Preview: IT3345 : Software Architecture

Syllabus

Course Overview

This course focuses on the foundation of software architecture and software development planning. A software development plan (SDP) establishes the standards and procedures to use for all software development phases within an organization. During course assignments, you create an SDP for a fictional company that uses Agile for software development. The SDP defines the process methodology, resources, staffing, organization, scheduling, technical standards, software development phases, maintenance, and verification and validation plans.

Technology Resources

Capella offers tutorials, labs, or a virtual desktop as part of this course. These resources offer software or guided practice in performing tasks related to achieving course competencies and completing assignments. If you require the use of assistive technology or alternative communication methods to participate in these activities, please contact DisabilityServices@capella.edu to request accommodations.

Course Competencies

(Read Only)

To successfully complete this course, you will be expected to:

1. Define business problems that can be solved using software architecture concepts and standards.
2. Explain fundamental concepts and principles of software architecture.
3. Create a Software Development Plan for an organization.
4. Apply software process model strategies within the software architecture process.
5. Communicate effectively.

Course Prerequisites

Learners who have received credit for IT3340 may not take IT3345 or IT-FP3345. Prerequisite(s): IT2230 or IT-FP2230; IT3225 or IT-FP3225.

Required

The materials listed below are required to complete the learning activities in this course.

Integrated Materials

Many of your required books are available via the VitalSource Bookshelf link in the courseroom, located in your Course Tools. Registered learners in a Resource Kit program can access these materials using the courseroom link on the Friday before the course start date. Some materials are available only in hard-copy format or by using an access code. For these materials, you will receive an email with further instructions for access. Visit the Course Materials page on Campus for more information.

Book

Tsui, F., Karam, O., & Bernal, B. (2018). *Essentials of software engineering* (4th ed.). Burlington, MA: Jones & Bartlett Learning. ISBN: 9781284106008.

Library

The following required readings are provided in the Capella University Library or linked directly in this course. To find specific readings by journal or book title, use Journal and Book Locator. Refer to the Journal and Book Locator library guide to learn how to use this tool.

- Abran, A. (2004). *Guide to the software engineering body of knowledge: 2004 Edition: SWEBOK.* Skillsoft, Ireland.
- Calnan, C. (n.d.). *Introduction to Agile software development* [Tutorial]. Skillsoft, Ireland.
- Castillo, D. (n.d.). *Getting started with Amazon Web Services* [Tutorial]. Skillsoft, Ireland.
- Girvan, L., & Paul, D. (2017). *Agile and business analysis: Practical guidance for IT professionals.* Swindon, UK: BCS Learning & Development Ltd.
- Keenan, C. (2014). *OOD: UML activity diagram* [Video]. Skillsoft, Ireland.
- Laporte, C. Y., & Alain, A. (2018). *Software quality assurance.* Hoboken, NJ: John Wiley & Sons.
- Mueller, J. P. (2017). *AWS for developers for dummies.* Hoboken, NJ: John Wiley & Sons.
- Pender, T. (2003). *UML bible.* Indianapolis, Indiana: Wiley Publishing, Inc.
- Saleh, H. (2015). *Agile project management fundamentals: Moving to Agile* [Video]. Skillsoft, Ireland.
- Skillsoft (n.d.). *Advanced architecting on Amazon Web Services: Data storage architecture* [Tutorial].
- Skillsoft. (n.d). *Adopting an agile approach to project management* [Tutorial].
- Skillsoft. (n.d). *Engaging Agile stakeholders and leading Agile teams* [Tutorial].
- Skillsoft. (n.d). *IT project management essentials: Introduction to IT project management* [Tutorial].
- Skillsoft. (n.d.) *Agile planning: Project initiating and requirements gathering* [Tutorial].
- Skillsoft. (n.d.) *Software practices (SCRUM): SCRUM roles* [Tutorial].
- Tripathy, P., & Naik, K. (2014). *Software evolution and maintenance: A practitioner's approach.* Hoboken, NJ: John Wiley & Sons.
- Walters, B. (n.d). *Agile stakeholder engagement and team development* [Tutorial]. Skillsoft, Ireland.

External Resource

Please note that URLs change frequently. While the URLs were current when this course was designed, some may no longer be valid. If you cannot access a specific link, contact your instructor for an alternative URL. Permissions for the following links have been either granted or deemed appropriate for educational use at the time of course publication.

- Agile Alliance (n.d.). Agile glossary. Retrieved from https://www.agilealliance.org/agile101/agile-glossary/
- Doig, C. (2015). The benefits of doing a detailed enterprise software requirements analysis. Retrieved from http://www.cio.com/article/2923225/enterprise-software/the-benefits-of-doing-a-detailed-requirements-analysis-before-selecting-enterprise-software.html
- Manifesto for Agile software development. (n.d.). Retrieved from http://Agilemanifesto.org/
- Scrum Case Studies.com. (n.d.). Scrum case studies. Retrieved from http://www.scrumcasestudies.com/

Suggested

The following materials are recommended to provide you with a better understanding of the topics in this course. These materials are not required to complete the course, but they are aligned to course activities and assessments and are highly recommended for your use.

Optional

The following optional materials are offered to provide you with a better understanding of the topics in this course. These materials are not required to complete the course.

Library

The following optional readings may be available in the Capella University Library. To find specific readings by journal or book title, use Journal and Book Locator. Refer to the Journal and Book Locator library guide to learn how to use this tool. If the full text is not available, you may be able to request a copy through the Interlibrary Loan service.

- Skillsoft. (n.d.). *Microsoft Azure: Introduction* [Tutorial].
- Welton, T. (n.d.). *Getting started with Visio 2016* [Tutorial]. Skillsoft.

External Resource

Please note that URLs change frequently. While the URLs were current when this course was designed, some may no longer be valid. If you cannot access a specific link, contact your instructor for an alternative URL.

Permissions for the following links have been either granted or deemed appropriate for educational use at the time of course publication.

- IEEE Computer Society. (n.d.). SWEBOK V3. Retrieved from https://www.computer.org/web/swebok/v3
- Oracle. (2012). Ten questions to ask your cloud vendor before entering the cloud [PDF]. Retrieved from http://www.oracle.com/us/products/applications/10-questions-for-cloud-vendors-1639601.pdf
- UACG. (2010). Software development process – Activities and steps [PDF]. Retrieved from https://www.uacg.bg/filebank/acadstaff/userfiles/publ_bg_397_SDP_activities_and_steps.pdf

Projects

Project >> Software Development Plan

Project Overview

The course project requires you to complete a Software Development Plan (SDP) for a fictional company (CapraTek) that is introduced in Unit 1. A template is provided to help you define components of your SDP. The final product is an SDP that is suitable for CapraTek's goals to internalize their software development process and introduce an Agile methodology to their software development. Details for the CapraTek scenario are defined in the CapraTek Overview document.

Your SDP is completed during unit assignments in which you do the following:

- Define a specific Agile methodology that will be used for CapraTek's development process.
- Define resources, scheduling, organization, and staffing.
- Define technical standards for the development environment, languages, and systems.
- Define various software development phases.
- Describe verification and validation processes and outline a maintenance plan.

The SDP that you create can serve as a template that may be valuable to you in your software development career.

Introduction

Traditional Project Management

Traditional project management is the root of most modern project management techniques. Projects that are managed traditionally are generally well-defined and provide a detailed scope for the project. A common traditional approach is the waterfall methodology, which focuses heavily on planning and control. These projects work very well for mission critical projects that must be fully defined from the start.

The typical waterfall approach is normally accompanied by a great deal of documentation, planning, and requirements gathering. The methodology has a very specific series of phases that need to be executed sequentially. These phases include defining requirements, design, development, integration, testing, and finally, implementation. Each of these phases must be fully completed before moving onto the next phase.

This traditional approach is not a realistic model for most software development. Although the waterfall approach can be helpful for companies that like to have the control and predictability over the costs and schedules of a project, or for projects that are completed on an all-or-nothing premise (such as setting up a new network), it is not always possible to have projects work exactly as initially expected. The fact that the requirements are rigid makes it very difficult to adapt to changes or user requests as the project progresses.

The Agile methodology evolved as an alternative to the waterfall methodology because of these limitations.

Agile

Origins

The origins of the Agile movement are found in the original *Manifesto for Agile Software Development*. This manifesto was quite simple and focused on four value statements:

> *We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*
>
> - *Individuals and interactions over processes and tools.*
> - *Working software over comprehensive documentation.*

- *Customer collaboration over contract negotiation.*
- *Responding to change over following a plan. (n.d.)*

Initially, it was a method of development without documentation, process or underlying methodology. However, the Agile methodology has significantly matured, and methodologies such as Scrum have become more than just development processes. There is a very strong base of knowledge and experience behind these methodologies in their current state. The waterfall methodology has also evolved and incorporated some Agile concepts, making the documentation and process requirements more effective.

In the software development world, "agile" has been a very popular keyword. There are a lot of misconceptions of what this process methodology is, as some have viewed specific forms of Agile to be the only options. Scrum and Extreme Programming (XP) are two examples of Agile processes that are frequently considered to be synonymous with Agile.

Scrum

Scrum is a specific Agile methodology based on running small teams working on aspects of a project in an intensive and interdependent way. The term "Scrum" is derived from the scrummage formation in rugby. This is what was used to restart the game when play is stopped (similar to a face-off in hockey). Scrum works in such a way. It requires the team to make decisions in real-time based on the current situation of the project and project tasks. The teams are normally well-trained and are able to manage themselves, communicate effectively, and make efficient decisions. Project teams work on separate deliverables but get together on a very regular basis to work together and keep the team focused.

Extreme Programming

Extreme programming (XP) is another Agile software development methodology that is based on simplicity, communication, and feedback. Project teams meet to provide feedback on where each member is and adapt appropriately. Team also generally works with a "customer," who is viewed as a business representative and meets with the team daily. In XP, the team focuses on business value and creates the software based on small integrated releases.

The movement to adopt more Agile methodologies has gained significant momentum for many reasons. In part, methodologies and tools for implementing Agile projects have become accepted and understood by a larger audience. There is also a better understanding by project management in understanding the balance between project control and agility.

Reference

Manifesto for Agile Software Development. (n.d.). Retrieved from: http://agilemanifesto.org/

Learning Activities

Readings

Read the following in your *Essentials of Software Engineering* text:

- Chapter 4, "Software Process Models," pages 57–79.
- Chapter 5, "New and Emerging Process Methodologies," pages 83–101.

Use the Internet to read the following:

- Manifesto for Agile software development. (n.d.). Retrieved from http://Agilemanifesto.org/
  - This resource explains the original tenets of the Agile philosophy.
- Agile Alliance (n.d.). Agile glossary. Retrieved from https://www.agilealliance.org/agile101/agile-glossary/
  - This resource lists Agile terms and definitions.
- Scrum Case Studies.com. (n.d.). Scrum case studies. Retrieved from http://www.scrumcasestudies.com/
  - This resource contains various case studies for projects that use Scrum.

Skillsoft Resources

Use the Capella University Library to complete the following:

- Pender, T. (2003). *UML bible*. Indianapolis, Indiana: Wiley Publishing, Inc.
  - Read Chapter 13, "Modeling Behavior Using an Activity Diagram."
- Keenan, C. (2014). *OOD: UML activity diagram* [Video]. Skillsoft, Ireland.
  - This video demonstrates how to create a Unified Modeling Language (UML) activity diagram.
- Saleh, H. (2015). *Agile project management fundamentals: Moving to Agile* [Video]. Skillsoft, Ireland.
  - This video will walk you through various considerations about moving to Agile.
- Calnan, C. (n.d.) *Introduction to Agile software development* [Tutorial]. Skillsoft, Ireland.
  - This tutorial will give you an introduction into Agile Software Development with considerations about all the organizational requirements to move to Agile.
- Skillsoft. (n.d). *Adopting an agile approach to project management* [Tutorial].
  - The tutorial will walk you through the conversion from traditional projects to Agile and discusses strategies that can be used to adopt agile practices.
- Girvan, L., & Paul, D. (2017). *Agile and business analysis: Practical guidance for IT professionals*. Swindon, UK: BCS Learning & Development Ltd.
  - Read Chapter 2, "Agile Philosophy and Principles."
    - This chapter will cover some of the origins of Agile and the Agile manifesto. It will also cover some of the various Agile approaches.
  - Read Chapter 5, "Understanding Agile Methods and Frameworks."

- This chapter will provide you a foundation about many of the popular agile methods and approaches including XP, Scrum, RAD, Lean, and Kanban.

Optional Resources

You may want to consider visiting the following optional resources:

- Welton, T. (n.d.). *Getting started with Visio 2016* [Tutorial]. Skillsoft.
  - This will help you learn Visio basics.
- Washington State Department of Licensing Case [DOCX].
  - View a description of an organization switching from traditional to Agile software development in this document.

u01s1 - Learning Components

- Examine traditional life cycle diagrams for Agile methodologies.
- Compare Agile methodologies with traditional methodologies of software architecture.
- Examine phases of Agile methodologies.
- Explain the needs and requirements of CapraTek's software development process.
- Examine examples of the steps performed at each phase within an Agile methodology.
- Examine examples of UML diagrams.

u01s2 - Software Preparation and Technology Access

In this course, you may be using software and technology that is needed to complete designated activities and assignments. There is no additional cost for this software and technology. Some software packages will be made available to you at no additional cost through Capella's subscription with Microsoft, while other software packages are available for free download through open-source licensing.

Capella University requires learners to meet certain minimum computer requirements. Please note that some software required for a course may exceed these minimum requirements. Check the requirements for the software you may need to download and install to make sure it will work on your device. Most software will require a Windows PC. If you use a Mac, refer to Installing a Virtual Environment and Windows on a Mac.

The software and technologies below are strongly recommended to support you in completing the course objectives. If you have access to other tools that you believe may still meet the requirements of this course, please discuss your selected alternatives with your instructor.

If you use assistive technology or any alternative communication methods to access course content, please contact DisabilityServices@capella.edu with any access-related questions or to request accommodations.

For this course, follow the instructions provided through the links below to download and install software or register for an account, as required.

Microsoft Visio

1. If you have a Capella MS Imagine account, go to Step 2. Otherwise, see the instructions for registering an account at MS Imagine - Registration.
2. Log into the Capella Microsoft Imagine WebStore.
3. Identify the version of MS Visio that is compatible with your operating system.
4. Download and install.

If you encounter any difficulties in the download and installation process, post a detailed question in the Ask Your Instructor section of the course. Your instructor should be able to help you or point you in the right direction for the answers you need.

u01d1 - Comparing Agile Methodologies

Review the CapraTek Overview found in the discussion Resources as required.

There are several different Agile methodologies that CapraTek might use for its new software development process. Briefly compare two Agile methodologies and provide arguments for the selection of one of them based upon your understanding of CapraTek's requirements and needs.

Why might your choice be superior to a waterfall approach?

Response Guidelines

Comment on the post of at least two other learners. Offer insights, solutions, examples, or opinions that add depth and value to the conversation.

Note regarding discussions in this course: The content topic should determine the length of your post; however, a minimum of 150 words is recommended. Refer to the discussion participation scoring guide for posting expectations.

Make your initial posts by midweek to allow sufficient time for peers to respond. The expectation within the course discussions is to respond to at least two posts by the end of the unit, but it is highly recommended that you extend the dialogue further. Responding over multiple days will help stimulate a lively discussion.

Course Resources

Undergraduate Discussion Participation Scoring Guide

CapraTek Overview [DOCX]

u01d1 - Learning Components

- Compare Agile methodologies with traditional methodologies of software architecture.
- Explain the needs and requirements of CapraTek's software development process.

u01a1 - Agile Methodology

Overview

In this assignment you focus on the process methodology of Section 5, Standards and Procedures of your software development plan (SDP)—choosing, diagramming, explaining and finally applying your choice of Agile development methodology to CapraTek. The choices that you make here drive the rest of the SDP, so it is an appropriate starting point.

Preparation

Use the assignment Resources to complete the following:

- View the CapraTek scenario in the CapraTek Overview document.
- Download the Software Development Plan Template. Save it as "CapraTek_SDP_u1" and use it to complete and submit your assignments for this course.

Directions

Consider the CapraTek scenario and address the items below in Section 5 of the CapraTek SDP document.

- Select an Agile methodology and briefly justify why your chosen Agile methodology is appropriate for CapraTek.
- Create a Unified Modeling Language (UML) activity diagram that illustrates the process methodology life cycle.
- Describe each of the phases depicted in the process methodology diagram with relation to the software development process. Keep your descriptions generic—the point is to explain the processes involved in each phase to illustrate the concept.
    - Partial Example: Scrum initial requirements steps: "During the initial requirements steps, the product owner examines the product backlog and gets feedback from the customer and other stakeholders. The product owner then informs the development team of the items from the product backlog . . ."
- Select two of the phases and describe how each would manifest itself in the context of CapraTek. Consider CapraTek's requirements, resources, time constraints, et cetera.

Save and submit your SDP.

Course Resources
CapraTek Overview [DOCX]
Software Development Plan Template [DOCX]

Unit 2 >> Software Development Plan Scope and Purpose

Introduction

Once the development methodology, standards, and procedures have been identified, it is time to consider the realities of the SDP's organization and its capacity to staff its development efforts. Proper planning with respect to the constraints and capabilities possessed by an organization is critical to creating an effective software development plan. Resources, scheduling, organization, and staffing are all quite interdependent and should be considered in concert with one another when forming the plan.

Resources and Staffing

Identification of project roles and responsibilities are dependent upon many factors. One factor is the choice of the development methodology that will be adopted, as different development methodologies will have specific unique roles. For example, Scrum, a popular Agile methodology has three key roles including product owner, scrum master and developer.

Organization and Scheduling

The organization and scheduling take the resources and staffing to be organized into logical groups for projects. Depending on the development methodology, certain roles may overlap between projects whereas other roles are distinctly separate. The organization and scheduling for projects must take into consideration the resources, staffing, and the number of projects.

Learning Activities

u02s1 - Studies

Readings

Read the following in your *Essentials of Software Engineering* text:

- Chapter 1, "Creating a Program," pages 7–20.
- Chapter 2, “Building a Program," pages 23–38.
- Chapter 13, "Software Project Management," pages 267–293.

Skillsoft Resources

Use the Capella University Library to complete the following:

- Skillsoft. (n.d.). *Software practices (SCRUM): SCRUM roles* [Tutorial].
  - This tutorial walks you through the SCRUM roles involved in an Agile Scrum project.
- Walters, B. (n.d). *Agile stakeholder engagement and team development* [Tutorial]. Skillsoft, Ireland.
  - This tutorial walks you through how to engage the stakeholders and build the team within an Agile project.
- Skillsoft. (n.d). *Engaging Agile stakeholders and leading Agile teams* [Tutorial].
  - This tutorial discusses Agile teams and team development.
- Girvan, L., & Paul, D. (2017). *Agile and business analysis: Practical guidance for IT professionals*. Swindon, UK: BCS Learning & Development Ltd.
  - Read Chapter 3, "Analyzing the Enterprise."
    - This chapter provides a business analysis perspective of how we can apply the Agile methodology for the organization.
- Skillsoft. (n.d.) *Agile planning: Project initiating and requirements gathering* [Tutorial].
  - This tutorial provides information on project scoping.

Optional Resources

You may want to consider visiting the following optional resources:

- Sample Software Development Plan [PDF].
  - Review the attached sample SDP document. It is based on traditional project management but will work as a good starting point to evaluate some of the key elements in an SDP. Unlike a software requirements specification (SRS), the structure of the SDP is not as strict.

u02s1 - Learning Components

- Describe how an Agile methodology specifies personnel deployment.
- Identify elements that should be included in the scope and purpose sections of an SDP document.
- Understand how critical points of the software development production affect business operations.
- Describe how an Agile methodology specifies organization and scheduling operations.

u02d1 - Agile and Staffing

Staffing can be a challenge when there are multiple development projects going on at the same time. Resource management is often one of the crucial factors in determining the success of a project.

Research and find two sample resource and scheduling samples for Agile development projects to share with the class. Explain how they relate to CapraTek based on the existing staffing. Based on your Agile methodology chosen, what would a typical breakdown of staffing look like?

Response Guidelines

Comment on the post of at least two other learners. Offer insights, solutions, examples, or opinions that add depth and value to the conversation.

Note regarding discussions in this course: The content topic should determine the length of your post; however, a minimum of 150 words is recommended. Refer to the discussion participation scoring guide for posting expectations.

Make your initial posts by midweek to allow sufficient time for peers to respond. The expectation within the course discussions is to respond to at least two posts by the end of the unit, but it is highly recommended that you extend the dialogue further. Responding over multiple days will help stimulate a lively discussion.

Course Resources
Undergraduate Discussion Participation Scoring Guide

u02d1 - Learning Components

- Describe how an Agile methodology specifies personnel deployment.
- Describe how an Agile methodology specifies organization and scheduling operations.

u02a1 - Resources, Scheduling, Organization, and Staffing

Overview

A proper SDP document must consider the organization's capabilities and constraints. This assignment focuses on the first three sections of the SDP:

- Section 1: Introduction: Scope, Purpose, and Business Challenges.
- Section 2: Resources and Scheduling.
- Section 3: Organization and Staffing.

Considering the following questions may help guide you in this assignment:

- Why is there a need for the software development plan?
- What are CapraTek's motivations to create the new architecture?
- What are the consequences of not adopting a new architecture?
- What business problem will the new architecture solve?
- What resources and scheduling would be required for each development project?
- What roles need to be filled within the Agile methodology that you have selected?

Preparation

- Review the CapraTek scenario found in the assignment Resources as required.
- Save a new version of your SDP document using this unit number and use it to complete the assignment.

Directions

Consider the CapraTek scenario and address the items below in sections 1–3 of the CapraTek SDP document:

- Introduction (SDP Section 1): Complete the following sections:
  - Scope.
  - Purpose.
  - Business challenges: Identify and prioritize the challenges facing CapraTek.
- Resources and Staffing (SDP Section 2): Specify the project roles and responsibilities based on your Agile process methodology. Assume three concurrent and similarly scoped projects.
- Organization and Scheduling (SDP Section 3): Specify the project allocation breakdown based on the required resources for anticipated projects. As CapraTek does plan to run three software development projects simultaneously, resources may need to be shared between projects. Note: You will need to make, and state, general assumptions regarding and based on the approximate size of each project.

Save and submit your SDP.

Course Resources
CapraTek Overview [DOCX]
Software Development Plan Template [DOCX]

Unit 3 >> Technical Standards

Introduction

Technical standards establish a uniform set of processes and practices that are to be used by all staff and help define what tools, software, hardware, and programming languages the organization should employ. These may be as simple as just the identification of the technology used by the organization, but they can be as detailed as identifying the specific naming structure used in a programming language. It is important for an SDP to consider the big picture of how all the technical pieces will interface.

Factors to Consider

There are many factors to consider when it comes to an organization's standards and how they relate to each architecture. There are considerations such as the programming language, existing architecture, protocols, requirements, responsiveness, availability, and the total cost of ownership.

The technology platform and product road map are also a consideration. Generally, the choice of technology platform is not only project specific but also applies to the organization as well. If an entire IT architecture has been built on a development platform, such as Microsoft .NET, Java, or J2EE, it would make sense to continue to use that platform unless there is a specific reason to make the switch. Keeping systems consistent in an organization will save time and money as the organization can use the same resources to help maintain those systems.

Learning Activities

u03s1 - Studies

Readings

Read the following in your *Essentials of Software Engineering* text:

- Chapter 3, "Engineering of Software," pages 41–49.

Skillsoft Resources

Use the Capella University Library to complete the following:

- Castillo, D. (n.d). *Getting started with Amazon Web Services* [Tutorial]. Skillsoft, Ireland.
    - This tutorial introduces the basics about getting started with Amazon Web Services (AWS). CapraTek will use this resource.
- Mueller, J. P. (2017). *AWS for developers for dummies*. Hoboken, NJ: John Wiley & Sons.
    - Read Chapter 1, "Starting Your AWS Adventure."
        - This chapter explores the AWS cloud and understanding when, why, and how to use AWS.
    - Read Chapter 3, "Choosing the Right Services."

        - This chapter explores an overview of the features in AWS and matching those services to an organization's needs.

- Skillsoft (n.d.). *Advanced architecting on Amazon Web Services: Data storage architecture* [Tutorial].
    - This tutorial looks at the storage architecture required on Amazon Web Services used for software architecture on the cloud.

- Girvan, L., & Paul, D. (2017). *Agile and business analysis: Practical guidance for IT professionals*. Swindon, UK: BCS Learning & Development Ltd.
    - Read Chapter 4, "Adopting an Agile Mindset."
        - This chapter relates many of the Agile principles to business analysis.
    - Read Chapter 7, "Working with Stakeholders and Roles."
        - This chapter introduces the various types of stakeholders and customers.

Optional Resources

You may want to consider visiting the following optional resources:

- Oracle. (2012). Ten questions to ask your cloud vendor before entering the cloud [PDF]. Retrieved from http://www.oracle.com/us/products/applications/10-questions-for-cloud-vendors-1639601.pdf
    - This white paper focuses on questions to ask a cloud vendor before making the transition.

- Skillsoft. (n.d.). *Microsoft Azure: Introduction* [Tutorial].
    - This tutorial introduces Microsoft Azure.

u03s1 - Learning Components

- Identify technology stack components and how they work together.
- Explain the hardware requirements for a business using an Agile methodology.
- Review development and support tool options appropriate for an Agile methodology.

u03d1 - Technology Stack

Although many organizations prefer to use a single technology stack, it is not always viable because of various application needs. However, it is important to understand all the pieces of technology involved with certain technology stacks. The two most common are Microsoft .NET and Java-based technologies. In the selection of either, there are specific operating systems, databases, programming languages, development tools and servers that are used with each technology stack.

Research all the necessary technical standards for a technology stack and make recommendations of what must be included for an organization to use that technology stack for software development.

Response Guidelines

Comment on the post of at least two other learners. Offer insights, solutions, examples, or opinions that add depth and value to the conversation.

Note regarding discussions in this course: The content topic should determine the length of your post; however, a minimum of 150 words is recommended. Refer to the discussion participation scoring guide for posting expectations.

Make your initial posts by midweek to allow sufficient time for peers to respond. The expectation within the course discussions is to respond to at least two posts by the end of the unit, but it is highly recommended that you extend the dialogue further. Responding over multiple days will help stimulate a lively discussion.

Course Resources
Undergraduate Discussion Participation Scoring Guide

u03d1 - Learning Components

- Explain the hardware requirements for a business using an Agile methodology.

u03a1 - Technical Standards

Overview

Technical standards are important to ensure that all development staff utilize the same technologies for consistency, maintenance, and support. By ensuring that all individuals follow a specific set of standards, development processes can be efficiently completed by many different individuals rather than having the dependency on a few.

In this assignment you complete SDP Section 6, Technical Standards, which is intended to provide developers a high-level view of technologies to be used.

Preparation

- Review the CapraTek Overview found in the assignment Resources as required.
- Save a new version of your SDP document using this unit number and use it to complete the assignment.

Directions

Consider the CapraTek scenario and address the items below in Section 6 of the CapraTek SDP document:

- Create a technology stack component diagram showing where each technical operation is performed.

- Describe the following components and justify why each is important for solving the identified technical challenges:
    - Servers: Describe the various types of servers that are appropriate for integration with the identified applications.
    - Development Software: Identify four development and support tools needed for software development that both address identified technical challenges and that interoperate with Java and .NET operations. Justify your choices.

Save and submit your SDP.

Course Resources
CapraTek Overview [DOCX]
Software Development Plan Template [DOCX]

Unit 4 >> Software Development Phases

Introduction

The Institute of Electrical and Electronics Engineers (IEEE) is an organization that helps develop standards for the computer and electronics industry. The IEEE has developed the *Software Engineering Body of Knowledge* (SWEBOK) to establish standards for software development. For software development, there are four key knowledge areas or phases to which most organizations adhere. These phases include the software requirements, software design, software construction, and software testing.

The software requirements phase focuses on the elicitation, analysis, specification, and validation of the software needs. It also considers management of all those requirements throughout the life cycle of the software.

The software design phase considers the software requirements to define the software architecture and how it should be organized into components and the interfaces that connect them. Software design also focuses on creating various models that help form a blueprint for software creation.

The software construction phase is generally the phase that most individuals are familiar with in terms of creating and developing the software. This is where the actual coding takes place. This phase is strongly affected by the choice of Agile process methodology.

The software testing phase includes the validation and verification of the software applications. This phase has changed and matured quite significantly with the adoption of Agile methodologies.

Learning Activities

u04s1 - Studies

Readings

Read the following in your *Essentials of Software Engineering* text:

- Chapter 6, "Requirements Engineering," pages 105–129.
- Chapter 7, "Design: Architecture and Methodology," pages 130–167.
- Chapter 9, "Implementation," pages 191-207.
- Chapter 10, "Testing and Quality Assurance," pages 208–235.

Use the Internet to read the following resources:

- Doig, C. (2015). The benefits of doing a detailed enterprise software requirements analysis. Retrieved from http://www.cio.com/article/2923225/enterprise-software/the-benefits-of-doing-a-detailed-requirements-analysis-before-selecting-enterprise-software.html
    - This article explains why it is beneficial to complete a detailed enterprise software requirements analysis.

Skillsoft Resources

Use the Capella University Library to complete the following:

- Skillsoft. (n.d). *IT project management essentials: Introduction to IT project management* [Tutorial].
    - This tutorial focuses on project phases.
- Abran, A. (2004). *Guide to the software engineering body of knowledge: 2004 edition: SWEBOK.* Skillsoft, Ireland.
    - Read Chapter 2, "Software Requirements."
    - Read Chapter 3, "Software Design."
    - Read Chapter 4, "Software Construction."
    - Read Chapter 5, "Software Testing."

Optional Resources

You may want to consider visiting the following optional resources:

- UACG. (2010). Software development process – Activities and steps [PDF]. Retrieved from https://www.uacg.bg/filebank/acadstaff/userfiles/publ_bg_397_SDP_activities_and_steps.pdf
    - This document discusses the software development requirements analysis stage and provides an example of an activity diagram showing the steps involved.
- IEEE Computer Society. (n.d.). SWEBOK V3. Retrieved from https://www.computer.org/web/swebok/v3

- SWEBOK V3.0 is the most recent completely revised and updated version of the internationally respected Guide to the Software Engineering Body of Knowledge. Visit us to learn more about SWEBOK V3.0.

u04s1 - Learning Components

- Identify steps in each software development phase.
- Identify SWEBOK software development standards.
- Understand how an Agile methodology directs software development phases.

u04d1 - Software Development Phases

The *Guide to the Software Engineering Body of Knowledge* (SWEBOK) describes accepted knowledge about software engineering and is an internationally accepted. The first four phases defined in the SWEBOK guide are the most commonly used phases across most software development projects. However, there can be differences based on the specific methodologies being used.

Based on the Agile methodology that you have selected, research and share resources that help define one of the software phases. How do those resources help define that phase for CapraTek?

Response Guidelines

Comment on the post of at least two other learners. Offer insights, solutions, examples, or opinions that add depth and value to the conversation.

Note regarding discussions in this course: The content topic should determine the length of your post; however, a minimum of 150 words is recommended. Refer to the discussion participation scoring guide for posting expectations.

Make your initial posts by midweek to allow sufficient time for peers to respond. The expectation within the course discussions is to respond to at least two posts by the end of the unit, but it is highly recommended that you extend the dialogue further. Responding over multiple days will help stimulate a lively discussion.

Course Resources
Undergraduate Discussion Participation Scoring Guide
Guide to the Software Engineering Body of Knowledge: 2004 Edition: SWEBOK

u04d1 - Learning Components

- Identify steps in each software development phase.

u04a1 - Software Phases

Overview

By adhering to the key software development phases defined in the SWEBOK, CapraTek has signaled its commitment to creating consistent software projects.

In this assignment you complete SDP Section 7, Software Phases.

Preparation

- Review the CapraTek Overview found in the assignment Resources as needed.
- Save a new version of your SDP document using this unit number and use it to complete the assignment.

Optional Resource

- You may refer to the *SWEBOK V3* available in the assignment Resources.

Directions

Consider the CapraTek scenario and address the items below in Section 7 of the CapraTek SDP document:

1. Create activity diagrams depicting the steps involved in each of the following software development phases:
    - Phase 1: Software Requirements.
    - Phase 2: Software Design.
    - Phase 3: Software Construction.
    - Phase 4: Software Testing.
2. Describe the high level steps involved with each phase. Note: Phase 3 has already completed this portion as an example.
3. Explain how your chosen Agile methodology affects the process flow of each software development phase.

Course Resources
CapraTek Overview [DOCX]
SWEBOK V3
Software Development Plan Template [DOCX]

Unit 5 >> Verification, Validation, and Maintenance

Introduction

The maintenance process of software is not as exciting as the construction of the software, but it is equally important to consider. Once the initial structure of the software has been implemented, the project does not simply end there. Validating, verifying, and maintaining operations of the project need to be well defined prior to commencing work.

At the end of the software construction and test phases, the project scope needs validation. It can be done by comparing the scope to the software requirements specifications (SRS), the work breakdown structure, or the schedule, based on the deliverable. Project managers should also obtain the approval from the customers at this point to ensure that they are satisfied with the results.

A transition plan should be defined to ensure that the individuals that worked on the software can properly hand off the project to others that will maintain it. Having adequate documentation ensures that there is adequate knowledge management sharing to inform future changes or fix bugs.

Learning Activities

u05s1 - Studies

Readings

Read the following in your *Essentials of Software Engineering* text:

- Chapter 12, "Software Support and Maintenance," pages 254–267.

Skillsoft Resources

Use the Capella University Library to complete the following:

- Tripathy, P., & Naik, K. (2014). *Software evolution and maintenance: A practitioner's approach*. Hoboken, NJ: John Wiley & Sons.
    - Read Chapter 2, "Taxonomy of Software Maintenance and Evolution."
        - This chapter considers common approaches for software maintenance in the long term.
    - Read Chapter 3, "Evolution and Maintenance Models."
        - This chapter considers some of the standard maintenance models that can be applied to your project.
- Laporte, C. Y., & Alain, A. (2018). *Software quality assurance*. Hoboken, NJ: John Wiley & Sons.

- Read Chapter 7, "Verification and Validation."
  - This chapter covers the verification and validation process and understand how verification and validation fits into the software quality assurance process.

Course Resources

Tsui, F., Karam, O., & Bernal, B. (2018). *Essentials of software engineering* (4th ed.). Burlington, MA: Jones & Bartlett Learning. ISBN:9781284106008.

u05s1 - Learning Components

- Explain principles of verification and validation plans.
- Describe the process to close a software project.
- Explain aspects of software maintenance.

u05d1 - Application Maintenance

Discuss the following in your post:

- Why software applications such as the ones developed by CapraTek need to be maintained.
- Which part of the software maintenance process you believe is the most complex, and which is the most important. Explain your reasoning.

Response Guidelines

Comment on the post of at least two other learners. Offer insights, solutions, examples, or opinions that add depth and value to the conversation.

Note regarding discussions in this course: The content topic should determine the length of your post; however, a minimum of 150 words is recommended. Refer to the discussion participation scoring guide for posting expectations.

Make your initial posts by midweek to allow sufficient time for peers to respond. The expectation within the course discussions is to respond to at least two posts by the end of the unit, but it is highly recommended that you extend the dialogue further. Responding over multiple days will help stimulate a lively discussion.

Course Resources

Undergraduate Discussion Participation Scoring Guide

u05a1 - Verification, Validation, and Maintenance

Overview

Once a software development project is completed at CapraTek, it is also important that there is a plan in place to have the system verified and validated to assure it is the right system and is developed correctly. The SDP needs to spell out how independent verification and validation is to be conducted.

Software is rarely completely done at the end of a project. There may be bugs that are uncovered or additional criteria that must be implemented. Planning out a process for the maintenance of the software is crucial. A large part of the software maintenance is to preserve its integrity as changes are made. Any changes should be logged and tracked. There needs to also be criteria put in place when certain software needs to be retired or migrated.

In this assignment you will complete SDP Section 8, Validation and Verification, and Section 9, Maintenance Process.

Preparation

- Review the CapraTek Overview found in the assignment Resources.
- Save a new version of your SDP document using this unit number and use it to complete the assignment.

Directions

Consider the CapraTek scenario and address the items below in Section 8 and Section 9 of the CapraTek SDP document:

- Create a plan for the independent verification and validation of software based upon your chosen Agile methodology per SWEBOK standards (SDP Section 8). It should be one page or less.
- Create an activity diagram demonstrating the software maintenance process (SDP Section 9).
- Create a maintenance plan that specifies how to maintain deployed software per SWEBOK standards (SDP Section 9). It should be one page or less.

Save and submit your SDP.

Course Resources
CapraTek Overview [DOCX]
Software Development Plan Template [DOCX]