SWE40006
Software Deployment and Evolution

Lecture 1

Lecture overview

This lecture will cover:

- **Learning Objectives**
- Motivation for this unit
- Assessment & Teaching Method
- In a Nutshell – Software Deployment and Evolution
- Environments

Learning objectives

1. Analyse the activities in software deployment and apply these in a problem context.
2. Apply knowledge of software environments to plan development and deployment.
3. Analyse and classify issues that drive software maintenance.
4. Reflect on latest practices in software development, deployment and evolution.
5. Draw on industry practice to evaluate the maintainability and design consistency of a software system.

Motivation

Situation- Your team has just finished development of a software system

- What are the next steps?
- This unit focuses on these “next steps"
- Typically ..... the team would:
  - Publish - Advertise - Distribute
  - Customers buy (hopefully) and install
  - Users use the software
  - And they ....

And they lived happily ever after?

Dina Goldstein, 2009

http://www.jpgmag.com/stories/11918

Lecture Roadmap

This lecture will cover,

- Learning Objectives
- **Motivation - Deployment**
- Motivation - Evolution
- Assessment & Teaching Method
- In a Nutshell – Software Deployment and Evolution
- Environments

Software Deployment

- What are the activities that ensure that software can be deployed successfully?
- Once it has an active user base ....
- What tends to happen?
- How can it be maintained effectively?

Life of the maintenance crew
Source: http://bit.ly/qq2u5

Deployment

Software deployment is a little bit more than worrying about installation scripts ...

Consider the following:

- A user downloads Crystal (a hypothetical photo editor) and runs the installer which successfully completes its task
- When the user runs it, the application reports a cryptic “DLL version error" and then quits!
- What just happened? Can this be avoided?

Deployment - Reality

- User wants to upgrade from MYOX 1.0 to 4.0
  - Prefers to retain all existing configuration
  - Expects data to migrate properly
  - Expects scripts to continue to work properly
  - Version 1.0 was released in 2010
  - Version 2.0 was released in 2019
  - Version 3.0 was released in 2024
- Is this reasonable?
- Can we do anything about it?
- Does MSWord provide this kind of support?

Deployment in the Real World

- You are part of a team that developed “Technomaly" - a Customer Relationship Management system
- You just released Version 2.2 (upgrade from v2. I)
- The new version has 3 new fields in the customer form, the upgrade has been rolled out to 20 existing customers (5000 users in total)
- The system has been in operation for 3 months
- After this period of time, a major flaw was identified when the “end of quarter" reports were executed -- What do you do?

Deployment - Larger Scale

- A new product called “NiNjA NetBank" has been purchased by ANX bank. This is an online bank module that claims to fully integrate with existing banking systems
- What is involved in rolling this out to customers?
- Can you create an installer script that will do everything and work out of the box?
- Can you just integrate with every bank in the world?
- Will standards help? Are they followed?

Deployment - Expectations

- NiNjA NetBank is almost perfect -- it works well. Its transactional reliability is rated at 99.99% per operational server.
- ANX bank will process 10 million transactions a day across the world
- Will you deploy this product?
- Is this acceptable?
- Can you deploy it to ensure that there is sufficient reliability?
- Is it worth pushing for additional reliability?
  - What do you think real banks do?

Not acceptable – 1000 accounts will be corrupted.

Demise of the NiNjA

- ANX bank wants to remove NiNjA NetBank after 3 years of use -- and wants to use a new product called Shadow NetBank
- Data is stored in a proprietary binary format
  - What now?
- Data is available as an XML export -- is this good enough?
- 8 Terabytes of data needs to be migrated, verified and checked to be accurate -- how do we do this?

Deployment - Games

- You have been playing “Blob from Mars" -- an exciting fast paced game for the last 2 weeks.
  - You have played one of the levels in the game for 30 hours -- you have saved it, and are yet to complete this level
- An upgrade has been released, with new levels -- new features -- new gizmos and du-dads.
- Should you be able to play a saved game after the upgrade?
- Can you think of a game that trashed saved games?
  - Why does this happen? What if it was a security patch that causes this to happen?

Need to be able to migrate from one version to the next

Deployment - Hardware

- You have upgraded your computer by adding a new high-end sound card
  - When the computer is re-started, what should the expected behavior of the O/S be to a change in the external environment
- Changes to external environment are very common in large enterprises
  - External software systems, libraries, data formats, hardware, protocols, files, CPU architectures etc. == all of these change

Who is responsible?

- Customers are NOT likely ever to worry (or) mention ‘deployment’ related requirements
  - You may get broad environment specifications (e.g. we use big and expensive IBM servers)
- Developers as well as Operations team needs to request, clarify, frame and articulate deployment requirements
- These are non-functional requirements, but can have a substantial impact on the architectural choices

What is an Operations Team?

Many IT organizations have three distinct technical focused groups:

1. Operations team: Deploy, Upgrade, Support, Raise and Track Issues
2. Development team: Analyze, Design, Construct, Deploy and Maintain
3. Research team

Why the split?

- Knowledge and Skill levels are different

- *Talk to all teams to gather deployment requirements — they are all stakeholders*

Both Dev and Ops teams do deployment. Dev team need to do it to patch/update/fix systems. Ops team monitor use, performance, complaints, feature requests and scalability. Makes sense to combine them.

Lecture Roadmap

Where are we?

- Learning Objectives
- Motivation - Deployment
- **Motivation - Evolution**
- Assessment & Teaching Method
- In a Nutshell – Software Deployment and Evolution
- Environments

Does Software Evolve?

- Software evolves -- in order to stay useful
- But, why study it?
  - How do developers make changes?
  - What type of modules get modified most?
  - Do developers avoid complexity?
  - Can software grow exponentially in size?
  - Are the changes large in size or small?
  - How often are architectural shifts made?
  - Does the eco-system matter?

Software Evolution - Why study it?

- You just joined a development company and will work on software that has been in development for 3 years.
- What can you expect to find?
- What would have happened as this software was built?
- What is the typical software change profile?

Software Evolution - How can it help?

- You need to select a third-party component for use in your application
- There are three possible libraries (all offer the features that you need)
- Would having access to how they evolved give you an indication about the development team?
- Are libraries that undergo frequent change good?
- Are libraries that undergo regular architectural re-wiring good?

Lecture Roadmap

Where are we?

- Learning Objectives
- Motivation - Deployment
- Motivation - Evolution
- **Assessment & Teaching Method**
- In a Nutshell - Evolution and Deployment
- Environments

References

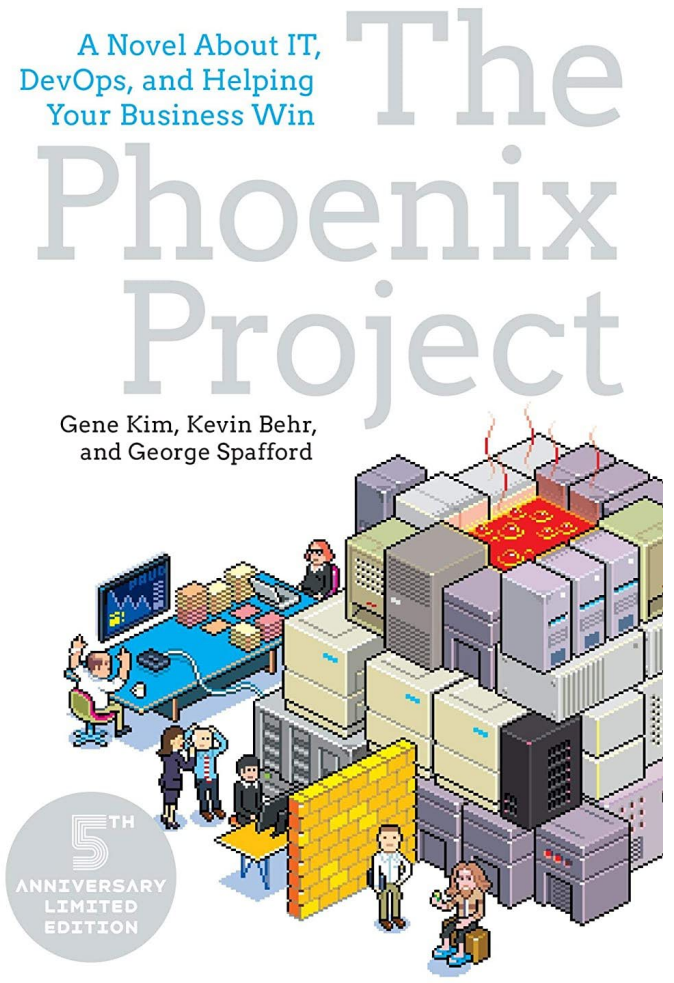
- **Majors:**
  - **“The Phoenix Project”,** Gene Kim, Kevin Behr, George Spafford. IT Rev (2018) (Please search/access through Swinburne Library to access online (need to be on Swinburne network). – **light reading**
    Here is a 10-minute summary: https://itrevolution.com/articles/10-minute-summary-of-the-phoenix-project/Links to an external site.
  - "**DevOps: Dive into the core DevOps strategies**", Sricharan Vadapalli, Packt Publishing Ltd (2018)
    https://ebookcentral.proquest.com/lib/swin/detail.action?docID=5322208
- **Additional readings/tools:**
  - "**DevOps: Puppet, Docker, and Kubernetes**", Uphill, Thomas ; Arundel, John. ; Hsu, Ke-Jou Carol. ; Khare, Neependra. ; Lee, Hui-Chuan Chloe. ; Saito, Hideto.; Packt Publishing, (2017)
    https://ebookcentral.proquest.com/lib/swin/detail.action?docID=4835403
  - "**Implementing DevOps with Ansible 2.**", McAllister, Jonathan. Packt Publishing (2017)
    https://ebookcentral.proquest.com/lib/swin/detail.action?docID=4924079
  - **… …**

A Novel?

Fictionalised account of how a business adopts DevOps principles to solve bottlenecks, waste, shortages and poor morale.

Does not contain anything about tools used for deployment, DevOps, CI/CD or continuous delivery.

Includes a few chapters of the DevOps Handbook.



Assessment

- Portfolio (individual) 50%
  - Tasks (individual) — 40%
    - 4 tasks (deploy stuff in different ways)
  - Report on guest lectures (individual) — 10%
- Test (individual) 20%
  - Week 9 (**on campus in tutorials**)
  - Examines material from first 7 lectures
  - On-line through Canvas – on campus in tutorial labs.
- Project (group) 30%
  - One big task including setting up and using an on-line CI/CD environment
  - Weeks 7-12,
  - Assessment includes presentations in lab
  - Details – week 7.

Portfolio

- Not quite what you're used to.
- Compulsory task
  - Guest lecture report – Write a report summarising, reviewing and contextualising the content – they discuss current industry practice.
  - Up to 10 marks (guest lectures are in Weeks 8 ~ 12)
- Deployment tasks
  - 4 deployment tasks
    - Deploy stuff for a Pass (easy/known tasks – fewer marks) *Up to 5 marks each – do all 4*
    - Go further – CR->HD more marks, *up to 10 marks per task*
  - All deployment tasks have Pass, Credit, Dist. and HD levels
  - You get to choose the level of depth of each task, but a**ssessment outcomes depend on quality of submission**

Portfolio Deadlines?

- Deadlines listed in the Syllabus & Assignments sections of Canvas.
  - Submission through Canvas
  - Extensions are rare and based on “severe impact” medical grounds.
  - You must clear a late submission with your convenor.
- Guest Lecture Report: end of semester
  - No late submissions
  - Missing these deadlines may cause you to fail (especially if you miss the “test”)

Time - resource management

- Walkthroughs, instructions may not actually work (as things change …)
  - Start early - plan for delays.
  - You're in final year now, so you should be able to solve problems as they arise.

- Some tasks may require you to have on-line accounts/sign up for trials
  - You may like to get a Microsoft developer account, and AWS account...
  - Good for your CV
  - A good investment
- Or use the free trials/free tiers.
  - Remember to shut down/delete/cancel any "free" resources which may expire and start costing you money.

Tutorials/Lab tasks

- There is **weekly tutorial** – discuss/explore questions related to topic
- There aren't any **labs**
- Tutors will provide general guidance, and assistance with your lab tasks and project.
- You are expected to install the relevant software on your own computer(s) or on-line accounts. Laptops and Desktops (64-bit) should be sufficient for you to complete the chosen portfolio tasks.
- Tutorials/labs are for feedback/advice/discussion. We will not be walking you through lab tasks – you're not in first year anymore.

Task submission

- See individual assignments and unit outline
- Make sure the task you submit is submitted under the correct assignment.
- Assessments will be listed (roughly) in order of due date
  - Deployment tasks first,
  - Test week 9,
  - Project week 8, 11 & 12 (presentations earlier),
  - The guest lecture report due after the last guest lecture.

Teaching Approach

- Lectures
  - We will cover principles, concepts, techniques
  - Case studies will be used to illustrate certain aspects
  - Practice quizzes every week
- Guest lectures
  - Latest practices in industry
- Tutorials (Labs)
  - Case studies and portfolio guidance
  - Questions and discussions will repeat and reinforce material in lectures
  - Quizzes

Lecture Roadmap

Where are we?

- Learning Objectives
- Motivation - Deployment
- Motivation - Evolution
- Assessment & Teaching Method
- **In a Nutshell – Software Deployment and Evolution**
- Environments

But Deployment is Evolving…

- 1970: Waterfall Model
- 1988: Spiral Model (Boehm, B. (1988), "A Spiral Model of Software Development and Enhancement," *IEEE Computer 21*, 5, 61-72.)
- 1991: Continuous Integration https://www.martinfowler.com/articles/continuousIntegration.html
- 1998: Carzaniga
- 1999: Extreme Programming https://www.computerworld.com/article/2585634/app-development/extreme-programming.html
- 2001: Agile Dev http://agilemanifesto.org/
- 2009: DevOps Bass, Len; Weber, Ingo; Zhu, Liming. *DevOps: A Software Architect's Perspective*.
- 2009: Continuous Deployment & Continuous Delivery https://puppet.com/blog/continuous-delivery-vs-continuous-deployment-what-s-diff
- 2012: DevSecOps http://www.devsecops.org/
- 2015: CI/CD (Continuous Integration/Continuous Delivery) https://dzone.com/articles/what-is-cicd

Software Maintenance

- “Modification of software after delivery, to correct faults, to improve performance or other attributes, or to adapt the product to a new environment" -- IEEE Standard 1219
- After Delivery Modification to Correct Faults
- Types: Adaptive, Perfective, Corrective, Preventive
- Many challenges in current software (“2.0 World")
  - “After delivery” in the beta forever world?
  - Definition of *fault* has changed -- esp. in Agile methods
  - *Modification* is a broad term

Software Evolution

- Software maintenance is the *activity*
- Software evolution is the *outcome*
- This evolution broadly fits certain ’general laws’
  - Not as firm as the physical laws (gravity…)
  - Qualitatively good and sufficient

Lehman's Laws of Software Evolution

1. The law of continuing change.
2. The law of increasing complexity.
3. The law of large program evolution.
4. The law of organizational stability.
5. The law of conservation of familiarity.

Original 1974-1978 laws

- Software grows
- Quality decreases
- Feedback drives the changes

Extra laws added 1991-1996

Deployment has Evolved too…

- Few software products are stand-alone applications which you can install from CD/Zip/MSI/DWG.
- Expectation is that you are connected to the internet permanently, no download limits, no speed limits.
- More likely that the thing you buy/download is itself a downloader
  - Connects to vendor repository
  - Scans target computer (OS, Hardware)
  - Determines what need to be downloaded
  - Downloads/pulls/builds software on your computer
  - Reports back to vendor

Deployment has Evolved too…

- Even more likely – software is hosted in the cloud
  - All you end up installing is a front-end and a bit of local variable storage
  - Or just use your phone's web browser for everything.
- So what do you have to deploy?
  - Deploy from Dev build server to cloud server
  - Potentially much easier to control, easier to maintain, easier to secure.

Old vs. New - Games

- 2000s - X-Box (original) – 3-core CPU, stripped-down Win2000 OS with a really good graphics card.
- Games supplied on DVD-ROM, crippled features
  - Spawned a whole generation of hardware hackers – modifying/replacing BIOS ROM to enable features (bypass copyright protection, execute unsigned home brew code, run emulators, XBMC (now Kodi)).
  - Microsoft updated hardware/software 6 times and still failed to stop hacking of the X-Box.
  - Impossible to update "deployed" software.
- 2010s Steam – Games kept on remote repository, subscription access only.
  - Use "standard" hardware (user's PC).
  - Much easier to control the games, deploy new versions.

But old school deployment still happens

- Enterprise systems, especially off-line or "leased-line" systems
- Middleware
- "Secure" (off the internet) systems (military, nuclear)
- Legacy systems (finance, banking, credit card systems)
- Manufacturing (robotics, automation, SCADA, PLC, Industry 4.0?)
- ATM, POS systems

- Autonomous vehicles?
- Aircraft fly-by-wire?

So we study old and new…

Trad. Software Deployment in a Nutshell

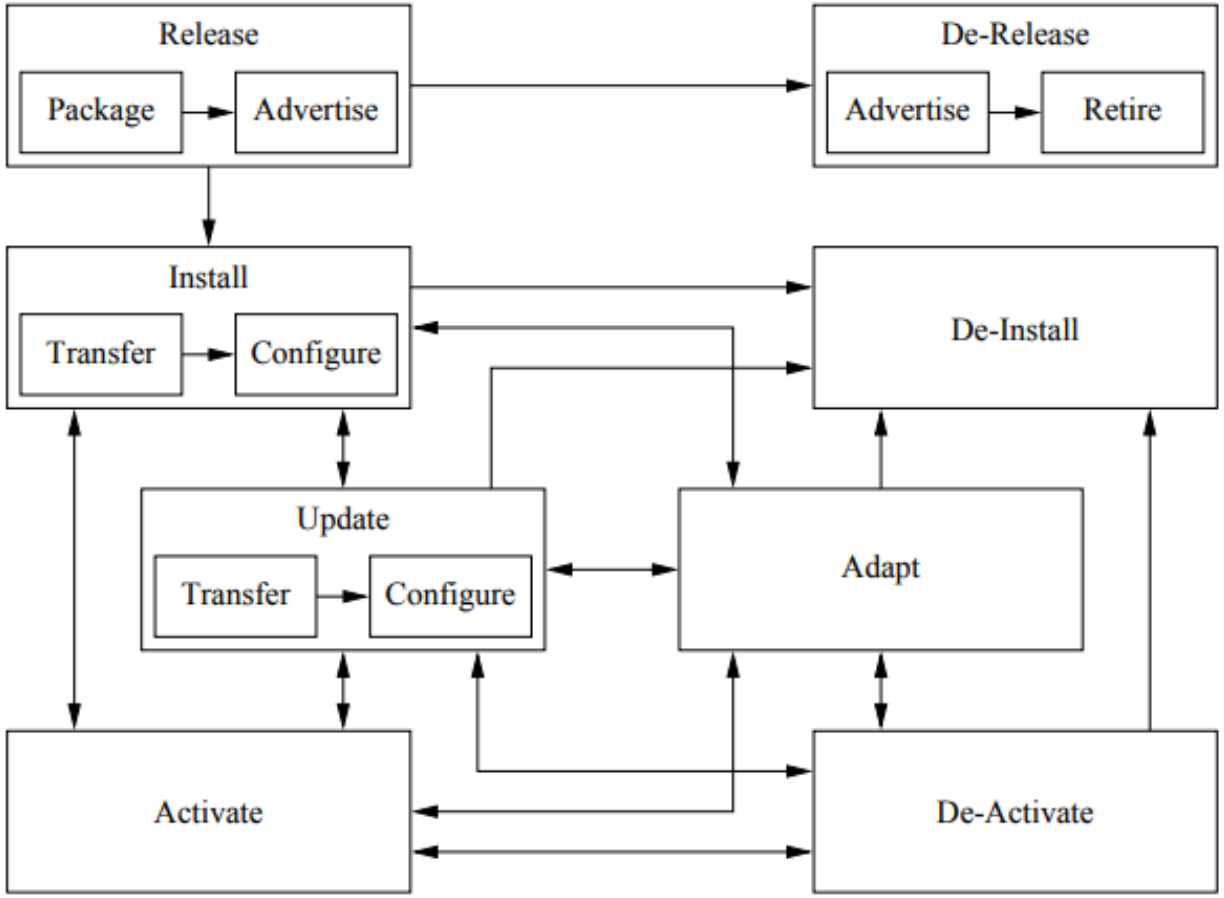


Figure 1: Activities of the Software Deployment Process.

https://www.ics.uci.edu/~andre/papers/T3.pdf

Lecture overview

Where are we?

- Learning Objectives
- Motivation - Deployment
- Motivation - Evolution
- Assessment & Teaching Method
- In a Nutshell – Software Deployment and Evolution
- **Environments**

Software Distribution Overview

- Conventional Software that needs packaging and distribution
- (zip, MSI, DMG, PKG, DEB, XPI…)
- Transfer is completed by the installation software e.g. Windows Installer

https://peacekeeping.un.org/en/deployment-and-reimbursement

Software Deployment - Release

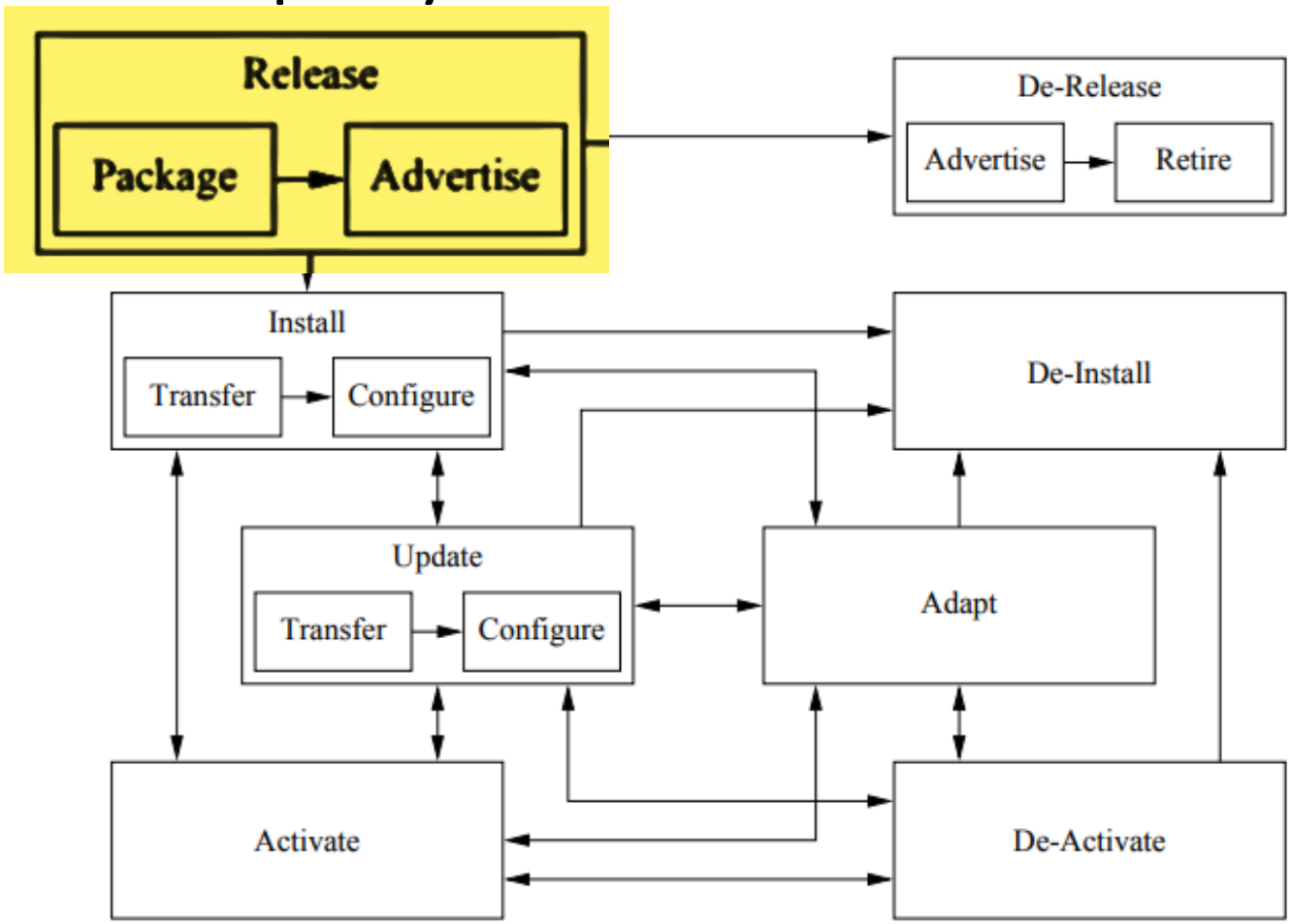


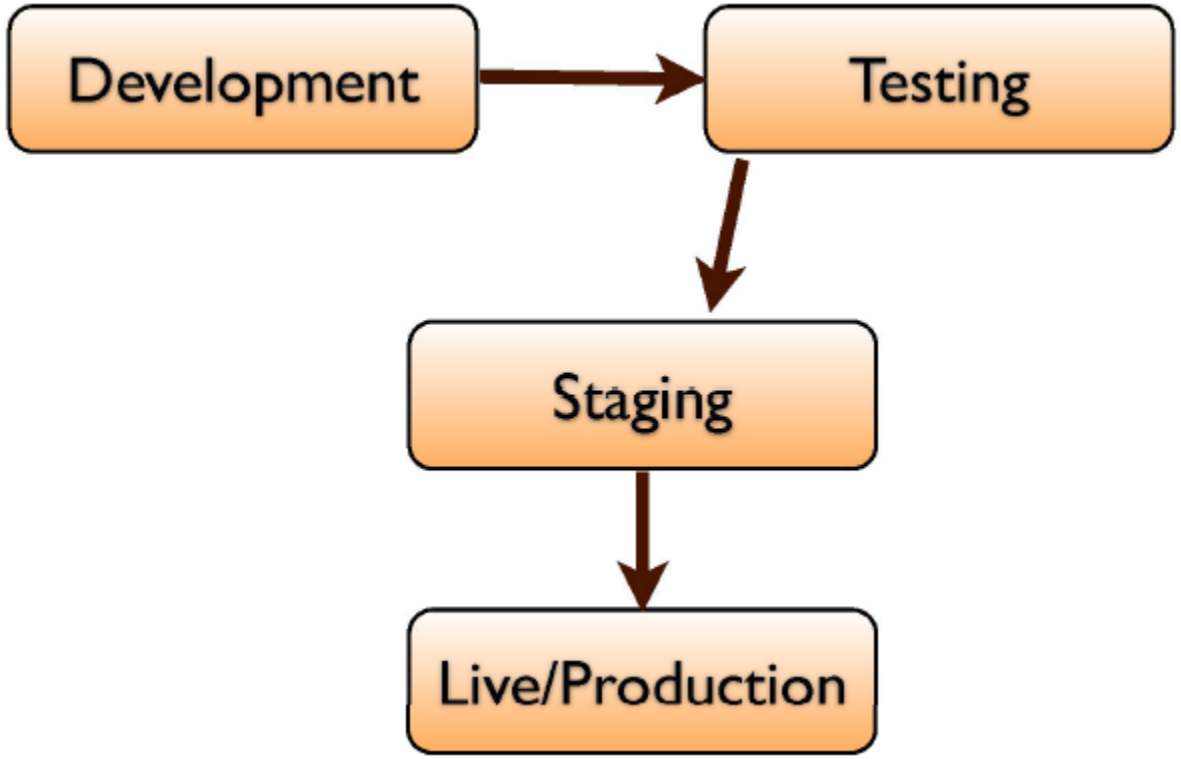
Figure 1: Activities of the Software Deployment Process.

https://www.ics.uci.edu/~andre/papers/T3.pdf

Releasing Software

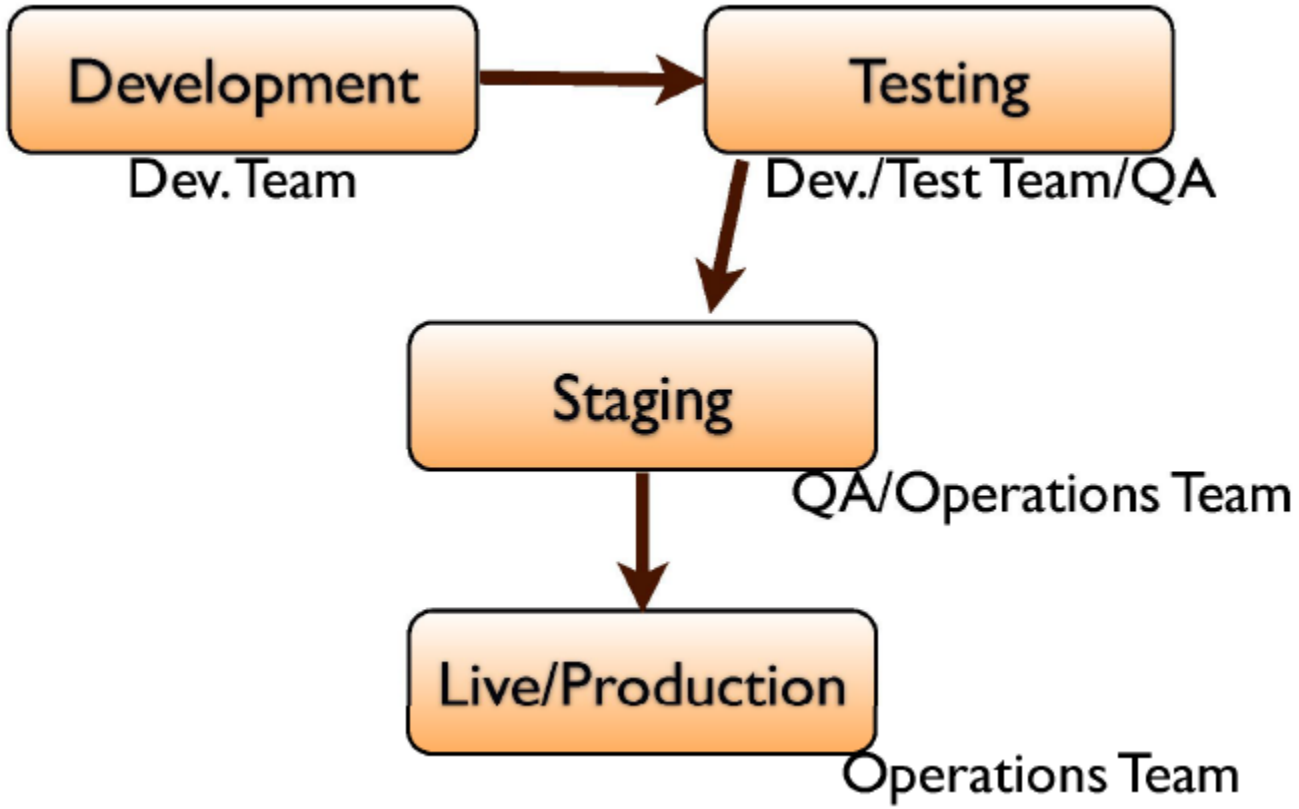
- Before software can be deployed it needs to be formally released
- The key steps in the Release activity are:
  - Packaging the software
  - Advertising
  - Setting up a distribution model/method
- A release depends on the ’target environment’

Different Types of Environments



- Software is released for different target environments
- Contents of software can change depending on environment

Different Types of Environments



- Teams that focus on different infrastructures

What constitutes an environment?

An environment includes:

1. Hardware and Operating System

- A set of software (including libraries)

2. Tools needed by teams operating in that environment
3. Developers need IDEs
4. Testers need testing tools

- Operations team is interested in monitoring

Development Infrastructure

Developers infrastructure is used by all developers

- Typical hardware is high-end workstations
- Contains all of the development tools

A release package is generally not formally created

- Developers tend to use scripts (build files) to create working software to allow them to progress

Testing Infrastructure

- Testing infrastructure is typically (ideally) separate
- Software is packaged for testing
  - With debug code
  - Extensive log files
  - Performance may be sacrificed for information (profiling may be turned on)
  - Test team will verify that the product satisfies requirements
- Releases to test infrastructure are frequent (daily is possible, but weekly is more common)

Testing Infrastructure

Releasing into a test infrastructure

Has to take into consideration certain issues:

- Test data – creation
- Connected software/services/APIs
- Automated testing
  - Do you populate databases with the same data for each test cycle?
- Regression testing
- Tools used for testing
- Database schema changes

Staging Infrastructure

This environment is identical (almost) to the actual “live/production" infrastructure

Software deployed into Staging for:

- Training operations team (or) customers
- Replication of issues and defects reported by customers
- Allows for a final verification on a realistic environments
- Final quality assurance is undertaken on staging infrastructure
- Performance testing …

Live/Production Infrastructure

- Once the system has been considered refined and perfect, it is deployed into a Live/Production infrastructure
- This is the environment on which real customers use the software
- It can be the customer's PC, server (or) a managed/hosted server
- Once a system goes live into this infrastructure, all issues and changes are much more expensive than otherwise
- The version of the product on live and staging is identical

Virtualization and Environments

Virtualization technology (e.g. VMWare) allows you to setup multiple virtual computers on top of a single physical computer

Many IT organisations are adopting virtualization technology to run 'Testing' and 'Staging' on a single physical machine -- it is cost/resource efficient

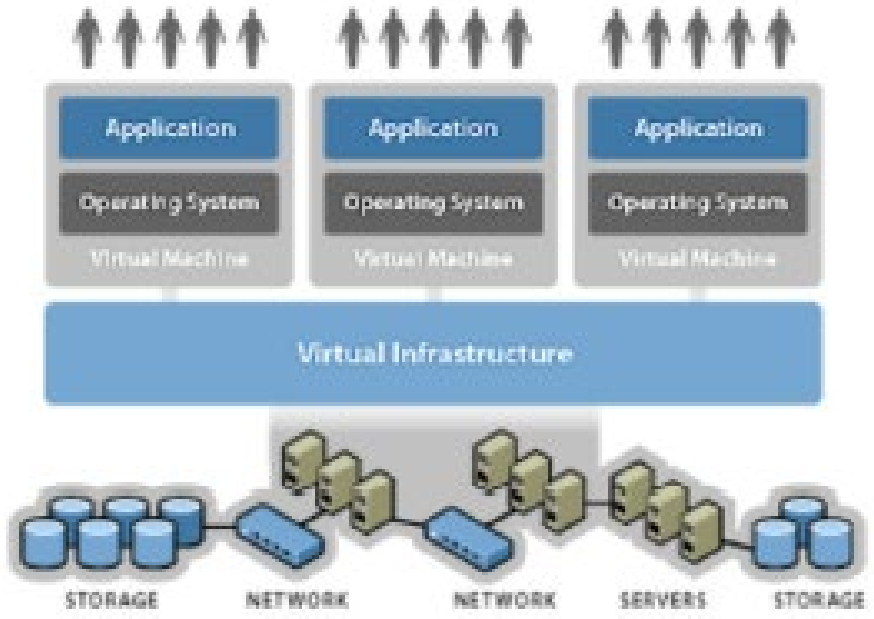


Image Source: VMWare Inc.

Short Problem - 1

You have the option to make use of virtual servers.

However, resources are limited.

Which infrastructure will you virtualize first? Why?

- Development
- Testing
- Staging
- Live/Production

Dev? Allows multiple teams on different platforms to share the work
Prod? Allows horizontal scaling (think of AWS ASGs spinning up extra instances when load gets too high)

Virtualization: PaaS and Iaas

- PaaS: Platform as a Service
  - Virtual computing hardware + Load Balancer + Operating System + Web/DB/Application Server Software
  - No direct control over the actual hardware, O/S
  - Example: Microsoft Azure, Google App Engine
- IaaS: Infrastructure as a Service
  - You get requested virtual hardware (CPU/RAM/ Disk) and operating system can be installed for you -- you do the rest
  - Example: Amazon EC2, Google Compute Engine, Rackspace

Platform as a Service (PaaS)

- Example: Google App Engine
  - You write software to the API provided by Google App Engine
  - Can install certain libraries (but not all)
  - Multiple language options: Java, Python, Go
  - Google's database management system (key-value based, i.e. a large hash table)
  - Pay for resources consumed
    - Network bandwidth, CPU cycles, Dish space, Messages sent/received

Infrastructure as a Service (IaaS)

Example: Amazon EC2

- Select either a pre-configured virtual machine (or) customise your own
- Can request different types of instances (virtual machines): High CPU, High I/O, GPU bound, High memory
- Can request O/S, DB server, Application Servers, CMS, Web server etc.
- Can cost between $ 0.05 - $3 / hour (only if instance is running)

Backend as a Service (BaaS)

- Offers another layer of abstraction on top of Infrastructure & Platform.
- Example: https://github.com/parse-community/parse-server

  Offers services like emails, login, account creation, user management, Basic Data management, Media management etc.
- Costs are minimal for small volume of transactions (increase after that)
- Quickest to start with when starting or testing a new idea.

Short Problem - 2

- To be ‘buzz word’ compliant, you decided to go cloud for your ’Facebook’ killer application -- “Me9Friends.com.
- It is like Facebook, but only allows you to have 9 friends. Features: Chat, Photos, Events, Stream
  - Will you go PaaS or IaaS or BaaS?
  - Will you try a combination?
  - Which environments will you virtualise?

Review

- Try the tutorial questions
- Attend tutorial
- More review …
- Do the Quiz (Canvas) – before and/or after tutorials
- Bask in the glory of your achievement*

*may not apply to all students

What's in the tutorial?

- Discussion, advice, explanation.
- Completely pointless unless you engage:
  - Attend and discuss in tutorials
  - Post on discussion boards (if available)
  - Interact with other students
  - Use other collaboration technologies
- Disagree with the quiz questions?
  - Raise your issues at the tutorial
- Ask about the assessments/tasks/reports