Upon completion of this week's activities, you will be able to:

- Discuss how software design is a consistent approach and method for the develop-ment of software requirements in defined designs of a work product.
- Identify how the software architecture definition provides a framework for the creation of the product design and at times can provide constrictions.
- Explain how the software design definition implements details about a software product's architecture, components, and interfaces.

عند الانتهاء من أنشطة هذا الأسبوع، سوف تكون قادرا على:

- ناقش كيف يكون تصميم البرمجيات نهجا متسقا وطريقة لتطوير متطلبات البرمجيات في تصاميم محددة لمنتج العمل.
- تحديد كيف توفر هندسة البرمجيات إطارا لإنشاء تصميم المنتج وفي بعض الأحيان يمكن أن توفر قيود.
- اشرح كيف ينفذ تعريف تصميم البرامج تفاصيل عن بنية منتج البرنامج ومكوناته و واجهاته.

SOFTWARE DESIGN

Purpose: develops software requirements in defined designs of a work product.

- Implements details about a software product's architecture, components, and interfaces.
- Used by software designers.
- documented according to program and project plans, ideas, processes, and procedures and applicable internal work instructions.

تصميم البرمجيات

الغرض: تطوير متطلبات البرامج في التصاميم المحددة لمنتج العمل.

- تنفيذ تفاصيل حول بنية منتج البرمجيات، والمكونات، والواجهات.
- يستخدم من قبل مصممي البرمجيات.
- توثيقها وفقا لخطط البرامج والمشاريع والأفكار والعمليات والإجراءات وتعليمات العمل الداخلية المعمول بها.

DEVELOPMENT PLAN

Purpose: a well-defined process useful for implementation and applicable standards.

Tasks: identify major software functions (components), functional hierarchy diagrams, and hardware/software interfaces.

خطة التطوير

الغرض: عملية محددة جيدا مفيدة للتنفيذ والمعايير المطبقة.

المهام: تحديد وظائف البرمجيات الرئيسية (المكونات)، مخططات التسلسل الهرمي الوظيفي، واجهات الأجهزة / البرمجيات.

SOFTWARE DESIGN DECISIONS

Purpose: provides design concepts and decisions for a work product.

- Analysis and integration of software requirements definition and the software operational concepts identify the capabilities and characteristics required to make key design decisions.
- Software designer uses software design tools for requirements, code development, configuration management (CM), and software documentation.

قرارات تصميم البرمجيات

الغرض: يوفر مفاهيم التصميم والقرارات لمنتج العمل.

- تحليل وتكامل تعريف متطلبات البرمجيات والمفاهيم التشغيلية للبرمجيات ،تحديد القدرات والخصائص المطلوبة لاتخاذ قرارات التصميم الرئيسية.
- مصمم البرامج يستخدم أدوات تصميم البرمجيات للمتطلبات، وتطوير التعليمات البرمجية، وإدارة التكوين (CM)، وتوثيق البرمجيات.

SOFTWARE REQUIREMENTS EVALUATION

Purpose: defines software operational scenarios to ensure problems affecting software design are identified, evaluated, and resolved.

- A risk analysis using prototype software is performed to support early requirements evaluations and design feasibility.
- If requirement is proven unusable and not to be implemented for use, data from evaluations is fed back into the output of the software requirements development phase.

تقييم متطلبات البرنامج

الغرض: يحدد السيناريوهات التشغيلية للبرنامج لضمان تحديد المشاكل التي تؤثر على تصميم البرمجيات وتقييمها وحلها.

• يتم إجراء تحليل للمخاطر باستخدام برمجية النموذج الأولي لدعم تقييم المتطلبات المبكرة وجدوى التصميم.

• إذا ثبت أن المتطلبات غير صالحة للاستعمال ولم يتم تنفيذها للاستخدام، فإن البيانات المستمدة من عمليات التقييم تتغذى مرة أخرى على ناتج مرحلة تطوير متطلبات البرمجيات.

SOFTWARE REUSE

Purpose: identifies evaluations by software architecture definitions on how to decide on the incorporation of components into the software design.

Reuse Criteria: identified in defined software plans

- Determines if the program and project re-use library or existing software work products can be used for near-term software design activities.

إعادة استخدم البرمجيات

الغرض: تحديد التقييمات حسب تعريفات معمارية البرمجيات حول كيفية اتخاذ قرار بشأن دمج المكونات في تصميم البرمجيات.

معايير إعادة الاستخدام: المحددة في خطط برمجية محددة

• يحدد ما إذا كان البرنامج والمشروع إعادة استخدام المكتبة أو منتجات العمل البرمجيات الحالية لأنشطة تصميم البرمجيات على المدى القريب.

PEER REVIEWS

Purpose: to find and correct as many errors as possible before test team integration or customers find problems.

- Starts with requirements, design models, and uninterrupted code and unit tests for the software designer.
- Applied at various stages during the software design/development life cycle
- Creates clean software work products and provides assurance that issues or errors are discovered and resolved.

أستعراض النظير

الغرض: لإيجاد وتصحيح العديد من الأخطاء كل ما أمكن قبل اختبار الفريق للتكامل أو عثور العملاء على المشاكل.

- يبدأ مع المتطلبات ونماذج التصميم، والترميز المتصل واختبار الوحدات لمصمم البرنامج.
- تطبق في مراحل مختلفة خلال دورة تصميم / تطوير البرمجيات
- يخلق منتجات عمل البرمجيات النظيفة ويوفر ضمان أن القضايا أو الأخطاء يتم اكتشافها وحلها.

Examples of peer review methods

- Inspections
- Structured walk-throughs
- Deliberate refactoring
- Pair programming

أمثلة على أساليب مراجعة النظراء

- التفتيش
- تنظيمات المشي
- إعادة البيع المتعمد
- البرمجة الثنائيه

PEER REVIEW CRITERIA

- Schedule the peer review at a convenient time
- Assign reviewers
- Prepare/update materials
- Provide checklists
- Introduce training materials
- Select software work products
- Provide entry and exit criteria

معايير استعراض النظراء

- جدولة استعراض النظراء في وقت مناسب
- تعيين المراجعين
- إعداد / تحديث المواد
- تقديم قوائم المراجعة
- إدخال مواد تدريبية
- اختيار منتجات عمل البرمجيات
- توفير معايير الدخول والخروج

SOFTWARE DESIGN/DEVELOPMENT METHODS

Concurrent Software/Design Development

- Requirement: software design expertise to anticipate where the defined design is going.
- Con: possible to delay commitment until the last moment when failure to make a decision eliminates an important alternative or decision.

Lean Software Design/Development

- Objective: to move as many changes as possible from the top curve to the bottom curve.
- Delays the freezing of all design decisions as long as possible.
- Emphasizes designing and managing changes throughout the life cycle.
- Provides a better understanding of software engineering and quick delivery to customers.

أساليب تصميم / تطوير البرمجيات

البرامج المتزامنة / تطوير التصميم

- البرامج المتزامنة / تطوير التصميم
- المتطلب: تصميم البرمجيات الخبرة لاستباق حيث التصميم المحدد هو الذهاب.
- con: من الممكن تأجيل الالتزام حتى آخر لحظة عندما يؤدي عدم اتخاذ قرار إلى إلغاء بديل أو قرار مهم.

Lean تصميم البرمجيات / التطوير

- الهدف: نقل أكبر عدد ممكن من التغييرات من المنحنى العلوي إلى منحنى أسفل.
- يؤخر تجميد جميع قرارات التصميم لأطول فترة ممكنة.
- تشدد على تصميم وإدارة التغييرات طوال دورة الحياة .
- يوفر فهم أفضل لهندسة البرمجيات والتسليم السريع للعملاء.

AGILE SOFTWARE PROCESSES

- Provides fewer defects.
- Supports numerous initiatives
- Provides a program and project with a manager's approach to emphasize short-term program and project planning.
- Adopts values that are consistently depicts processes and makes decisions that may reject a software design.
- More effective than the traditional models due to perfection versus good-enough concepts for software design practices.
- Provides capability to understand information first before jumping into software design and development.

العمليات البرمجية agile

- يوفر عيوب أقل
- يدعم العديد من المبادرات
- يوفر برنامجا ومشروعا مع نهج المدير للتأكيد على تخطيط البرامج والمشاريع على المدى القصير
- يعتمد القيم التي تصور باستمرار العمليات واتخاذ القرارات التي قد ترفض تصميم البرمجيات
- أكثر فعالية من النماذج التقليدية بسبب الكمال مقابل مفاهيم جيدة بما فيه الكفاية لممارسات تصميم البرمجيات.
- يوفر القدرة على فهم المعلومات أولا قبل القفز إلى تصميم البرمجيات والتطوير.

FOUR KEY ELEMENTS OF AGILE SOFTWARE ENGINEERING

1. The team has control of work assignments
2. Communication with team members and customers is needed
3. Change is good: "Think outside the box"
4. Customer satisfaction and expectations are achieved

أربعة عناصر رئيسية للهندسة البرمجيات Agile

١. الفريق لديه السيطرة على مهام العمل
٢. هناك حاجة للتواصل مع أعضاء الفريق والعملاء
٣. التغيير جيد: "فكر خارج الصندوق"
٤. يتم تحقيق رضا العملاء والتوقعات

CONFIGURATION MANAGEMENT

CM methods are a supporting discipline not directly involved in creating executable code. In the Agile process, CM methods are not referenced for specific routines.

CM Purpose: trim process and provide more automation in tools

- Brings back focus to configuration control objectives.
- Software tools common to other team members are adapted to the process

- When Agile processes lack configuration control, Lean principles are a waste of time and lead to chaos. As a result, there is no progress in software design/development.
- The ability to control change is the foundation of design/development activities. CM methods should not limit change or it will become a barrier for program and project plans.

إدارة التكوين

طرق CM هي دعم النضباط وليس مشاركه مباشرة في إنشاء رمز قابل للتنفيذ. في عملية Agile ، لا يتم الإشارة إلى أساليب CM لروتينات محددة.

الغرض: عملية تقليم وتوفير المزيد من الأتمتة في الأدوات (أتمتة؛ الآلية الذاتية : تقنيه يستطاع بها جعل عملية ما اوتوماتيكيه)

- يعيد التركيز على أهداف التحكم في التكوين.
- يتم تكييف أدوات البرمجيات المشتركة لأعضاء الفريق الآخرين للعملية

• عندما عمليات Agile تفتقر إلى التحكم في التكوين، مبادئ العجاف هي مضيعة للوقت وتؤدي إلى الفوضى. ونتيجة لذلك، لم يحرز أي تقدم في تصميم البرمجيات وتطويرها.

• القدرة على التحكم في التغيير هي أساس أنشطة التصميم / التطوير. لا ينبغي أن تحد أساليب CM من التغيير أو أنها ستصبح عائقا أمام خطط البرامج والمشاريع.

SOFTWARE STANDARDS

Purpose: ensures development processes are in accordance with identified process models.

Minimum software standards consist of the following:

- Documented and maintained plans and procedures
- Peer reviews
- Standard software tools

معايير البرنامج

الغرض: التأكد من أن عمليات التطوير تتماشى مع نماذج العملية المحددة.

الحد الأدنى من معايير البرمجيات تتكون مما يلي:

- توثيق وصيانة الخطط والإجراءات
- آراء Peer
- أدوات البرمجيات القياسية

CAPABILITY MATURITY MODEL INTEGRATION

Purpose: provides opportunity to address software design/development with support to customers after delivery.

- Provides a systematic approach to software engineering tasks in programs and projects.
- Enhance the knowledge base for software designers.
- Provides content for performance during the software life cycle.

CAPABILITY MATURITY MODEL INTEGRATION

الغرض: يوفر فرصة لمعالجة تصميم البرمجيات / التطوير مع دعم للعملاء بعد التسليم.

• يوفر نهجا منظم لمهام هندسة البرمجيات في البرامج والمشاريع.

• تعزيز قاعدة المعرفة لمصممي البرمجيات.

• يوفر محتوى للأداء خلال دورة حياة البرنامج.

CMMI software engineering tasks

- Identify internal and external interfaces
- Establish infrastructure abilities with software design
- Develop plans, processes, and procedures
- Reuse capabilities for identified software

cMMI المهام هندسة البرمجيات

• تحديد الواجهات الداخلية والخارجية

• إنشاء قدرات البنية التحتية مع تصميم البرمجيات

• وضع الخطط والعمليات والإجراءات

• إعادة استخدام قدرات البرامج المحددة

LEAN SIX SIGMA

Purpose: reduces process variation, resulting in fewer errors and defects.

Goal: Zero defects

Six Sigma Process:

1. Define
2. Measure
3. Analyze
4. Improve

LEAN SIX SIGMA

الغرض: يقلل من تغير العملية، مما يؤدي إلى عدد أقل من الأخطاء والعيوب.

الهدف: العيوب صفر

Six Sigma Process

١. تحديد
٢. قياس
٣. تحليل
٤. تحسين

Lean Goal: Eliminate eight wastes

1. Defects
2. Overproduction
3. Waiting
4. Nonutilized talent
5. Transportation
6. Inventory
7. Motion
8. Excess processing

Lean Goal: Eliminate eight wastes

١. العيوب
٢. الإفراط
٣. الانتظار
٤. المواهب غير المستخدمه
٥. النقل
٦ - الجرد
٧. الحركة
٨. المعالجة الزائدة

Software Design/Development

Purpose: secure databases related to software.

- Effective software and systems integration methods create profit from inside the software design/development sector.

تصميم وتطوير البرمجيات

الغرض: قواعد بيانات آمنة تتعلق بالبرمجيات.

- تؤدي أساليب تكامل البرمجيات والنظم الفعالة إلى تحقيق الربح من داخل قطاع تصميم / تطوير البرمجيات.