**שאלות פרק 2 Sommerville**

.10 מהם 4 התהליכים הבסיסיים שקיימים בכל פיתוח תוכנה?

.11 **תאר** את המודלים הבאים, **הסבר** את ההבדלים בניהם ו**הדגם** באמצעות מערכת מתאימה:

מודל מפל המים, מודל אינקרמנטאלי, מודל פיתוח, מוכוון שימוש חוזר, מודל בוהם, RUP .

.12 אחד מהתהליכים המשמעותיים ביותר הוא הגדרת דרישות. מהם 4 השלבים ומתי הם מתרחשים.

.13 מהו תהליך עיצוב המערכת, מהם הקלטים והפלטים לשלב זה?

.14 מה ההבדל בין אימות לתיקוף, הסבר והדגם באמצעות דוגמה.

.15 בחר בשיטה המתאימה ביותר לניהול שינויים והסבר כיצד תומכת השיטה בכך.

.16 מהו המודל המתאים ביותר לפיתוח מערכות "זמן אמת"?

**שאלה 10:** מהם 4 התהליכים הבסיסיים שקיימים בכל פיתוח תוכנה?

1. מפרט תוכנה- רשימת הפעולות שהתוכנה יכולה לבצע ואילוצים הנובעים תוך כדי פיתוח התוכנה חייבים להיות מוגדרים מראש

2. עיצוב תוכנה ויישום- התוכנה חייבת להיווצר על פי המפרט שהוגדר לה ללא חריגות

3. אימות תוכנה- התוכנה חייבת להיות מאומתת על מנת לוודא שהיא עושה מה שהלקוח ביקש בהזמנתו, כלומר מותאמת לצורכי הלקוח, וכמובן לבדוק שהיא פותחה נכון ועונה על הדרישות של התהליכים שהוגדרו בעת אפיון התוכנה

4. אבולוציית התוכנה- התוכנה חייבת להתפתח כדי לענות על הצרכים המשתנים של הלקוח, כלומר חייבים להוציא גרסאות חדשות לתוכנה כל הזמן על מנת לענות על הצרכים של הלקוחות, צורכיהם עלולות להשתנות בכל פרק זמן בגלל שינויים ארגוניים או שינויים במטרות השיווקיות של הלקוח

**שאלה 11:** תאר את המודלים הבאים, הסבר את ההבדלים בניהם והדגם באמצעות מערכת מתאימה:

מודל מפל המים, מודל אינקרמנטאלי, מודל פיתוח, מוכוון שימוש חוזר, מודל בוהם. RUP,

**11.**

Waterfall model

Because of the cascade from one phase to another, this model is known as the ‘waterfall model’ or software life cycle. The waterfall model is an example of a plan-driven process—in principle, you must plan and schedule all of the process activities before starting work on them.

The principal stages of the waterfall model directly reflect the fundamental development

activities:

1. *Requirements analysis and definition* The system’s services, constraints, and goals are established by consultation with system users. They are then defined in detail and serve as a system specification.
2. *System and software design* The systems design process allocates the requirements to either hardware or software systems by establishing an overall system architecture. Software design involves identifying and describing the fundamental software system abstractions and their relationships.
3. *Implementation and unit testing* During this stage, the software design is realized as a set of programs or program units. Unit testing involves verifying that each unit meets its specification.
4. *Integration and system testing* The individual program units or programs are integrated and tested as a complete system to ensure that the software requirements have been met. After testing, the software system is delivered to the customer.
5. *Operation and maintenance* Normally (although not necessarily), this is the longest life cycle phase. The system is installed and put into practical use. Maintenance involves correcting errors which were not discovered in earlier stages of the life cycle, improving the implementation of system units and enhancing the system’s services as new requirements are discovered.

In principle, the result of each phase is one or more documents that are approved. The following phase should not start until the previous phase has finished. In practice, these stages overlap and feed information to each other. During design, problems with requirements are identified. During coding, design problems are found and so on. The software process is not a simple linear model but involves feedback from one phase to another. Documents produced in each phase may then

have to be modified to reflect the changes made. Because of the costs of producing and approving documents, iterations can be costly and involve significant rework. Therefore, after a small number of iterations, it is normal to freeze parts of the development, such as the specification, and to continue

with the later development stages.

**Advantages :**

1. Easy to understand and implement.

2. Widely used and known (in theory!).

3. Reinforces good habits: define-before- design,

design-before-code.

4. Identifies deliverables and milestones.

5. Works well on mature products and weak teams.

**Disadvantages :**

1. Idealized, doesn’t match reality well.

2. Doesn’t reflect iterative nature of exploratory

development.

3. Unrealistic to expect accurate requirements so

early in project.

4. Software is delivered late in project, delays discovery

of serious errors.

5. Difficult to integrate risk management.

6. Difficult and expensive to make changes to

documents, ”swimming upstream”.

7. Significant administrative overhead, costly for small

teams and projects

Incremental development

Incremental development is based on the idea of developing an initial implementation, exposing this to user comment and evolving it through several versions until an adequate system has been developed. Specification, development, and validation activities are interleaved rather than separate, with rapid feedback across activities. Incremental development reflects the way that we solve problems.

We rarely work out a complete problem solution in advance but move toward a solution in a series of steps, backtracking when we realize that we have made a mistake. By developing the software incrementally, it is cheaper and easier to make changes in the software as it is being developed.

Each increment or version of the system incorporates some of the functionality that is needed by the customer. Generally, the early increments of the system include the most important functionality. This means that the customer can evaluate the system at a relatively early stage in the development to see if it delivers what is required. If not, then only the current increment has to be changed and, possibly, new functionality defined for later increments

**Advantages :**

1.The cost is reduced.

2.The amount of analysis and documentation that has to be redone is much less.

3. It is easier to get customer feedback.

4. Customers can comment on demonstrations of the software

5. Customers can see how much has been implemented.

6. More rapid delivery and deployment of useful software to the customer

**Disadvantages:**

1.Needs good planning and design.

2.Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.

3.Total cost is higher than [waterfall](http://istqbexamcertification.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/).

Reuse-oriented software engineering

In the majority of software projects, there is some software reuse. This often happens informally when people working on the project know of designs or code that are similar to what is required. They look for these, modify them as needed, and incorporate them into their system.

Reuse-oriented approaches rely on a large base of reusable software components and an integrating framework for the composition of these components. Sometimes, these components are systems in

their own right that may provide specific functionality.

Stages of reuse-oriented model are:

1. *Component analysis* Given the requirements specification, a search is made for components to implement that specification. Usually, there is no exact match and the components that may be used only provide some of the functionality required.
2. *Requirements modification* During this stage, the requirements are analyzed using information about the components that have been discovered. They are then modified to reflect the available components. Where modifications are impossible, the component analysis activity may be re-entered to search for alternative solutions.
3. *System design with reuse* During this phase, the framework of the system is designed or an existing framework is reused. The designers take into account the components that are reused and organize the framework to cater for this. Some new software may have to be designed if reusable components are not available.
4. *Development and integration* Software that cannot be externally procured is developed, and the components and COTS systems are integrated to create the new system. System integration, in this model, may be part of the development process rather than a separate activity.

Reuse-oriented software engineering has the obvious advantage of reducing the

amount of software to be developed and so reducing cost and risks. It usually also

leads to faster delivery of the software. However, requirements compromises are

inevitable and this may lead to a system that does not meet the real needs of users.

Furthermore, some control over the system evolution is lost as new versions of the

reusable components are not under the control of the organization using them.

**Advantages :**

1.The cost is reduced.

2. Reducing the amount of software to be developed.

3. Usually also leads to faster delivery of the software.

4. Customers can comment on demonstrations of the software

**Disadvantages:**

1. May lead to a system that does not meet the real needs of users.

2. over the system evolution is lost as new versions of the

reusable components are not under the control of the organization using them.

**The Rational Unified Process(RUP)**

The Rational Unified Process (RUP) is an example of a modern process model that has been derived from work on the UML. It is a good example of a hybrid process model. The RUP is a phased model that identifies four discrete phases in the software process. However, unlike the waterfall model where phases are equated with process activities, the phases in the RUP are more closely related to business rather than technical concerns.

The practice perspective on the RUP describes good software engineering practices

that are recommended for use in systems development. Six fundamental best

practices are recommended:

1. *Develop software iteratively* Plan increments of the system based on customer priorities and develop the highest-priority system features early in the development process.
2. *Manage requirements* Explicitly document the customer’s requirements and keep track of changes to these requirements. Analyze the impact of changes on the system before accepting them.
3. *Use component-based architectures* Structure the system architecture into components, as discussed earlier in this chapter.
4. *Visually model software* Use graphical UML models to present static and dynamic views of the software.
5. *Verify software quality* Ensure that the software meets the organizational quality standards.
6. *Control changes to software* Manage changes to the software using a change management system and configuration management procedures and tools.

The most important innovations in the RUP are the separation of phases and workflows, and the recognition that deploying software in a user’s environment is part of the process. Phases are

dynamic and have goals. Workflows are static and are technical activities that are not associated with a single phase but may be used throughout the development to achieve the goals of each phase.

**Advantages:**

1. This is a complete methodology in itself with an emphasis on accurate documentation
2. It is proactively able to resolve the project risks associated with the client's evolving requirements requiring careful [change request management](http://www.my-project-management-expert.com/change-request-management.html)
3. Less time is required for integration as the process of integration goes on throughout the software development life cycle.
4. The development time required is less due to reuse of components.
5. There is online training and tutorial available for this process.

**Disadvantages:**

1. The team members need to be expert in their field to develop a software under this methodology.
2. The development process is too complex and disorganized.
3. On cutting edge projects which utilise new technology, the reuse of components will not be possible. Hence the time saving one could have made will be impossible to fulfill.
4. Integration throughout the [process of software development](http://www.my-project-management-expert.com/process-of-software-development.html), in theory sounds a good thing. But on particularly big projects with multiple development streams it will only add to the confusion and cause more issues during the stages of testing

12: אחד מהתהליכים המשמעותיים ביותר הוא הגדרת דרישות. מהם 4 השלבים ומתי הם מתרחשים?

1. *Feasibility study* An estimate is made of whether the identified user needs may be

satisfied using current software and hardware technologies. The study considers

whether the proposed system will be cost-effective from a business point of view

and if it can be developed within existing budgetary constraints. A feasibility

study should be relatively cheap and quick. The result should inform the decision

of whether or not to go ahead with a more detailed analysis.

2. *Requirements elicitation and analysis* This is the process of deriving the system

requirements through observation of existing systems, discussions with potential

users and procurers, task analysis, and so on. This may involve the development

of one or more system models and prototypes. These help you understand

the system to be specified.

3. *Requirements specification* Requirements specification is the activity of translating  
the information gathered during the analysis activity into a document that  
defines a set of requirements. Two types of requirements may be included in this  
document. User requirements are abstract statements of the system requirements  
for the customer and end-user of the system; system requirements are a  
more detailed description of the functionality to be provided.

4. *Requirements validation* This activity checks the requirements for realism, consistency,

and completeness. During this process, errors in the requirements document

are inevitably discovered. It must then be modified to correct these problems.

Requirements occur at the beginning of the project

.13 מהו תהליך עיצוב המערכת, מהם הקלטים והפלטים לשלב זה?

**A software design** is a description of the structure of the software to be implemented,

the data models and structures used by the system, the interfaces between system components

and, sometimes, the algorithms used.  
this process showing the **inputs** to the design process, process activities, and the documents produced as **outputs** from this process.

These activities lead to a set of design outputs, The detail and representation of these vary considerably,in addition the outputs of the design process may not

be separate specification documents but may be represented in the code of the program

14. מה ההבדל בין אימות לתיקוף, הסבר והדגם באמצעות דוגמה?

**אימות (Verification):**

בדיקות תוך כדי התהליך- נועדו לוודא ששלב הסתיים כראוי

בדיקות ללא הרצה

:(tests executional-non) סקירה (walkthrough), ביקורת(inspection), סקר עמיתים (peer review)  
בדיקות הרצה ליחידות ולשילובים  
אימות אוטומטי: הוכחת נכונות

**תקפות (Validation)**

בדיקת המוצר, או חלקיו לאחר פיתוח בסביבה אמיתית.

בדיקה בהרצה execution-based testing

כאשר בודק צריך לאשר כי דרישה מסויימת או סעיף מסויים באפיון אכן תקין ומולא עליו לבדוק את המערכת, להריץ בדיקה כלשהי ורק לאחר בדיקה הוא יוכל לטעון האם הדרישה מולאה והאם היא תקינה. מקרה מסוג זה הינו ולידציה- הבודק מוודא שהדרישה מולאה על ידי הרצה של המערכת- בדיקה דינמית.

במקרה מסוג הזה, על הבודק לוודא כי דרישות הגורם החיצוני אכן מולאו על פי דרישתו ולשביעות רצונו.

לעומת זאת כאשר הבודק צריך לאמת את הדרישה, ולבדוק שאכן הדרישה מופיעה בצורה תקינה במסמכים והיא אכן קיימת, ותהליך העבודה להשגת התוצאה מתבצע בצורה תקינה עליו רק לאמת את הנושא, הבודק יבצע וריפיקציה- יאמת את הדרישה/ תהליך/ סטנדרטים וכדומה.  
אימות מתבצע בדרך של בדיקות סטטיות ולא נדרשת הפעלת או הרצת המערכת.  
במקרה מסוג זה, התהליך הוא תהליך פנימי של הארגון ועל הבודק החובה לאמת כי הארגון עומד בסטנדרים שנקבעו, תהליכי העבודה תקינים וכדומה.

דרך פשוטה להבין את ההבדל בין שני המושגים הינה ההבנה כי ולידציה בוחנת ש"בנית את המערכת הנכונה" ואילו וריפיקציה בוחנת ש"בנית את המערכת כנדרש". ולידציה אכן מאשרת שהמערכת נבנתה לשביעות רצונו של הלקוח/ המשתמש.

15 . בחר בשיטה המתאימה ביותר לניהול שינויים והסבר כיצד תומכת השיטה בכך.

Incremental development reflects the way that we solve problems.

We rarely work out a complete problem solution in advance but move toward

a solution in a series of steps, backtracking when we realize that we have made a

mistake.  
The cost of accommodating changing customer requirements is reduced. The

amount of analysis and documentation that has to be redone is much less than is

required with the waterfall model

.16 מהו המודל המתאים ביותר לפיתוח מערכות "זמן אמת"?

**מערכת זמן אמת** היא מערכת [מחשב](https://he.wikipedia.org/wiki/%D7%9E%D7%97%D7%A9%D7%91) שיש לה, מעבר לדרישות הפונקציונליות הרגילות, דרישה של עמידה ב[זמנים](https://he.wikipedia.org/wiki/%D7%96%D7%9E%D7%9F). מערכת זמן אמת לא חייבת להגיב מהר, אך היא חייבת להגיב תוך הזמן שהוגדר מראש ובעקביות.  
מפל המים זה המודל המתאים ביותר מפני שהוא מייצר בטיחות וביטחון,ואלה מדגימים ללקוחות או רגולטורים שהמערכת למעשה עומדת בדרישות הבטיחות אשר הוגדרו, בנוסף, התיעוד המדויק עוזר עם שקיפות המערכת כך שהתהליכים ברורים וכשל בתקשורת ובתפעול של המערכת מצומצם יותר