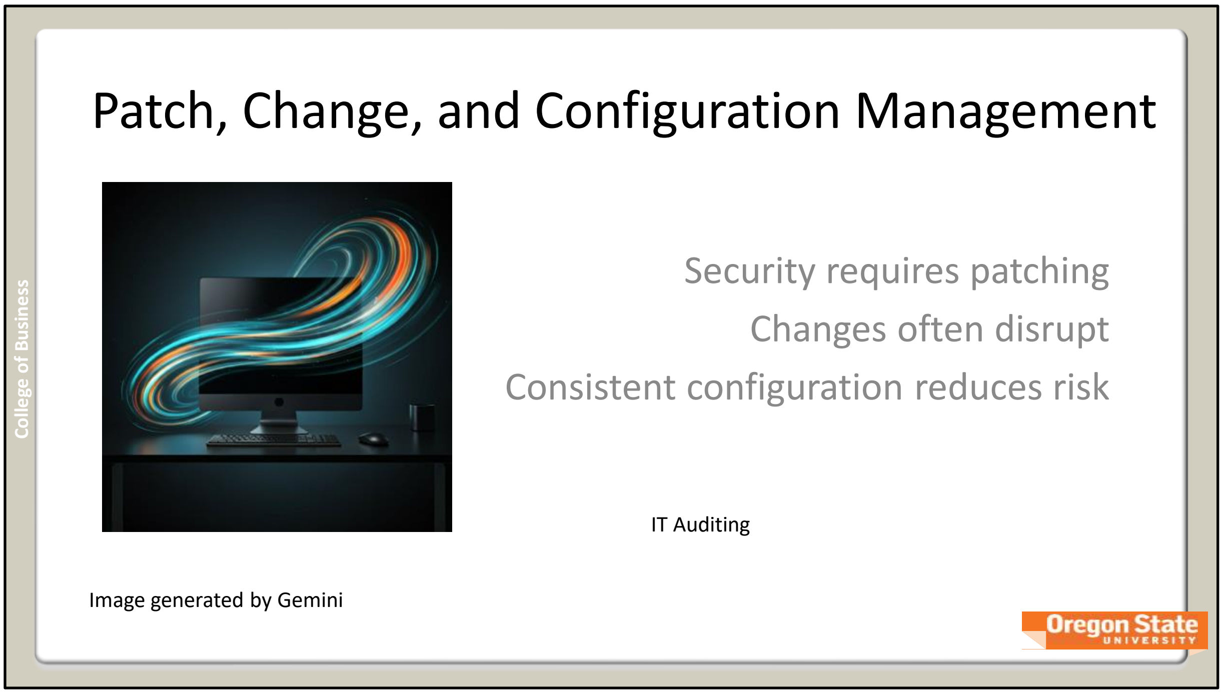


College of Business

Patch, Change, and Configuration Management

Security requires patching

Changes often disrupt

Consistent configuration reduces risk

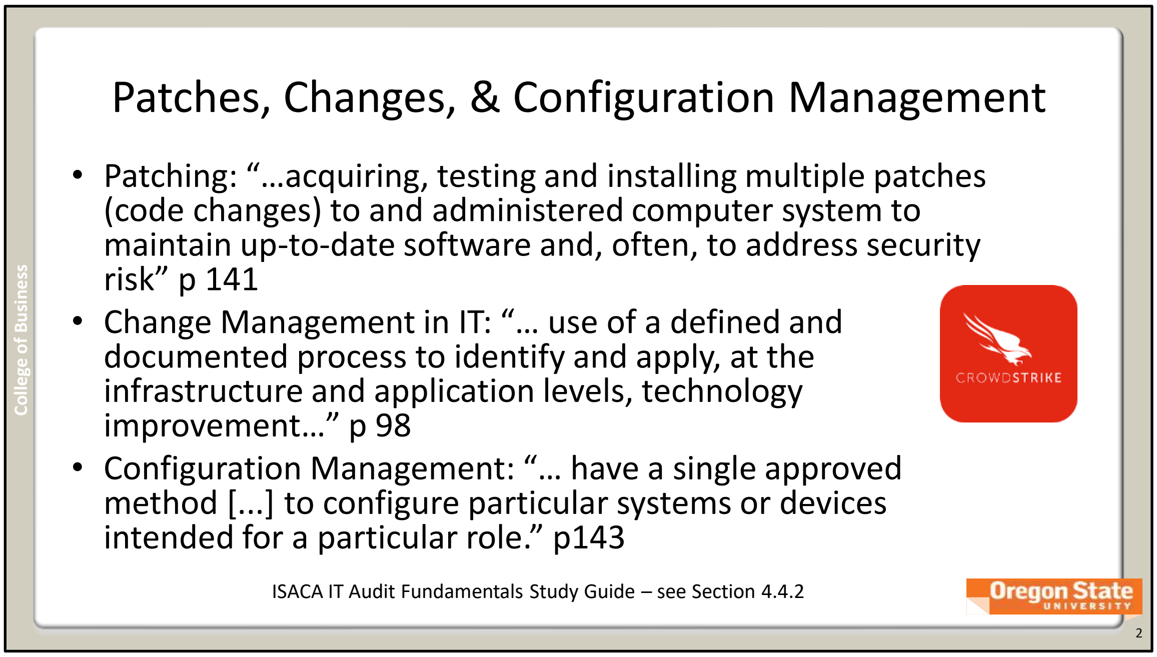
IT Auditing

Image generated by Gemini

Oregon State UNIVERSITY

Organizations face substantial risk in acquiring new systems and maintaining old ones.

This slide deck presents an overview of SDLC risks and controls. Process descriptions and purpose statements for some SDLC-relevant COBIT sections are included as we go along.



College of Business

Patches, Changes, & Configuration Management

- Patching: "…acquiring, testing and installing multiple patches (code changes) to and administered computer system to maintain up-to-date software and, often, to address security risk" p 141
- Change Management in IT: "… use of a defined and documented process to identify and apply, at the infrastructure and application levels, technology improvement…" p 98
- Configuration Management: "… have a single approved method [...] to configure particular systems or devices intended for a particular role." p143

ISACA IT Audit Fundamentals Study Guide – see Section 4.4.2

CROWDSTRIKE

Oregon State UNIVERSITY

2

The IT landscape is dynamic. When one part changes, other parts need to adapt if organizations are to achieve their objectives.

IT management frameworks have a lot to say about processes that involve change management in on form or another.

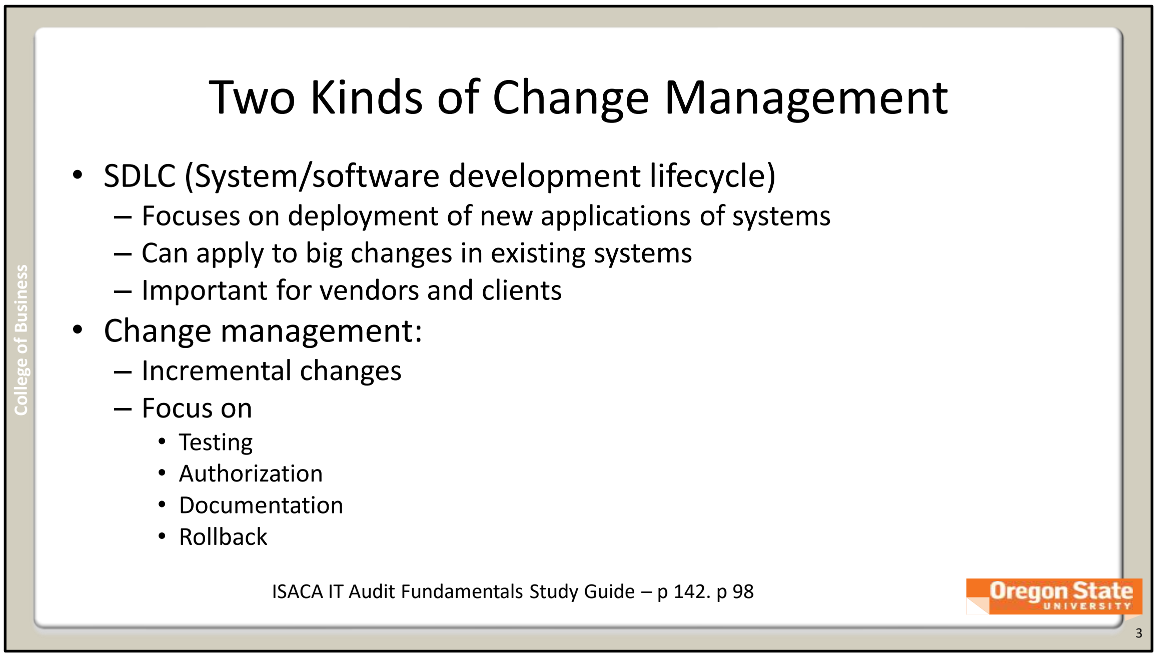
Summarizing: the goal is to balance the need to deploy needed changes quickly so that processes can approve and known risk can be mitigated while avoiding disruptions that often come with changes.

Whenever new programming or new devices or new device settings are deployed there is a risk of disruption. The change may have flaws or the change may negatively impact how systems interact.

In a staggeringly relevant example from 2024, CrowdStrike deployed a code change to its security protection system. IT had a flaw that caused millions of windows systems to crash causing, among other effects, massive disruptions in air travel in mid-July of 2024.

This event relates to patching, changing, and configuration. The direct cause of the damage was the automated installation of a "patch". The vendor's change management (software development) controls did not catch the problem. The affected systems were systematically configured to automatically deploy this kind of patch without any testing.

This is a cautionary tale. Auditors might well have approved of all the controls that were intended to protect against this kind of problem, but controls only provide reasonable assurance. It may be that organizations will reconsider what is reasonable in the future.


College of Business

Two Kinds of Change Management

- SDLC (System/software development lifecycle)
  - Focuses on deployment of new applications of systems
  - Can apply to big changes in existing systems
  - Important for vendors and clients
- Change management:
  - Incremental changes
  - Focus on
    - Testing
    - Authorization
    - Documentation
    - Rollback

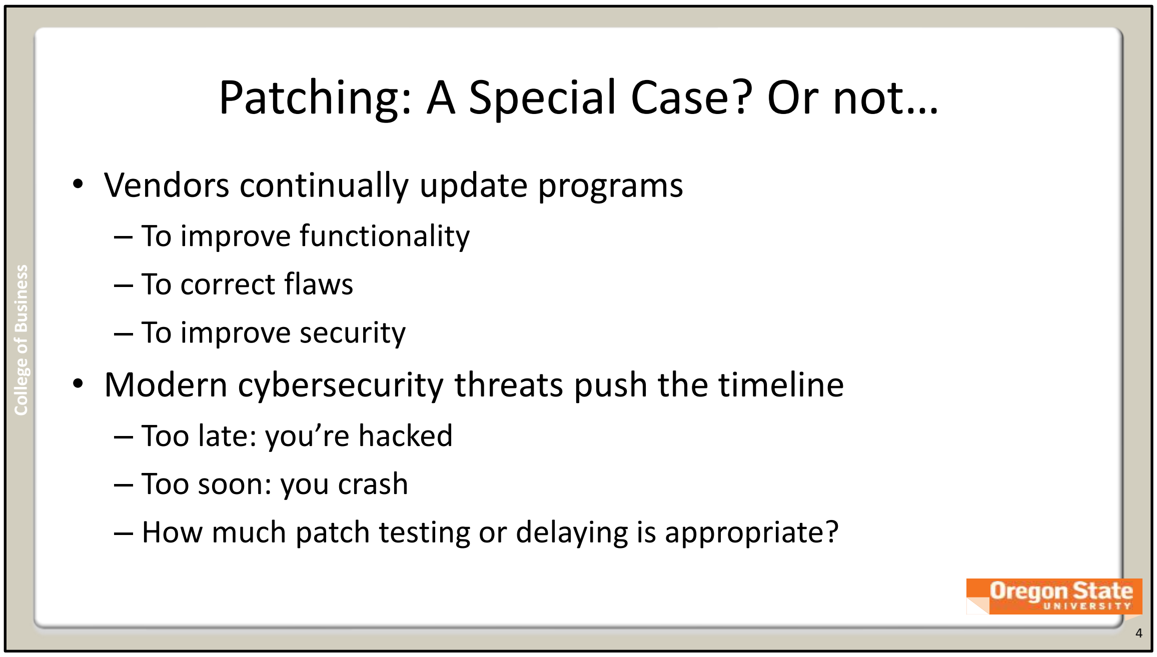
ISACA IT Audit Fundamentals Study Guide – p 142. p 98

3

The ISACA study guide does a good job of outlining the kinds of controls expected for change management. (p142)
- Informing relevant personal when a change is coming and what is involved
- Updating documentation to reflect the change (including job scheduling and operations)
- Test results have been reviewed and approved by users and IT
- Accurate and complete data conversion has been approved
- Thorough testing is done post change
- Legal and compliance elements have been considered
- Adverse effects have been considered and rollback plans are in place to pull out changes is necessary

Auditors might determine whether: (p 98)
- Formal change management procedures are in place
- Change requests are categorized, prioritized, and authorized
- There is a process for emergency changes
- Changes are tracked as they move through the systematic process

To do this they can:
- Validate authentication policies and methods (IAM! Unverified logins mean approvals and deployments are suspect)
- See if the folders containing executable code are protected
- See if change requests are reviewed, approved, and documented
- See if risks are assessed
- See if changes are tested prior to deployment
- Verify that change documentation includes required elements including:
  - Description and cost analysis
  - Appropriate and documented approvals
  - Documented consideration of a change's impact on other systems
  - Roll back or back out procedures have been developed as appropriate



College of Business

Patching: A Special Case? Or not…

- Vendors continually update programs
  - To improve functionality
  - To correct flaws
  - To improve security
- Modern cybersecurity threats push the timeline
  - Too late: you're hacked
  - Too soon: you crash
  - How much patch testing or delaying is appropriate?

Oregon State UNIVERSITY

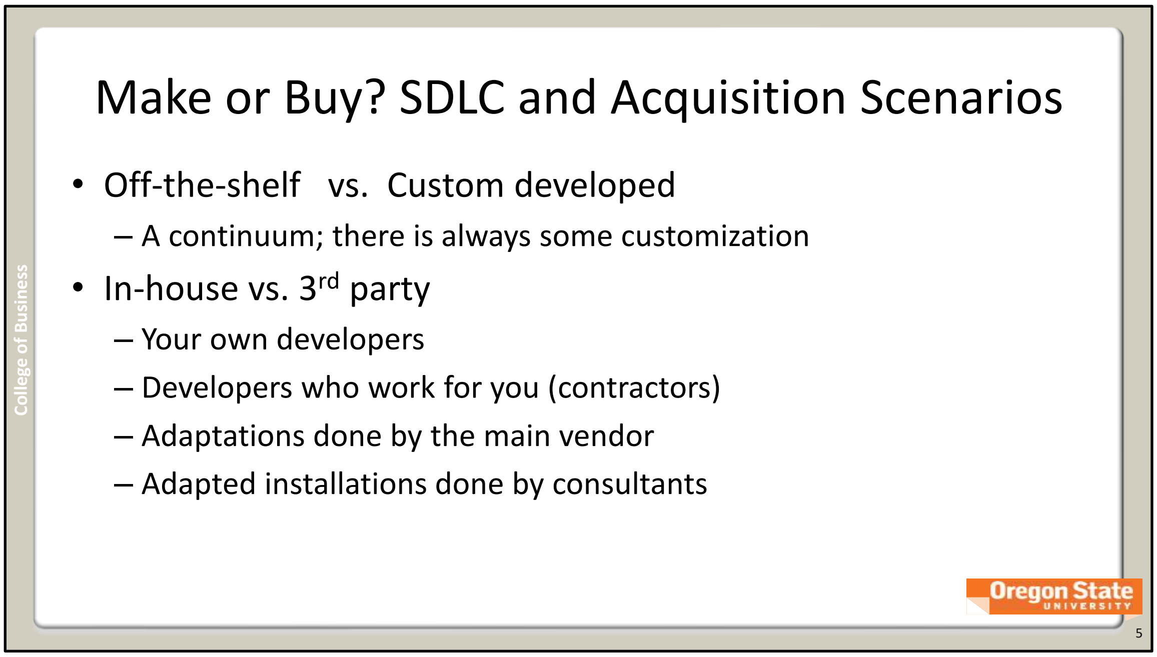
4

Back in the day, organizations could wait a month or two before upgrading to a new version of a software program.
- Changes can disrupt things - true!
- Let someone else try it first.
- A little delay won't matter much.

Cybercriminals work fast. Once a flaw is identified, bad actors will begin trying to use the flaw to compromise thousands or millions of devices within a few days.
Vendors don't want their clients (or their own reputations) to suffer so they try to fix vulnerabilities quickly. Going fast can result in errors, going slow can increase the chance that clients will be negatively impacted by an attack. In an ironic twist, wide-spread efforts to communicate about identified vulnerabilities means that the "good guys" publish information on how to exploit unpatched systems. Reputable researchers usually inform a vendor of flaws and allow time for the vendor to produce a patch before they publish their results. Still, the information is often available long before many organizations have gotten around to patching their systems. That results in lots of successful cyber attacks. This reality of the modern cybersecurity landscape has made the logic of delay less palatable.

Organizations need to have thoughtful patching procedures and policies, but there are few "right" answers. Auditors can look for systematic action, even without knowing what is the "best" approach to patching. If management has taken a systematic approach, based on reasonable analysis, that is not too far out or alignment with authoritative guidance, the auditor will probably not assert a negative finding.

Some key ideas:
- "Feature" patches that fix non-security bugs or improve functionality are often differentiated from security patches that fix exploitable security flaws.
- Organizations may insist on prompt (or immediate) application of security patches but allow a longer delay for feature patches. Many vendor systems support this distinction.
- Organizational patching strategy should adapt for variations in risk across devices, systems, and organizations
- Auditors should see if such factors are included in control design but shouldn't impose their conclusions about the right thing to do on organizations.


Make or Buy? SDLC and Acquisition Scenarios

- Off-the-shelf vs. Custom developed
  - A continuum; there is always some customization
- In-house vs. 3rd party
  - Your own developers
  - Developers who work for you (contractors)
  - Adaptations done by the main vendor
  - Adapted installations done by consultants

College of Business

Oregon State UNIVERSITY

5

An application is a computer program or a group of programs that supports an organizational function. Programs that work together are considered an 'application' when they address the 'same' need or process. An application can be:

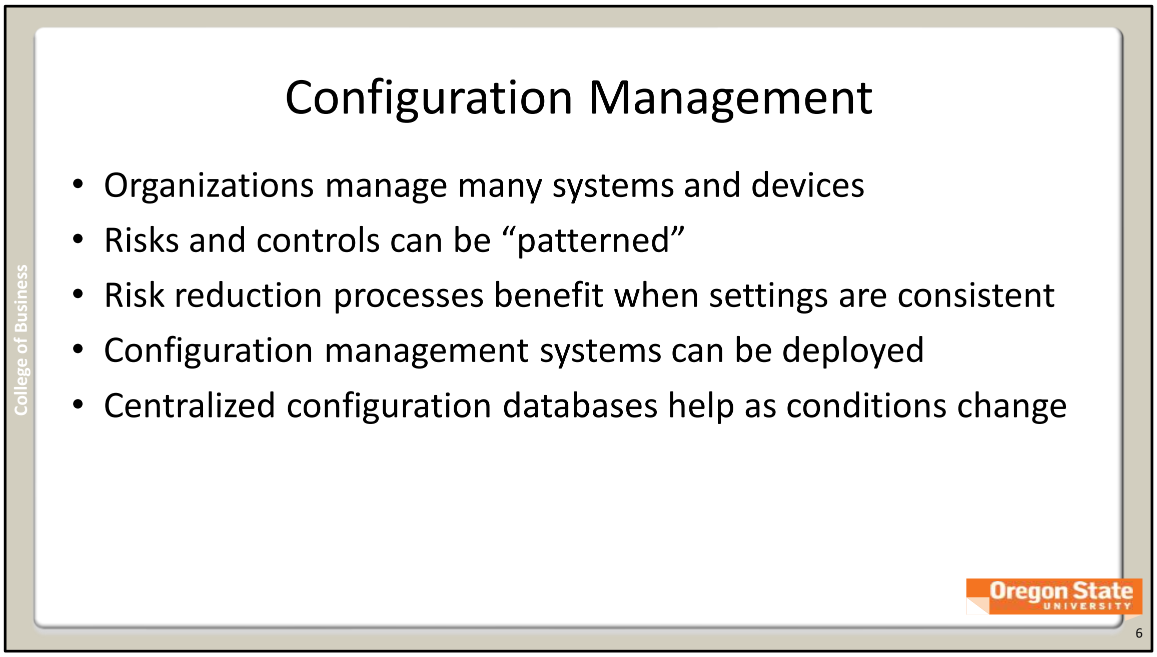
- relatively coherent – addressing a focused function - for example, Word (for word processing);
- moderately complex, supporting several related functions (email and calendaring); or
- immensely capable as in ERP (Enterprise Resource Planning) or CRM (Customer Relationship Management) applications which support a vast array of processes in a somewhat integrated fashion.

Don't over-simplify the distinction between system acquisition and in-house development of applications. Oversimplifying, you can usually purchase a vendor's application or build it yourself. But very few organizationally important systems can be deployed without a significant customization. Even simple general ledger systems are configured with a chart of accounts. If you get the chart wrong, the system won't work right. Customization risks are not unlike new-development risks.

SDLC: System (or software) development lifecycle concepts:

- What is 'developed'?
  - Entire applications (collections of related programs) can be custom written for an organization.
  - Extensive customizations are often required for systems such as ERP (Enterprise Resource Planning) or CRM (Customer Relationship Management). *It's often a trade off: change the organizational process or customize the system.*
  - Every organization has multiple systems. Connecting them generally involves programming of some sort.
  - Even solid vendor tools require installation and configuration. It can take years and lots of expertise.
- Who does the development?
  - In-house: an organization may have the technical people to design and build (usually in computer code) applications they need to support operations.
  - Contractors can be hired as temporary employees to do the work.
  - You may be able to pay the main vendor (e.g., SAP) to do needed customizations.
  - Consulting firms (often consulting arms of accounting firms) also do installations, development, or customization.

While we won't systematically compare these options. But you can probably see how they relate to the risks and controls discussed in previous slides.


College of Business

Configuration Management

- Organizations manage many systems and devices
- Risks and controls can be “patterned”
- Risk reduction processes benefit when settings are consistent
- Configuration management systems can be deployed
- Centralized configuration databases help as conditions change

6

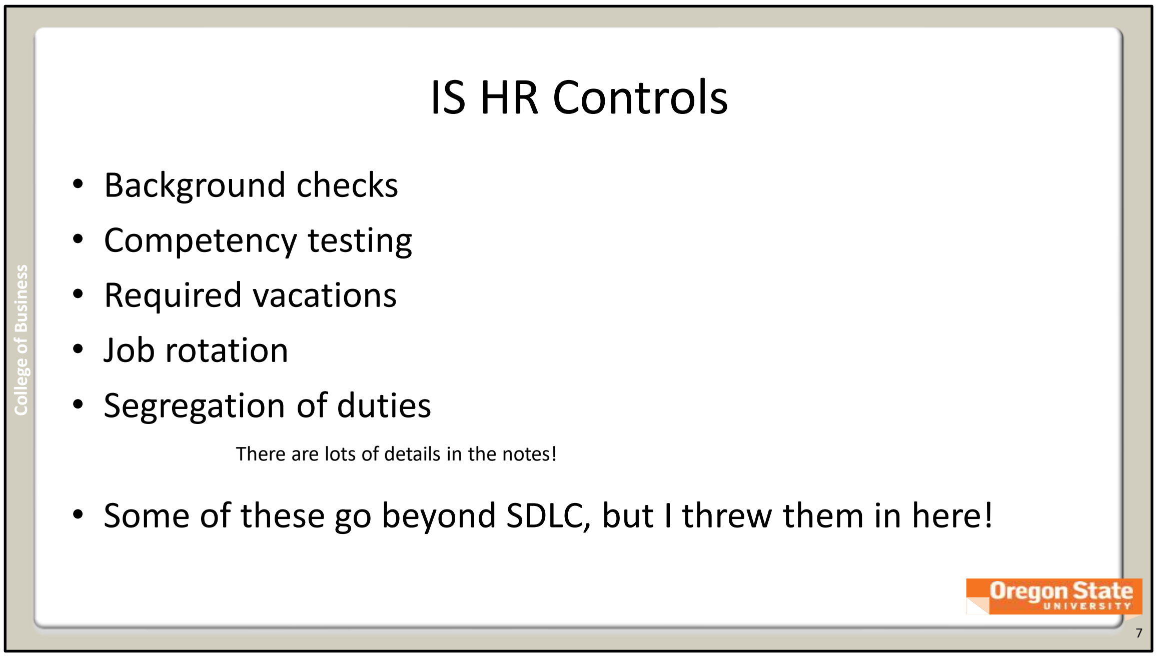
Organizations manage many systems and devices.
Manually configuring each device using expertise and judgement seems like a good idea – each device can be tailored for maximum usefulness and minimum user friction. But being systematic as advantages too.

Risks, and therefore controls, are or should be patterned. For example, you should expect that if a person manually sets up 10 user laptops, connecting them to organizational resources, setting up automatic updates, and installing various software components, mistakes will happen. Controls such scripts, checklists and installable images will reduce the number of configuration mistakes and increase setting consistency.

Risk reduction processes, such as testing and endpoint management, become less costly and more effective when settings are consistent across multiple devices. If each device uses different settings, it becomes harder to effectively test incoming updates. A particular change may, for example, affect a windows machine that accesses shared storage differently depending on which of several versions of the storage software app are installed or whether files are replicated on the local hardware. Variations that are purposeful and valuable are one thing, testing might appropriately adapt, but variations that arise because consistent configurations were not initially or incrementally applied can be costly and counter productive.

Configuration management systems can be deployed to automatically assess compliance with configuration management policies. Actually, there are numerous processes to help with configuration management issues. A vendor can, for example pre-install the bulk of needed software and initially configure many important settings before a new device is shipped. That can reduce acquisition costs and reduce configuration variations. Mobile device management (MDM) tools can help the organization validate that phones, laptops, and other mobile devices meet minimum standards. And Endpoint systems such as Microsoft InTune can be installed to monitor organizational devices to monitor and even enforce organizational configuration policies.

When new conditions (threats) arise, having a centralized database of device settings can help speed risk assessment and response. This can be a very hard problem. For example, in 2023, a security flaw found in the widely used log4j package embedded in numerous application programs was very problematic because that code was embedded in billions of devices and the owners of the devices had no reasonable way to tell. Standard tools for configuration management might not have made it easy to see which devices were at risk, but they could have helped a lot. And many less-obscure issues might be easily addressed if a centralized database tracked the configuration of a large majority of devices deployed in an organization.


College of Business

IS HR Controls

- Background checks
- Competency testing
- Required vacations
- Job rotation
- Segregation of duties

There are lots of details in the notes!

- Some of these go beyond SDLC, but I threw them in here!

7

Having competent IT people and system developers is important.
IS staff present some unique risks because systems often rely on them, because they often need broad access you would not allow other users, and because non-IS managers often lack the expertise to assess IS competency. Retention can also be a challenge.

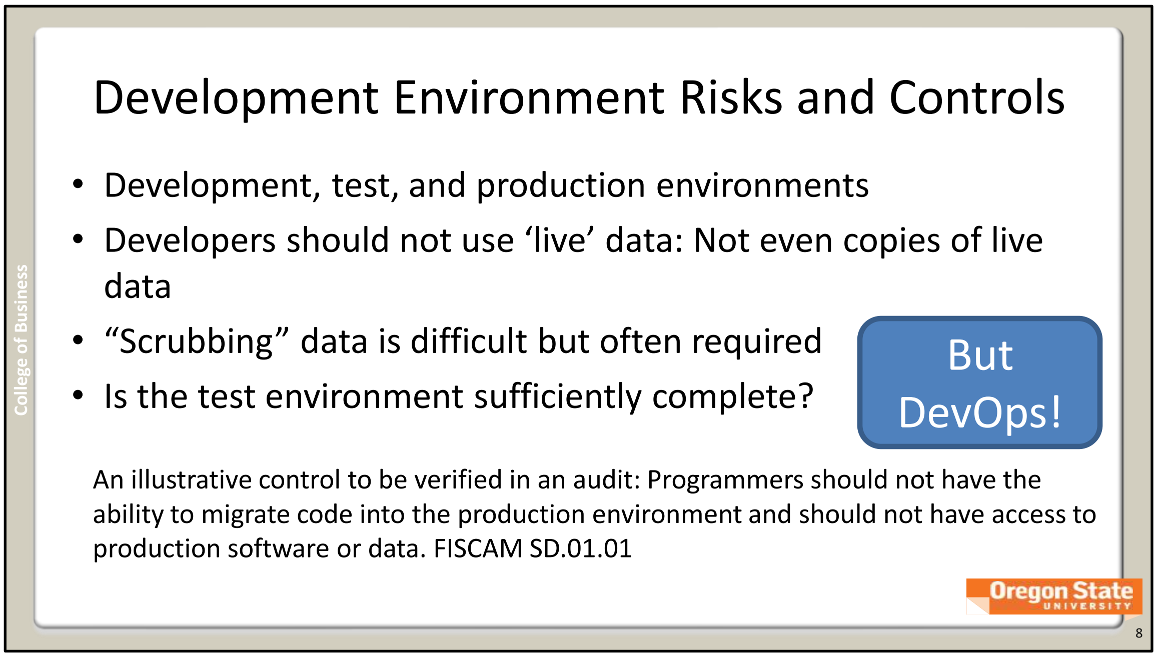
Internal controls for IS staff matter. Common controls include:
Background checks:
- Are they fraud risks?
- Do they have the credentials they claim? Legal/compliance restrictions or documented controls may require certifications – failing to realize someone is claiming false credentials could be both expensive and embarrassing.

Competency testing may be performed as part of the hiring process.
Required vacations: If someone has to cover a key person's responsibilities for a couple of weeks various risks may come to light:
- If the person is doing something nefarious and they are barred from activity for a while, discrepancies may show up.
- If only one person knows how to perform important activities, the organization is at risk. This can happen because of time constraints, skill sets, or organizational history/knowledge but sometimes it happens because an employee wants to be difficult to replace so as to have job security.

Job rotation:
- As with required vacations, cross training reduces excessive risk related to an individual.
- Good tech employees are often mobile. They can get jobs elsewhere. Working on new systems can help maintain interest.
- System documentation is often weak. Having new people work with the system while the current staff is still with the organization can identify documentation short comings before a crisis arises.

Segregation of IS duties
- Separating development and maintenance groups is considered a leading practice, but this practice is often neglected or purposely not implemented. The original developers of a system may have unique understanding that can help them commit fraud if they can later make subtle changes during maintenance.
- Individuals who develop or maintain systems should not also be allowed to promote code to production in part because they may not be inclined to test sufficiently (believing their work is great) or because they may slip in nefarious code. A librarian often has this responsibility.
- Security tasks (such as accessing cybersecurity encryption keys) often requires two separate authenticated individuals. A single individual is more likely be coerced, act nefariously, or make an error. This is only one example.

If asked: could you identify risks/controls/audit procedures based on this material? Given a control, could you explain risks and how the control mitigates that risk?



College of Business

Development Environment Risks and Controls

- Development, test, and production environments
- Developers should not use 'live' data: Not even copies of live data
- "Scrubbing" data is difficult but often required
- Is the test environment sufficiently complete?

But DevOps!

An illustrative control to be verified in an audit: Programmers should not have the ability to migrate code into the production environment and should not have access to production software or data. FISCAM SD.01.01

Oregon State UNIVERSITY

8

Although it is not always feasible, best practices call for separate development, test, and production environments.

- Three environments:
  - Developers create and initially test code in the development environment.
  - Once a new component (usually a program) is ready, it is migrated to the test environment where more rigorous and systematic tests can be performed.
  - The production environment is where the organization stores operational programs and data.
- Generally, people who do 'real work' are not supposed to have access to the test environment except when testing. More importantly, developers are not supposed to be able to touch the programs in the production environment.
- Leading practices call for a 'librarian' function where someone who is NOT the developer is charged with transferring (promoting) code into production. They are expected to ensure that appropriate controls such as source evaluation, testing, and management approval are in place. They not only promote the programs. They document the related activities and monitor the programs in production for unauthorized changes.

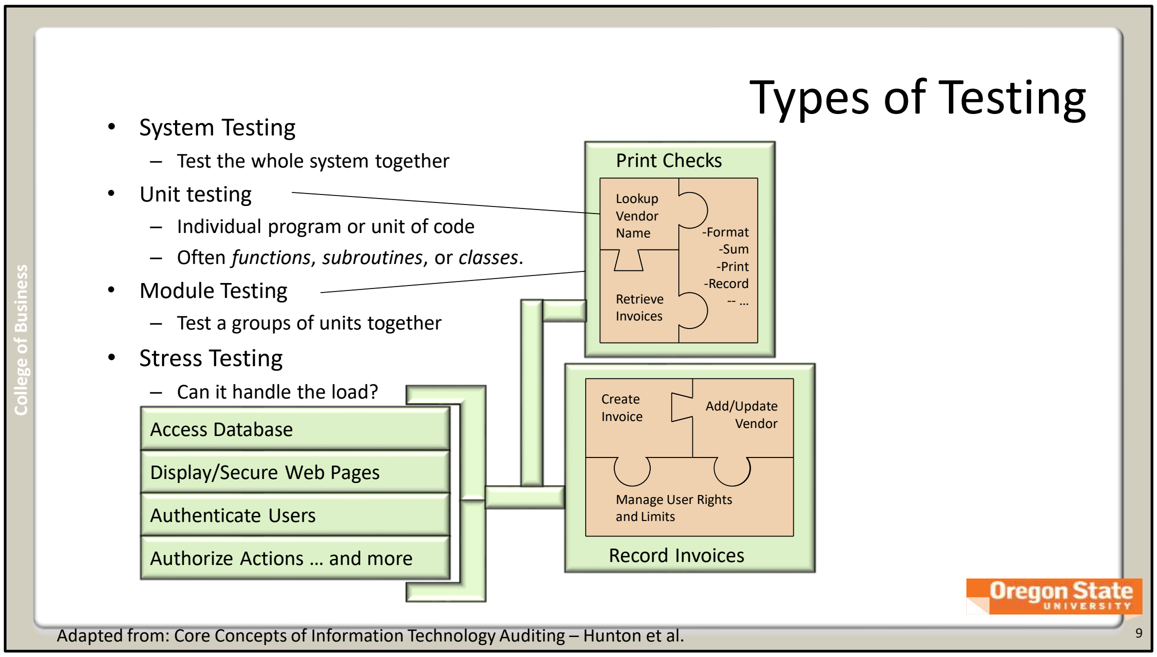
Increasingly, organizations are seeking to speed delivery of new programs and reduce deployment errors through "DevOps" (development operations). It is not too far wrong to say that, in DevOps the librarian functions are delegated to a largely automated system that verifies conditions (are tests passed, are approvals in place) and then automatically transfers newly developed components into the development or production environments. As with all automation efforts, this brings advantages and risks. An auditor needs to understand the process and assess controls, whether or not a DevOps system is in place.

The use of data in the development and testing process is often subject to controls:

- Leading practices say that developers should not have access to live data or live systems.
  - They might be able to learn things to help them implement nefarious code.
  - They might learn things that are supposed to be kept private.
  - Subtle logic inserted by developers might allow fraud to occur in ways that would be missed even by systematic tests.
  - Data sometimes needs to be scrubbed before it is used in a test environment. Privacy policies may forbid allowing developers to 'play with' real information about real customers. Scrubbing might involve replacing names, addresses, and network addresses as a start. But scrubbing can sometimes remove important features, hiding errors or patterns.

The sufficiency of the test environment can be a concern. For example:

- The test environment may not have as much storage or processing capacity. This could mean that issues are missed because the test data do not happen to contain examples that result in computational errors.
- Or testers may assume a program will be faster in production and only find out later that performance is inadequate.



Types of Testing

- System Testing
  - Test the whole system together
- Unit testing
  - Individual program or unit of code
  - Often *functions*, *subroutines*, or *classes*.
- Module Testing
  - Test a groups of units together
- Stress Testing
  - Can it handle the load?

Print Checks
Lookup Vendor Name
-Format -Sum -Print -Record -- …
Retrieve Invoices
Access Database
Display/Secure Web Pages
Authenticate Users
Authorize Actions … and more
Create Invoice
Add/Update Vendor
Manage User Rights and Limits
Record Invoices

College of Business

Adapted from: Core Concepts of Information Technology Auditing – Hunton et al.

Oregon State University

This slide and its animations list and illustrates various kinds of testing.

Do you see how the tests work together? No one type is enough by itself.

A key idea here is that the development team should test – not just the end users.