DevOps Practitioner

Lesson 1—DevOps Overview

What's in It for Me

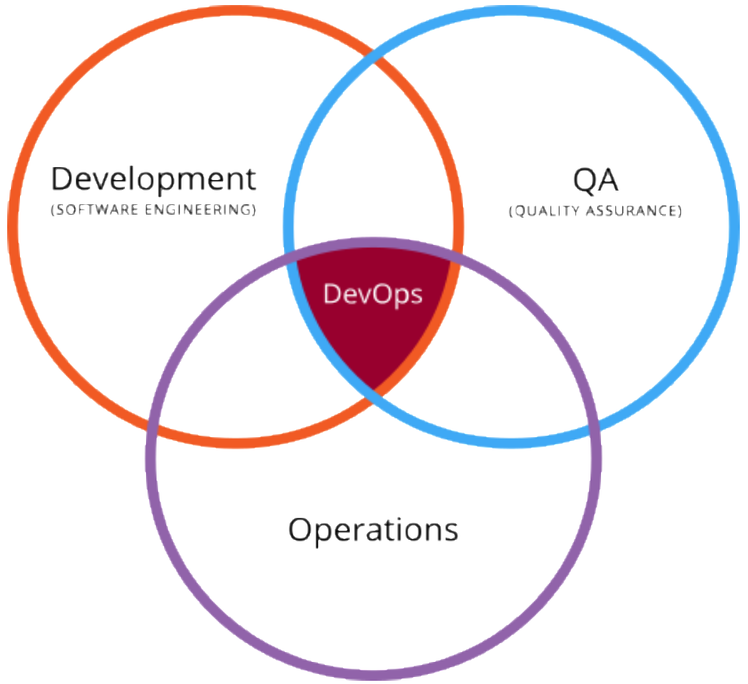
- An understanding of DevOps
- An understanding of the technical challenges in DevOps
- How to choose the best tools to use
- How to choose when to deploy applications into the Cloud
- An understanding of security issues
- An understanding of the difference between requirements and architecture
- How to write user acceptance tests

What is DevOps?

Introduction to DevOps

DevOps

- DevOps is short for **Dev**elopment **Op**eration**s**
- It focuses on collaboration between developers and other parties involved in building, deploying, operating, and maintaining software systems
- It is about defining a flow from development through full scale operation of a system
- It is about Systems Thinking with feedback to earlier stages of a DevOps workflow
- The emphasis is on automating processes required to release and change software
    - With a view to rapid, frequent, and reliable releases



Agile

In the late 1990's, several methodologies began to gain public attention; each had a different combination of old and new ideas

Development Team

Internal Stakeholders

Agile (Contd.)

These methodologies emphasized frequent delivery of business value, tight, self-organizing teams; and smart ways to craft, confirm, and deliver code.

Agile (Contd.)

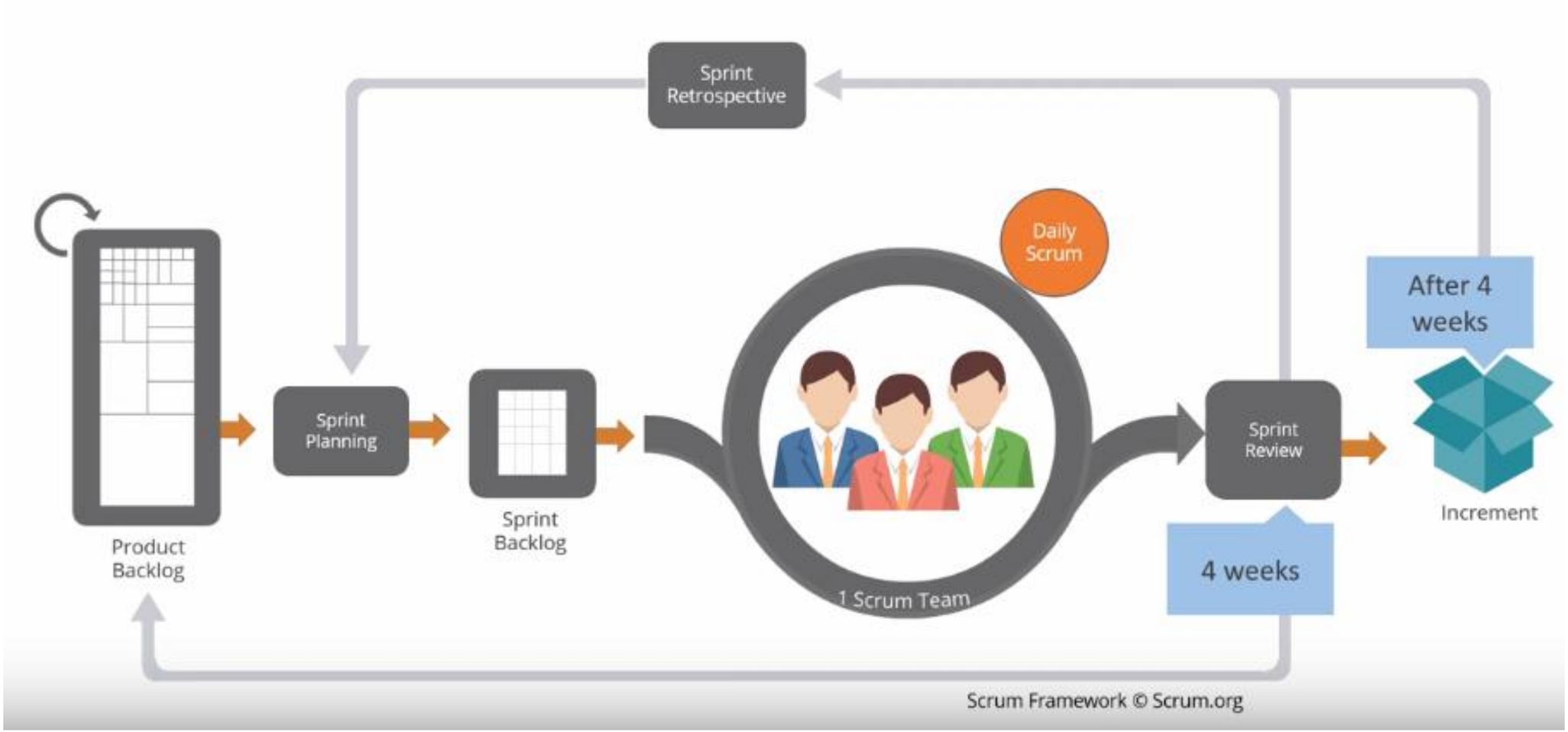
As expressed in the manifesto, Agile is a joint collection of values and principles for agile software development

Seventeen software development practitioners gathered in Snowbird, Utah, and shared ideas and approaches to software development.

Solutions evolve through collaboration between self-organizing, cross-functional teams utilizing the appropriate practices for their context.

Source: www.agilealliance.org

Agile—Example



Agile—Example (Contd.)

Agile and DevOps—How Are They Related?

Some of the principles of Agile manifesto that lead to a DevOps environment are:

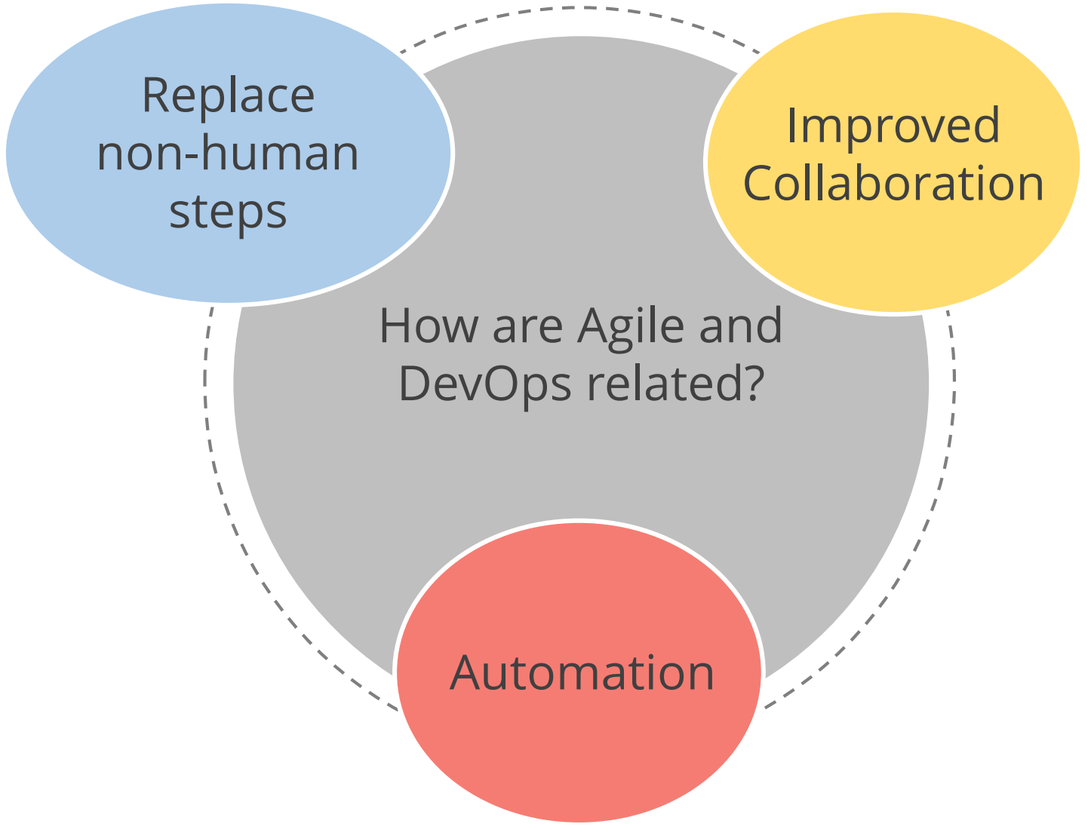
Satisfy the customer through early and continuous delivery of valuable software

Deliver working software frequently with a preference for the shorter timescale

Business people and developers must work together daily throughout the project

Agile and DevOps—How Are They Related? (Contd.)

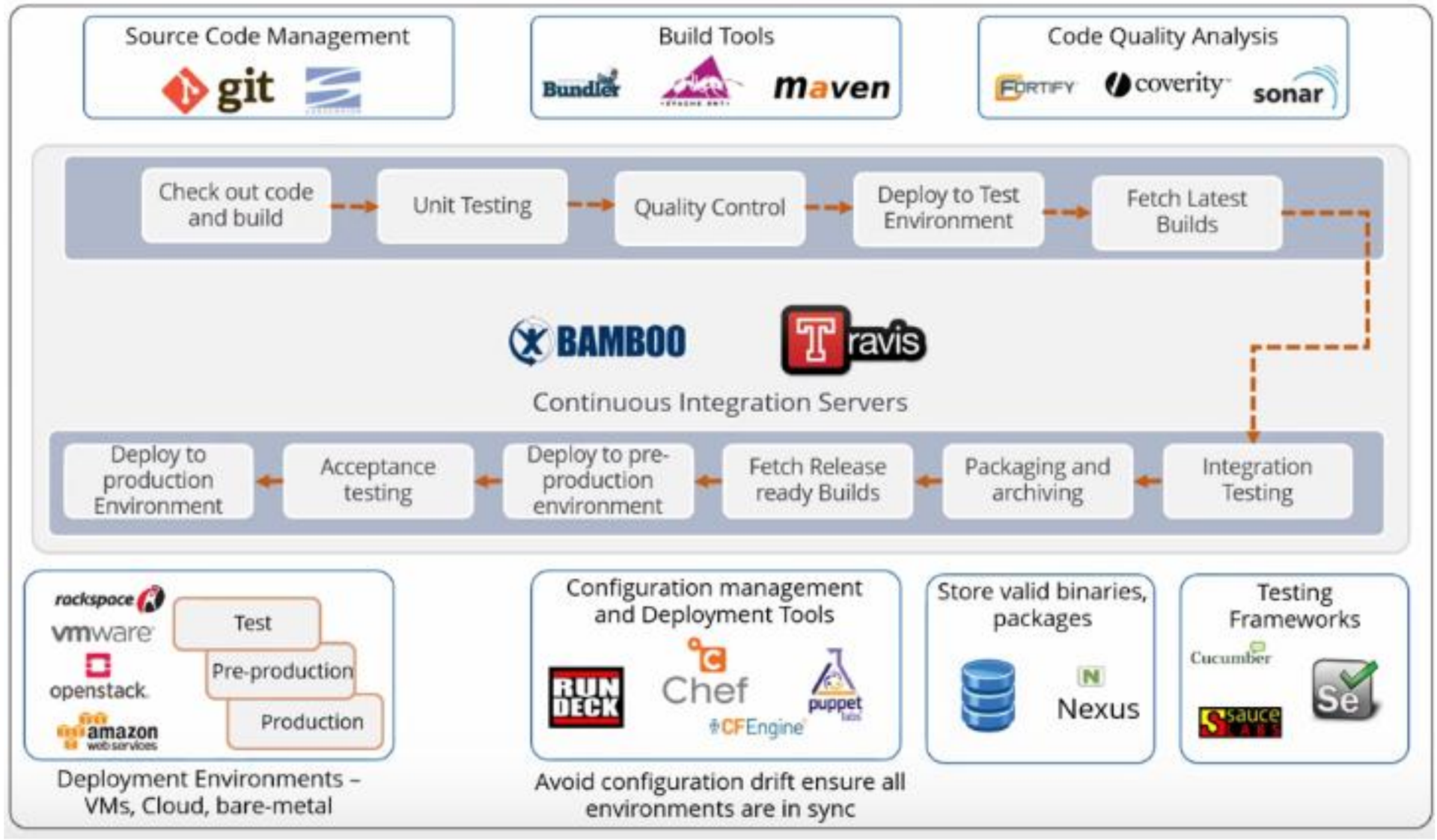
Using tools to replace non-human steps such as build creation, environment setup, trigger test cases, and automated deployments multiple times in a day



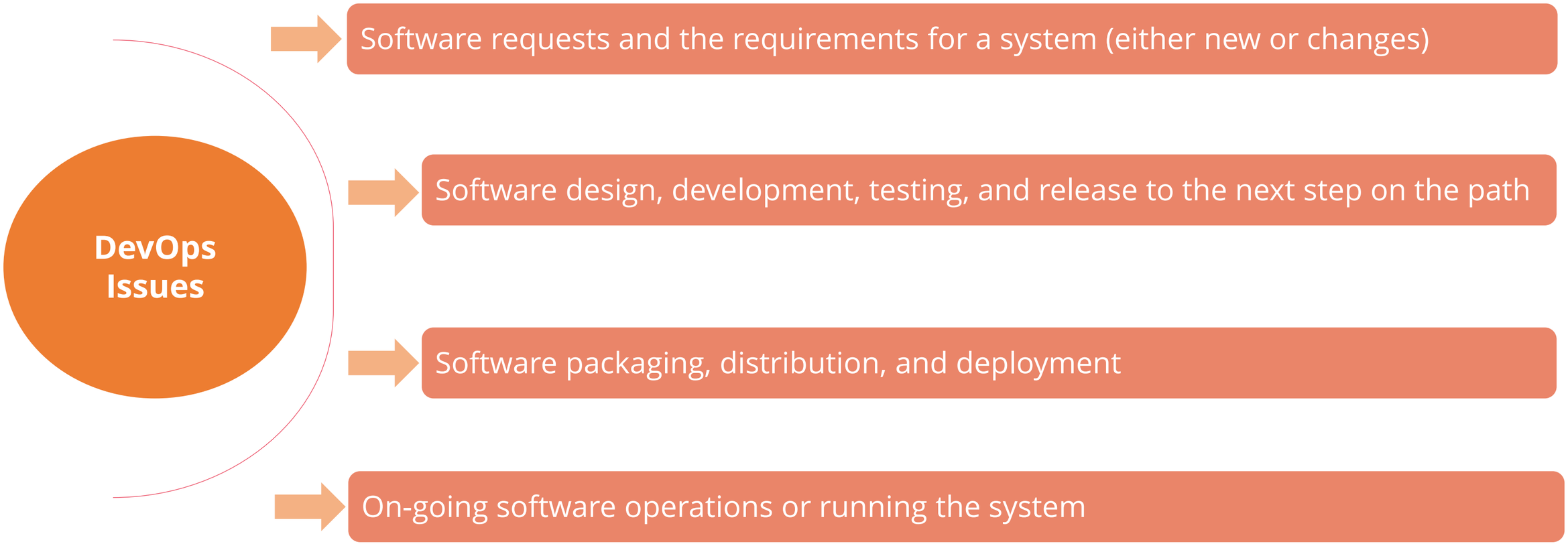
DevOps not only talks about collaboration between Development and Operations, but improved collaboration between all teams, including business

DevOps also promotes using a lot of automation to complete the many tasks and steps that go into making a potentially shippable increment

Agile and DevOps—Example



DevOps Issues



Deployment Issues

Skill gaps exist that may have been hidden

- The deployment of Web and Enterprise applications to servers need special skills

Developers write the application code

- They often have little knowledge of the application server

Operations Teams install and manage the applications servers

- They are usually not developers

What happens when the application is deployed into the server?

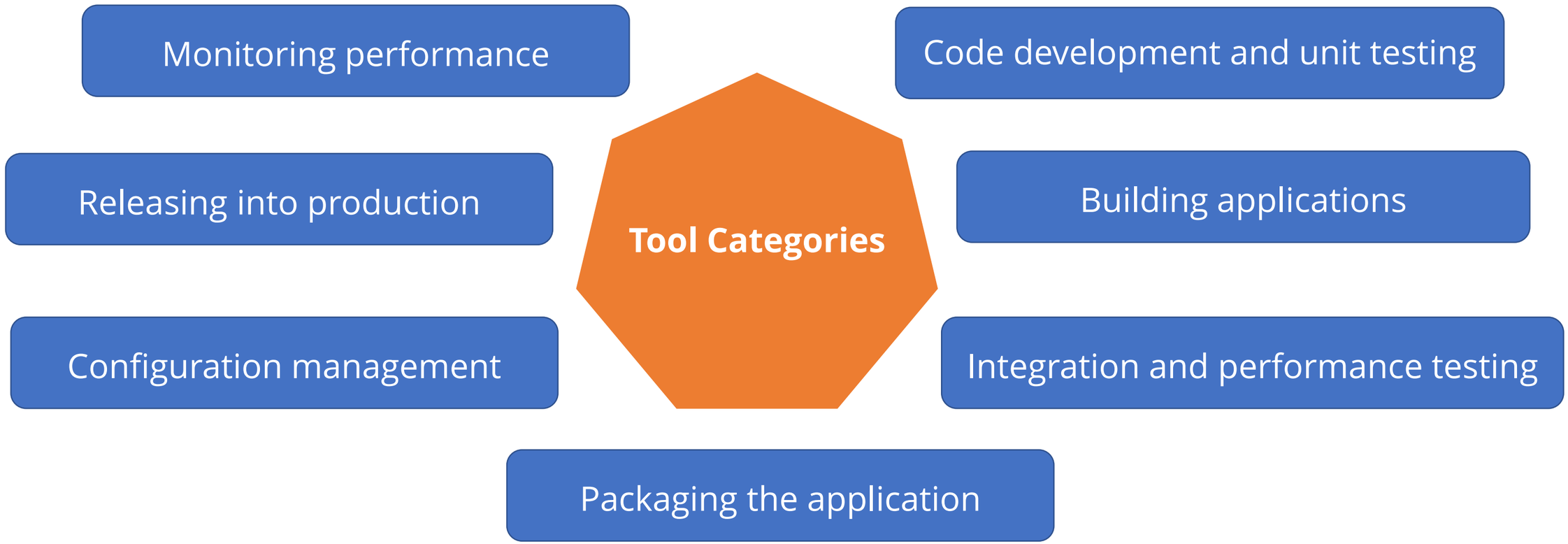
- Need people who are both developers and operations to resolve issues

DevOps Toolchain

DevOps utilizes a number of different tools

- There are a number of categories each with its own selection of tools

Activities that have their own Category of tooling are:



DevOps Goals

- DevOps is about systems thinking, process definition, team coordination, and feedback loops
- DevOps is also about continuous improvement

Aims to improve quality and reduce the number of incidents in production

Overall efficiency in the process will increase as the team learns best practices

DevOps Goals (Contd.)

Some key improvements are:

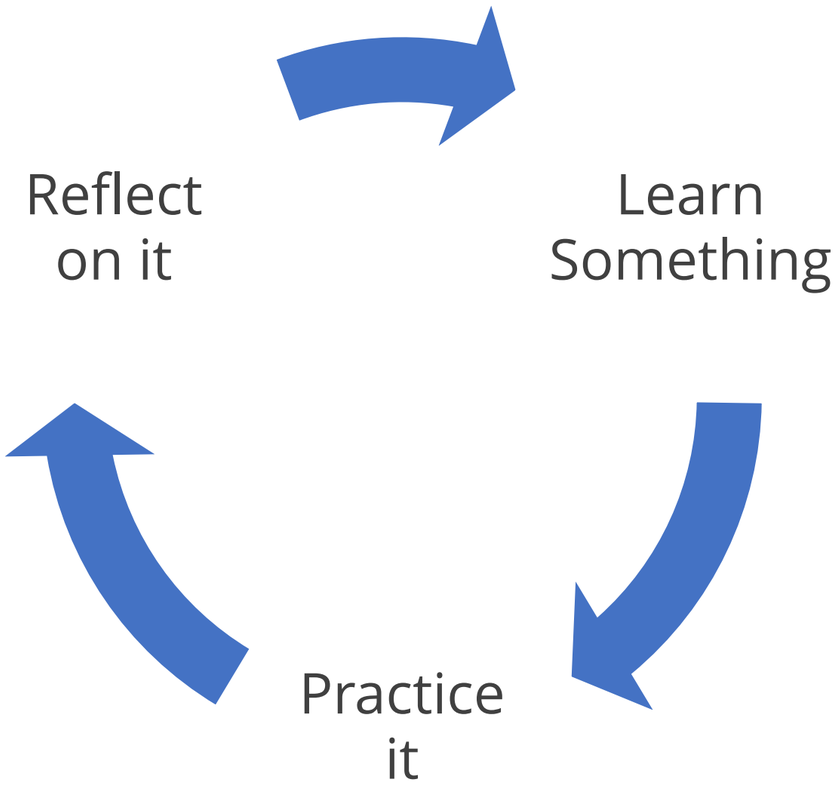
- Reduce testing time
- Reduce the amount of changes required due to incomplete or conflicting requirements
- Design the simplest solution required to meet requirements
- Leverage standardized integrated software tools
- Create robust production systems which lower maintenance costs
- Produce software to the highest quality
- Improve the performance of individuals and teams

Behaviors

- Adopting good working behaviors is key to success
  - Strive for clarity – challenge uncertainties in requirements
  - Challenge the validity of assumptions
  - Seek simplicity and reliability; do not strive to produce the optimal design
  - Seek the least risky design that meets requirements
  - Avoid Gold Plating – do not add extra code or features beyond the bare minimum
  - Fix the root cause of each issue instead of patching to fix the symptoms

Working Practice

Working practice should be constantly evolving.



Knowledge Check

KNOWLEDGE CHECK

What is Gold Plating?

a. Writing complicated code

b. Making user interfaces visually appealing

c. Work which adds little or no value

d. Writing too many tests

KNOWLEDGE CHECK

What is Gold Plating?

a. Writing complicated code

b. Making user interfaces visually appealing

c. Work which adds little or no value

d. Writing too many tests

The correct answer is **c.**

Gold Plating is working on a task beyond the point where the extra effort is worth any value it adds.

Technical Challenges

Technical challenges in DevOps

Technical Challenges

- DevOps requires addressing technical challenges
- The challenges are constantly changing like:

Technology is constantly evolving

Working environment is evolving

External influences: economic, legal, and social

DevOps Culture

- A DevOps culture needs to be built and maintained
- A DevOps culture adapts to continuous change by continually changing
    - The organization's culture needs to evolve with it
    - People are resistant to change
    - People need to work together and not in isolation

Software Tools

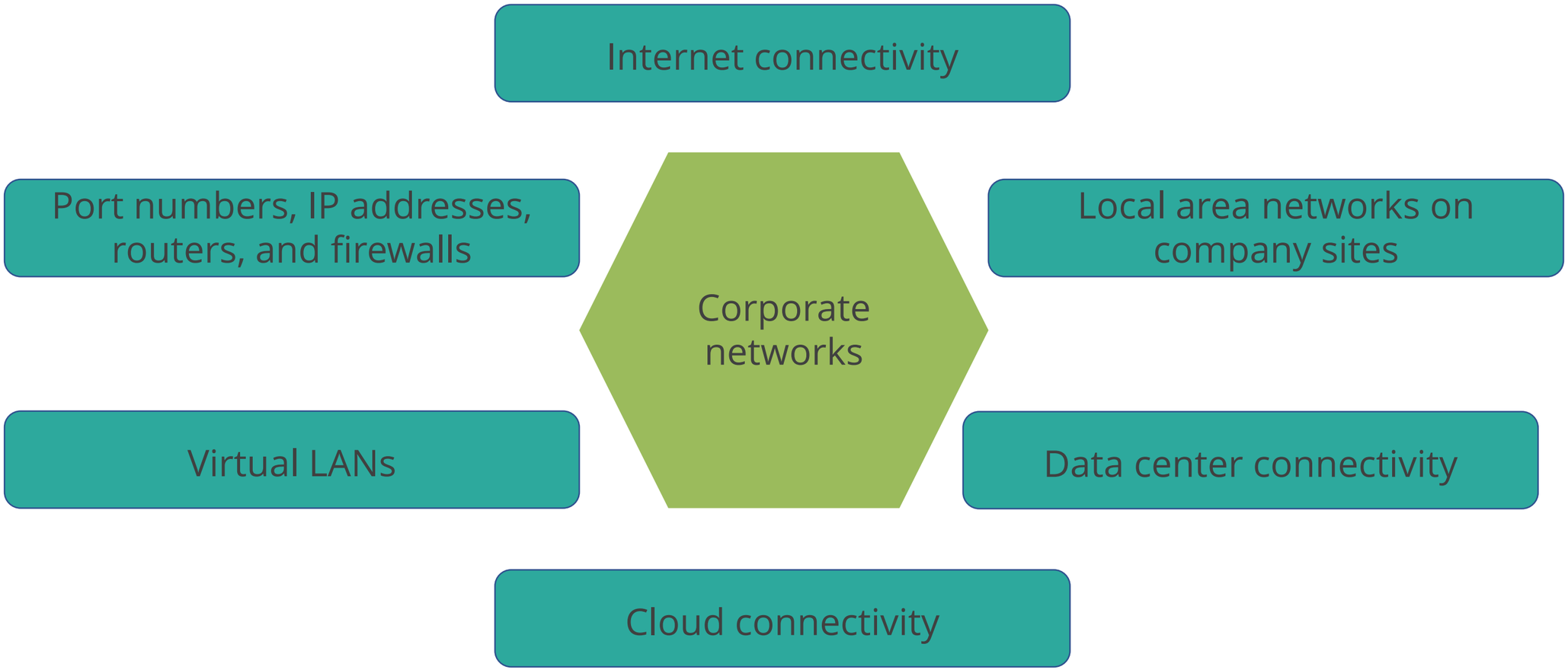
- Individuals and teams will have preferred tools
- Open source tools are readily available
- It is important to have a standardized set of tools across the organization
- Different users or teams using different tools leads to major issues

Tool Compatibility

- Organizations have legacy systems
  - New tools need to be compatible with new and older systems
- Hybrid environments are evolving
  - Existing solutions
  - Cloud-based solutions are getting increasingly attractive
  - Tools must be Cloud ready or Cloud compatible
- Old and new systems need to work together
  - Complete with production support

Networking

Corporate networks are getting increasingly complex:



Networking (Contd.)

- Changing network configuration is complex and risky
- Cloud computing with software define networking

Less complex

More adaptable

More auditable

More secure (deny all by default and only allow by design)

Cost

- Changing a corporate culture incurs costs
    - New hardware requirements
    - Software licenses and Cloud usage costs
    - Training and retraining
    - Reduced efficiency while learning new skills
- There needs to be a budget for ongoing change
- Improving efficiency and quality will reduce costs in the long term

Legacy

- Introducing DevOps is not just about writing and deploying new code
- Existing code needs to be improved too
    - New tools generally support new code; they need to also support legacy code
    - Legacy code may need to be refactored or restructured over time
        - Make it manageable
        - Fit with new processes and tooling
- Bug fixes and enhancements need to be performed using new tools and procedures

Knowledge Check

KNOWLEDGE CHECK

What is the cultural challenge for DevOps?

a. People need to learn new languages

b. People need to work together across traditional role boundaries

c. People need to work different hours

d. People need to move desks frequently

KNOWLEDGE CHECK

What is the cultural challenge for DevOps?

a. People need to learn new languages

b. People need to work together across traditional role boundaries

c. People need to work different hours

d. People need to move desks frequently

The correct answer is **b.**

People need to work together across traditional role boundaries. Developers need to work with operations and testing teams.

Software Tools

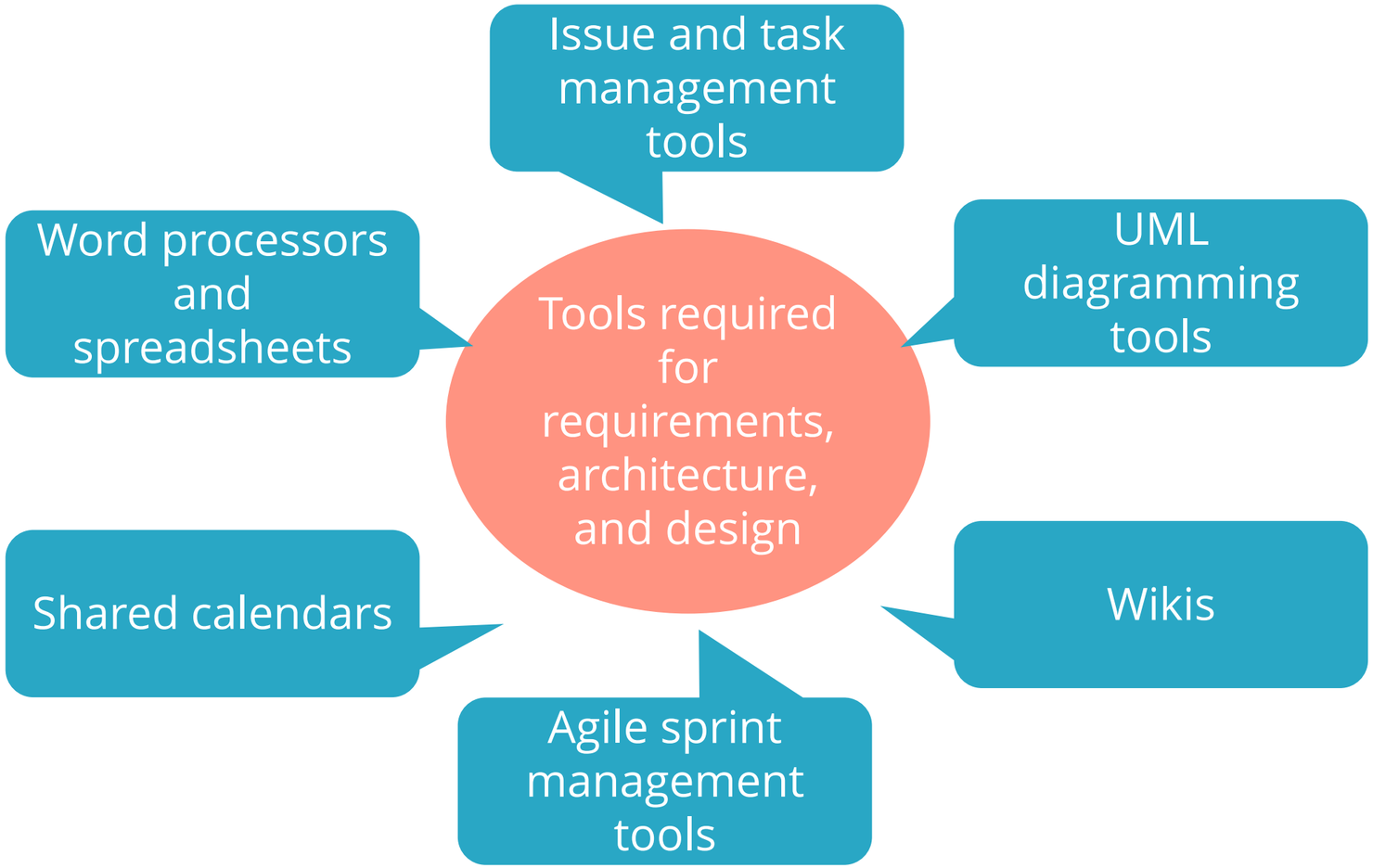
Overview of the required software tools

Tool Chains

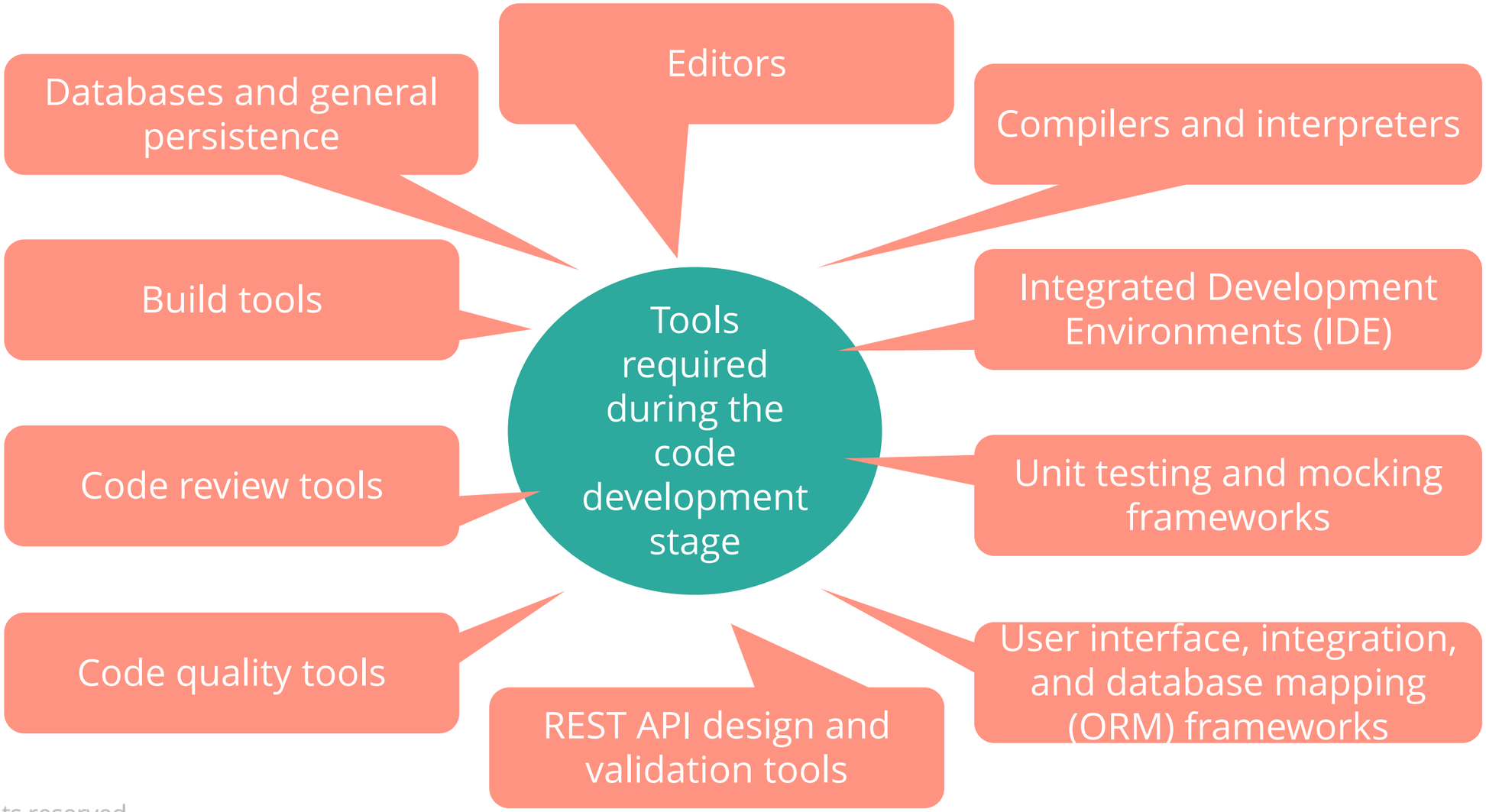
- One of the DevOps challenges is to select the right tools
- There are a number of categories of tools for different SDLC phases
- There are a number of tools that perform similar functions
- Tools need to work well together

Requirements Tools

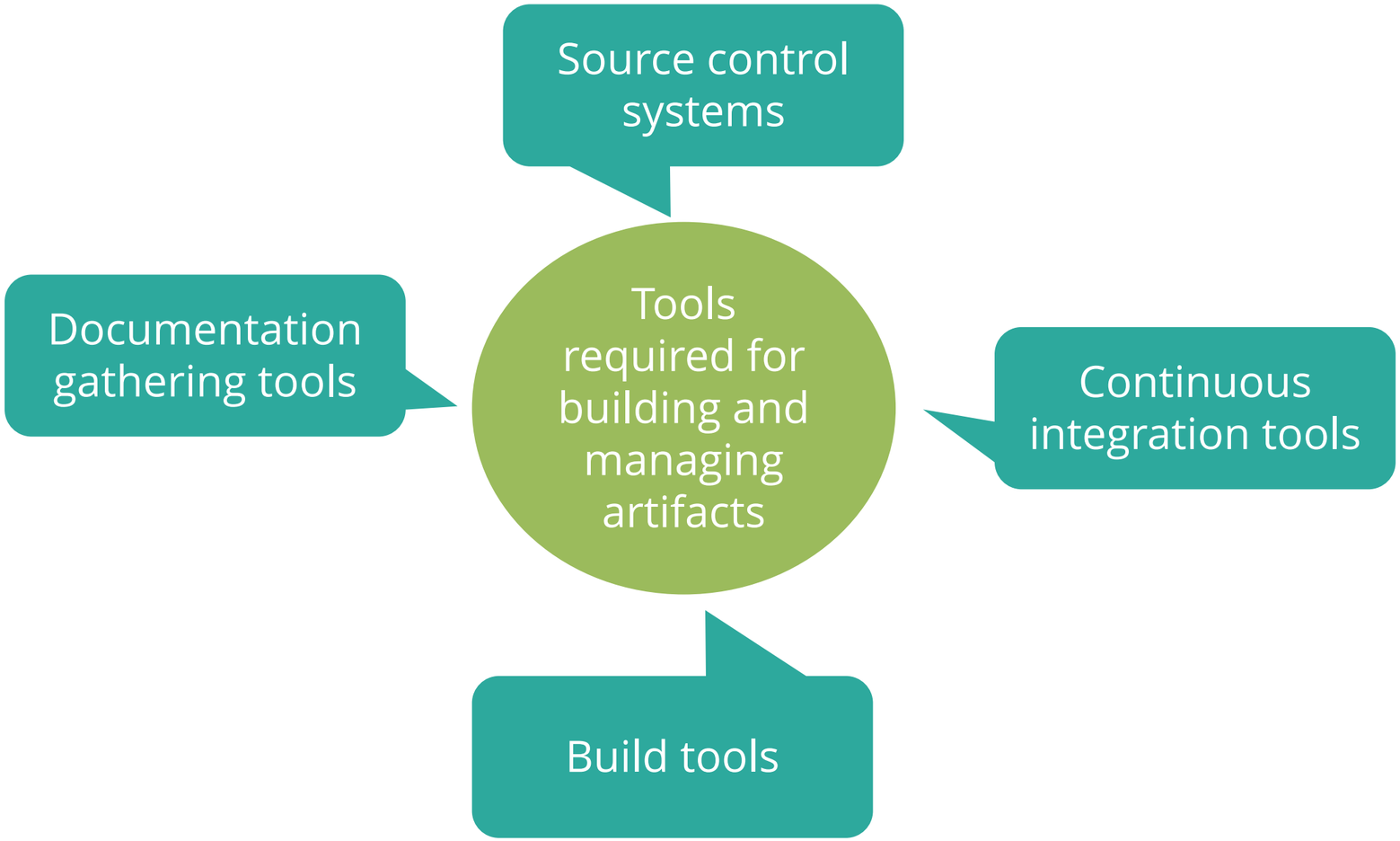
Tools are about sharing and communicating within a team, and with others.



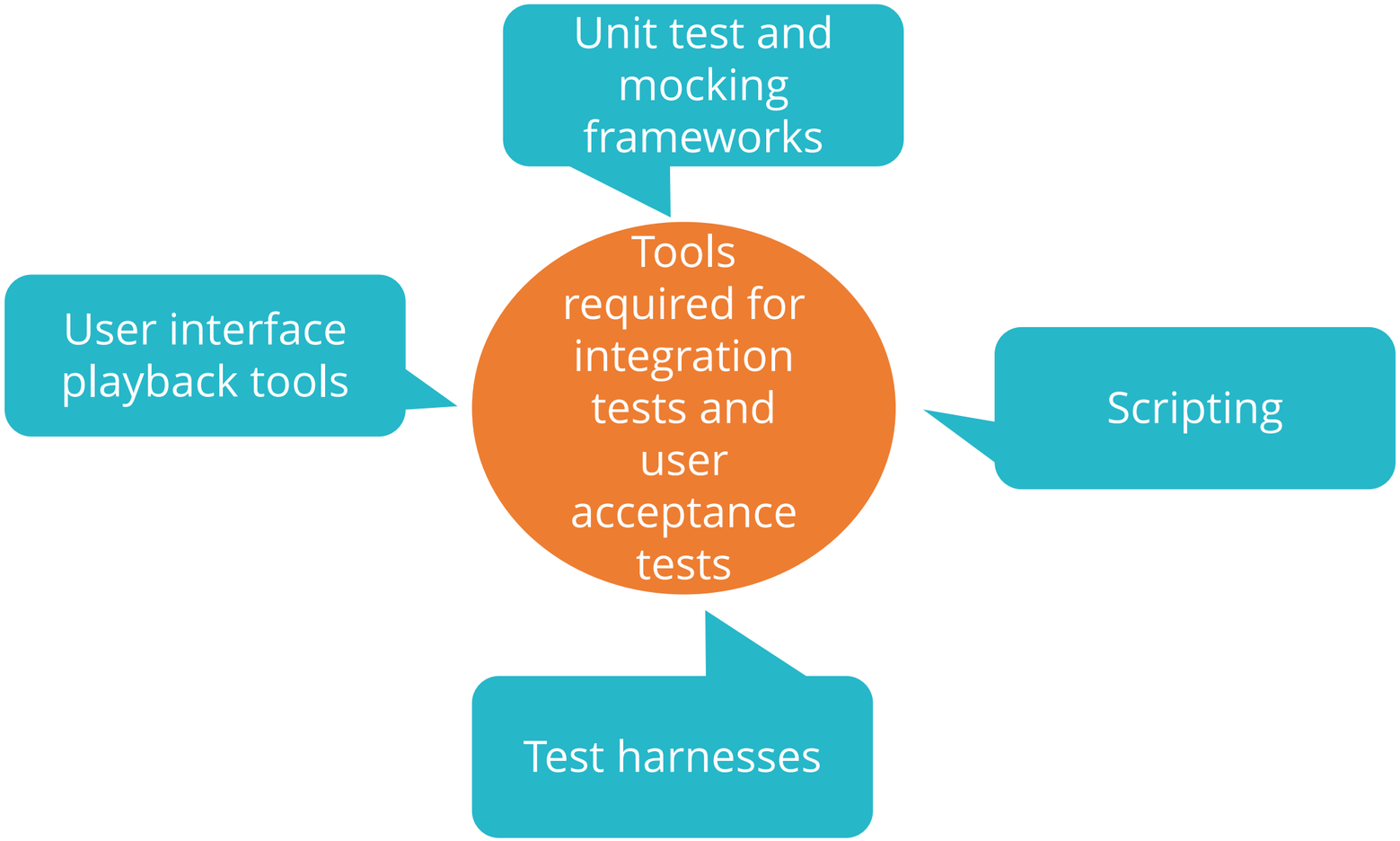
Code Development Tools



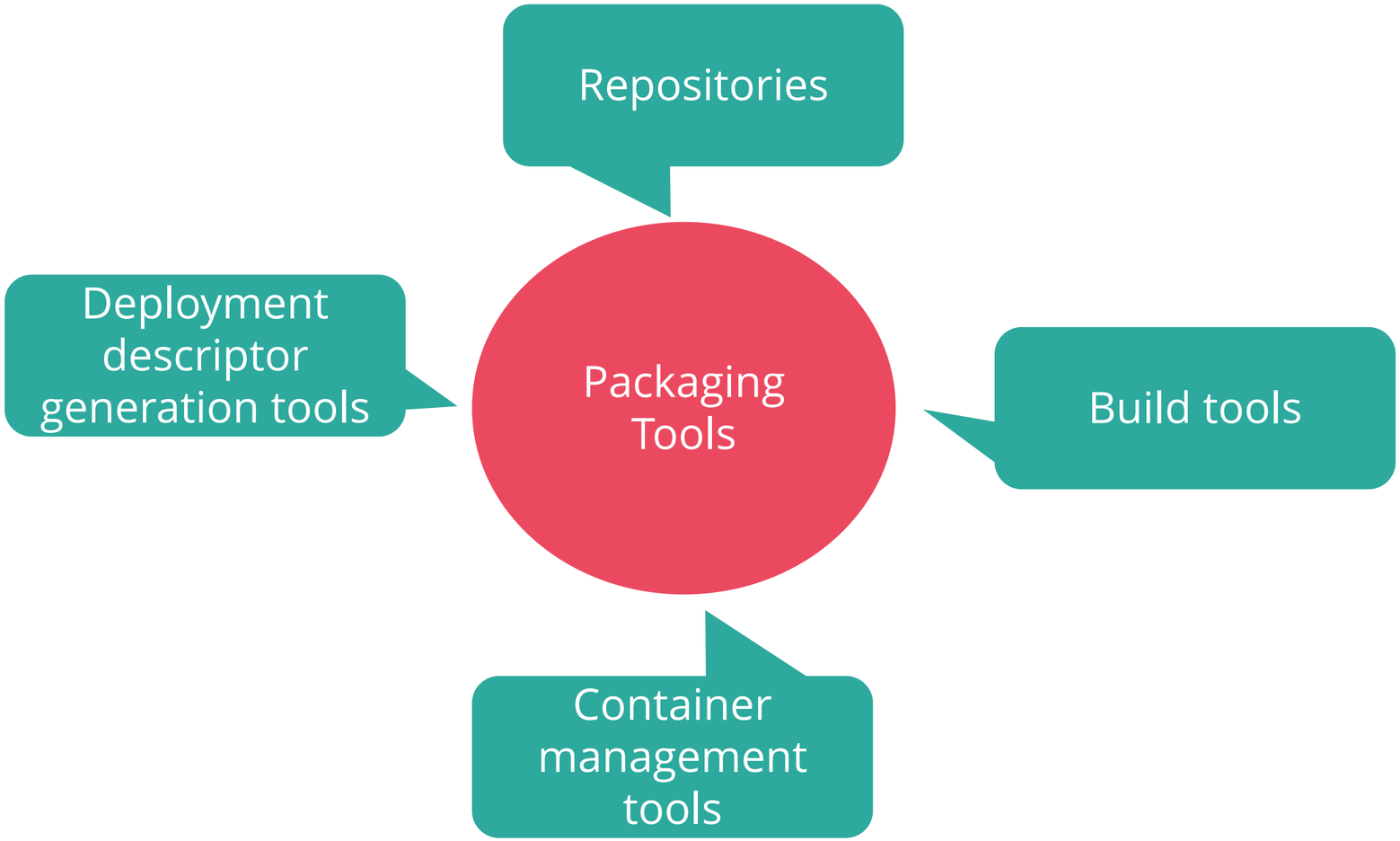
Artifact Creation Tools



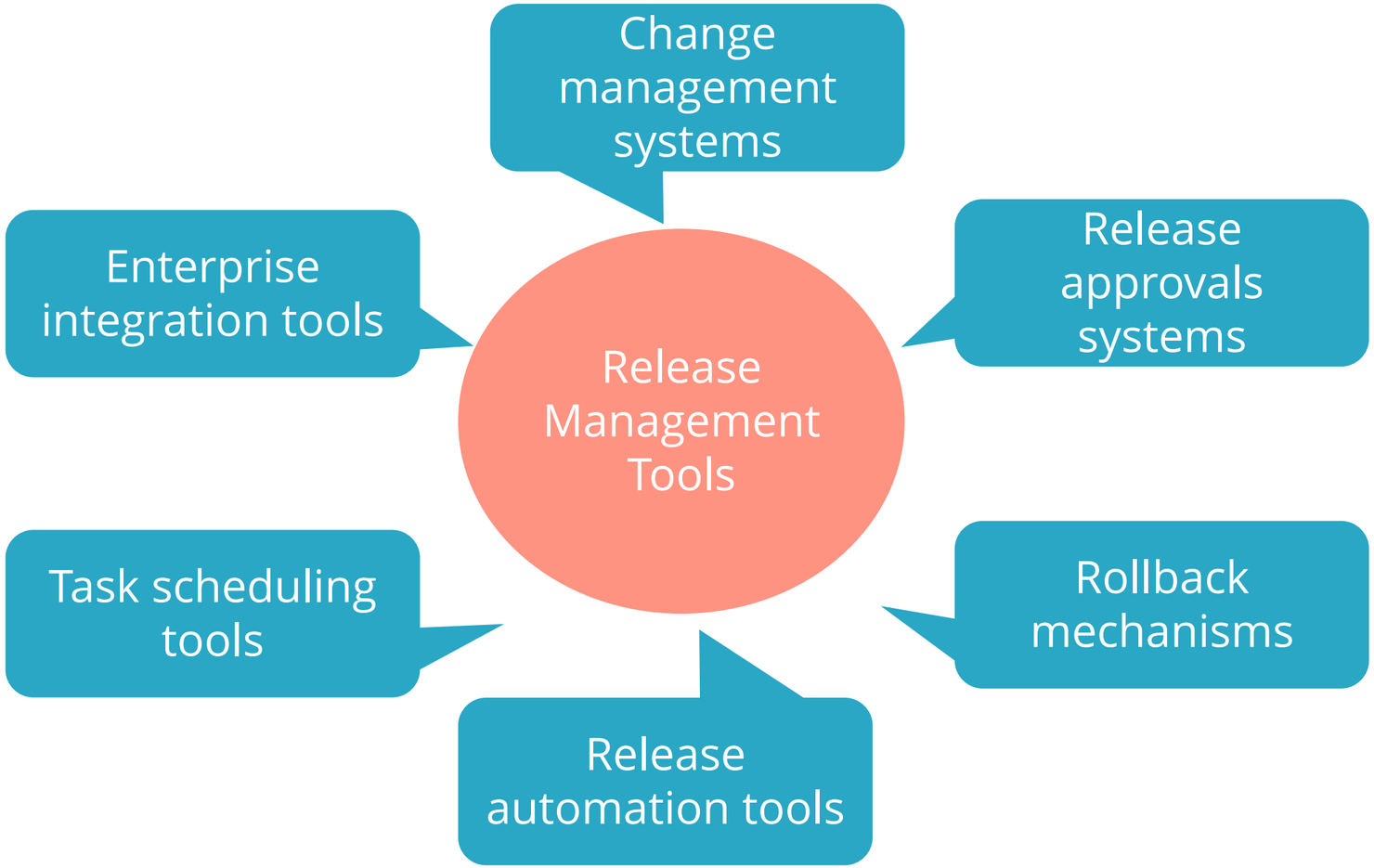
Test Tools



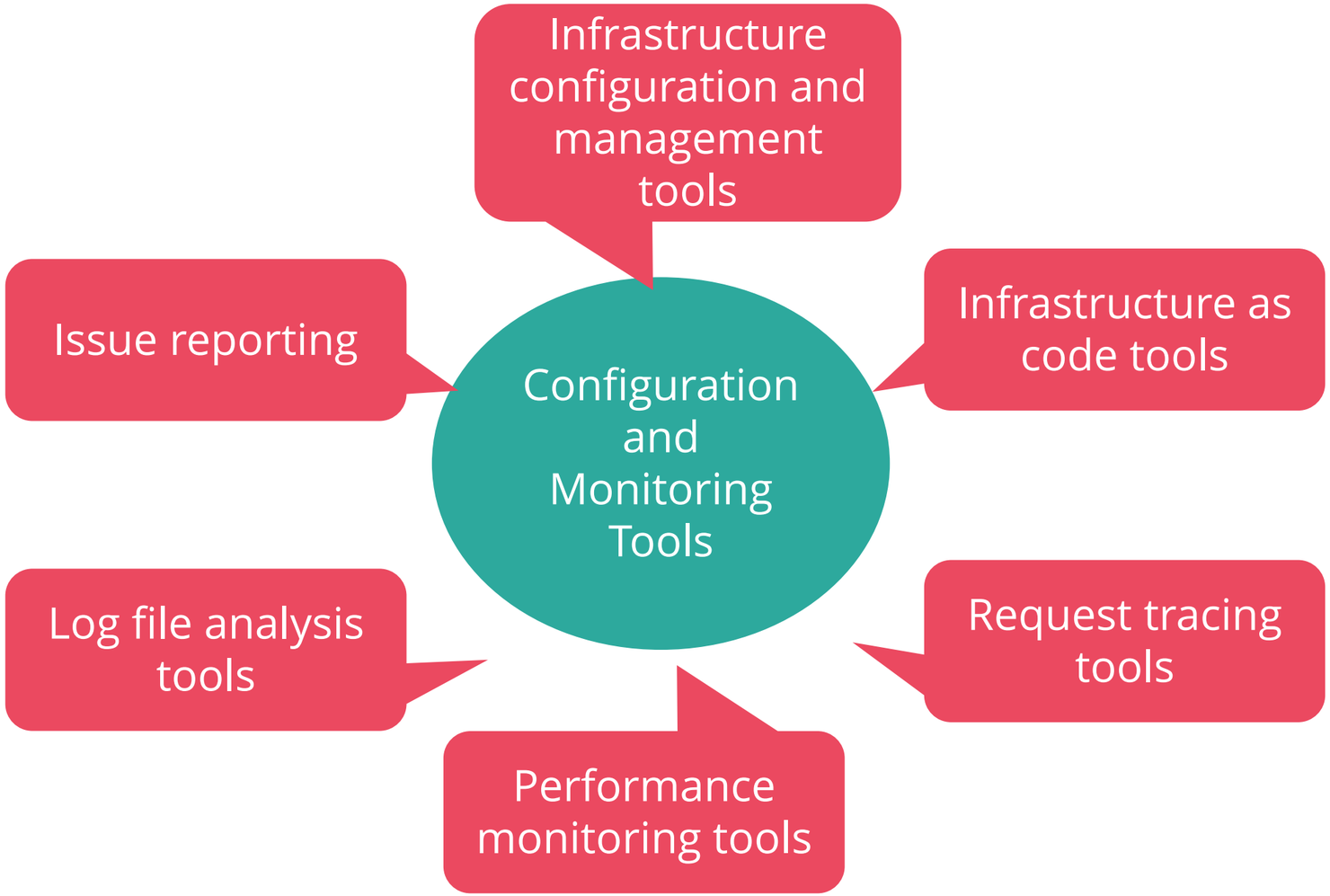
Packaging Tools



Release Management Tools



Configuration and Monitoring Tools



Knowledge Check

KNOWLEDGE CHECK

At which phase would container management tools be required ?

a. Release

b. Test

c. Packaging

d. Code development

KNOWLEDGE CHECK

At which phase would container management tools be required ?

a. Release

b. Test

c. Packaging

d. Code development

The correct answer is **c.**

Containers are created in the packaging phase.

Cloud Computing

Moving into the Cloud

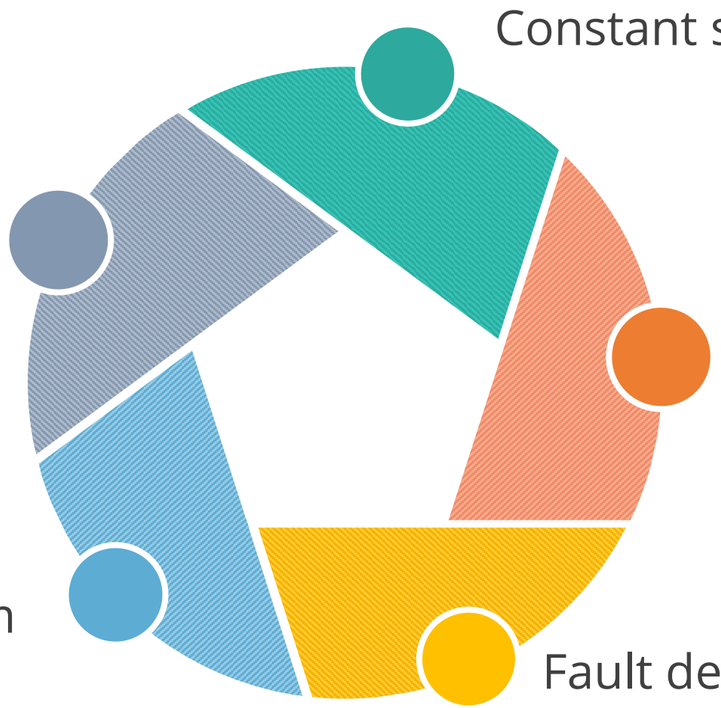
Data Centers

Data centers are expensive to run and maintain:

- Building costs and maintenance
- Hardware cost
- Power costs
- Cooling
- Physical security
- Connectivity
- Maintenance teams
- Security teams, policies, and mechanisms

Failures

- The mean time between failure (MTBF) of a SATA hard drive is 1.2 million hours
- A data center with 100,000 hard drives can expect two drives to fail every day on average!
- What is the impact of this?



Cloud Platforms

- Cloud platform providers have invested heavily in infrastructure
    - Highly scalable
    - Secure
    - Reliable
- They provide a number of services
    - Storage
    - Computing power
    - Services
    - Platforms (as Platform as a Service)
    - Applications
- They focus on doing one thing extremely well – providing a compute platform

More about Data Centers

- Cloud providers have large high-performance data centers
    - Geographically distributed
    - There may be legal requirements related to the location of sensitive data storage
- Access points are available at many locations
    - High speed networks connect access point to data centers
    - High speed fiber cable connect data centers
        - Some Cloud vendors own their own entire fiber optic global network
    - Caching at the edge improves performance

Infrastructure as a Service (IaaS)

- Raw computer power
- Raw storage
- Raw networking
- Virtually build a computer or networked computer system

Infrastructure as a Service (IaaS)

- Payment is for resources provisioned
    - When you use a component, no one else can use it
    - Fair to pay for components requested even if unused
    - Main difference is that virtual components are easy to return to the Cloud vendor
        - Short term “rental” can be very economic
        - Easy to reconfigure to smaller or larger computers

Platform as a Service (PaaS)

Provisioned preconfigured run time environments

Cloud-based software components

Cloud provider responsible for updates to hardware, operating systems, etc.

Customers provide business logic only – reducing and focusing staff

Payment is for resources used.

Software as a Service (SaaS)

Full web-based applications

Maintained by the SaaS provider

- Pay for actual usage
  - Message sent/received
  - Storage of information
  - Other factors

Exercise 1

Set up and examine the Cloud base virtual machine environment

Exercise 1

Demonstrate how to set up and examine the Cloud base virtual machine environment

To set up and examine the Cloud base virtual machine environment, you need to:

1. Create a project in the Cloud on GCP
2. Create a Debian Linux Virtual machine on GCP
3. Give the VM a static IP address and set up SSH on GCP

Knowledge Check

KNOWLEDGE CHECK

Which is an example of IaaS?

a. Web based applications

b. Packaged run time environments

c. Raw computing power

d. Caching

KNOWLEDGE CHECK

Which is an example of IaaS?

a. Web based applications

b. Packaged run time environments

c. Raw computing power

d. Caching

The correct answer is **c.**

IaaS provides raw computing, storage, and networking.

Information Security

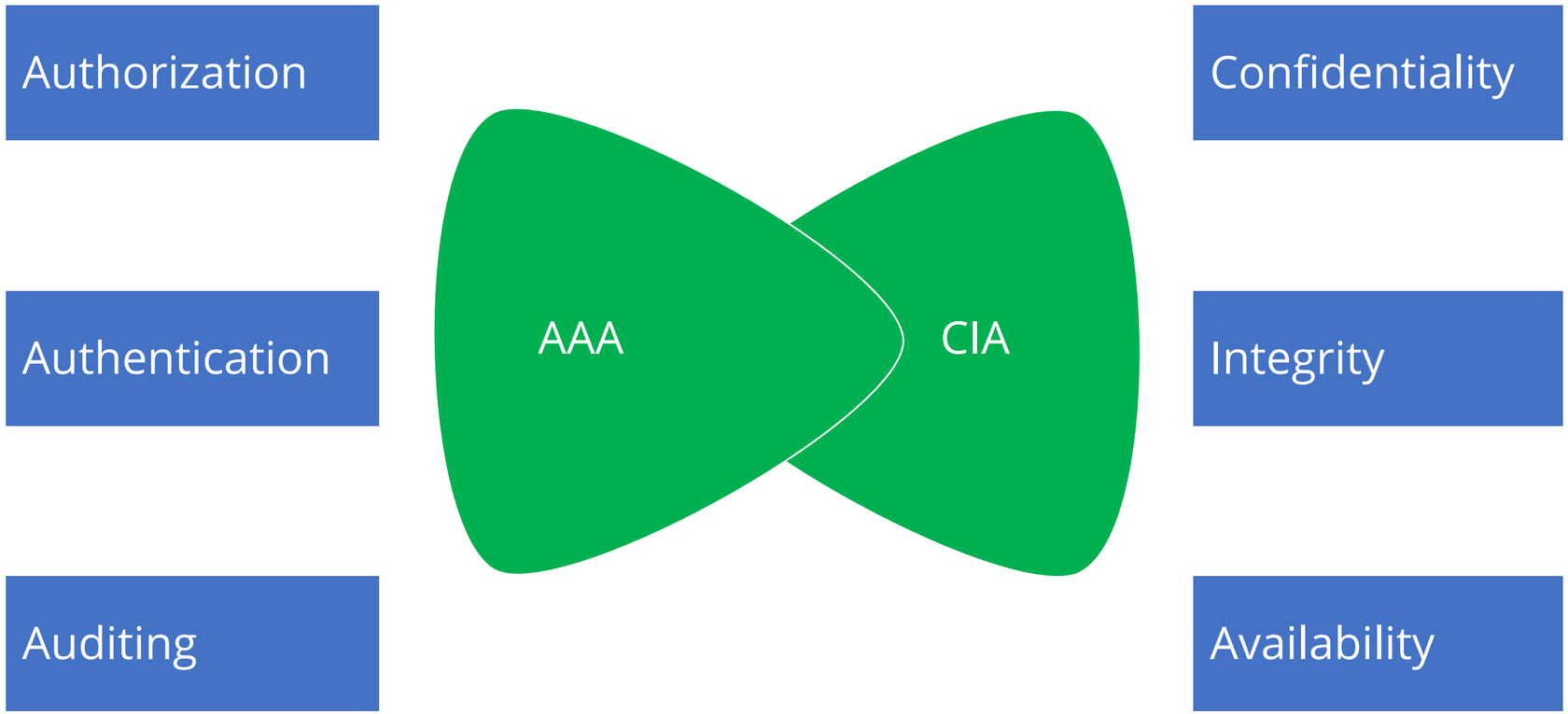
Securing your resources

Security Threats

What are the security threats to a system?

- Physical access
- Social Engineering
- Crackers
- Eavesdroppers and man-in-the-middle attacks
- Viruses and Trojan horses
- Buffer overflow attacks
- Denial of service attacks
- Human error
- Configuration errors
- Malicious employees

AAA and CIA



Security Principles

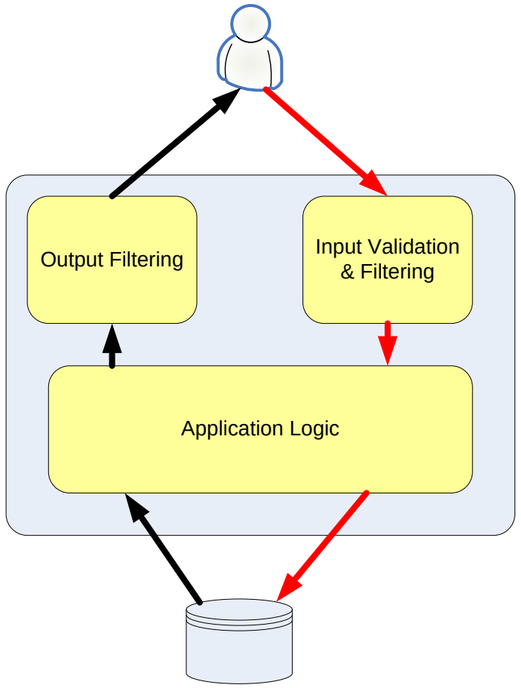
- Information Security is vital
    - Information security teams often need to be engaged during development
- Security comes at a cost
    - It makes systems more difficult to use
- There are a number of security principles which help reduce risks
    - Cover in the subsequent slides

Defense in Depth

- Defense in depth means having several layers of defense
- If the first layer is compromised, the second layer still holds

Defense in Depth (Contd.)

- Cross site scripting (XSS)
    - Accounts for over 80% of web site vulnerabilities
- An email or a web site contains a link to another web site
    - The link contains embedded HTML controls or JavaScript
    - Clicking on the link causes the browser to execute the code
    - It can email sensitive information from the victim's browser
- Web server stores raw user data
    - Need to remove or escape HTML control



Minimize Attack Surfaces

- Minimize the number of entry points into the system
- The fewer entry points the fewer vulnerabilities

Fail Secure

- Fail secure versus fail safe
  - A fail safe door will open when power fails
  - A fail secure door will lock when power fails
  - Systems should fail secure
- What happens when a store's connection to credit card validation systems fails?

Least Privilege

Give programs and users the minimum privileges they need to do their job

- It improves security
- It improves stability
- It requires less testing
- It makes deployment easier

For example, a backup system only needs read access to the data it is backing up.

Security Through Obscurity

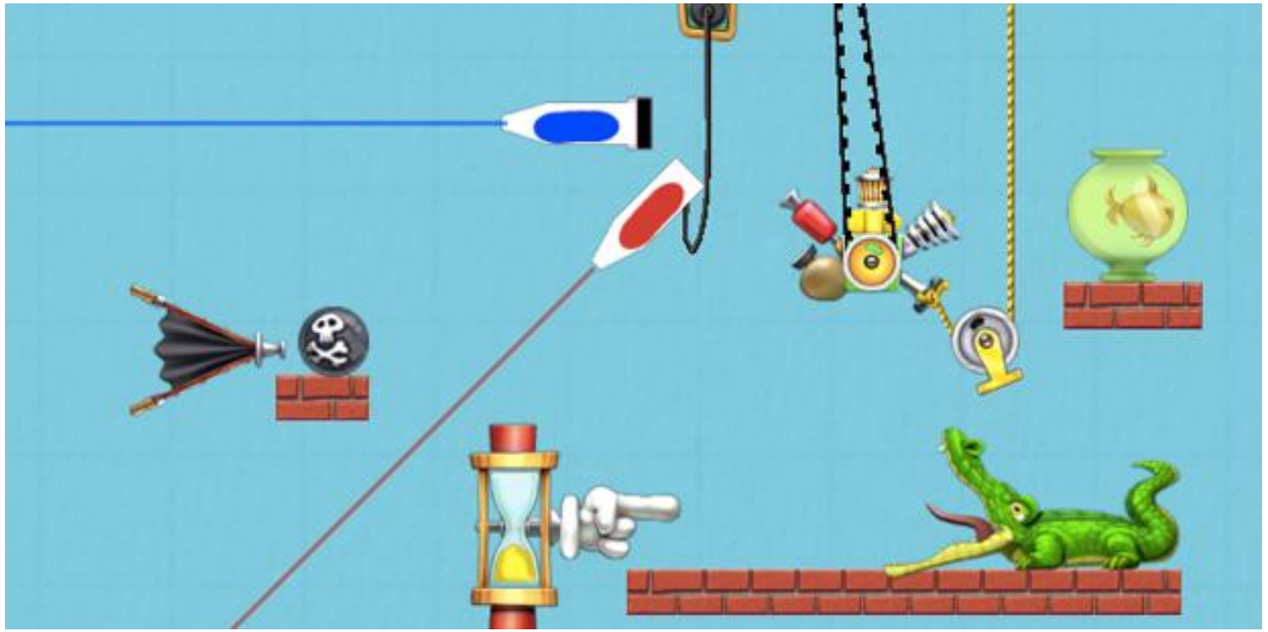
- Hiding something doesn't make it more secure
- Putting a web server on a non-standard port
    - Port scanners will find it
- Cryptographers used to be only government employees
    - The algorithms were secret
- Now all cryptographic algorithms are published
    - Security is in the encryption keys, not the encryption algorithms

Use Modern Languages

- We understand the root cause of many security breaches and language design has changed
- Modern languages make it harder to make mistakes
- The Rust language aims to eliminate memory allocation failures that allow hacks easily created with C and C++
- The Kotlin language aims to eliminate null pointer errors that can crash systems
- REST based system de-emphasize the use of any programming

Keep it Simple

- Simple solutions have fewer security risks
- Over complicated solutions have obscure failure paths
- Sendmail has a very cryptic and complex configuration file
    - It ran 80% of mail servers in 1996
    - It ran 24% of mail servers in 2015 in favor of easier-to-configure alternatives



Segregation of Duty

- Ensure that critical functions require several entities (programs or people) to complete
- Divide a critical function into steps and assign each step to a separate entity

Trust and Complete Mediation

- Trust is when two parties successfully exchange credentials and require no further checks
- Complete mediation is where credentials are checked on every access
- Systems often generate a time limited token on authentication
    - The token is verified on each subsequent access
- Single Sign On is an accepted compromise
    - It eliminates the need for multiple, forgettable passwords

Don't Reinvent the Wheel

- Use existing, known, and reliable security software
  - Writing correct security software is very difficult
  - Building software from scratch will re-implement previously discovered issues
- The UNIX TCP/IP stack has been around a long time
  - It is stable and reliable
- Other OS venders subsequently implemented their own stack
  - These had vulnerabilities which had been discovered and fixed in UNIX

Passwords

- Many people have very insecure passwords
  - Memorable dates
  - People’s names
  - Car registration numbers
  - Based on words
- Passwords are stored as a one way hash
  - Software exists which can find insecure passwords that match a hash
  - Rainbow tables are precomputed hashes of passwords up to a certain length

Securing Passwords

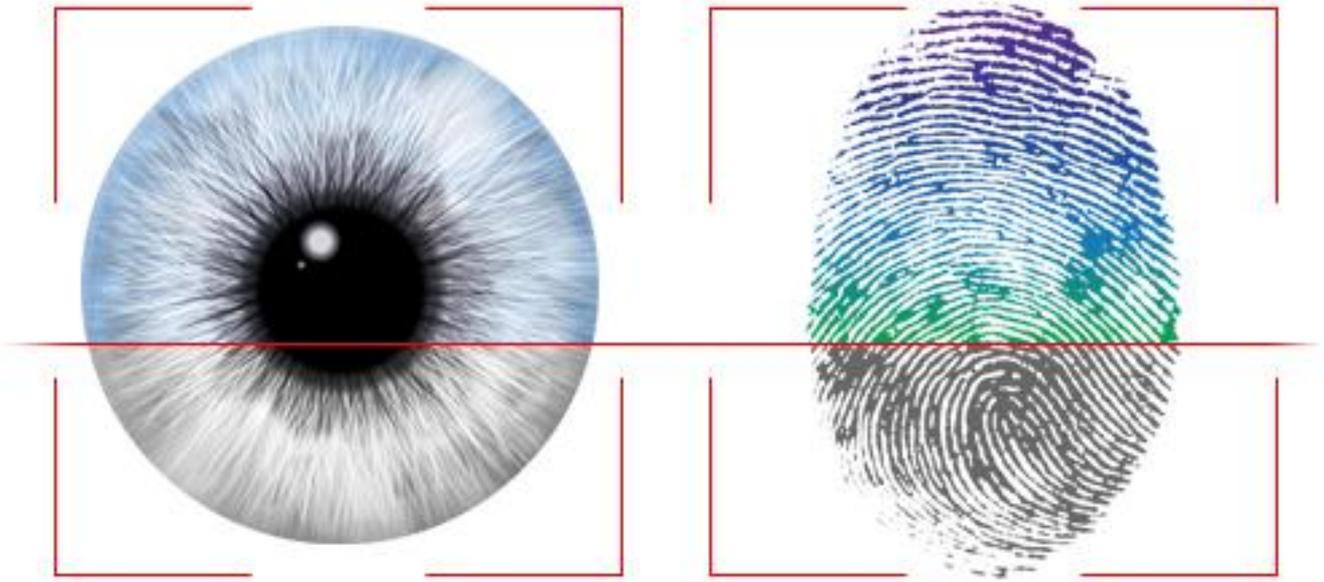
- Salt can help obscure passwords
    - A random fixed size number is added to the password and stored with the resulting hash
    - It reduces the chance that identical passwords will generate the same hash
    - It defends against rainbow table attacks
- There are ways of making secure but memorable passwords
    - Take two short unrelated words and put numbers and symbols in between – bond42!frogs

SecurID

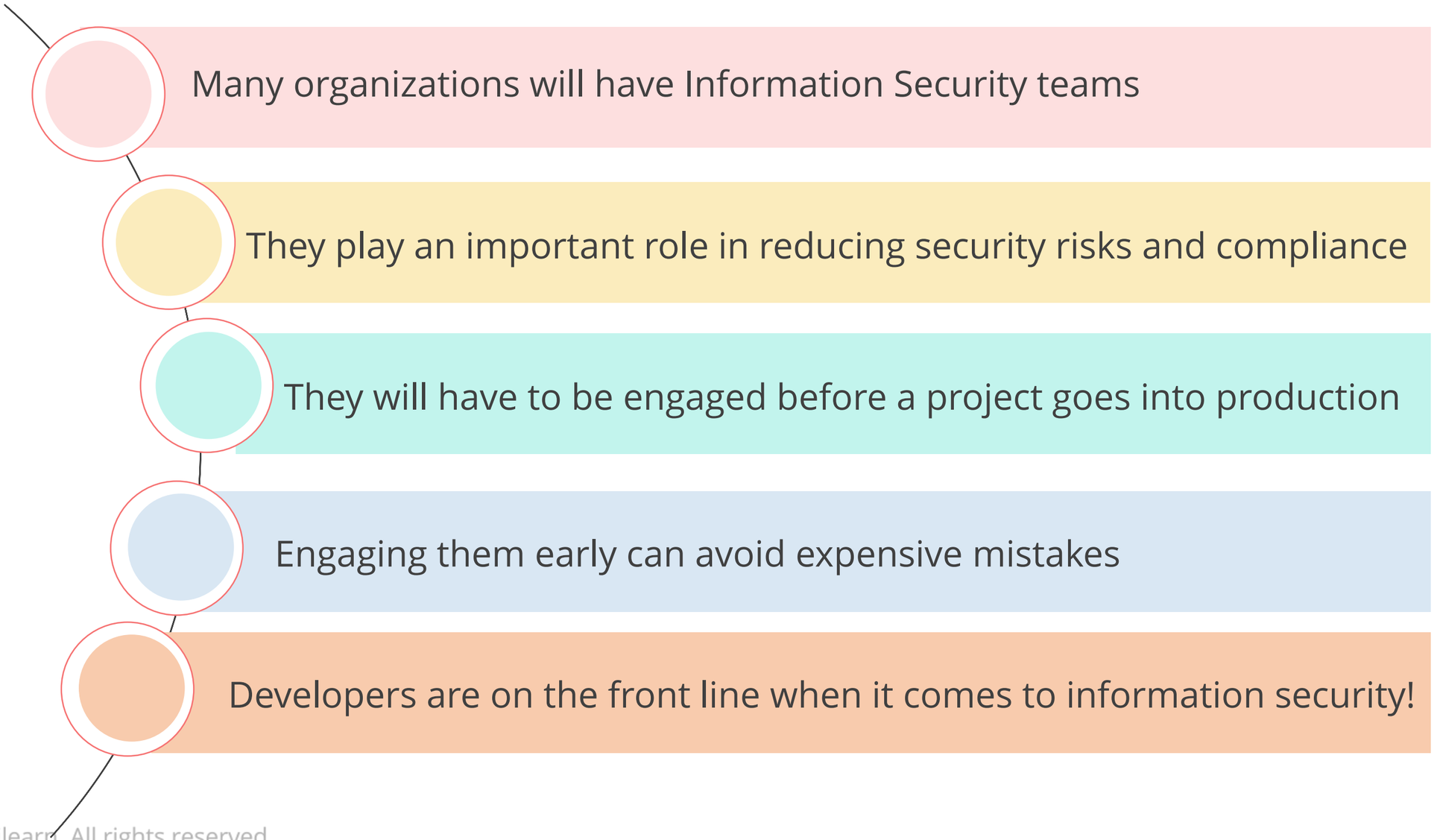
- SecurID is often used for remote authentication
- The user is issued with a device which has a clock and a factory encoded random seed
- The device displays a sequence of tokens that change every minute
- The server has a clock and a database of seeds for each supported SecurID
- The server calculates which token is currently being displayed by the device
- Users authenticate with a personal identification number (PIN) and the currently displayed token
- Problems can occur if the clocks get out of sync, which can be corrected by storing drift values on the server

Biometrics

- Authentication can also be done using an intrinsic characteristic of a user:
  - Finger prints
  - Retinal scans
  - Face recognition
  - Voice prints
  - Typing patterns



Information Security



Knowledge Check

KNOWLEDGE CHECK

What is segregation of duty?

a. Require several entities to complete a sensitive operation

b. Several entities can perform all operations

c. Only one person can perform any particular sensitive operation

d. Giving an entity several different roles

KNOWLEDGE CHECK

What is segregation of duty?

a. Require several entities to complete a sensitive operation

b. Several entities can perform all operations

c. Only one person can perform any particular sensitive operation

d. Giving an entity several different roles

The correct answer is **a.**

Require several entities to complete a sensitive operation

Requirements

Requirements analysis and user stories

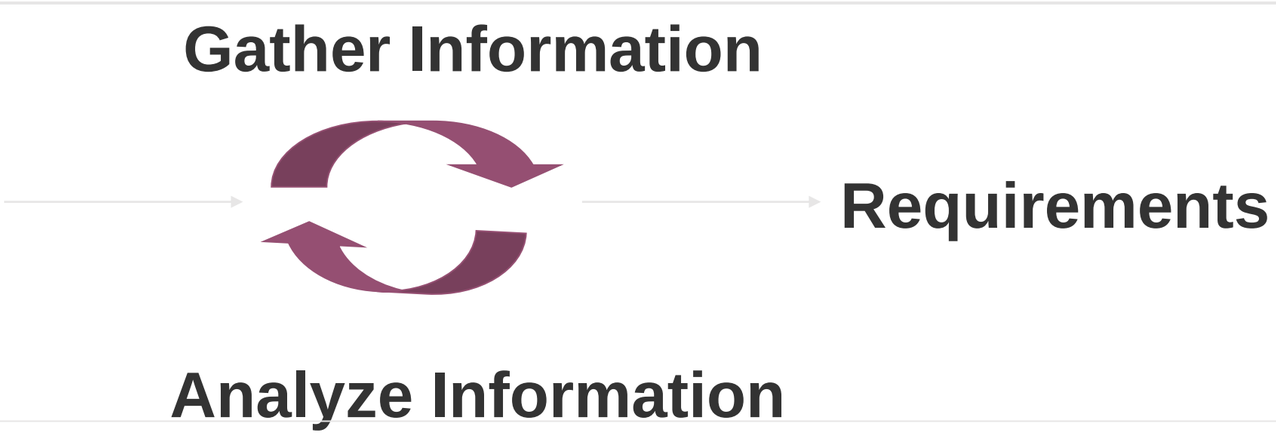
Requirements

From IEEE Standard 610.12-1990:

1. A condition or capability needed by a user to solve a problem or achieve an objective
2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents
3. A documented representation of a condition or capability as in (1) or (2)

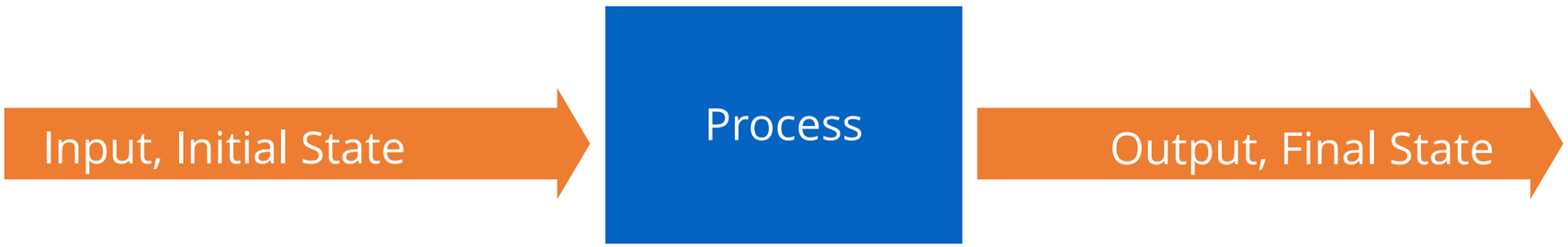
Requirements Definition

- Requirements gathering is an iterative process
  - The problem needs to be defined
  - The requirements are then defined or developed
- Defining requirements is not the same as design



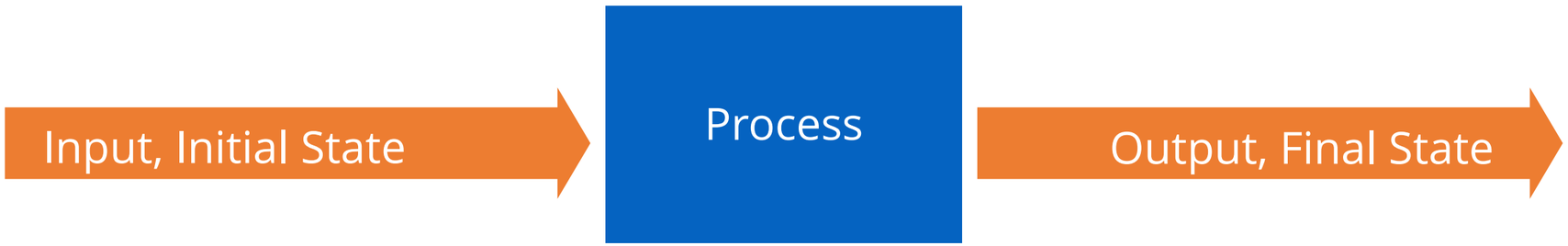
Functional Requirements

- Functional requirements are the relationship between input to a system and the outputs created by this system for the inputs
- There are four parts of a functional requirement
    1. inputs
    2. initial state of the system
    3. outputs
    4. final state of the system
- All four must be specified or the functional requirement is wrong
    - Warning: It is here that assumptions are often made



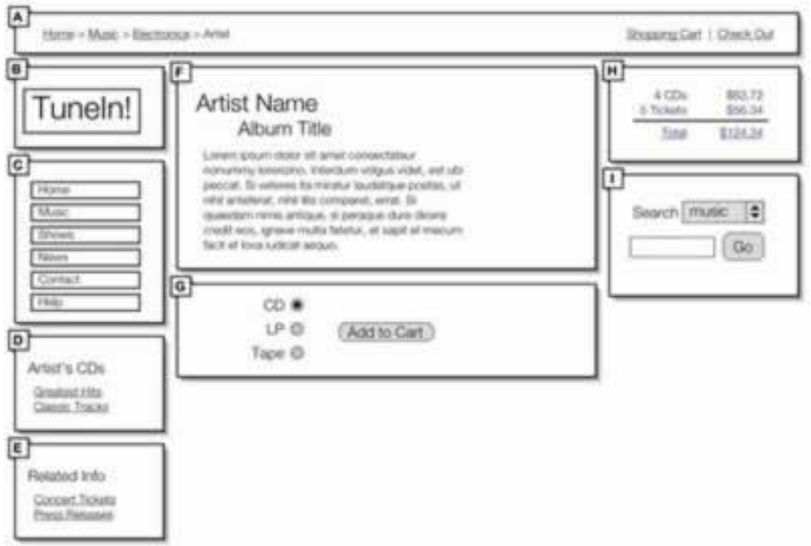
Interface Requirements

- Interface requirements identify all inputs and output to the process
- Interface requirements to other machines or processes are called integration requirements
- Interface requirements to users are called user interface requirements



User Interface Requirements

- User Interface (UI) requirements specify information content of each screen, without showing screen layout
- Correlate screen content with individual functional requirements (use case steps)
- Quite commonly, UI specifications also include screen layouts
    - Wireframe diagrams
    - These are not the final appearance as a UI design will be required



Integration Requirements

- Identifies all sources of input and destinations of output
    - For each source or destination
    - Nature of data flowing on the connection
    - Data transmission rate requirements, if known
    - Transmission protocols, if known
    - Database data volumes
    - Network access requirements
- Integration with third-party components
    - Third-party components must be well documented, with well-defined interfaces at programming level
    - Should be addressed in architecture specification, not in requirement

Performance Requirements

- Performance requirements specify the behavior of the system
- Capacity relates to the number of concurrent sources/destinations connected to the system, and the data capacity of the system database
- Availability is the availability of the system to perform its functions
    - Mean Time Between Failures (MTBF)
    - Mean Time to Restore (MTTR)
    - Availability = MTBF/(MTBF+MTTR) * 100
- Response time is related to the required time between the arrival of a given input and the appearance of the corresponding output
- Throughput is a measure of the amount of data (or the number of transactions or events) that arrive at the input port in a given time interval, and must be handled without error, without data loss, and within the response time requirements.

Assumptions

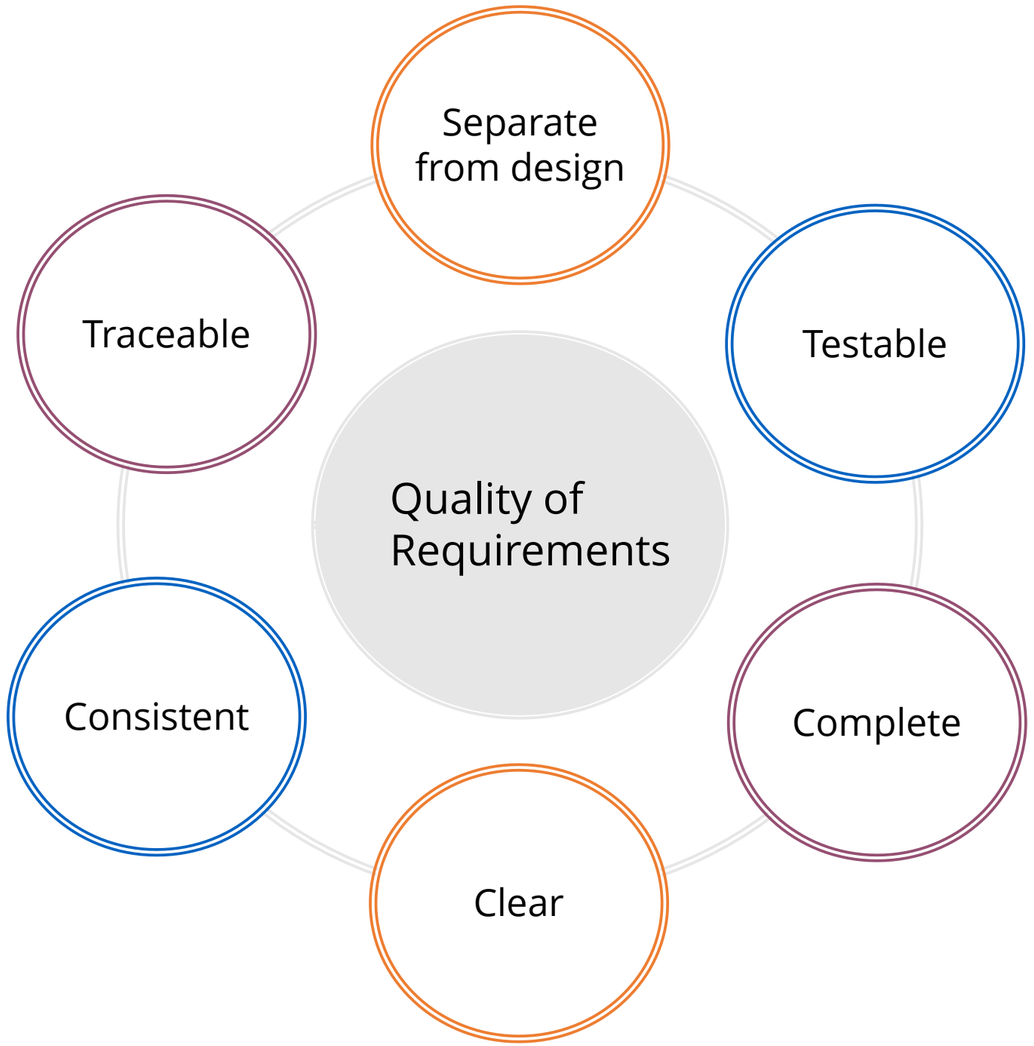
- An assumption is a proposition that is taken for granted, as if it were true based upon presupposition without preponderance of the facts
- Assumptions are often not stated
- Your job is to find the assumptions, state them, and justify them
- Assumptions are everywhere!

Jargon

- Jargon is a terminology which is especially defined in relationship to a specific activity, profession, group, or event
- It is a domain-specific language designed for unambiguous communication
    - Natural languages are inherently ambiguous
    - Lawyers don't use commas as a misplaced comma can change the meaning of a sentence
    - As opposed to slang, which is a language between peers, it enables them to communicate in public without being understood by others
- Developers must think in the domain to communicate with others
    - Use jargon only where the audience understands it

Quality of Requirements

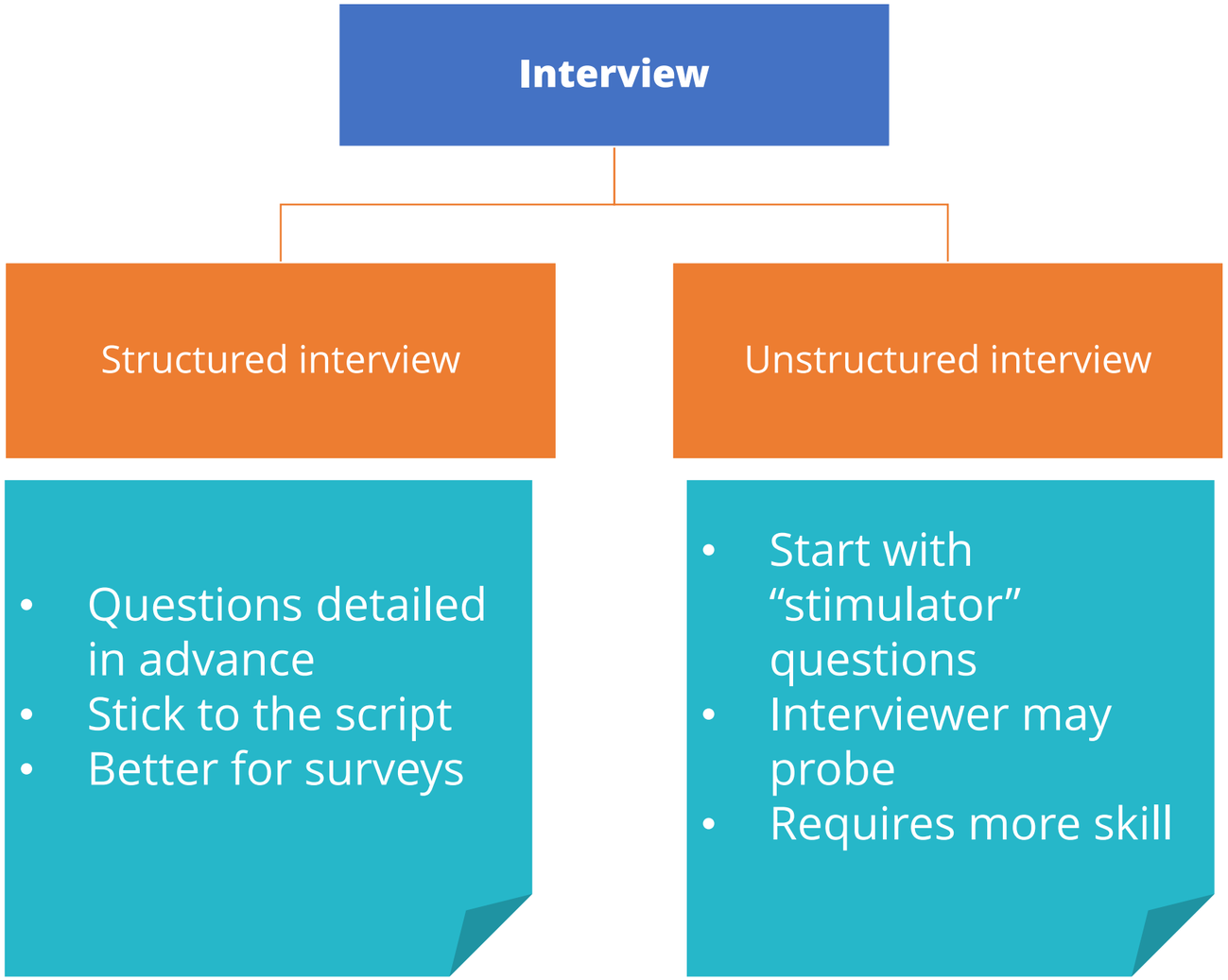
There are a number of quality metrics associated with requirements.

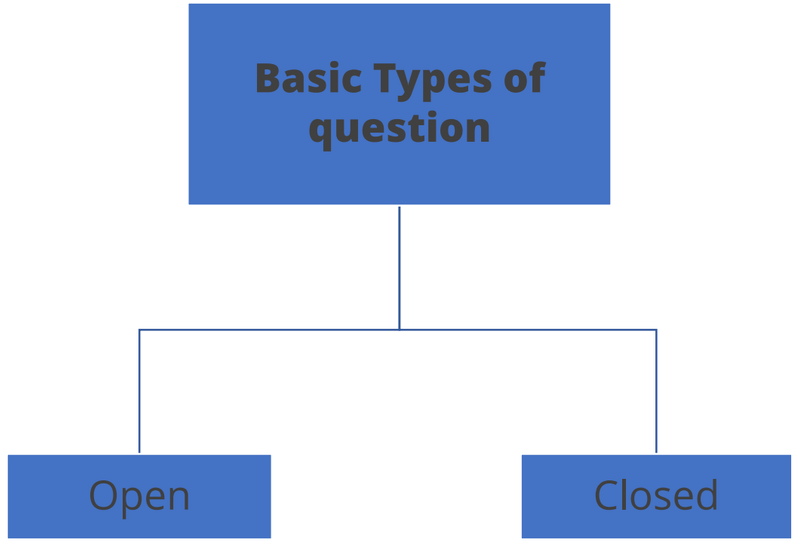


Incomplete Requirements

- Many requirements documents are inconsistent or incomplete
- Never try and resolve issues without consulting the customer or business analyst
- A good way of resolving issues is by means of an interview

Interview





Interview Questions

Situation questions

- What is the justification for the problem?
- What is the vision of the solution?

Problem questions

- What are the underlying problems?

Implication questions

- What does the impact of the solution mean?
- What are the benefits of the solution?

Need-payoff questions

- What are the priorities?

Knowledge Check

KNOWLEDGE CHECK

What is NOT true about assumptions?

a. They must be eliminated

b. They are everywhere

c. They need to be justified

d. They need to be documented

KNOWLEDGE CHECK

What is NOT true about assumptions?

a. They must be eliminated

b. They are everywhere

c. They need to be justified

d. They need to be documented

The correct answer is **a.**

It is not possible to eliminate all assumptions. They must be justified and documented.

Architecture

Joining things together

Architecture

UML

A collection of connected units that are organized to accomplish a specific purpose, which can be described by one or more models, possibly from different viewpoints

IEEE Std. 610.12-1990

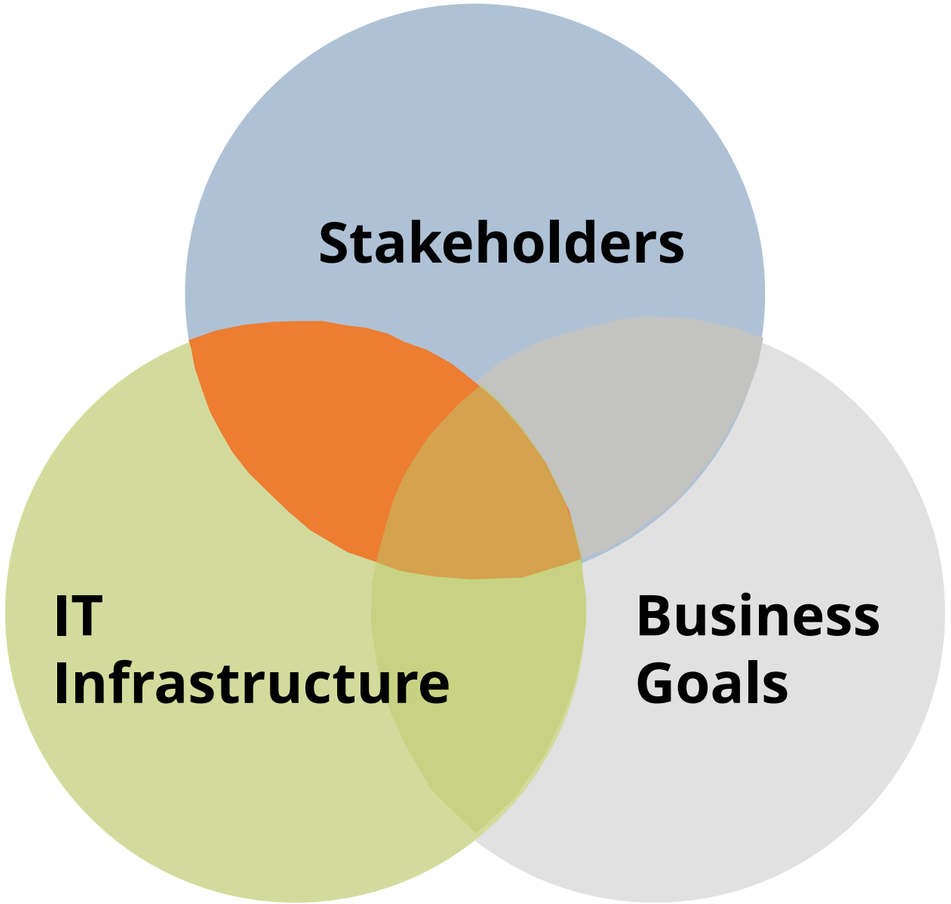
A system is a collection of components organized to accomplish a specific function or set of functions

Software architecture in practice 2nd edition, Addison-Wesley 2003:

The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them

Architecture Goals

- Architecture needs to take into account the goals and requirements of stakeholders, IT infrastructure, and business goals
- IT infrastructure provides the implementation in terms of technology and data



What is an IT Architecture?

- An IT architecture is the set of rules, guidelines, and patterns of system development including:
  - Managing the problem and the solution system as discrete pieces
  - Creating communication interfaces between the discrete pieces
  - Managing the overall structure and flow of data
  - Integrating the system with other systems in the environment
  - Developing software modules based on best practices and design patterns
  - Controlling the delivery of software into the system environment

Architecture versus Design

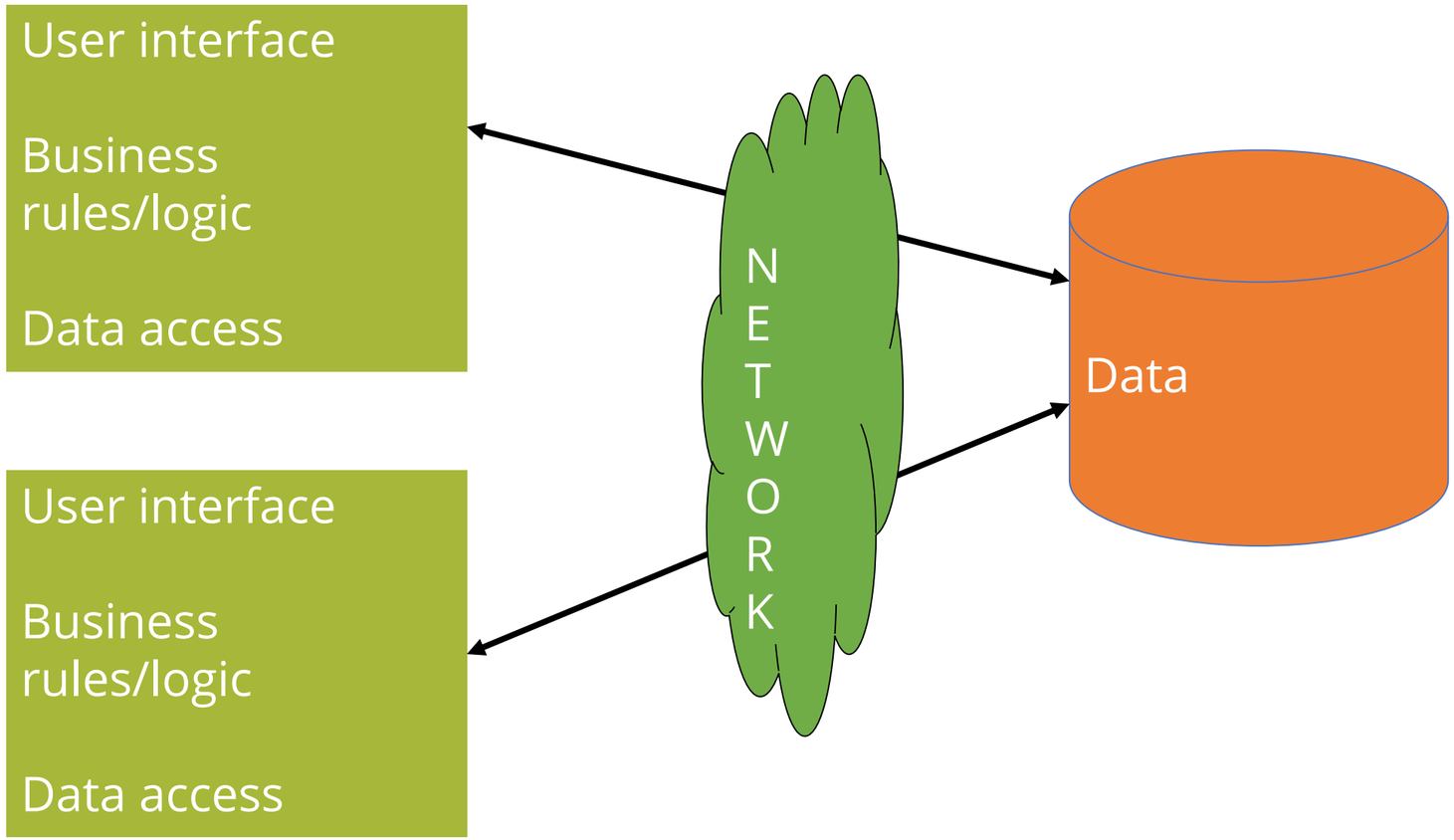
Architecture	Design
Strategic design	Tactical design
Global – “how”	Local – “what”
Programming paradigms, architectural patterns	Algorithms, design patterns, programming idioms
Non-functional requirements	Functional requirements
Represented in UML as component, deployment, and package diagrams	Represented in UML as class, object, and behavior diagrams which appear in detailed functional design documents

Architecture Patterns

- Client Server Architecture
- N-Tier Architecture
- Service Oriented Architecture
- REST Microservice Architecture
- 4+1 Architecture View Model

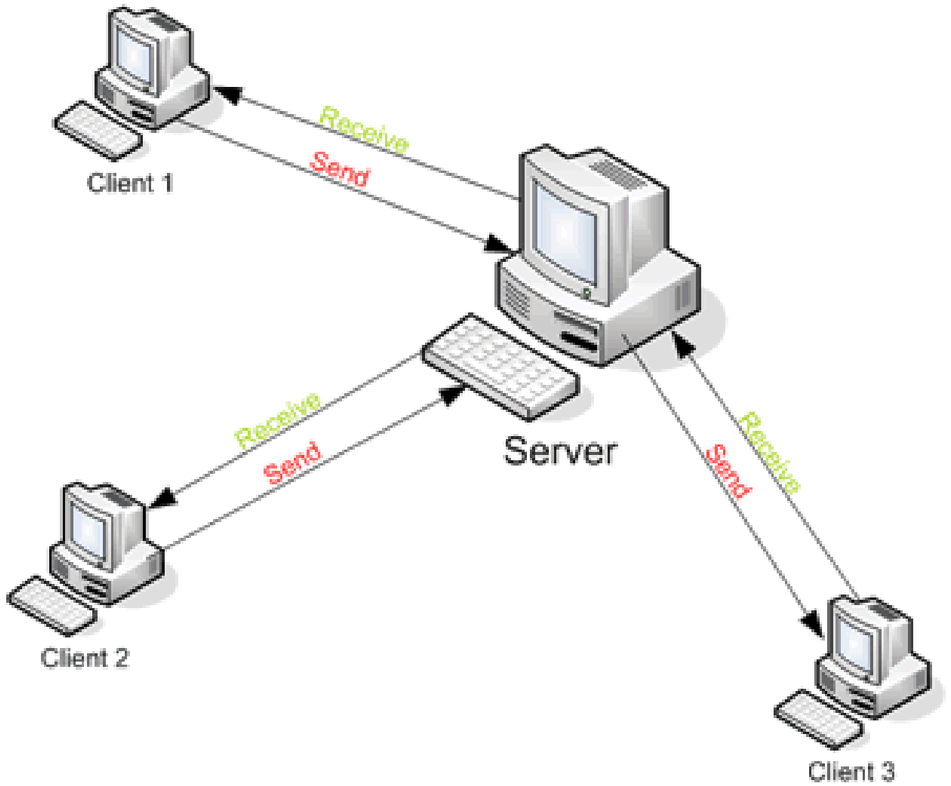
Client Server Architecture

Client server architecture utilizes a thick client communicating with data storage.



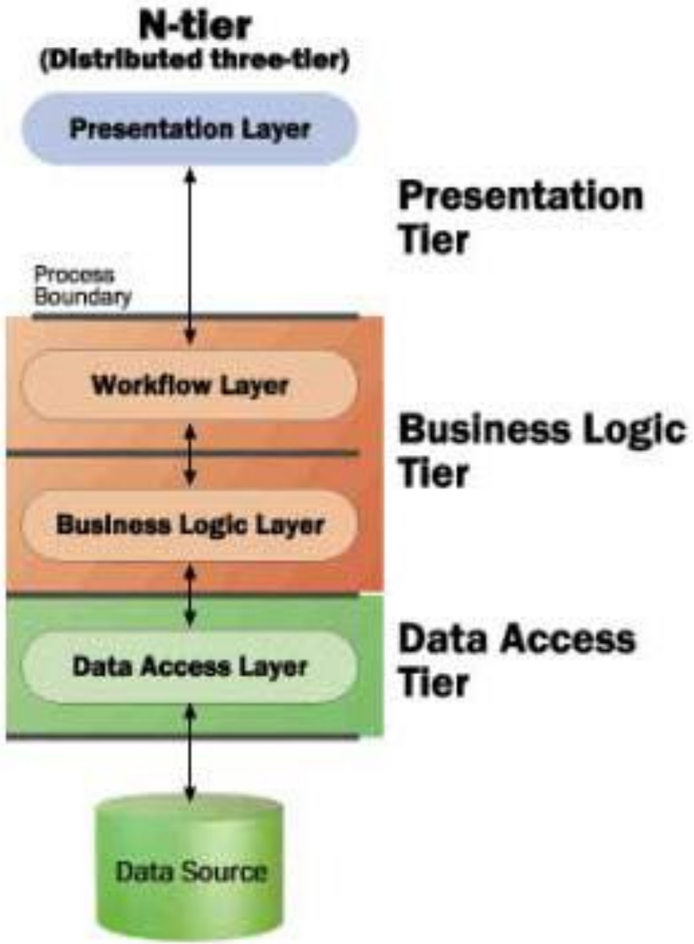
Client Server Architecture (Contd.)

- Advantages
  - Separation of user interface presentation and business logic processing from the centralized data layer
  - Reusability of server components
  - Ease of managing security of centrally located data
  - Optimize infrastructure usage
  - Scalable
- Disadvantages
  - Lack of infrastructure for dealing with requirements changes
  - Security
  - Server availability and reliability as it is a single point of failure
  - Testability and scalability
  - Presentation and business logic in same place



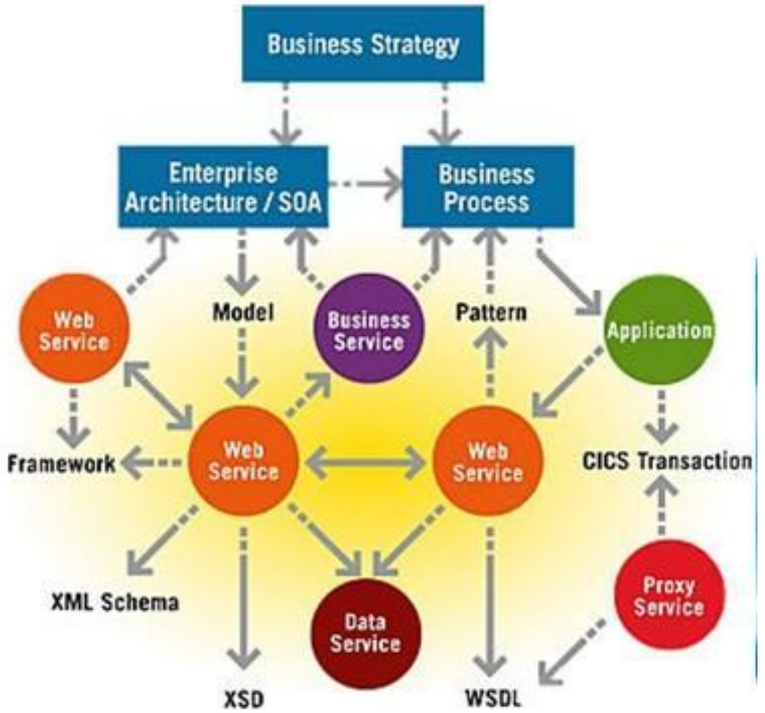
N-Tier Architecture

- Advantages
  - Scalable multi-threaded support
  - No need to update client code for business logic changes
  - Reduced network traffic
  - Can add additional tiers for reuse, failover, and abstraction
- Disadvantages
  - Lack of infrastructure to deal with the requirement changes
  - Security complications
  - Server availability and reliability
  - Testability and scalability



Service Oriented Architecture

- Service Oriented Architecture (SOA)
- Advantages
  - Loose coupling
  - Interoperability—business services across platforms
  - Location transparency
  - Reuse of IT Services—can expose legacy applications
  - Development cost reductions
  - Speed to market
  - Better business and IT integration
- Disadvantages
  - Costly to migrate
  - Need good control system
  - Requires complex service auditing and monitoring
  - Additional development and design

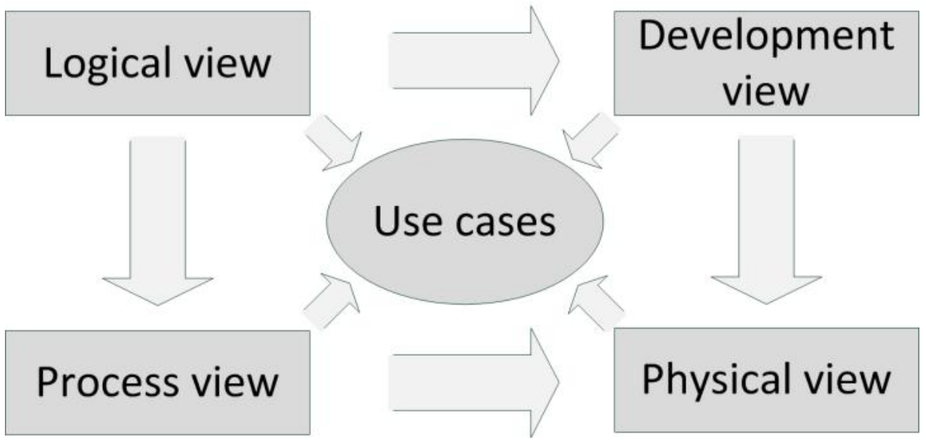


REST Micro-Service Architecture

- REST foundation for all API designs
- Advantages
  - Works the way the World Wide Web works
  - Evolvable architecture
  - Focus on resources and information, not code
  - Separate evolution of components
  - Very high scalability
  - Reuse of micro-services
  - Development cost reductions
  - Makes data/information a first class architectural citizen
  - Separate data semantic from code and place it in the data
- Disadvantages
  - Not a well known approach
  - Few infrastructures support this
  - Requires significant changes in the way developers think

4+1 Architecture View Model

- The 4+1 architecture view model describes the architecture in terms of four different views:
  - Logical view is end user functionality
  - Development view is software management
  - Process view is system processes and communication
  - Physical view or deployment view is software topology on hardware
  - The resulting scenarios or use cases form the +1 view



Knowledge Check

KNOWLEDGE CHECK

What is architecture?

a. Infrastructure which interconnects system components

b. System design

c. The physical layout of hardware

d. A set of functional requirements

KNOWLEDGE CHECK

What is architecture?

a. Infrastructure which interconnects system components

b. System design

c. The physical layout of hardware

d. A set of functional requirements

The correct answer is **a.**

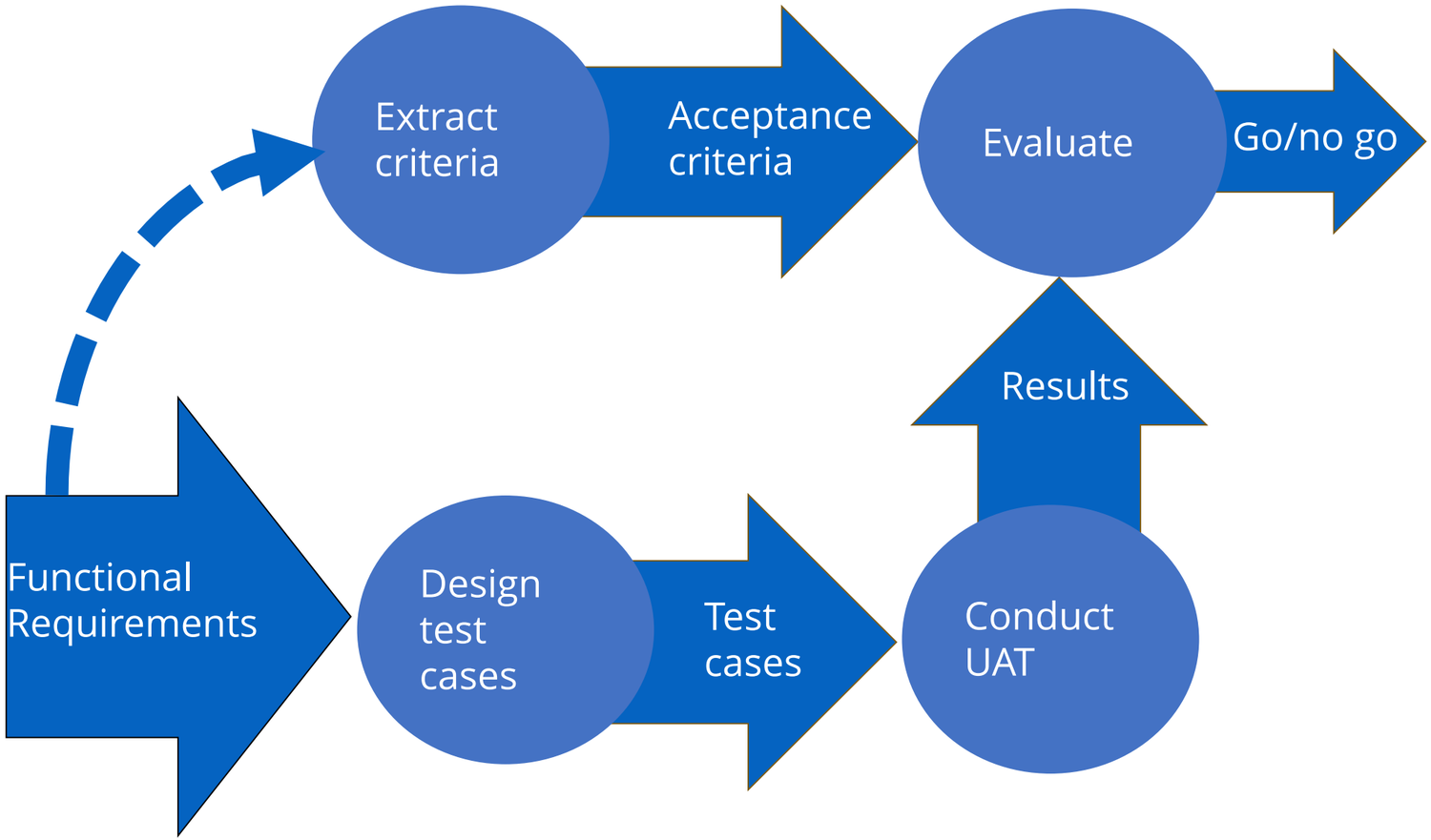
Architecture is infrastructure which interconnects system components. It is often realized as messaging system and associated systems.

User Acceptance Tests

Have we built the right thing?

User Acceptance Tests

- A formal set of tests conducted to determine whether or not a system satisfies its acceptance criteria
    - Enables the customer to determine whether or not to accept the system
- Acceptance criteria
    - Fitness for purpose
    - Solves the user's problem
    - Works with all interfacing systems



UAT Design

- Ideal acceptance testbed is a clone of the production environment
    - Changes in production are mirrored in testbed
    - Minimize risk of compatibility issues during transition from acceptance test to production
- Make compromises to satisfy resource constraints
    - Restricted number of remote users for a distributed system – or none at all
    - Focus on functionality and information flow, not performance
    - Simulate external interfacing systems
    - Populate databases fully with recent production data
- Consider synthesizing personal information to make sure private data is not compromised

For example, Replace all customer names with “John Doe”;
Use your company address as shipping addresses for items to be shipped

Use Cases and UAT

- A structured use case description is written as a test case
- Each step has:
  - Initial conditions (pre-conditions)
  - Inputs
  - Expected outputs
  - Final conditions (post-conditions)
- Each use case must be instantiated for execution in acceptance testing

Use case says, “Customer selects a product code for purchase at current market price and designates the number of items to purchase”

UAT must specify the product code, and other details of the purchase transaction (current price, exact quantity purchase, etc.)

Use Cases and UAT (Contd.)

- UAT designer must establish initial values for all system database elements at the start of the UAT session
  - Roadmap dictates execution sequence for use cases
  - UAT design must specify values of all input fields in all use case steps
  - Data values in expected outcomes must be similarly specified, step by step, for each use case
  - Also specify final content of database at the end of UAT to show that system database changed as expected during UAT
- Additional details are provided in system design:
  - Screen layouts including data formats
  - Instructions for entering input
  - Users must be trained in advance of UAT
- Combine use case scenarios, database and transaction values, and design details to produce user acceptance test scenarios

Traceability

- There needs to be traceability between Use Case and User Acceptance Tests
- Create a traceability matrix

	R1	R2	R3
T1	X		
T2	X		
T3	X	X	
T4		X	
T5		X	
T6			X
T7			X
T8	X	X	X
T9		X	X

A traceability matrix is an index to a parent and child specification

R	T
R1	T1
R1	T2
R1	T3
R1	T8
R2	T3
R2	T4
R2	T5
R2	T8
R2	T9
R3	T6
R3	T7
R3	T8
R3	T9

T	R
T1	R1
T2	R2
T3	R1
T3	R2
T4	R2
T5	R2
T6	R3
T7	R3
T8	R1
T8	R2
T8	R3
T9	R2
T9	R3

Testable User Stories

- Agile development often specify requirements in terms of user stories
- Effective user stories need to be testable
- Features of effective user stories are:
    - It needs to describe an action which has value to a specific user
    - It needs to target a specific user or role
    - It needs to have clearly stated acceptance criteria which can easily be tested
    - It needs to be small enough to implement in a few days
    - It needs to be short and precise

Case Study

- An example user story for the case study. Is it testable? Can it be improved?
- As a Docker administrator, I need to be able to obtain a list of all containers that terminated in the last hour.
- The output web page needs to contain the event type, the container name, and the date and time of the event sorted in reverse order by event time.

KNOWLEDGE CHECK

Tests to verify that every functional requirement has been implemented

a. Testing that the user interface works

b. Testing that system components works together

c. Testing user interface usability

d. What are user acceptance tests?

KNOWLEDGE CHECK

User acceptance tests are tests to verify that every functional requirement has been implemented.

a. Testing that the user interface works

b. Testing that system components works together

c. Testing user interface usability

d. What are user acceptance tests?

The correct answer is **a. Tests to verify that every functional requirement has been implemented**

Tests to verify that every functional requirement has been implemented

Key Takeaways

- DevOps requires a cultural change to improve quality and reliability.
- There are many constantly changing technical challenges facing DevOps.
- There are a number of categories of software tools, each with a number of choices.
- Cloud computing eliminates the need for expensive data centers and supporting groups.
- Information security is important to protect sensitive assets.
- Architecture is structure that defines how systems communicate and work together.
- It is important to ensure that requirements are complete and consistent.
- User acceptance test are essential to ensure that all functional requirements have been correctly implemented.

Quiz

QUIZ 1

What is DevOps?

a. An integrated development environment

b. Collaboration between developers and operations

c. A form of software development lifecycle

d. A coding methodology

QUIZ 1

What is DevOps?

a. An integrated development environment

b. Collaboration between developers and operations

c. A form of software development lifecycle

d. A coding methodology

The correct answer is **b.**

DevOps is the collaboration between developers and operations. Developers need to work with operations to achieve optimal results.

QUIZ 2

What is defence in depth?

a. A type of firewall

b. A design strategy

c. An encryption algorithm

d. Layered applications that limit the scope of an intrusion

QUIZ 2

What is defence in depth?

a. A type of firewall

b. A design strategy

c. An encryption algorithm

d. Layered applications that limit the scope of an intrusion

The correct answer is **d.**

Layered applications to limit the scope of an intrusion. N tier architecture means that the data has several layers of protection above it.

QUIZ 3

What is a rainbow table?

a. A form of database construct

b. A lookup table of hashes of common passwords

c. A means of representing requirements

d. A software data structure

QUIZ 3

What is a rainbow table?

a. A form of database construct

b. A lookup table of hashes of common passwords

c. A means of representing requirements

d. A software data structure

The correct answer is **b.**

Rainbow table is a lookup table of hashes of common passwords used to break password security.

QUIZ 4

How should assumptions be treated?

a. They must be justified and documented

b. They must be eliminated

c. They are by definition implicit

d. They must form part of tests

QUIZ 4

How should assumptions be treated?

a. They must be justified and documented

b. They must be eliminated

c. They are by definition implicit

d. They must form part of tests

The correct answer is **a.**

Assumptions must be justified and documented.

QUIZ 5

What is a traceability matrix?

a. A data structure

b. A mechanism for locating employees

c. A design pattern

d. A mapping between functional requirements and UAT

QUIZ 5

What is a traceability matrix?

a. A data structure

b. A mechanism for locating employees

c. A design pattern

d. A mapping between functional requirements and UAT

The correct answer is **d.**

Traceability matrix is a mapping between functional requirements and UAT to ensure that all requirements are tested.

Capstone Project

Introducing a capstone project which will be used during the course.

Capstone Project

- We are going to introduce a capstone project for use in the course
- DevOps techniques and tools will be used to take the project from requirements to deployment
- Part of the project will be implemented during the course
- A large combination of technologies and tools can be used for the project
- A typical set of technologies and tools will be chosen as examples

Scenario

- A large organization has a requirement for a monitoring and reporting tool. It will need to monitor other tools by means of their APIs.
- The first two tools to be monitored will be Docker and Jenkins, both of which have REST APIs to access their functionality.
- The application will poll the other tools' APIs on a periodic basis and examine the output for important events. Each event will be stored in a relational database.
- Events are classified as normal operations, warnings, and error conditions.
- The application has a web-based front end. The web interface allows filtering by tool and by time interval.
- The web interface also allows alerts to be set up. An alert is triggered when a tool generates an event of a certain type, usually an error condition. An email is sent whenever an alert condition is detected.
- The application is self monitoring and generates its own events.

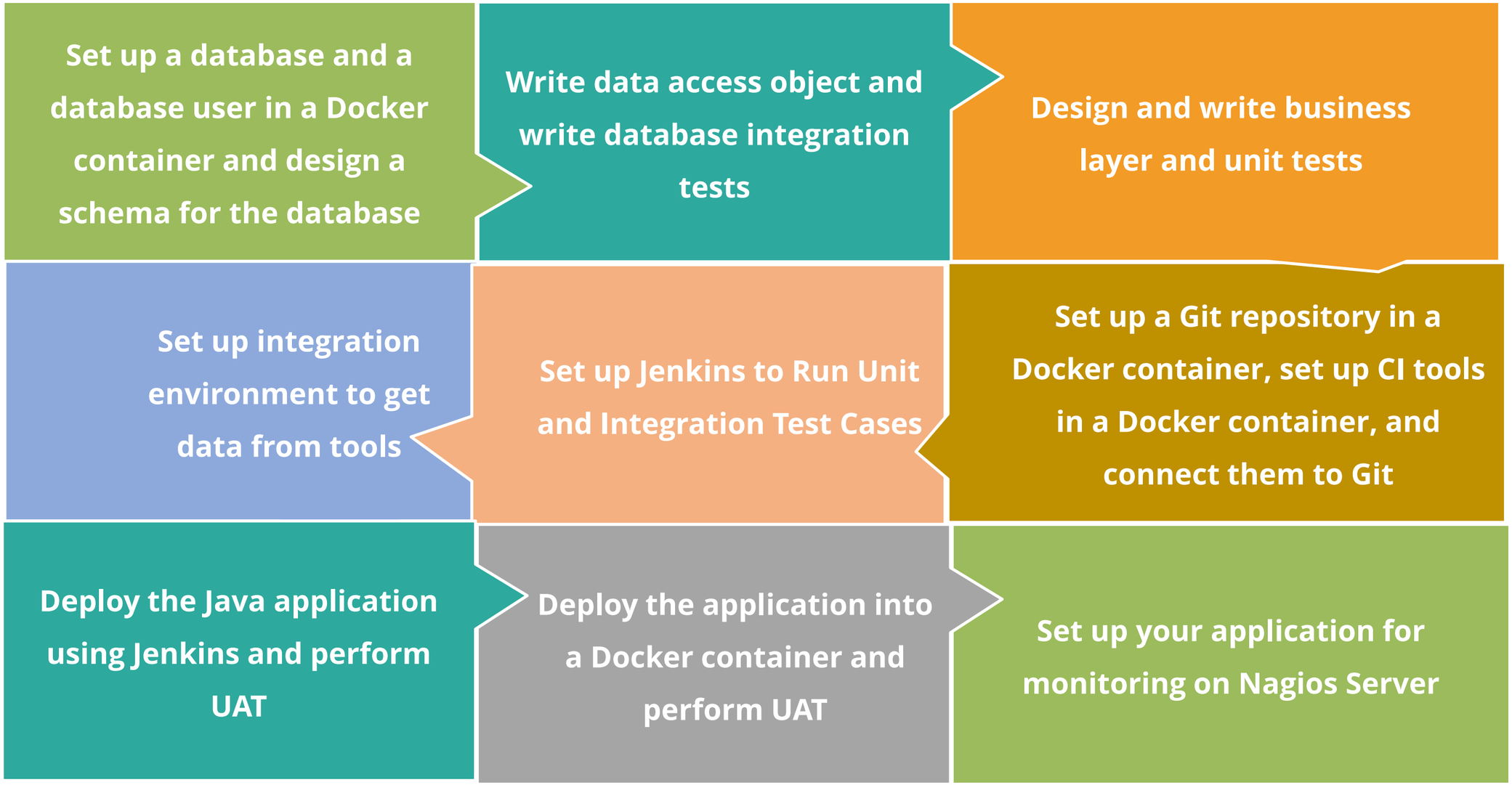
Capstone Project—Definition

We are going to revisit the DevOps process using the scenario we looked at earlier.

- The requirement is to build a monitoring tool that will interface with Docker, Git, and Jenkins and produce reports on activities.
- The application will be written in Java and use Apache Camel as the integration tool.
- Use Jenkins to create build, deploy jar file, and trigger the unit test cases that are part of your project and for deployment of your Java application.
- Set up Nagios Server to monitor your Java application.

Capstone Project Steps

DevOps process steps required for the case study are:



Capstone Project Important Points

- Tools and technologies should not be considered in the early stages of a project.
- Some tools and technologies may be dictated by the corporate environment. These should simply be stated as requirements.

This concludes “DevOps Overview.”

The next lesson is “Containerization Using Docker.”