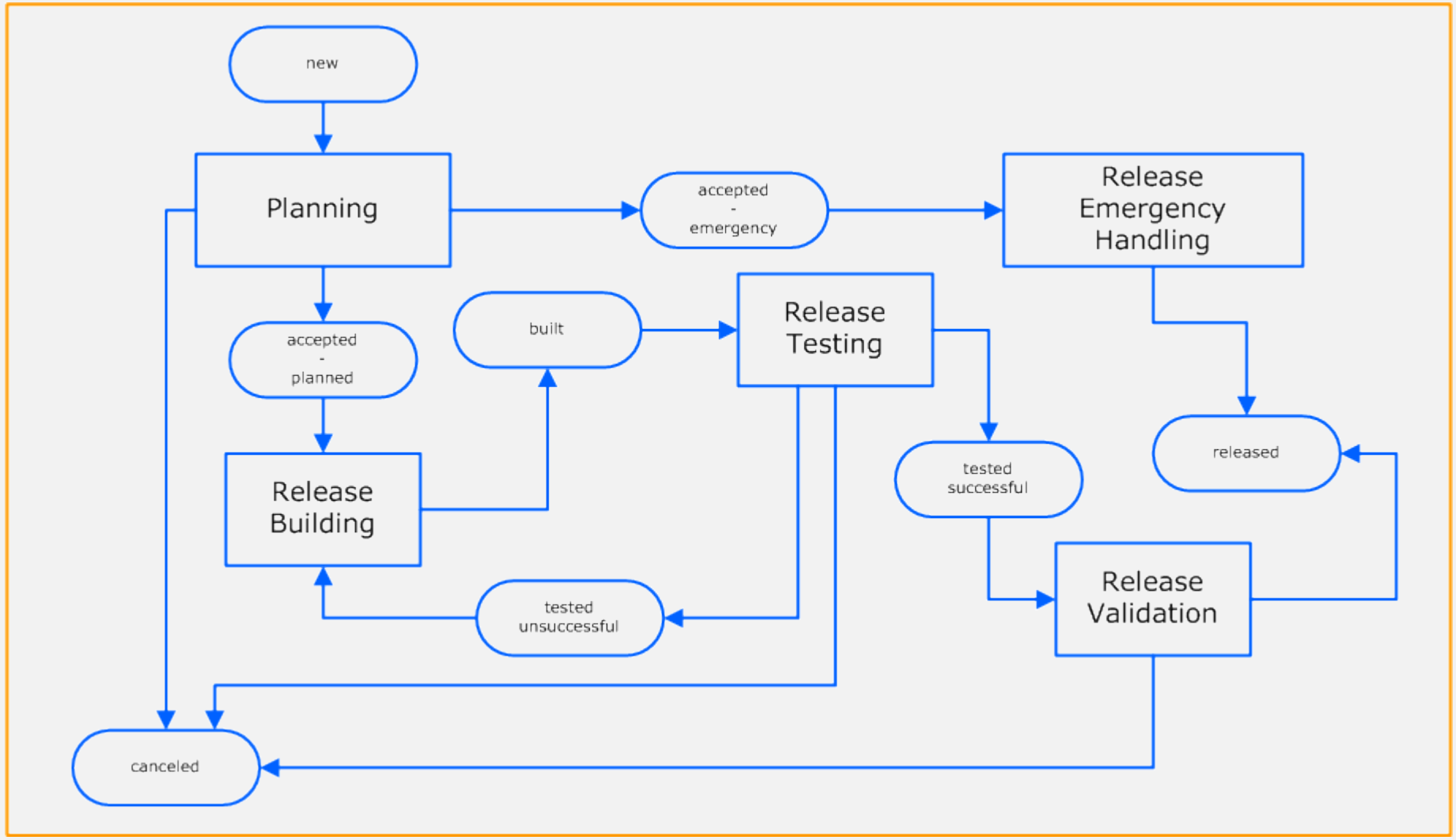
- verify if all conditions are met for the new service to be activated, and to obtain a binding consent from the customer that the new service fulfills the agreed Service Level Requirements

Initial scope ("skateboard first")

Outside of initial scope

* Text is from http://wiki.en.it-processmaps.com/index.php/Service_Validation_and_Testing

End-to-end process flow: first phase (simplified)



Source - https://www.mitsm.de/release-management-en

Backup slides

IDEAS FROM 3RD-PARTY SOURCES

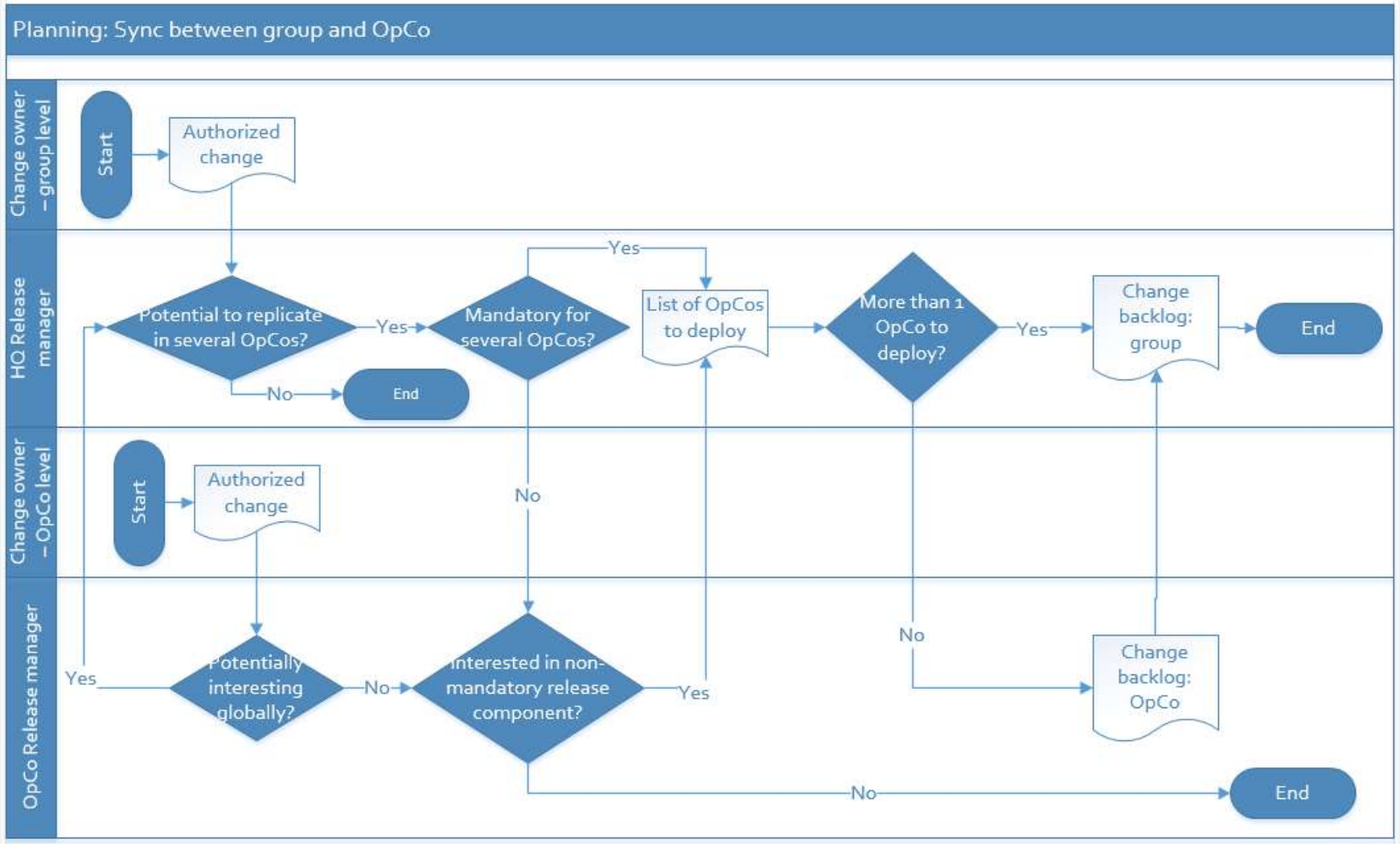
AND VARIOUS ARTEFACTS REDESIGNED LATELY

Release management & testing: process flow details

LIFECYCLE MANAGEMENT PROCESS FLOW STEP BY STEP

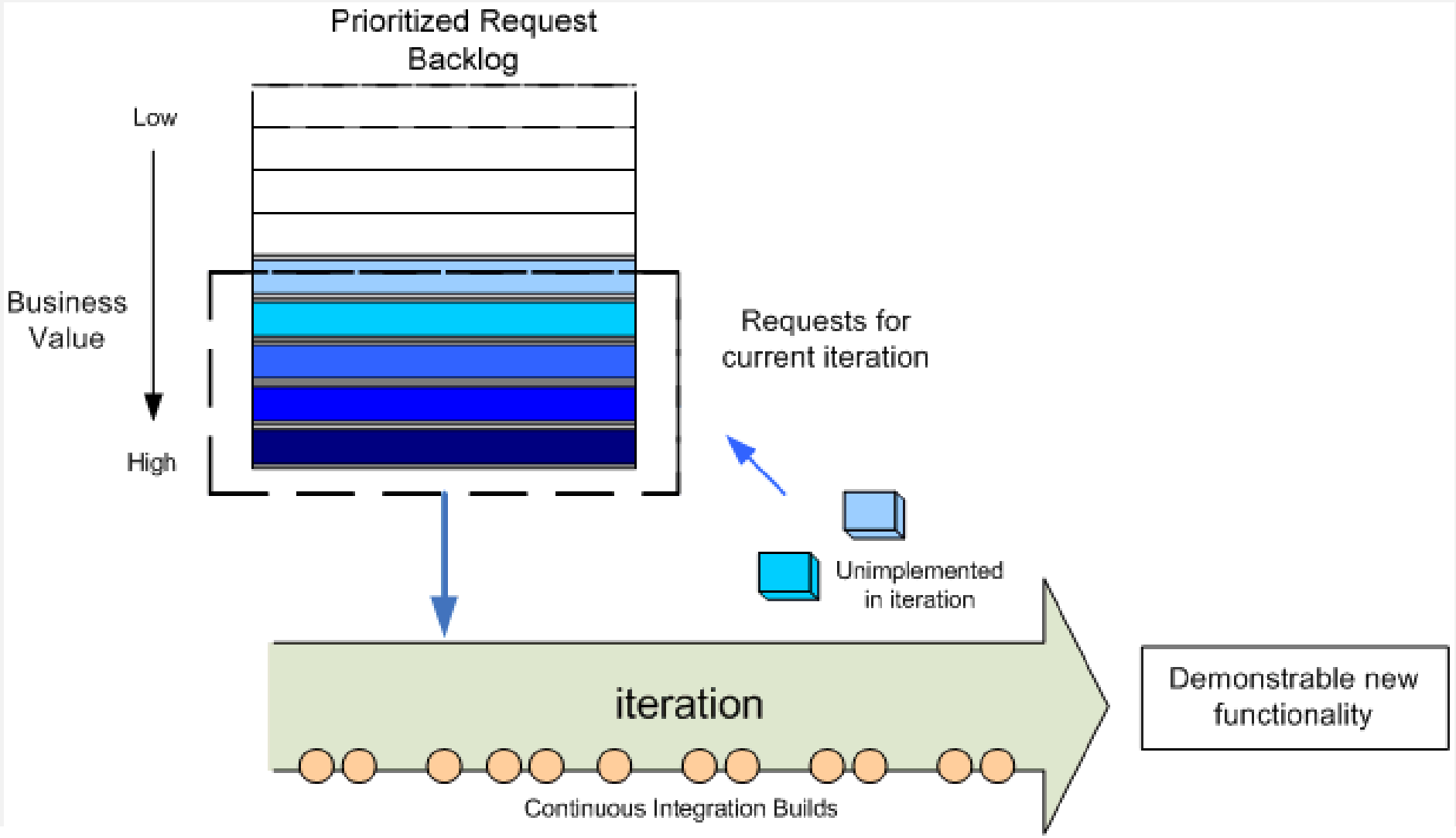
Planning: step 1: create change backlog

Sync between group and OpCos



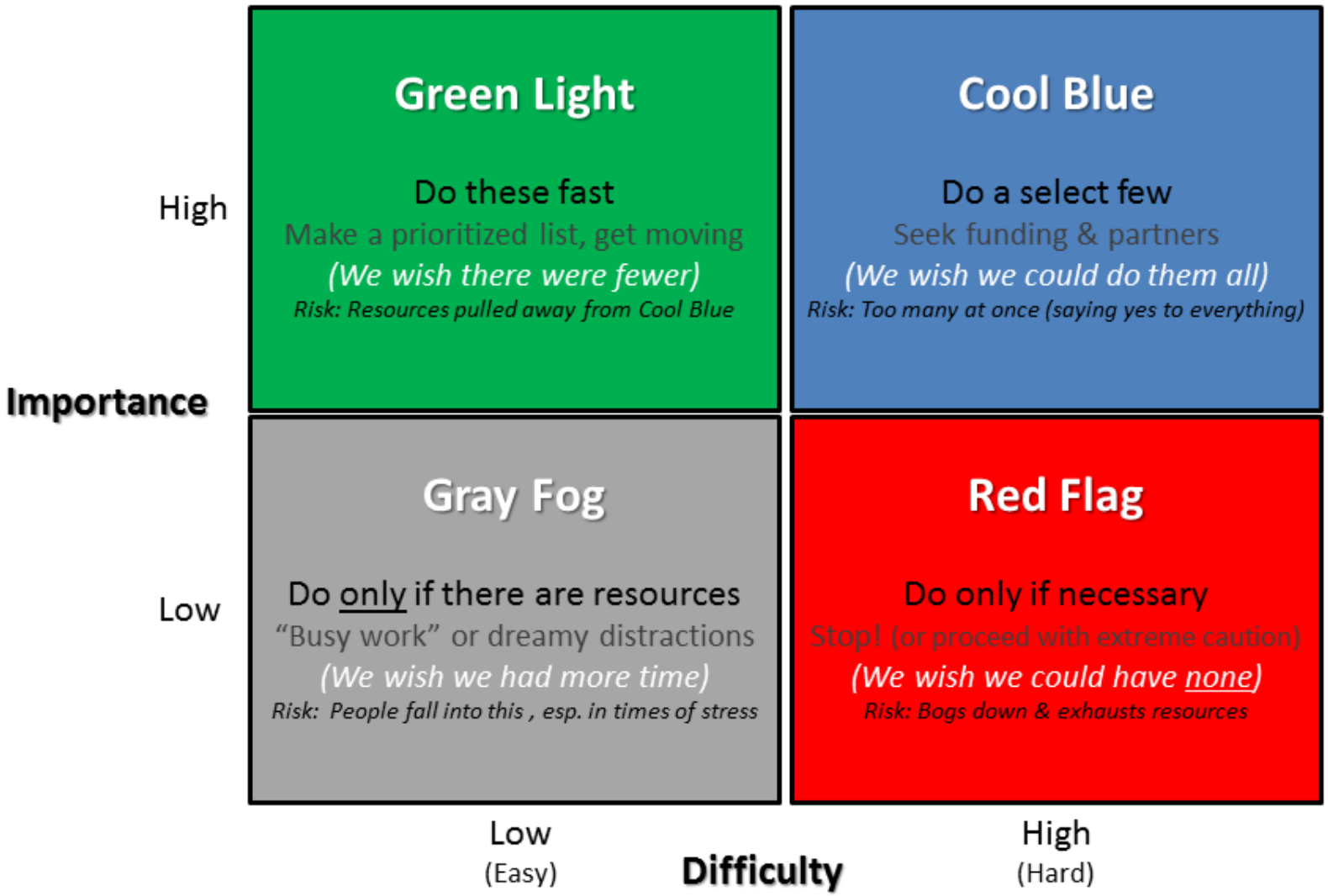
Planning: step 2: prioritize planned changes in backlog

Choose what to implement first (what we can process in next timeframe with available resources)



Source - http://www.ibm.com/developerworks/rational/library/jun06/lee/

Making Decisions

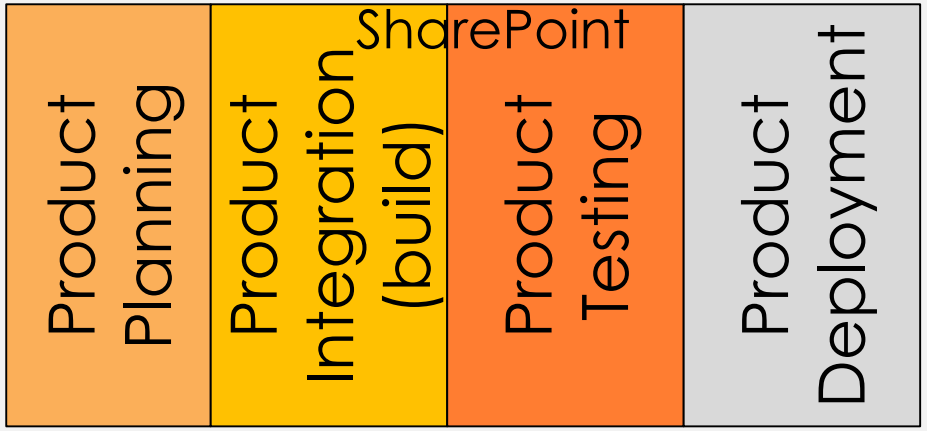


Source - http://mw2016.museumsandtheweb.com/paper/the-agile-museum/

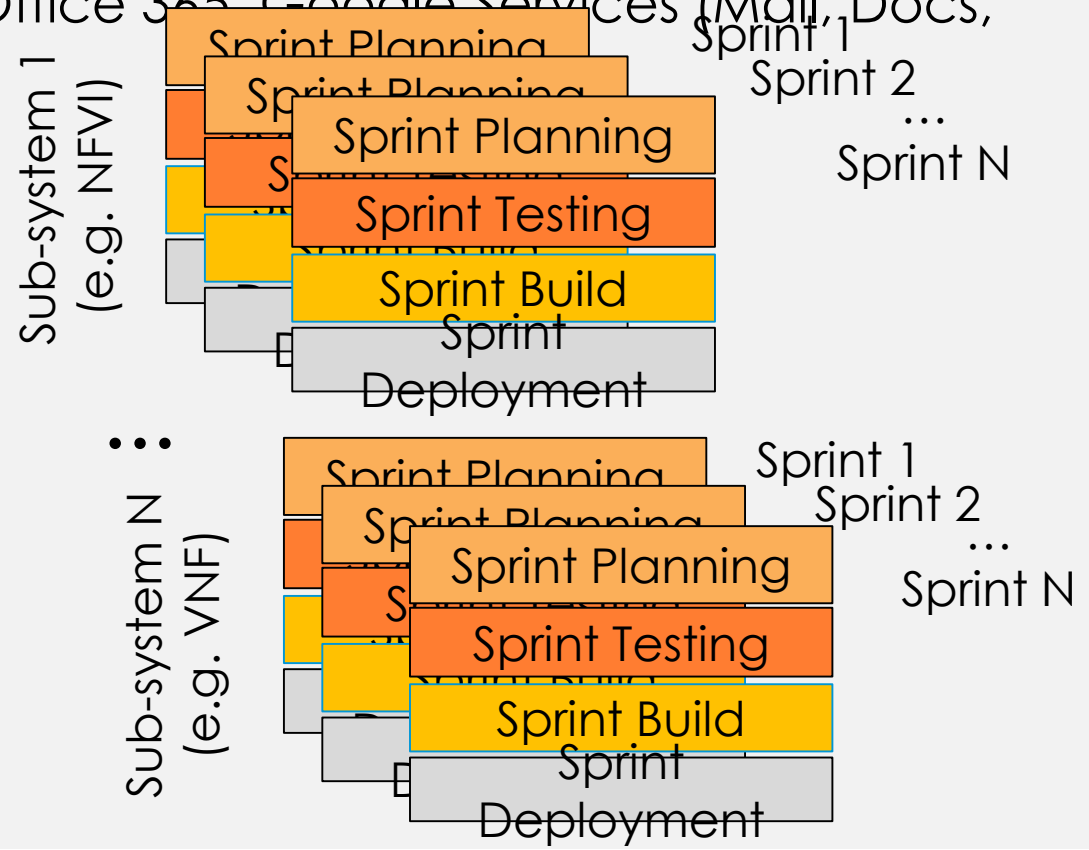
Planning: step 3: assign changes to small change sets (sprints)

Each update (or set of closely related updates) assigned to separate sprint (ideal case with infinite resources)

Traditional release cycle – one huge release in several years. Examples: Microsoft Office for Desktops, Exchange, SharePoint



Agile release cycle – continuous upgrades of platform components. Examples: Microsoft Office 365, Google Services (Mail, Docs,



What should a sprint cover?

A single certification / verification scenario from some group SLA*

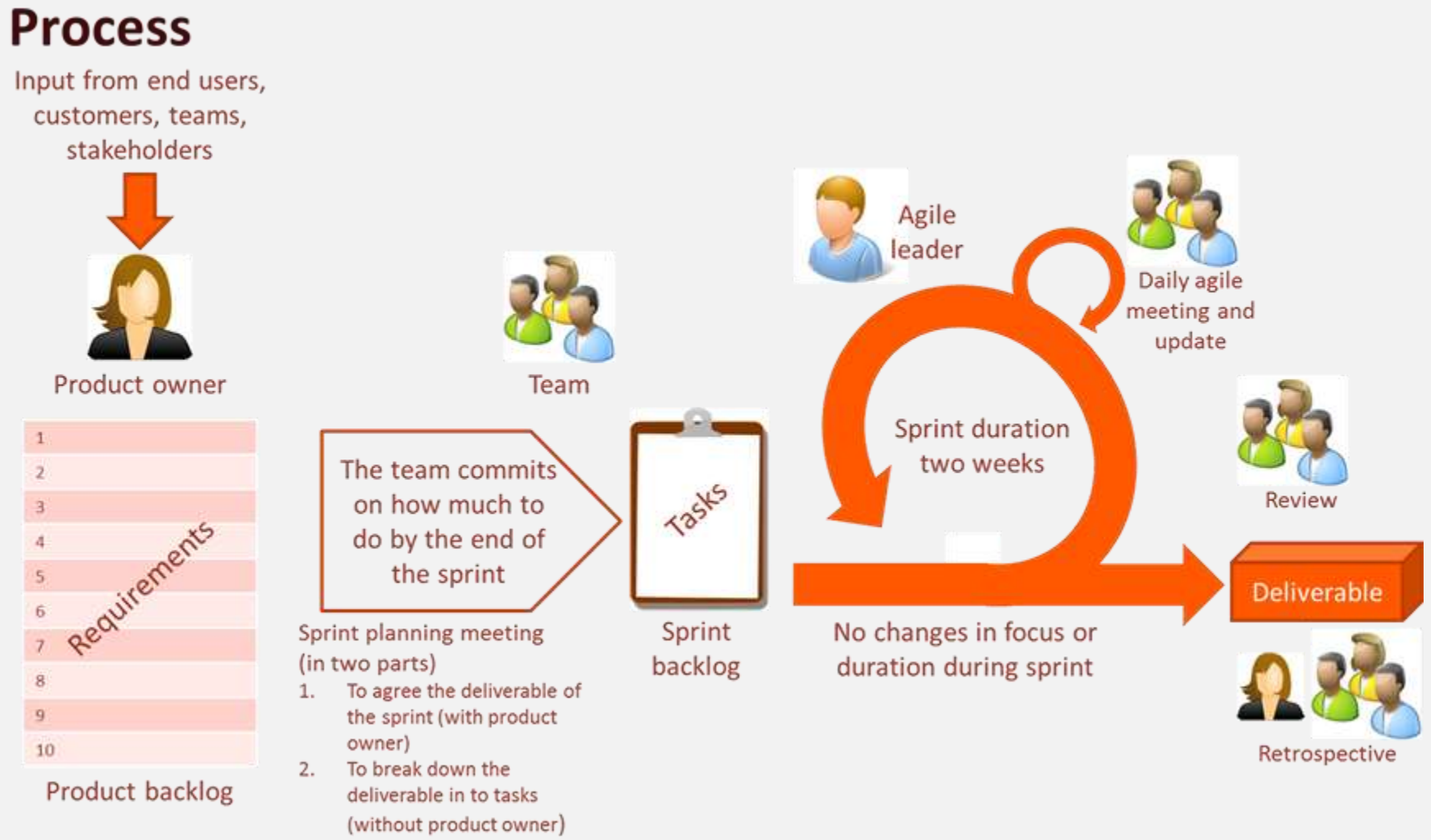
	Buyer	NFVI Supplier	VNF Supplier
New customized by vendor OpenStack release or **updates (test subject) available for testing**	I	R	I
Resources allocation	A	R	A
To provide detailed information regarding new functionality (Product description, Manuals, List of Bugs (if exists)	I	R	C
To provide current **trafic model** (common for overall Virtualized solution)	R	A	A
To provide **SW environment for Lab purposes (NFVI and VNFs)**	I	R	A
To provide **snapshots from the production** Virtualized solution	R	A	A
To provide a **traffic generator(s)** (for performance testing; for production environment emulation); for testing program execution	I	C	R
Provide **Measurement tools** for performance and load testing	I	C	R
Developing and harmonization of the **testing methodology and acceptance (success) criteria**	A	R	A
Developing of expected **KPIs**	R	A	A
Tests execution	A	R	A
Test Report submission and provide key recommendations	A	R	A
Pilot launching/babysitting during stability period and KPI insurance	A	R	A

* - Scenario 5: Certification/verification of Improvements of overall NFVI solution

R – Responsible
A – Accountable
C – Consulted
I – Informed

Building: step 1: conduct each sprint individually

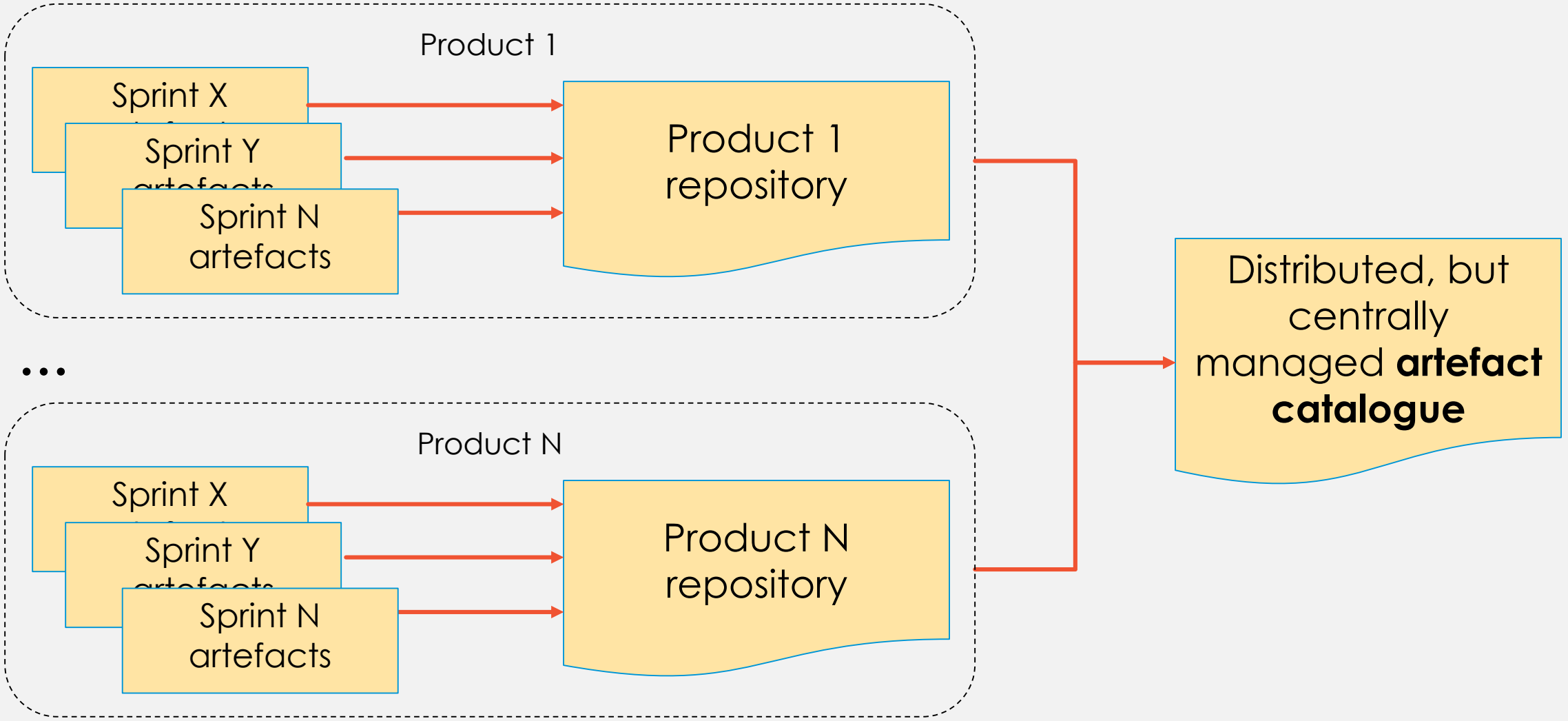
2-4 weeks for each sprint; several sprints in parallel



Source - http://www.redquadrant.com/news-and-events/news/redquadrant-certification-local-government-agile-practitioner/

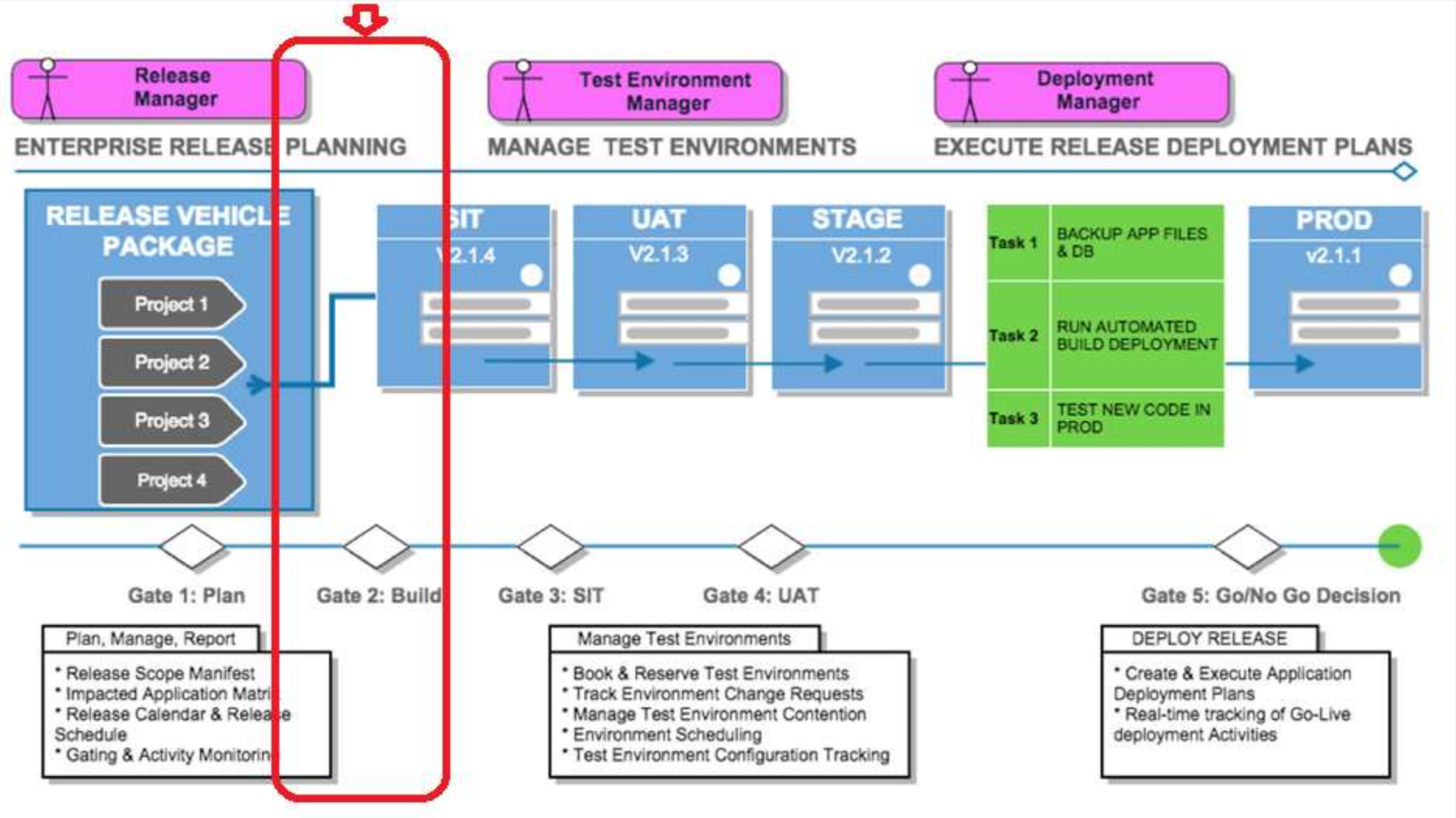
Building: step 2: populate repositories & catalogues

With sprints' outcomes



Building: step 3: integrate artefacts

From repositories and catalogues into release package

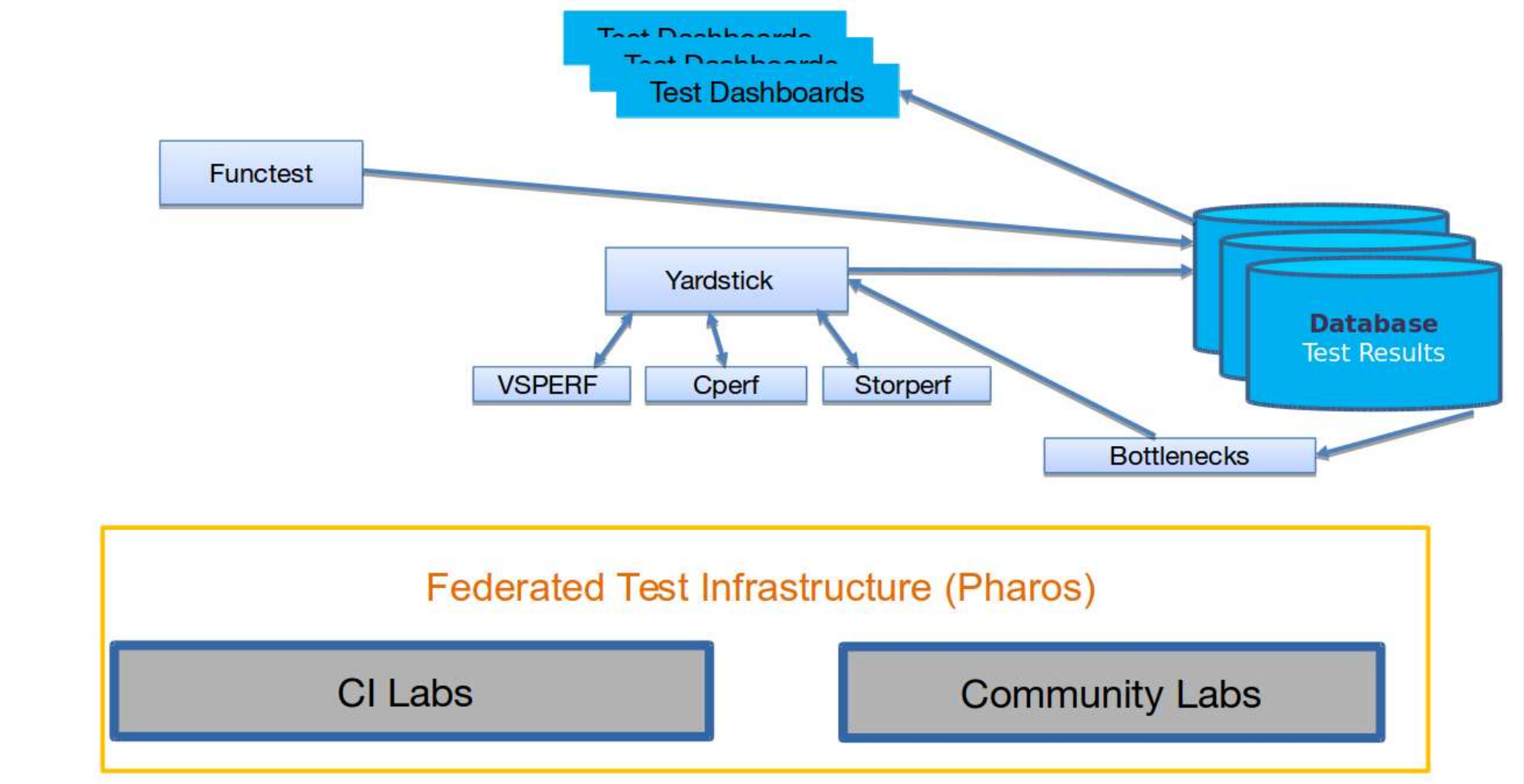


SIT – System Integration Testing
UAT – User Acceptance Testing

Source - http://www.plutora.com/release-management-software/

Testing: step zero: modify test model (if required)

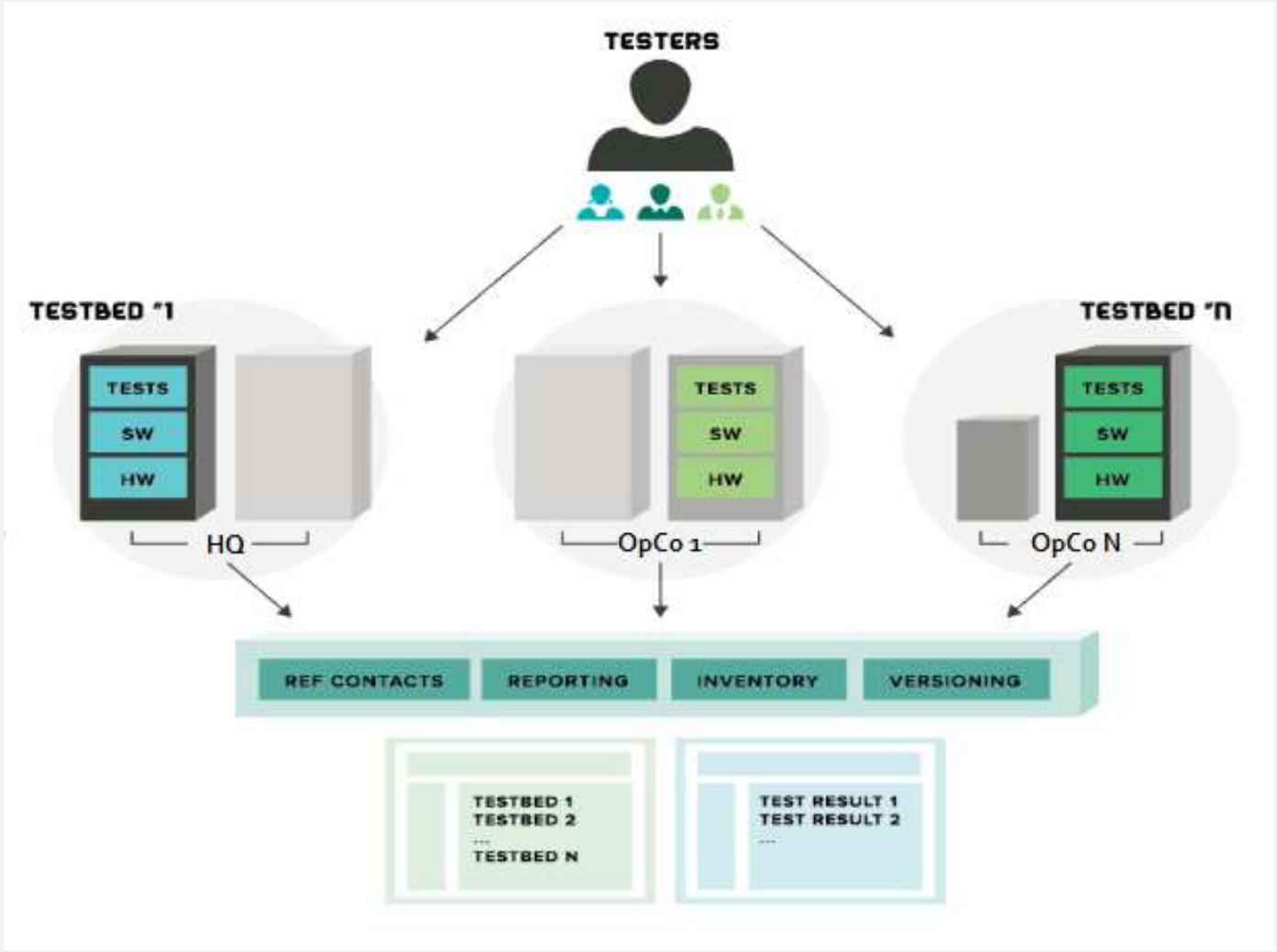
Test model should be automated - leverage industry best practices (OPNFV, RefStack and alike)



Source - OPNFV Test Ecosystem

Testing: step 1: conduct automated tests

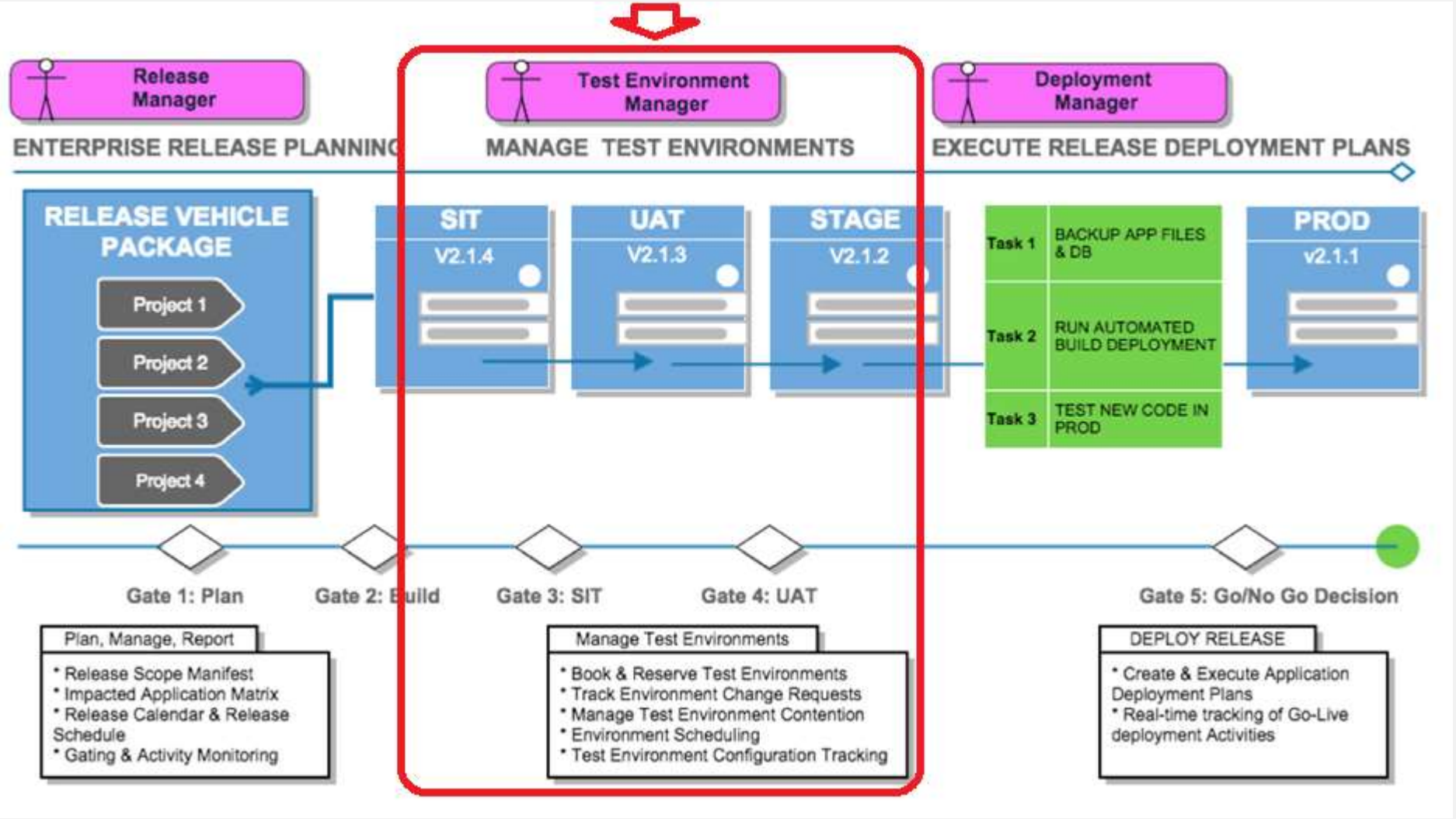
In distributed environment



Original picture is from https://www.opnfv.org/developers/pharos

Testing: step 1: conduct tests

In distributed automated environment



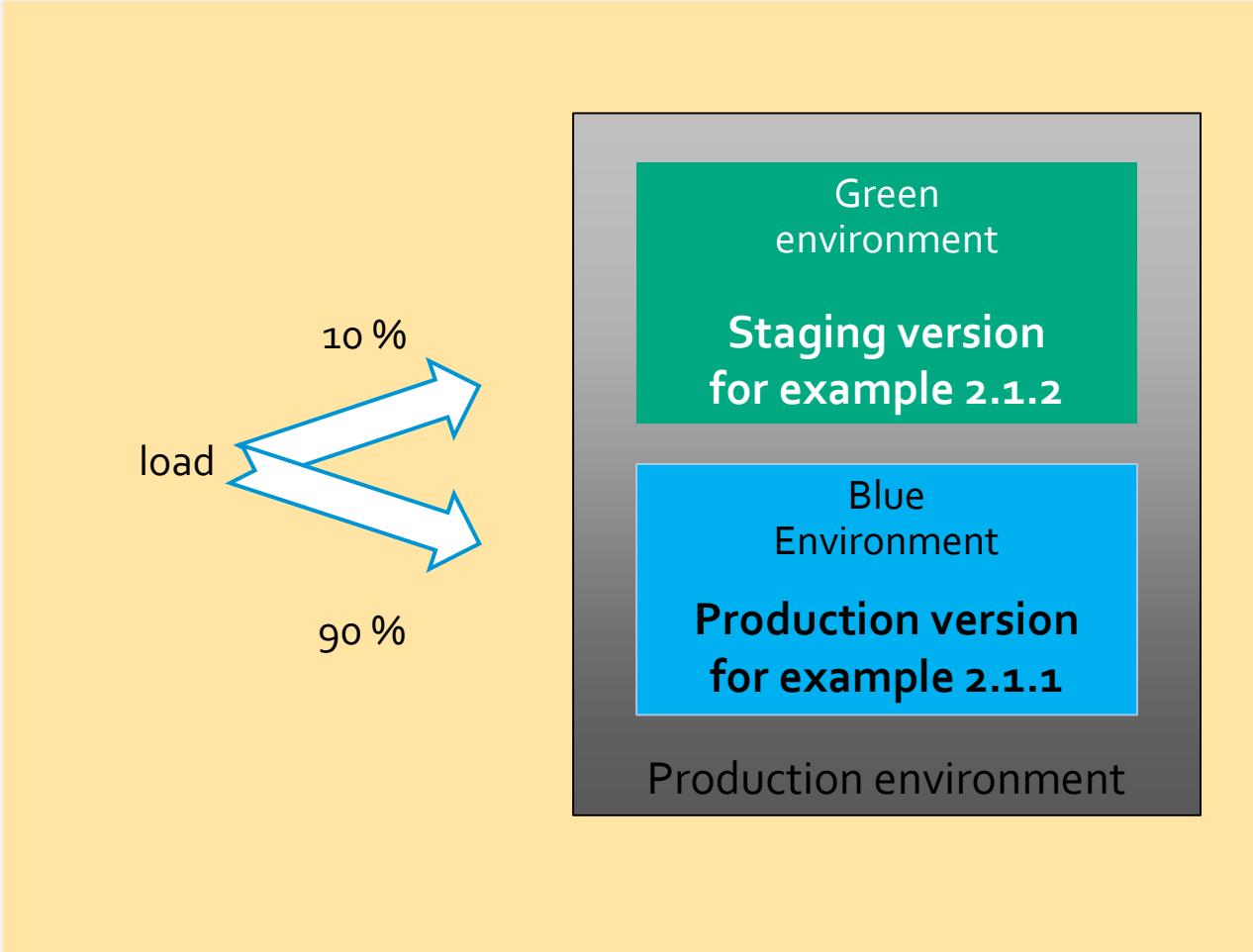
SIT – System Integration Testing
UAT – User Acceptance Testing

Source - http://www.plutora.com/release-management-software/

Deployment

Gradual new version scale up with synchronous scale down of the old one

- Start with current production (in this case it is **blue**) environment – carrying 100% load
- Deploy an update to a parallel (staging - **green**) environment
- Do the final tests in green environment
- Redirect small portion of the load (for example, 10%) to the staging environment
- Monitor the green environment
- **Continue redirecting load from the blue environment to green**
- If needed, roll back to blue
- **When 100% of the load migrated to green environment, it becomes production and blue becomes staging**



Roles and responsibilities

- For release management and testing

Responsibilities of contractor and buyer

from some standard group SLA

- Software support
  - The Contractor shall be responsible for implementation of Software Updates, patches, fixes, corrections in the one respective site of the Buyer
- Software Update implementation
  - Buyer may request the Contractor to provide Software update, fix, patch correction implementation. Depending on Buyer's demand, Software Update implementation may be ordered remote via Remote connection or On-Site.

Roles and responsibilities

For release management and testing

Process	Inputs	Outputs	Release manager		Project manager		Test manager		Service owner (OpCo)	Service user (OpCo)
			HQ	OpCo	HQ	OpCo	HQ	OpCo		
Planning: HQ	Changes	Release plan	AR	R	R					
Planning: OpCo	Changes	Release plan	R	AR		R				
Building: HQ	Artefacts	Integrated HW/SW	AR	R	R					
Building: OpCo	Artefacts	Integrated HW/SW	R	AR		R				
Test model definition: HQ	Requirements	Test model					AR	R		
Test model definition: OpCo	Requirements	Test model					R	AR		
Testing: HQ	Test model, Integrated HW/SW	Released component					AR	R		R
Testing: OpCo	Test model, Integrated HW/SW	Released component					R	AR		R
Deployment, early life support (outside of initial scope)			AR						R	

A: Accountable according to the RACI Model: Those who are ultimately accountable for the correct and thorough completion of the ITIL Release & Deployment process.

R: Responsible according to the RACI Model: Those who do the work to achieve a task within Release Management.

Main roles and responsibilities

For release management and testing

Process	Inputs	Outputs	Release manager		Project manager		Test manager		Service owner (OpCo)	Service user (OpCo)
			HQ	OpCo	HQ	OpCo	HQ	OpCo		
Planning: HQ	Changes	Release unit backlogs, release plan	AR	R	R					
Planning: OpCo	Changes	Release unit backlogs, release plan	R	AR		R				
Release unit processing: HQ	Release unit backlog	Release unit	AR	R	R					
Release unit processing: OpCo	Release unit backlog	Release unit	R	AR		R				
Integration and testing: HQ	Release units	Release package	AR	R			R			
Integration and testing: OpCo	Release units	Release package	R	AR				R		
Deployment, early life support (outside of initial scope)				AR					R	

A: Accountable according to the RACI Model: Those who are ultimately accountable for the correct and thorough completion of the ITIL Release & Deployment process.

R: Responsible according to the RACI Model: Those who do the work to achieve a task within Release Management.

Release and deployment management: reference roles and responsibilities

Release Manager - Process Owner

- The Release Manager is responsible for planning and controlling the movement of Releases to test and live environments. His primary objective is to ensure that the integrity of the live environment is protected and that the correct components are released.

Responsibility Matrix: ITIL Release Management				
ITIL Role / Sub-Process	**Release Manager**	**Project Manager[3]**	**Service Owner[3]**	**Other roles involved**
Release Management Support	A[1]R[2]	-	-	-
Release Planning	AR	R	-	-
Release Build	AR	-	-	-
Release Deployment	AR	-	R	R[4]
Early Life Support	AR	-	R	R[5]
Release Closure	AR	-	-	-

Remarks

[1] *A: Accountable* according to the RACI Model: Those who are ultimately accountable for the correct and thorough completion of the ITIL Release & Deployment process.

[2] *R: Responsible* according to the RACI Model: Those who do the work to achieve a task within Release Management.

[3] see → Role descriptions

[4] Process Owner, IT Operator, Facilities Manager (and others, as appropriate)

[5] ... and others, as appropriate.

Source - http://wiki.en.it-processmaps.com/index.php/Release_and_Deployment_Management

Testing: reference roles and responsibilities

Test Manager - Process Owner

- The Test Manager ensures that deployed Releases and the resulting services meet customer expectations, and verifies that IT operations is able to support the new service.

Service User

- A person who uses one or several IT services on a day-to-day basis. Service Users are distinct from Customers, as some Customers do not use IT services directly.

Responsibility Matrix: ITIL Service Validation & Testing							
ITIL Role / Sub-Process	Test Manager	Information Security Manager[3]	Compliance Manager[3]	Service User	IT Operator[3]	Customer[3]	Other roles involved
Test Model Definition	A[1]R[2]	-	-	-	-	-	-
Release Component Acquisition	AR	-	-	-	-	-	-
Release Test	AR	R	R	R	R	-	-
Service Acceptance Testing	AR	-	-	R	R	R	R[4]

Remarks

[1] *A: Accountable* according to the RACI Model: Those who are ultimately accountable for the correct and thorough completion of the Service Validation and Testing process.

[2] *R: Responsible* according to the RACI Model: Those who do the work to achieve a task within Service Validation.

[3] see → Role descriptions...

[4] Service Level Manager, Release Manager and Service Owner. → Role descriptions...

Source - http://wiki.en.it-processmaps.com/index.php/Release_and_Deployment_Management

Changes required

Changes required in HQ

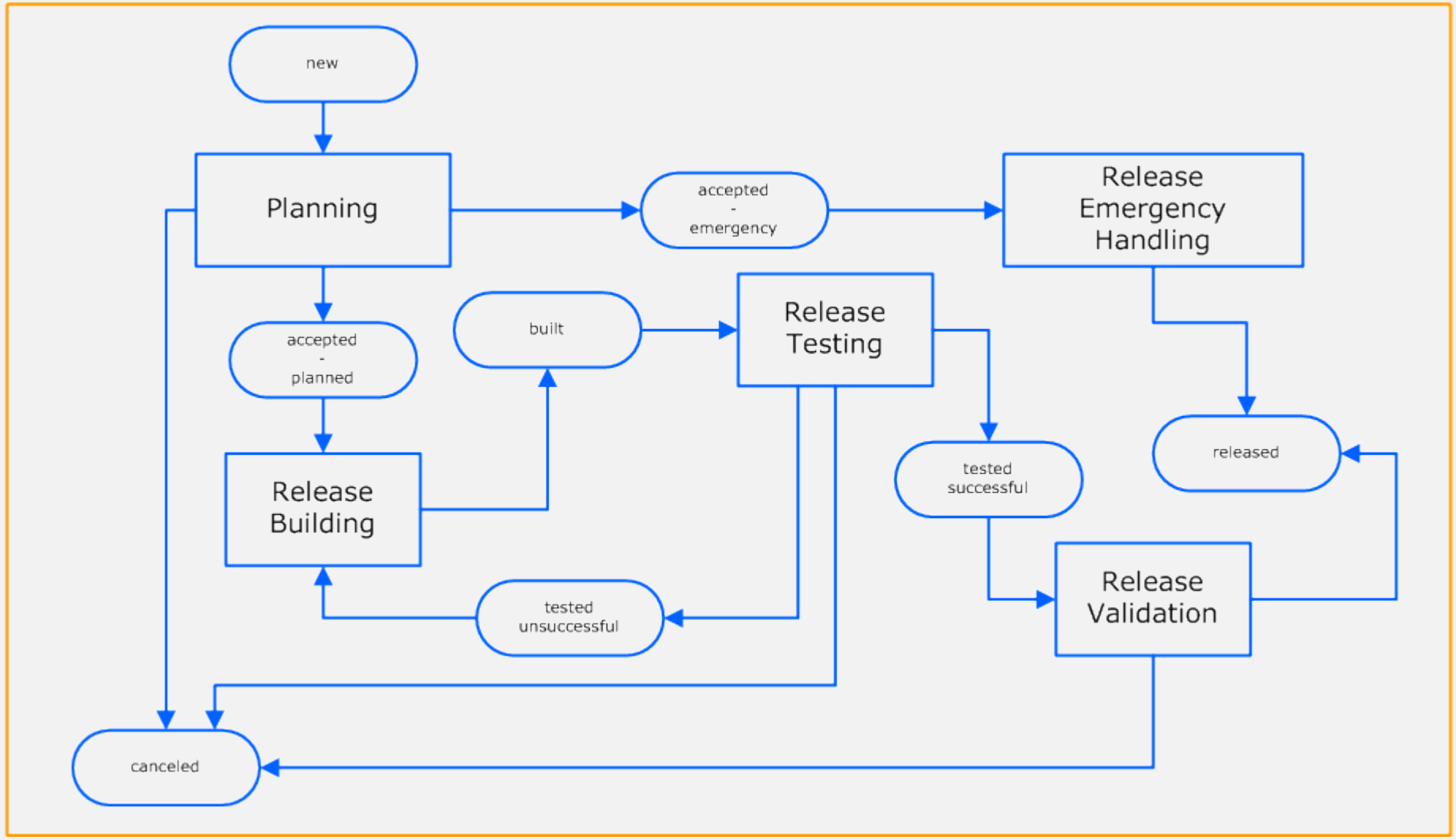
- Testing environment
- Somebody responsible for new roles
  - Release manager
  - Project manager (release management and testing)
  - Test manager
- If the release management and testing activities are processed by current employees, they should be refocused / hired / trained

Changes required in OpCos

- Testing environment
- Somebody responsible for new roles
  - Release manager
  - Project manager (release management and testing)
  - Test manager
- If the release management and testing activities are processed by current employees, they should be refocused / hired / trained

Summary

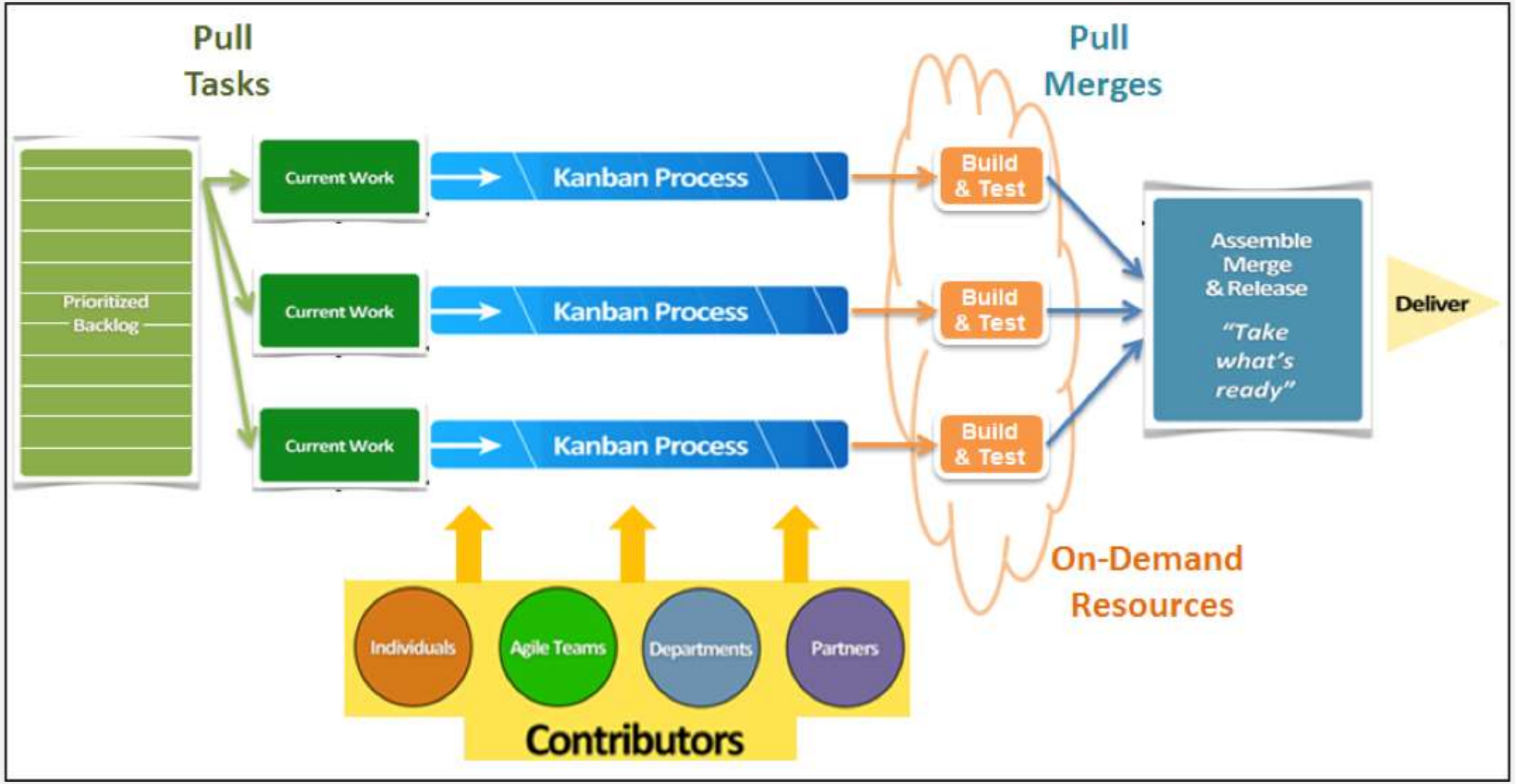
Single sprint end-to-end flow



Source - https://www.mitsm.de/release-management-en

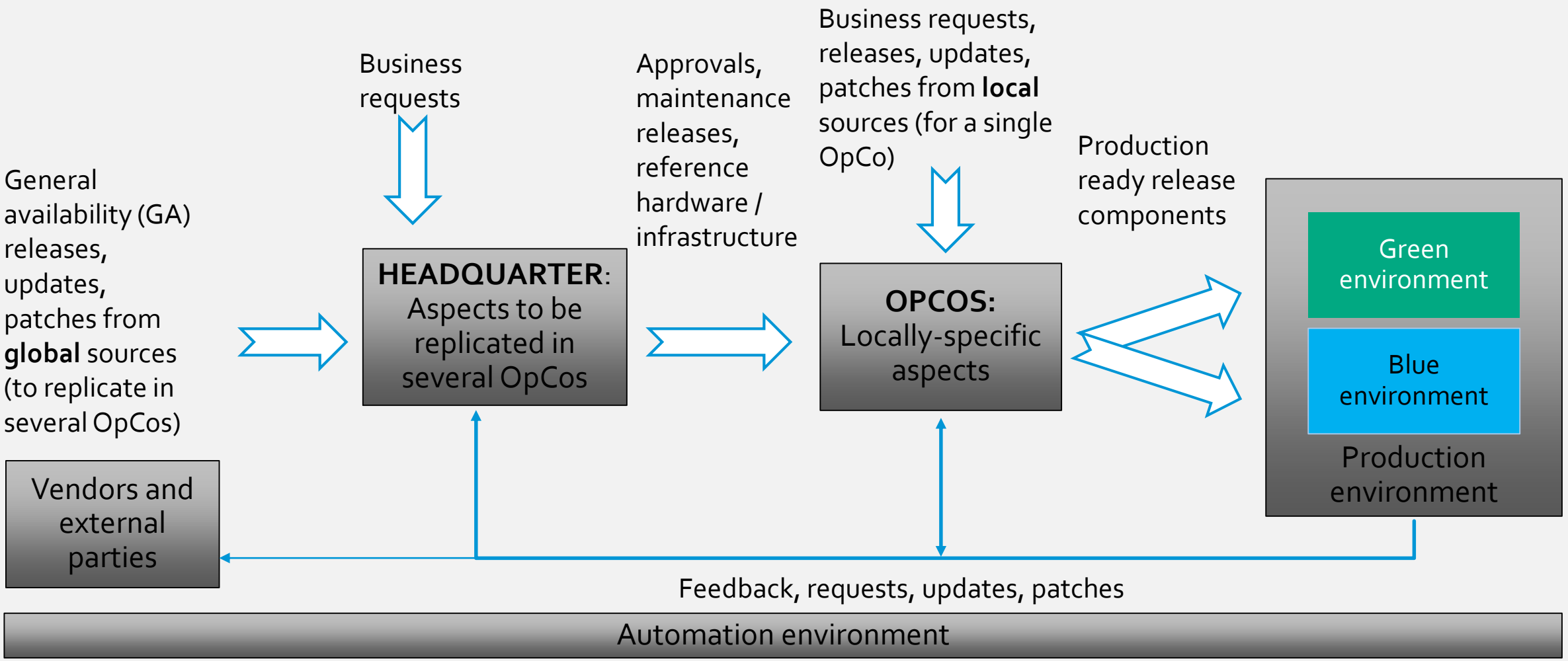
Parallel sprints

ПРЕДСТАВЛЕНО СХЕМАТИЧНО для первоначального обсуждения – БУДЕТ ПЕРЕРИСОВАНО



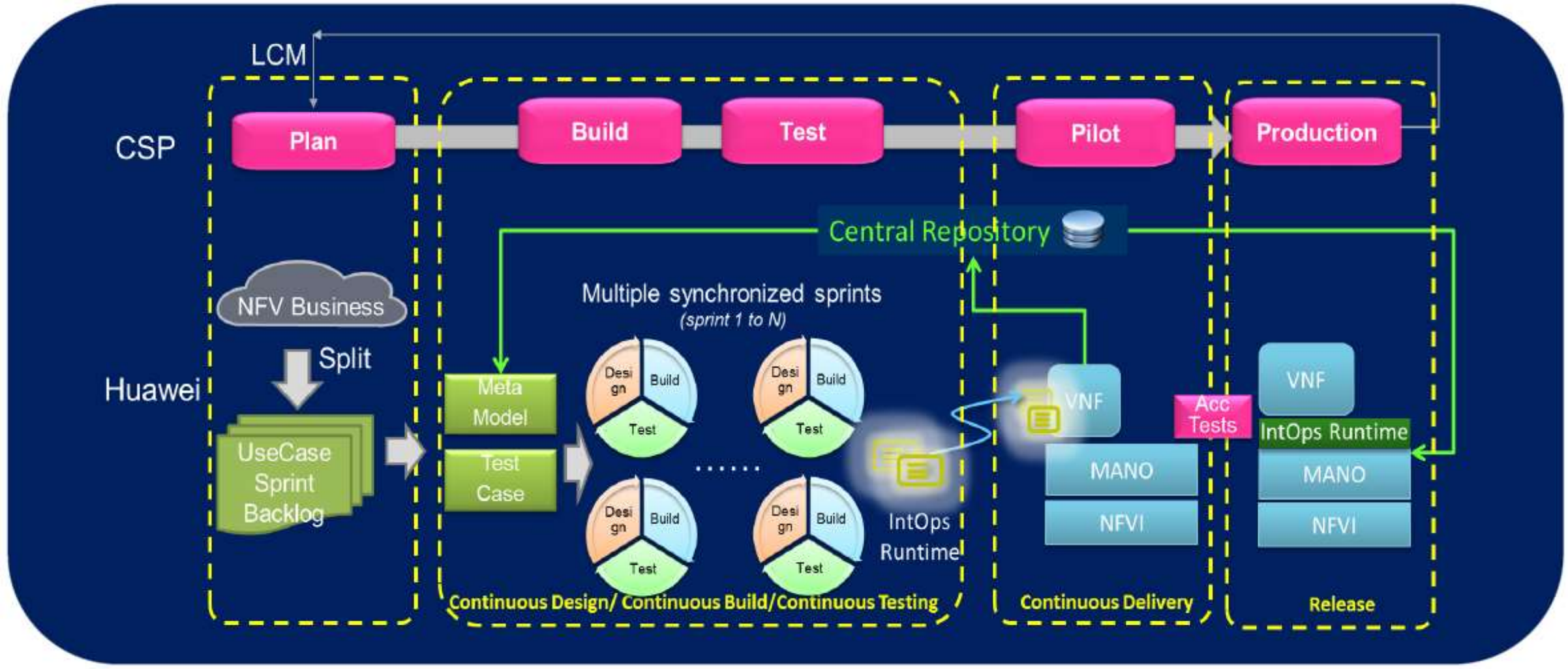
Source - https://www.perforce.com/company/newsletter/2013/04/beyond-scrum-continuous-integration-build-and-test-automation

Release management and testing: organizational view



How others see lifecycle management

Huawei: release management on virtualized multivendor infra

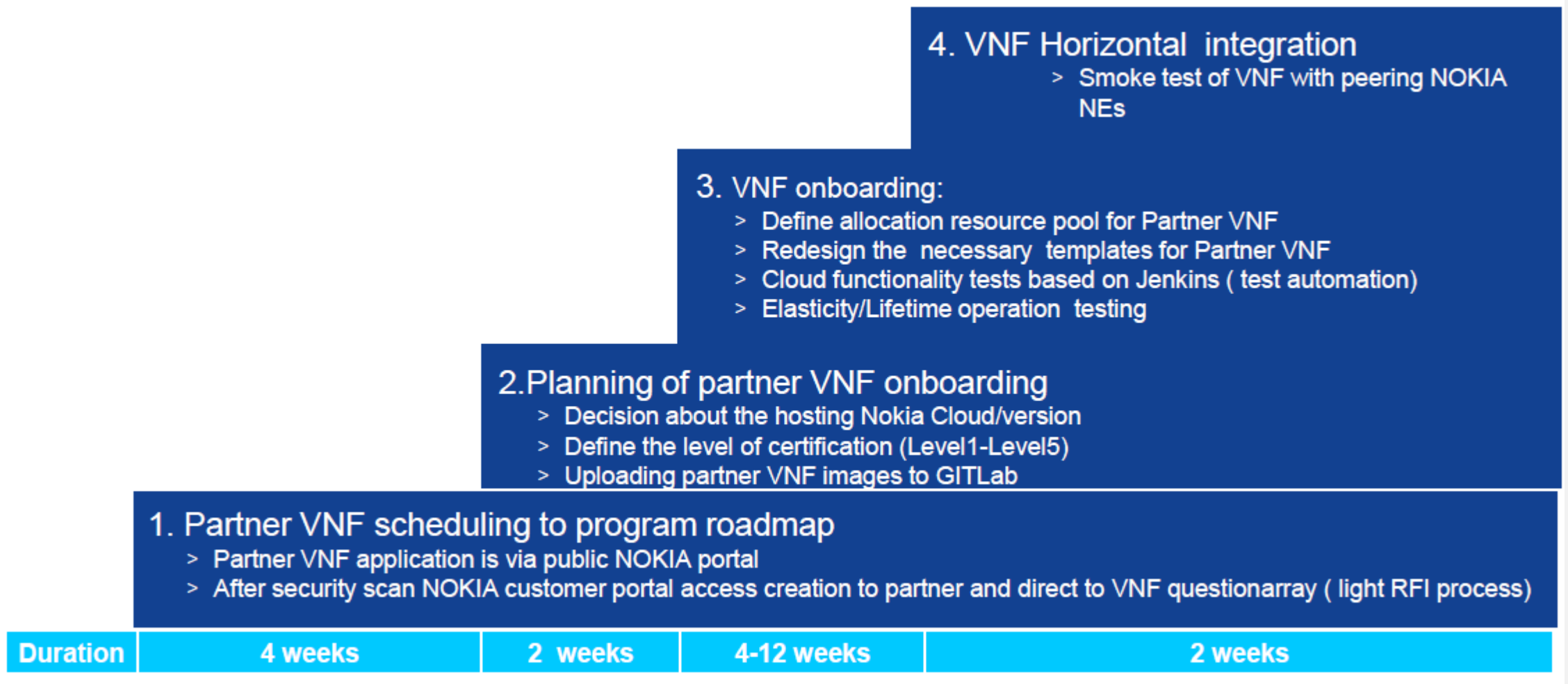


Nokia: cloud verification (clover) service overview

Service Items:

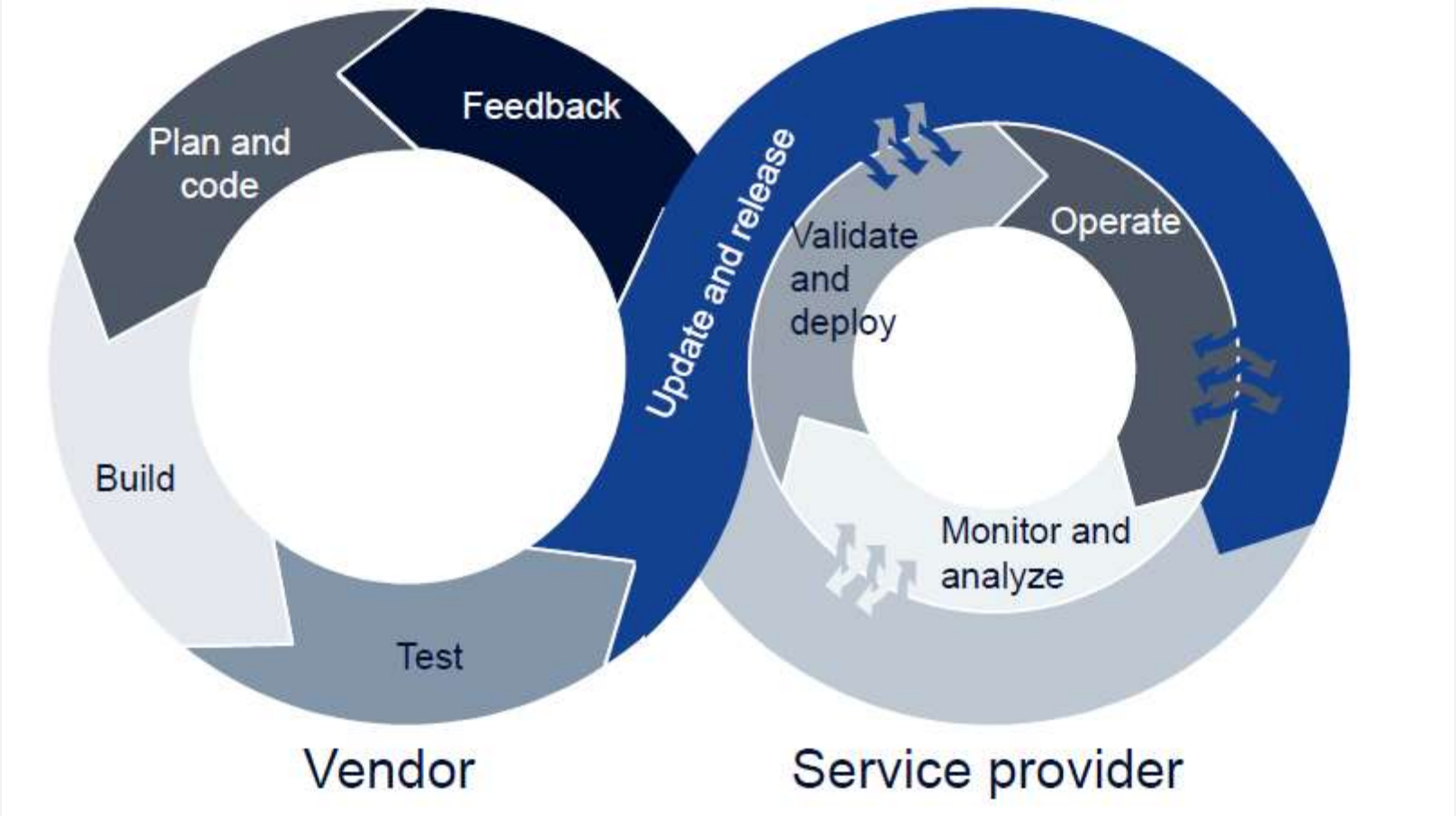
1. **Define**
   - Customer Questionnaire
   - Questionnaire analysis and workshop preparation
   - Customer workshop
2. **Schedule**
   - Testing Plan (Testing strategy, test plan, timelines)
   - Site Preparation/remote connectivity testing
3. **Testplan execution**
4. **Recommendation report**
5. **Customer workshop**
   - (test execution presentation)

Nokia: VNF onboarding methodology



Nokia: Vendor and service provider as a virtual team

Vendor and service provider as a virtual team



Automation

Lifecycle management is to resolve business challenges

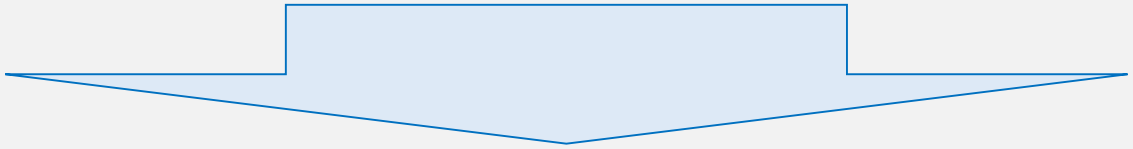
Stakeholders	Interested in	Obstacle
Business, marketing	• Significantly decrease time to market for new solutions and services • Increase number of business offerings • Decrease costs	• Lack of features in production environment • Long release cycle • High cost of changes in production environment
Procurement	• Increase competition between suppliers • Decrease dependency on key suppliers • Introduce more potential suppliers	• Highly integrated solutions are difficult to disintegrate • Technical experts see disintegration as risky
Technology, Operations, Vendors	• Predictable, convenient way of doing things: long release cycle and very limited number of suppliers (ideally best in class)	• Business environment is pushing towards further disintegration and decreased release cycle

There will be demand from business to continuously:

- **decrease release cycle duration**
- **increase number of suppliers**

What will we manage

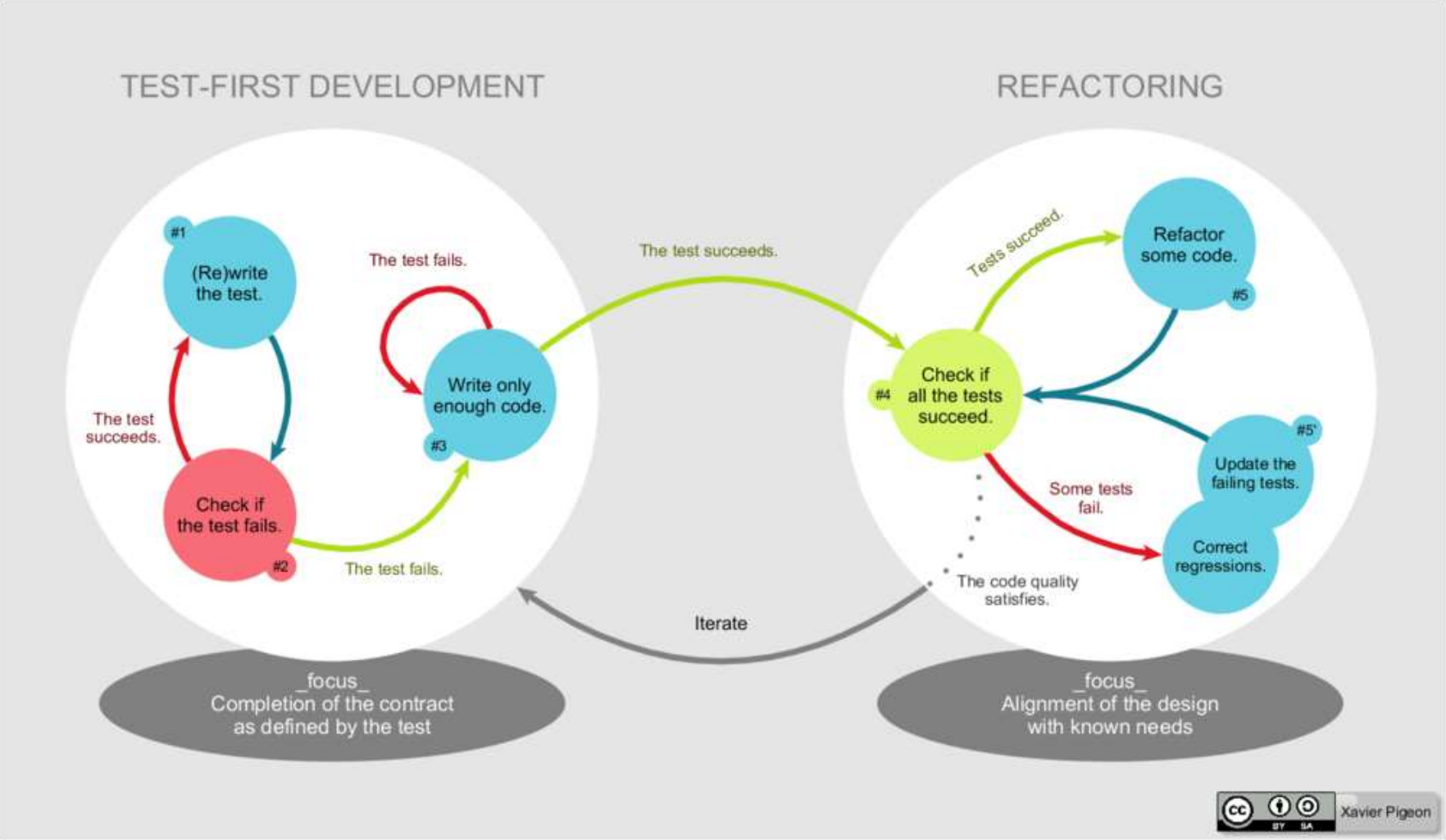
Objects to manage	Major challenges
Traditional applications (with stable release cycles) from major vendors	• Delays: patches might be delayed until next release; • Vendor locks • High cost;
Cloud applications (and many software-defined) for agile environments; Some cheaper alternatives	• Frequent patches, • Imperfect documentation
Many hardware and software components from different vendors with independent release cycles	• Interoperability



Too many things change too often – automation required

Test-driven development (TDD)

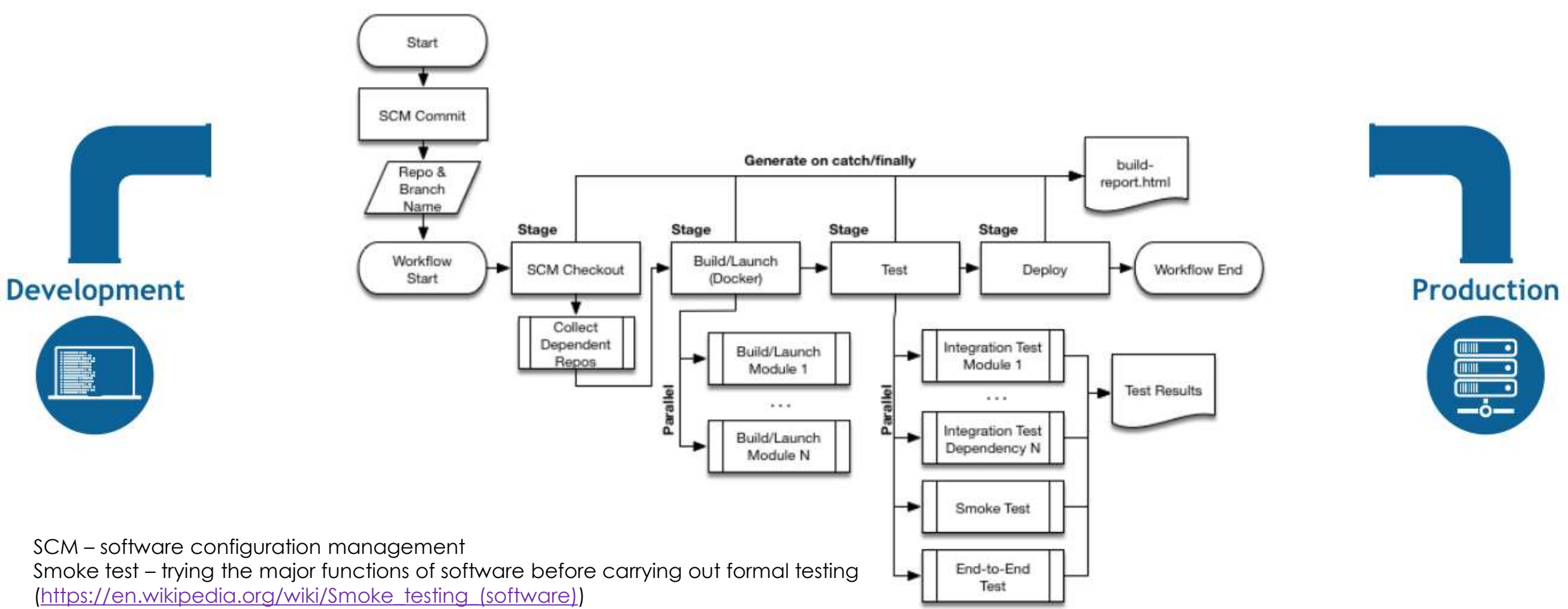
"Requirements are turned into specific test cases, then the software is improved to pass the new tests, only"



- "**Decomposition** in computer science, **also known as factoring**, is breaking a complex problem or system into parts that are easier to conceive, understand, program, and maintain"
- "**Refactoring** – process of **restructuring** existing computer code – changing the factoring – **without changing its external behavior**. Refactoring improves nonfunctional attributes of the software."

Source: https://en.wikipedia.org/wiki/Test-driven_development, https://en.wikipedia.org/wiki/Decomposition_(computer_science), https://en.wikipedia.org/wiki/Code_refactoring,

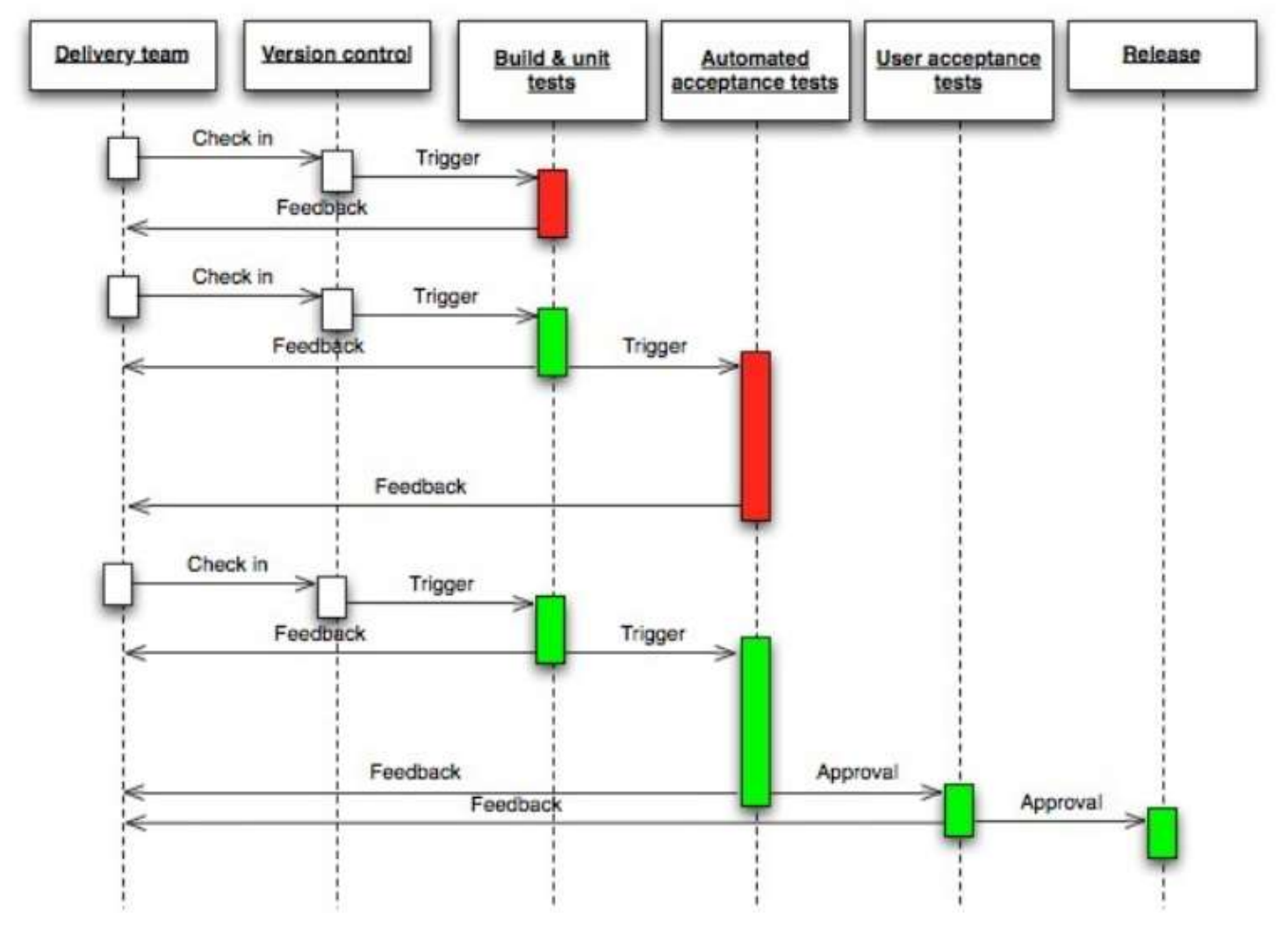
Build and test automation: sample flow



SCM – software configuration management
Smoke test – trying the major functions of software before carrying out formal testing (https://en.wikipedia.org/wiki/Smoke_testing_(software))

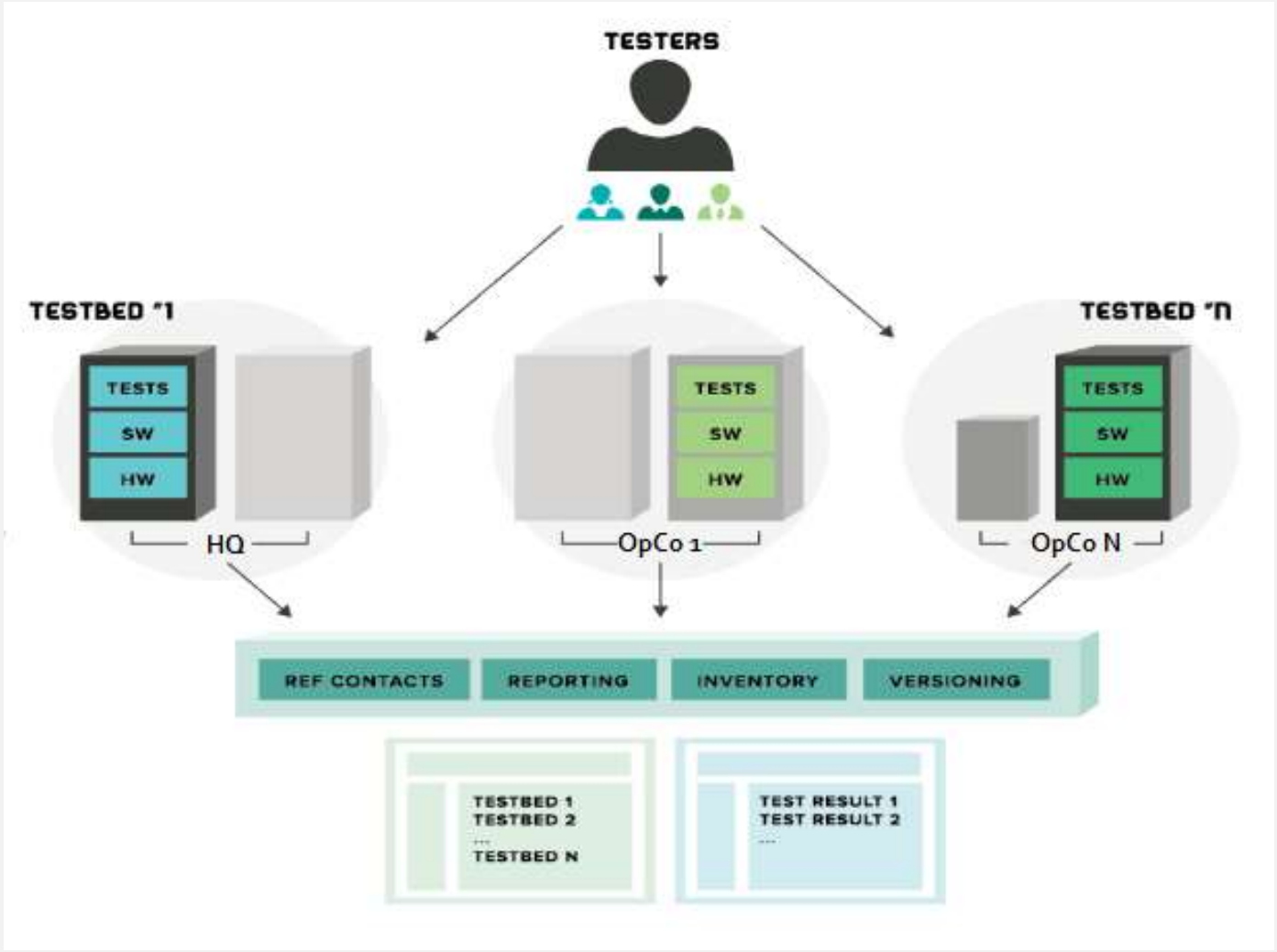
Source - https://jenkins.io/doc/pipeline/

Feedback and approval loops in automated workflow



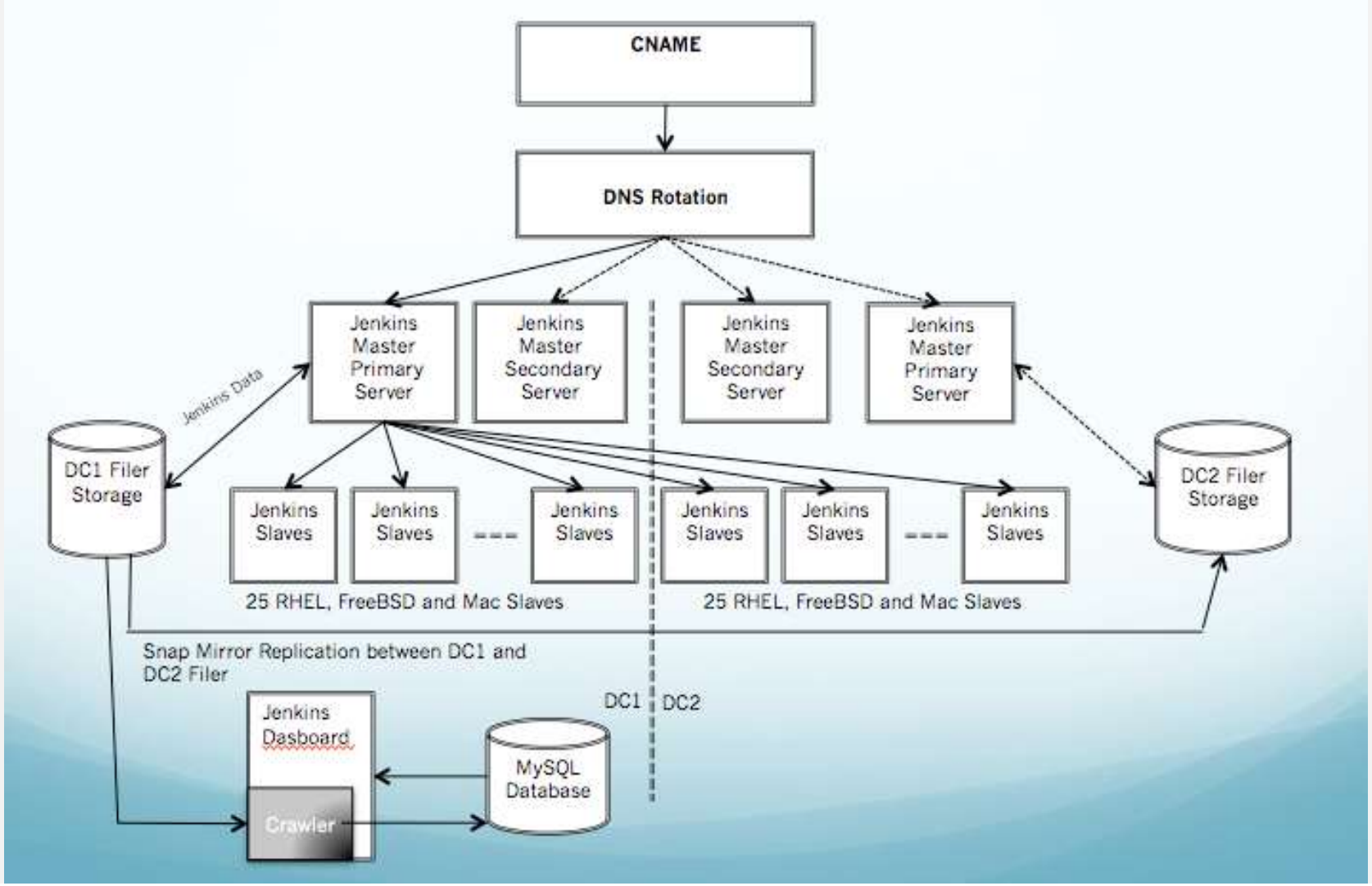
Source - http://www.slideshare.net/martinmalek92/ci-and-cd-with-jenkins

Distributed test automation environment example: OPNFV Pharos



Original picture is from https://www.opnfv.org/developers/pharos

Distributed test automation software example: Yahoo



Source - https://go.cloudbees.com/docs/cloudbees-documentation/cookbook/ch01.html#_hardware_recommendations

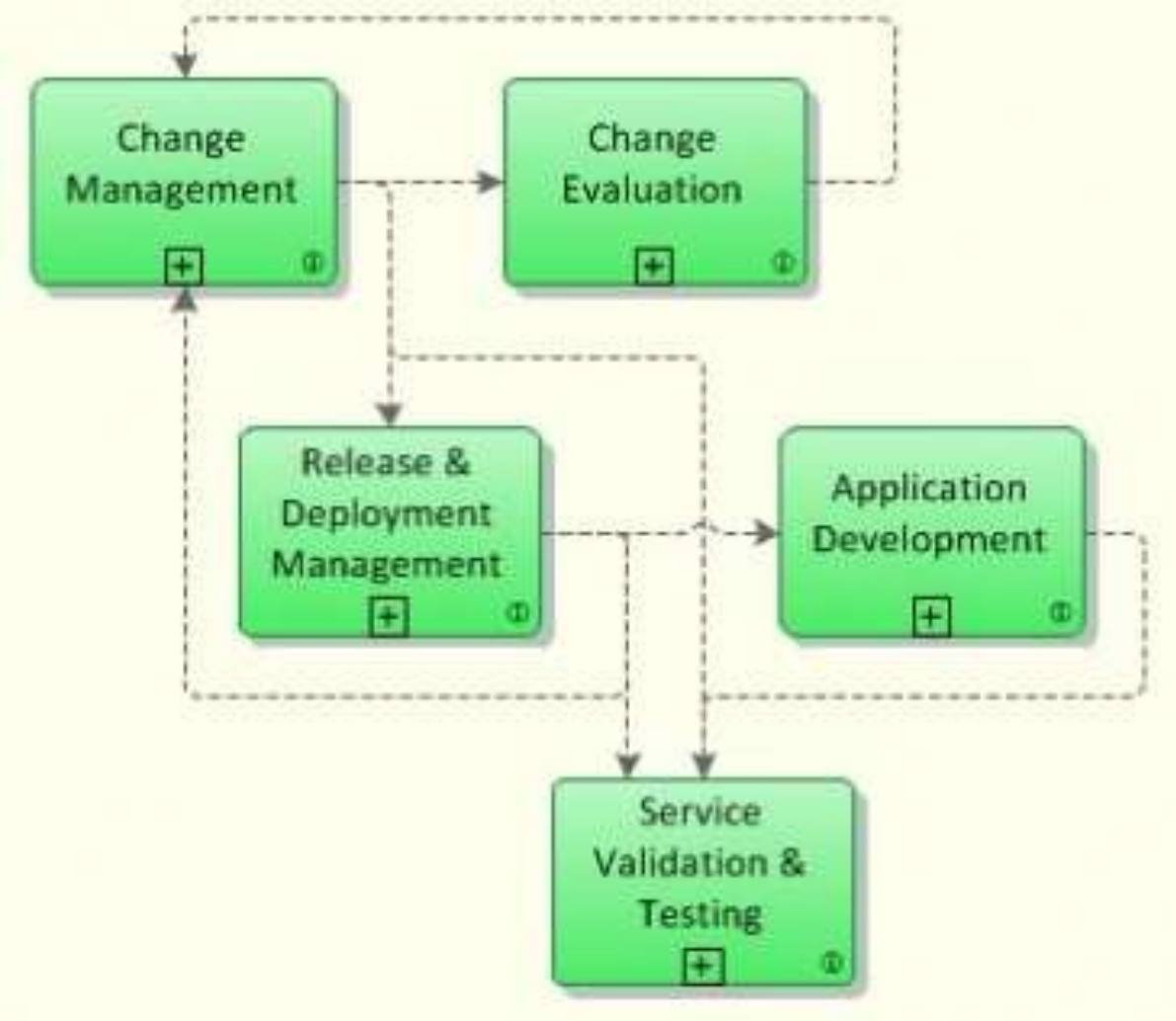
Application development & customization

AN INSIGHT FROM ITIL EXPERTS:

APPLICATION DEVELOPMENT IS BARELY MENTIONED IN THE ITIL BOOKS, AS ITIL FOCUSES ON DIFFERENT TOPICS LIKE SERVICE DESIGN AND ROLLOUT. AT IT PROCESS MAPS WE DECIDED TO ELIMINATE THAT GAP BY INTRODUCING AN APPLICATION MANAGEMENT PROCESS WHICH TAKES CARE OF THE ACTUAL APPLICATION CODING AND THE CUSTOMIZATION OF STANDARD SOFTWARE PACKAGES

Application development and customization

Development and maintenance of custom applications as well as the customization of products from vendors



Source - http://wiki.en.it-processmaps.com/index.php/ITIL_Service_Transition

Infrastructure as a Code

IaC

- Infrastructure as Code (IaC) is the process of **managing and provisioning** computing **infrastructure** (processes, bare-metal servers, virtual servers, etc.) **and their configuration through machine-processable definition files**, rather than physical hardware configuration or the use of interactive configuration tools.
- https://en.wikipedia.org/wiki/Infrastructure_as_Code

DevOps toolchain

- is a set or combination of tools … , which supports specific DevOps initiatives: Plan, Create, Verify, Preprod, Release, Configure, and Monitor.
- https://en.wikipedia.org/wiki/DevOps_toolchain

Releases

Release types*

Release	Periodicity	Origin	Description
General availability (GA) release	Half a year	NFVI Vendor	“A fixed set of objects and artefacts marked at a specific point in time that will be supported as such”
Maintenance release	Triggered by: • GA release or major (security) update from NFVI vendor • Major feature request from VNF vendor / OpCo / Business	Headquarter	“"Maintenance release" content is the same as that of a "release" with the corrected features / functions”
Latest release	Regularly (weekly?)	Headquarter	Integrates “the latest available components (could be binary artefacts or sources) from upstream repositories... Main objective … is to give developers and testers immediate feedback at system level. "Latest" does not imply a “supported” release, rather is the “latest” working merge working towards the next target ‘release’ (future or maintenance)

*Example. OPNFV terminology used

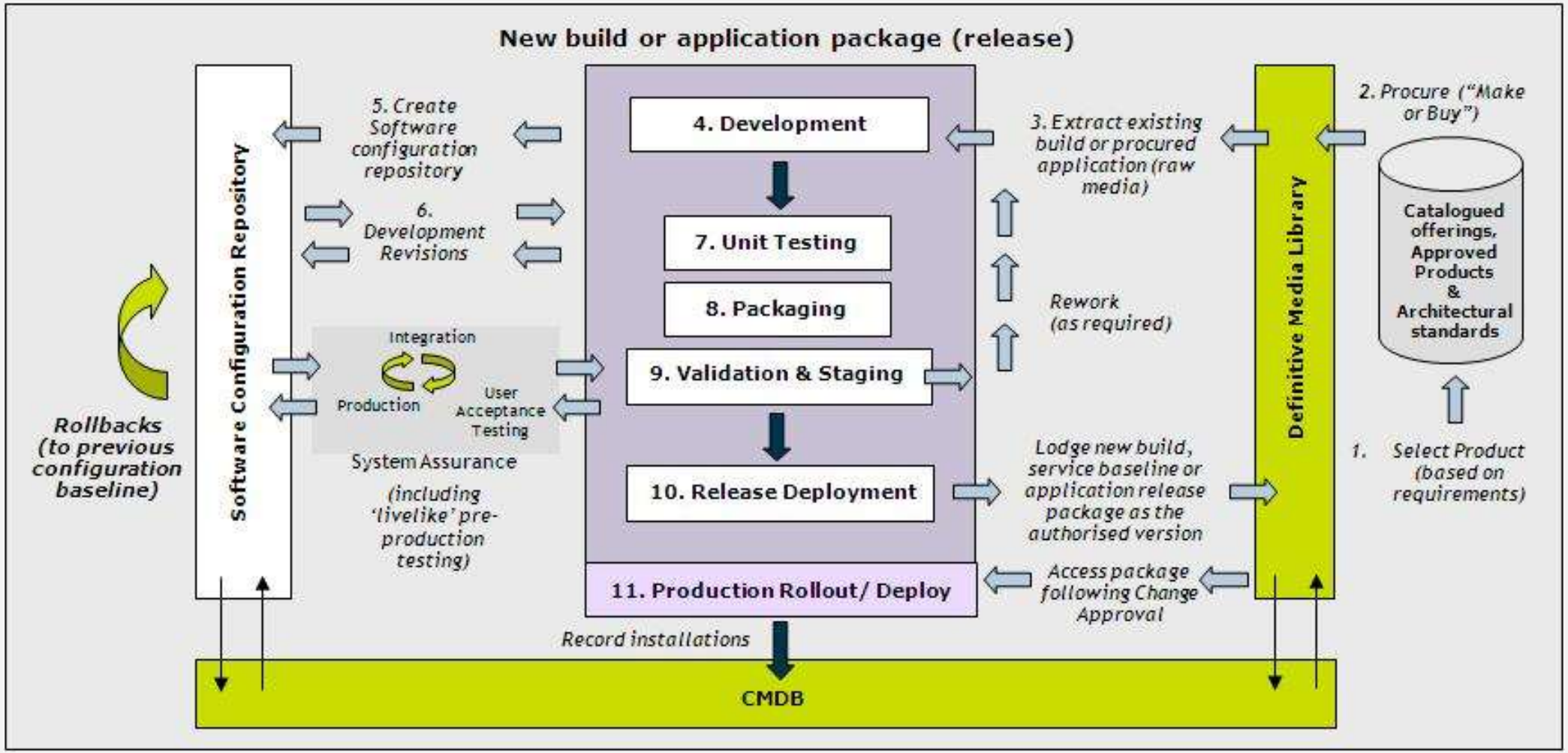
Major release milestones*

Project planning start	March	OpCos, vendors, other stakeholders indicate intent to participate in the release
Planning complete	April	Close project inclusion and plans are established
Feature Code freeze	…	Close feature project development phase - freeze
Installation Code freeze	…	Installer/plugins code complete
Test Code freeze	…	Test infra and scenario tests complete
Integration ready	…	Scenario integration complete
Release	August	Release

*Example. Based on OPNFV Colorado release milestones

Populating release components catalogue

ITIL DML population workflow



* Source - https://en.wikipedia.org/wiki/Definitive_Media_Library

Thank you!