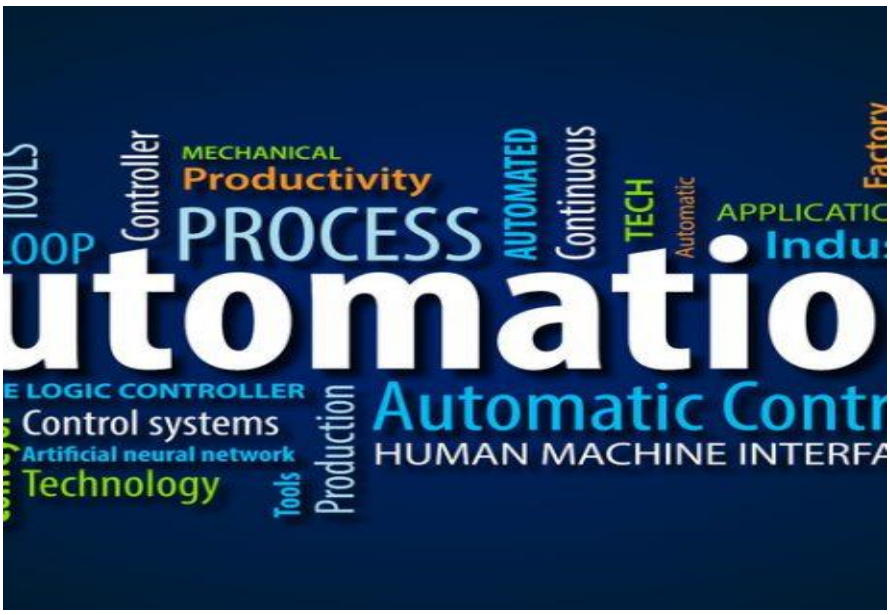
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

ЗЕМЛЕВПОРЯДНИЙ ФАКУЛЬТЕТ КАФЕДРА ІНОЗЕМНИХ МОВ

Н. ГОРОДЕЦЬКА, Н. ГАВРИШКІВ

Навчальний посібник з англійської мови для студентів напряму підготовки «Автоматизація та комп'ютерно-інтегровані технології»



Львів-2020

УДК 004.4(075.8) =111

Рекомендовано до друку
вченою радою Львівського НАУ
протокол №10 від 20.05.2020 р.

Автори: к.п.н., доцент кафедри іноземних мов ЛНАУ **Н.Г. Городецька**
ст. викладач кафедри іноземних мов ЛНАУ **Н.Б. Гавришків**

Рецензенти:

Семко Н.М. – кандидат філологічних наук, доцент кафедри менеджменту та соціально-гуманітарних дисциплін Львівського інституту ДВНЗ «Університет банківської справи».

Дерпак О.В. - кандидат філологічних наук, доцент кафедри гуманітарної освіти Львівського національного аграрного університету.

Городецька Н.Г., Гавришків Н.Б. Навчальний посібник з англійської мови для студентів напряму підготовки «Автоматизація та комп'ютерно-інтегровані технології». Львів: ЛНАУ, 2020.165с.

Навчальний посібник з англійської мови для студентів напрямку підготовки «Автоматизація та комп'ютерно-інтегровані технології» містить тексти фахового змісту. Завдання спрямовані на засвоєння лексичного матеріалу за фахом, розуміння інформації з англомовних джерел, а також розвиток мовленнєвих навичок.

ЗМІСТ

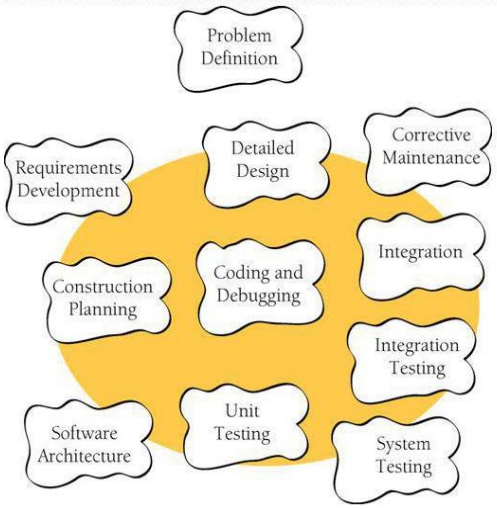
ПЕРЕДМОВА

Запропонований навчальний посібник призначений для студентів напрямку підготовки «Автоматизація та комп'ютерно-інтегровані технології». Мета цього посібника – формування у студентів навичок та вмінь з різних видів читання оригінальної фахової літератури, опанування фаховою лексикою, вміння вести бесіду за фахом, розвиток навичок анотування.

Посібник складається із шести розділів, кожний з яких містить основний текст для детального опрацювання, а також додаткові тексти для самостійної роботи студентів та перевірки вмінь і навичок роботи з текстами. До текстів додається ряд вправ на ознайомлення та семантизацію лексичних одиниць; використання лексичних одиниць у словосполученнях/реченнях; заміну/вставку лексичних одиниць; укладання синонімічних/антонімічних рядів; співвіднесення слова з дефініцією; переклад; складання питань до тексту; відповіді на запитання різних типів; доповнення речення; вправи на розпізнавання і диференціацію нових граматичних структур; на зміну граматичної форми.

Окрім того, навчальний посібник містить словник-мінімум термінів за фахом.

UNIT 1. SOFTWARE CONSTRUCTION



Translate and study the basic vocabulary.

coding	
configuration item	
content	
control structure	
debugging	
formatting	
hardware	
integration testing	
low(high)-level design	
named constant	
routine	
software	
software configuration	
software construction	
software design	
software engineering	
software life cycle	
software project	
software system	
source file	
statement	

test case	
unit testing	
variable	
verification	
verify	

Exercise 1. Choose verbs among the following words. Read and translate the word.
Vary, particular, involve, routine, enter, boundary, refer subsystem, facilitate, separately, tell, instruct, success, operate, meaningful, navigate, integrate, few, create, significant, access, specific.

Exercise 2. Give synonyms (a) and antonyms (b) for the following words:
a) refer (to), detailed, combination, meaningful, link, involve, vary, component, need, integrate, specific, verify, proceed, create, select, polish, carefully, tune, manage, strongly, typical;

b) meaningful, integration, strongly, output, carefully, separately,
fast, few, closely.

Exercise 3. Write derivatives of the words below and explain their meanings.
Model: construct – construction – constructor – constructive – constructively
Construct, mean, combine, verify, integrate, act, strong, system, separate,
success, add, select, care, close, detail, create, engine.

Exercise 4. Give Ukrainian equivalents for the following word combinations.
The term software construction refers to; the detailed creation of working, meaningful software; a combination of coding, verification, unit testing, integration testing, and debugging; software engineering; detailed boundaries between design, construction, and testing; to depend upon the software life cycle processes; the integration of separately constructed routines; to lay the groundwork; to proceed successfully; to create the named constants; to select control structures; to organize blocks of statements; to integrate software components; to make the code faster and use fewer resources; to produce such configuration items as source files, content and test cases; software configuration management.

Text 1. Software Construction Activities

The term software construction refers to the detailed creation of working, meaningful software through a combination of coding, verification, unit testing, integration testing, and debugging.

The software construction is linked to all the other software engineering activities, most strongly to software design and software testing. This is because the software construction process itself involves significant software design and test activity. It also uses the output of design and provides one of the inputs to testing. Detailed coundaries between design, construction, and testing (if any) will vary depending upon the software life cycle processes that are used in a project. One of the key activities during construction is the integration of separately constructed routines, classes, components, and subsystems. In addition, a particular software system may need to be integrated with other software or hardware systems. The following specific tasks are involved in software construction:

– verifying that the groundwork has been laid so that construction can proceed successfully;
– determining how your code will be tested;
– determining and writing classes and routines;
– creating and naming variables and named constants;
– selecting control structures and organizing blocks of statements;
– unit testing, integration testing, and debugging your own code;
– reviewing other team members' low-level designs and code and having them review yours;
– polishing the code by carefully formatting and commenting it;
– integrating the software components that were created separately;
– tuning the code to make it faster and use fewer resources.

Software construction typically produces configuration items that need to be managed in a software project (source files, content, test cases, and so on). Thus, the software construction is also closely linked to the oftware configuration management.

Exercise 5. Find in the text the English for:

конструювання програмного забезпечення; детальне створення робочого, змістовного програмного забезпечення; поєднання кодування, перевірки, тестування компонентів системи, тестування взаємодії компонентів системи та налагодження; бути пов'язаним з розробленням програмного забезпечення; проектування та тестування програмного забезпечення; чіткі межі; залежати від процесів життєвого циклу програмного забезпечення; поєднання підпрограм, класів, компонентів та підсистем; закласти основу; відбуватися успішно; створення іменованих констант; вибір керівних конструкцій; створення блоків операторів; налагодження коду; перевірка низькорівневих програмних структур та коду; відрегулювати код так, щоб він був швидший та використовував менше ресурсів; керувати такими елементами

конфігурації, як вихідні файли, зміст та набори тестових даних; керування конфігурацією програмного забезпечення.

Exercise 6. Say whether the statements below are true or false. Correct the false ones.

1. The term software construction refers to the detailed creation of working, meaningful software through a combination of coding, verification, unit testing, integration testing, and debugging.
2. The software construction process itself involves significant software design and test activity.
3. The software construction is linked to all the other software engineering activities, most strongly to software design and software modelling.
4. Detailed boundaries between design, construction, and testing (if any) will vary depending upon the hardware life cycle processes that are used in a project.
5. The software construction also uses the output of testing and provides one of the inputs to design.
6. One of the key activities during construction is the integration of separately constructed routines, classes, components, and subsystems.
7. Besides other things, software construction involves testing control structures and routines.
8. In addition, a particular software system may need to be integrated with other software or hardware systems.
9. The code can be polished by carefully formatting and commenting it.
10. Software construction typically produces configuration items that need to be managed in a software project (source files, content, test cases, and so on).
11. The code needs tuning to make it faster and use fewer resources.
12. The software construction is also closely linked to the project management.

Exercise 7. Form all possible word combinations with the words from both columns. Translate them.

1) to refer to	a) successfully
2) to depend on	b) tasks
3) to review	c) the creation of software
4) to be linked to	d) configuration items
5) to create	e) software engineering activities
6) to be integrated with	f) the software life cycle process
7) to test	g) low-level designs
8) to involve	h) software or hardware systems
9) to produce	i) named constants
10) to proceed	j) the code

Exercise 8. Fill in the blanks with prepositions *in,(up)on, with, of, to, through, between* where necessary.

1. The term software construction refers … the detailed creation … working, meaningful software … a combination … coding, verification, unit testing, integration testing, and debugging.
2. The software construction is linked … all the other software engineering activities, most strongly … software design and software testing.
3. It also uses the output … design and provides one … the inputs … testing.
4. Detailed boundaries … design, construction, and testing (if any) will vary depending … the software life cycle processes that are used … a project.
5. … addition, a particular software system may need to be integrated ... other software or hardware systems.
6. Some configuration items that need to be managed … a software project (source files, content, test cases, and so on).

Exercise 9. Fill in the blanks with proper terms *(software engineering, debugging, source file, routine, verification, coding, software, software construction)* to complete the sentences.

1. _____is establishment of the correctness of a theory, fact, activity, etc.
2. _____is the detailed creation of working, meaningful software through a combination of coding, verification, unit testing, integration testing, and debugging.
3. _____ is locating and removing defects in a evice, system, plan, program, etc.
4. _____is a set of instructions, called a program, which tells a computer what to do.
5. _____ is writing texts of programs.
6. _____ is a part of a program performing a specific function.
7._____ is the original form of a program before it is converted into a machine-readable form.
8._____ is a systematic and disciplined approach to developing software which applies both computer science and engineering principles to the creation, operation and maintenance of the computer systems.

Exercise 10. Answer the questions on the text.

1. What does the term software construction refer to?
2. What software engineering activities is the software construction linked to?
3. What do the detailed boundaries between design, construction, and testing depend on?

4. What is one of the key activities during construction?
5. What may a particular software system need to be integrated with?
6. What specific tasks are involved in construction?
7. How can the code be polished?
8. What is the purpose of code tuning?
9. What does software construction typically produce?
10. What is software construction also closely linked to?

Exercise 11. Put all possible questions to the sentences below.

1. The term software construction refers to the detailed creation of working, meaningful software.
2. The software construction is linked to all the other software engineering activities.
3. The software construction process itself involves significant software design and test activity.
4. Detailed boundaries between design, construction, and testing depend upon the software life cycle processes.
5. Software construction typically produces configuration items that need to be managed in a software
project.
6. The software construction is also closely linked to the software configuration management.

Exercise 12. Translate into English.

1. Термін «конструювання програмного забезпечення» означає детальне створення робочого, змістовного програмного забезпечення шляхом поєднання кодування, перевірки, тестування компонентів системи, тестування взаємодії компонентів системи та налагодження її роботи.
2. Конструювання програмного забезпечення пов’язане з усіма іншими видами роботи з розроблення програмного забезпечення, найбільше з його проектуванням та тестуванням.
3. Сам процес конструювання програмного забезпечення передбачає багато роботи з проектування та тестування.
4. Також у процесі конструювання програмного забезпечення використовують вихідні дані проектування, які забезпечують один з вхідних параметрів тестування.
5. Чіткі межі між розробленням, конструюванням та тестуванням залежать від процесів життєвого циклу програмного забезпечення, що використовуються в проекті.
6. Одним з ключових процесів конструювання є поєднання окремо створених підпрограм, класів, компонентів та підсистем.

7. Крім цього, окремі системи програмного забезпечення можуть потребувати поєднання з іншими програмними чи апаратними системами.
8. Спочатку необхідно перевірити, чи закладені основи для успішного конструювання.
9. Потім слід визначити спосіб тестування коду, написати класи та підпрограми, створити та назвати змінні та іменовані константи.
10. Після цього потрібно вибрати керівні конструкції та створити блоки операторів.
11. Всі члени команди повинні перевірити низькорівневі програмні структури та код один одного.
12. Код має бути налагоджений, «відшліфований» та відрегульований для його пришвидшення та використання меншої кількості ресурсів.
13. Під час конструювання програмного забезпечення зазвичай створюють елементи конфігурації, що потребують керування в програмному проекті.
14. Такими елементами конфігурації є вихідні файли, зміст та набори тестових даних.
15. Отже, конструювання програмного забезпечення також тісно пов'язане з керуванням конфігурацією програмних засобів.

Exercise 13. Write a summary of the text "Software Construction" using the phrases:
The text deals with the problem of …
…is devoted to …
…describes …
…focuses on …
…gives detailed information on …
…informs the readers of …
The key note of the text is …

Text 2. Software Development

Developing computer software can be a complicated process, and in the last 25 years researchers have identified numerous distinct activities that go into software development. They include:
- problem definition;
- requirement development;
- construction planning;
- software architecture or high-level design;
- detailed design;
- coding and debugging;
- unit testing;
- integration testing;

- integration;
- system testing;
- corrective maintenance.

These activities may be grouped together as “programming” or “creating a software product”. If you create software on informal projects you deal with the activity the researchers refer to as “construction”. Construction focuses on coding and debugging but also includes detailed design, unit testing, integration testing, and other activities. Construction s also sometimes referred to as “coding” or “programming”.

But “coding” isn’t really the best word because it implies the mechanical translation of preexisting design into a computer language. Construction is not at all mechanical and involves substantial creativity and

judgement. But what activities are not part of construction? Important nonconstructional activities include management, requirements development, software architecture, user-interface design, system testing, and

maintenance. Each of these activities affects the ultimate success of a project as much as construction – at least the success of any project that calls for more than one or two people and lasts longer than a few weeks.

Exercise 14. Find in text 2 the English for:

складний процес; визначати багато окремих процесів; випрацювання вимог; визначення завдання; високорівневе проектування; кодування та налагодження; тестування компонентів системи та тестування взаємодії компонентів системи; коригувальний супровід; мати справу з роботою, яку дослідники називають конструюванням; потребувати значної креативності та розсудливості; проектування інтерфейсу користувача; кінцевий успіх проекту; передбачати участь більш як двох людей, продовжуватися більш як кілька тижнів.

Exercise 15. Answer the questions on text 2.

1. What activities does software development include?
2. What may these activities be named as?
3. What does software construction focus on?
4. What is construction sometimes referred to as?
5. Is construction just a mechanical process?
6. What activities are not a part of construction?
7. How do these activities affect the ultimate success of a project?

Exercise 16. Use the proper tense form of the verbs in brackets. (Present, Past or Future Indefinite).

1. Software (be) just instructions which (tell) the computer what to do.
2. The IBM 360 (be) the first commercially successful computer family.
3. The Internet (be) the biggest network in the world.
4. Computer professionals (decide) which hardware, software, and networks endure.
5. The PC (start) a revolution which affects nearly everything we do today.
6. Some common operating systems still in use (be) Windows 2000, Windows XP and Windows Vista.
7. Most software (change) over time and the anticipation of change (drive) many aspects of software construction.
8. Our former network administrator (relate) with routing technology such as Cisco.
9. Distance learning and videoconferencing (be) concepts made possible with the use of an electronic classroom.
10. As computers (evolve) throughout the late 20th century, they (become) more interactive.
11. The objective functions (depend) on the perspective of the model's user.
12. The international company (store) their customer information in a central database in Brussels.
13. Today, computers in security systems (result) in safer environments.
14. Our homes and even objects on the street probably (interact) with our smartphones seamlessly.
15. Computer languages that (require) an interpreter often (run) slower than languages that (require) a compiler.
16. The system administrator (need) to upgrade the machine hardware.
17. The iPhone (have) all the features of a PDA, mobile phone, and an MP3 player in one package.
18. The evolution of nanocomputer technology (enable) us to build microscopic machines.
19. Operating system software (run) on laptop computers, cell phones and other so-called embedded devices.
20. During next ten years the size and shape of the computer (become) a major design issue.
21. Steve Jobs (be) famous for making high quality computers.
22. Computers can (help) people work more creatively.
23. The continued success of mainframes likely (depend) on the functionality.
24. Programs written in Java can (run) on many different computer architectures and operating systems.
25. The popularity of computer networks sharply (increase) with the creation of the World Wide Web (WWW) in the 1990s.

26. The Internet's technical changes (have) an increasing effect on our social and political structures.
27. The early versions of Microsoft Windows (not provide) any computer networking support.
28. With small computing devices people (be) able to spend more time doing what they often do best.
29. The specialist (use) Bluetooth technology to create a personal area network (PAN).
30. Without innovations in the areas of microprocessor and software reliability future systems (face) continuous failure.

Exercise 17. Choose the right form of the verbs in brackets and translate the sentences. Mind the sequence of tenses.

1. The IT support technician asked the end user how often he (update/is updating/updated) his device drivers.
2. The programmers know that PC servers (cost/will cost/costs) in the range of a few thousand dollars.
3. Analysts predict that our homes, cars and even objects on the street (interact/interacts/will interact) with our smart phones and with each other.
4. Teachers say that educational interaction (includes/was in15 cluded/will include) many factors, among them are students, curriculum, parents, teachers, administrators and more.
5. The woman said she always (is searching/searches/searched) for freeware versions of an application before buying one.
6. I'm sure the company (releases, will release, released) a new software next year.
7. The IT manager said the new information about architect's project (was/were/will be) unacceptable.
8. We know that online applications and services (will transform/are transforming/transform) the consumer technology market.
9. The student asked his professor if the course books (was/were/are being) available as hypertext.
10. I suppose that the specialist (is knowing, knew, knows) several foreign languages.

Exercise 18. Change the sentences into indirect speech.

1. The lecturer told his students, "Microsoft Windows has a significant majority of market share in the notebook computer markets."
2. The professor warned our group, "Don't forget to review the material of the previous lecture on software engineering".
3. "You have computer problems that involve your operating system or an application", the IT specialist said to his client.

4. “Did you study any programming computer codes yesterday?” he asked his groupmates.
5. “Our analysts didn’t establish computer security programs to prevent attacks”, she complained.
6. My friend told me, “Tablets will take over from smartphones as the technology of choice for shopping”.
7. “Last week we developed specialized databases for company needs,” replied the programmer.
8. “What Internet provider will you choose?” asked him his colleague.
9. “Our technicians will help users deal with hardware and software problems”, explained the support engineer.
10. I asked the networking specialist, “What kind of networks do you maintain?”

Exercise 19. Use Present or Future Simple of the verbs in brackets. Mind that in subordinate clauses of time and condition, Present Simple is used instead of Future Simple. Such clauses begin with conjunctions: if, when, while, since, before, after, unless.
1. After I (finish) school, I (enter) the University.
2. Since nobody (like) to wait for a computer, high-quality computers (have) fast processors and lots of quick memory.
3. Before she (get) to the theatre, she (go) past the computer centre.
4. If you (not have) previous experience, good contacts, or a good degree from a well-known university, you (be) more successful in getting a lower-level job.
5. If students (keep) learning something new every day, they eventually (be) competent enough to get a high-skill job.
6. Hundreds of books (come) in the future as technologies (mature) and (evolve).
7. If you (work) on networks for a living and you (be) a network engineer, you probably (take) certification exams by networking companies such as Cisco.
8. Unless nanotechnology or some other technology actually (become) operational, this trend (end) according
to some predictions around 2022.
9. While both mainframe and other platforms (evolve), there (be) some cost advantages to retaining the old technology.
10. When software (have) a bug the program (crash) and (terminate) with a confusing message.

Text 3. Software Construction Fundamentals

The fundamentals of software construction include:

- Minimizing complexity
- Anticipating change
- Constructing for verification
- Standards in construction

The first three concepts apply to design as well as to construction. We will define these concepts and describe how they apply to construction.

Minimizing Complexity

A major factor in how people convey intent to computers is the severely limited ability of people to hold complex structures and information in their working memories, especially over long periods of time. This leads to one of the strongest drivers in software construction: minimizing complexity. The need to reduce complexity applies to essentially every aspect of software construction, and is particularly critical to the process of verification and testing of software construction. In software construction, reduced complexity is achieved through emphasizing the creation of the code that is simple and readable ratherthan clever.

Anticipating Change

Most software will change over time, and the anticipation of change drives many aspects of software construction. Software is unavoidably part of changing external environments, and changes in those outside

environments affect software in diverse ways. Anticipating change is supported by many specific techniques:

- Communication methods (for example, standards for document formats and contents)
- Programming languages (for example, language standards for languages like Java and C++)
- Platforms (for example, programmer interface standards for operating system calls)
- Tools (for example, diagrammatic standards for notations like UML (Unified Modelling Language))

Constructing for Verification

Constructing for verification means building software in such a way that faults can be ferreted out readily by the software engineers writing the software, as well as during independent testing and operational activities.

Specific techniques that support constructing for verification include coding standards to support code reviews, unit testing, organizing code to support automated testing, and restricted use of complex or

hard-to-understand language structures.

Standards in Construction

Standards that directly affect construction issues include the use of external standards. Construction depends on the use of external standards for construction languages, construction tools, technical interfaces, and interaction between software construction and other software engineering. Standards come from numerous sources, including hardware and software interface specifications such as the Object Management Group (OMG) and international organizations such as the ISO.

The Use of Internal Standards

Standards may also be created on an organizational basis at the corporate level or for use on specific projects. These standards support coordination of group activities, minimizing complexity, anticipating change, and constructing for verification.

Exercise 20. Answer the questions on the text 3.

1. What do the fundamentals of software construction include?
2. What does the need to reduce complexity apply to?
3. What is reduced complexity achieved through?
4. What specific techniques is anticipating change supported by?
5. What does constructing for verification mean?
6. What do specific techniques that support constructing for verification include?
7. What does construction depend on?
8. What do standards in construction come from?
9. May standards also be created on an organizational basis?

Exercise 21. Make up questions to the italicized parts of the sentences.

1. The first three concepts apply to **design as well as to construction**.
2. This leads to **one of the strongest drivers in software construction**.
3. **Reduced complexity** is achieved through **emphasizing the creation of code**.
4. **Most software** will **change** over time.
5. Anticipating change is supported by **many specific** techniques.
6. Standards **that directly affect construction issues** include **the use of external standards.**
7. **Construction** depends on **the use of external standards**.
8. Standards come from **numerous** sources.
9. **These standards** support coordination of group activities, minimizing complexity, anticipating change, and constructing for verification.

Exercise 22. Give nouns corresponding to the following verbs. Translate them.

Construct, verify, apply, define, reduce, achieve, create, anticipate, communicate, program, operate, automate, restrict, direct, manage, interact, organize, specify, coordinate, act.

Exercise 23. Translate the word combinations below into Ukrainian.

Waterfall and staged-delivery life cycle models; to treat construction as an activity; significant prerequisite work has been completed; extensive design work; to emphasize the activities that precede construction (requirements and design); to create more distinct separations between the activities; to be more iterative, such as evolutionary prototyping and extreme programming; to occur concurrently with other software development activities; the approaches, which tend to mix design, coding,

and testing activities; the choice of construction method; to affect the extent to which construction prerequisites are performed; to affect the project's ability to reduce complexity, anticipate change, and construct for verification; to be addressed at the process, requirements, and design levels; to be influenced by the choice of construction method; to define the order in which components are created and integrated; the software quality management processes; the allocation of task assignments to specific software engineers.

Text 4. Construction Models and Construction Planning

Numerous models have been created to develop software, some of which emphasize construction more than others. Some models are more linear from the construction point of view, such as the waterfall and staged-delivery life cycle models. These models treat construction as an activity which occurs only after significant prerequisite work has been completed – including detailed requirements work, extensive design work, and detailed planning. The more linear approaches tend to emphasize the activities that precede construction (requirements and design), and tend to create more distinct separations between the activities. In these models, the main emphasis of construction may be coding. Other models are more iterative, such as evolutionary prototyping and extreme programming. These approaches tend to treat construction as an activity that occurs concurrently with other software development activities, including requirements, design, and planning, or overlaps them. These approaches tend to mix design, coding, and testing activities, and they often treat the combination of activities as construction. Consequently, what is considered to be "construction" depends to some degree on the life cycle model used.

The choice of construction method is a key aspect of the construction planning activity. The choice of construction method affects the extent to which construction prerequisites are performed, the order in which they are performed, and the degree to which they are expected to be completed before construction work begins. The approach to construction affects the project's ability to reduce complexity, anticipate change, and construct for verification. Each

of these objectives may also be addressed at the process, requirements, and design levels – but they will also be influenced by the choice of construction method. Construction planning also defines the order in which components are created and integrated, the software quality management processes, the allocation of task assignments to specific software engineers, and the
other tasks, according to the chosen method.

Exercise 24. Find in text 4 the English for:

численні моделі; розробляти програмне забезпечення; надавати особливого значення конструюванню; водоспадні та каскадні моделі життєвого циклу; розглядати конструювання як процес; важлива попередня робота; лінійний підхід; операції, що передують конструюванню; чітке розмежування; еволюційне моделювання; екстремальне програмування; поєднувати вимоги, проектування та планування; впливати на рівень виконання передумов конструювання; підхід до конструювання; вибір методу конструювання; порядок створення та поєднання компонентів; процес керування якістю програмного забезпечення; розподіл завдань.

Exercise 25. Make five key questions to the text "Construction Models and Construction Planning".

Text 5. Construction Measurement and Design

Numerous construction activities and artifacts can be measured, including code development, code modification, code reuse, code destruction, code complexity, code inspection statistics, fault-fix and fault-find rates, effort, and scheduling. These measurements can be useful for purposes of managing construction, ensuring quality during construction, improving the construction process, as well as for other reasons.

Construction is an activity in which the software has to come to terms with arbitrary and chaotic real-world constraints. Due to its proximity to these constraints, construction is more driven by practical considerations and software engineering is perhaps most craft-like in the construction area. Some projects allocate more design activity to construction; others to a phase explicitly focused on design. Regardless of the exact allocation, some detailed design work will occur at the construction level, and that design work tends to be dictated by immovable constraints imposed by the real-world problem that is being addressed by the software. Just as construction workers building a physical structure must make smallscale modifications to account for unanticipated gaps in the builder's plans, software construction workers must make modifications on a smaller or larger scale to flesh out details of the software design during construction.

Exercise 26. Find in text 5 the English for:

оцінювати численні процеси та артефакти конструювання; включати розроблення, модифікацію, повторне використання, руйнування та складність коду; включати статистику перегляду коду, коефіцієнт визначення та знаходження помилки, обсяг робіт та планування; з метою керування конструюванням, забезпечення якості в процесі конструювання та вдосконалення конструювання; пристосовуватися до довільних та хаотичних обмежень реального світу; через подібність до обмежень реального світу; розроблення програмного забезпечення; бути найбільш майстерним у конструюванні; не звертаючи уваги на чіткий розподіл роботи; відбуватися на рівні конструювання; зумовлюватися незмінними обмеженнями завдань реального світу; вносити незначні зміни; конкретизувати деталі розроблення програмного забезпечення.

Exercise 27. Write derivatives of the verbs below and explain their meanings.

Measure, develop, use, modify, destruct, act, improve, consider, move, anticipate, allocate, require, restrict, specify, design.

Exercise 28. Find in the text words that can function both as nouns and verbs. Translate them.

Model: work – 1) праця 2) працювати

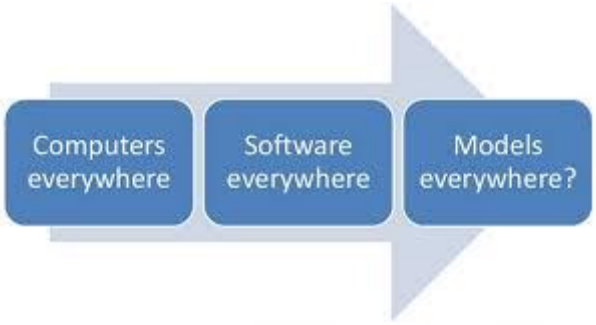
Exercise 29. Answer the questions on text 5.

1. What construction activities and artifacts can be measured?
2. What can these measurements be useful for?
3. What constraints does the software have to come to terms with?
4. What is construction driven by?
5. Why must software construction workers make modifications on a smaller or larger scale design?

Exercise 30. Find some additional information and speak on:

1. Software design.
2. Software testing.
3. Coding and debugging.

UNIT 2. BASICS OF SOFTWARE MODELLING



Translate and study the basic vocabulary.

behavioural perspective	
data architecture	
dependent quantity	
diagram	
external perspective	
fix a bug	
legacy system	
object-oriented design	
process model	
software model	
software modelling	
structural perspective	
Unified Modelling Language (UML)	

Exercise 1. Choose nouns among the following words. Read and translate the word.

Permeate, however, modification, responsible, fortunately, object, inherently, conjure up, diagram, dependent, environment, unlike, probably, language, unwieldy, legacy, identify, inherently, interface, briefly, network, evaluate, whenever, generation.

Exercise 2. Give synonyms (a) and antonyms (b) for the following words:

a) diagram, apply to, bug, architecture, create, permeate, implement, allow, support, feature, cause, perspective, environment, activity, wallsized, unwieldy, identical, unique, uniform, quickly, fortunately, briefly, probably, apply;

b) dependent, destruction, out of date, internal, unwieldy, unique, responsible, quickly, fortunately, unlike.

Exercise 3. Write derivatives of the words below and explain their meanings.

Model: create – creation – creator – creativity – creature – creative

Create, internal, fortune, simple, dependence, identify, violate, probable, brief, responsible, change, behaviour, differ, develop, architect, structure, apply, quick, cause, active.

Exercise 4. Give Ukrainian equivalents for the following word combinations.

To conjure up images of wall-sized UML diagrams; diagrams are usually out of date by the time they are printed; large-scale and often unwieldy methods; to be not the only kind available for modelling software; to start from scratch; to permeate the entire system; to apply to all software models; ownership, dependency, interface and identity; to apply to legacy systems as well as to new projects; to think back to the last serious bug that you fixed; to identify a violation of one of these rules as the cause; responsible for creation and destruction; to be not transferable; to be up to date whenever referenced; to implement some set of interfaces; to inherently identify an object; to access an identical object; to expect uniform behaviour; communication among stakeholders; to present the system from different perspectives; to show the system's context or environment; to show the system development process as well as activities supported by the system.

Text 1. Basics of Software Modelling

Modelling is simply the practice of creating a small system that has some of the same features of a larger system. When applied to software, the word modelling usually conjures up images of wall-sized UML diagrams. Internal software, however, changes so quickly that such diagrams are usually out of date by the time they are printed. Fortunately, such large-scale and unwieldy methods are not the only kind available for modelling software. A software model does not have to start from scratch. It does not have to permeate the entire system. The only thing that a software

model needs is a set of rules. The same four rules apply to all software models.

The four rules of software modelling are:

1) ownership

2) dependency

3) interface

4) identity

Unlike the rules of object-oriented design, these four apply to legacy systems as well as to new projects. Think back to the last serious bug that you fixed. You can probably identify a

violation of these rules as the cause. Briefly, ownership means that every object has exactly one owner responsible for its creation and destruction, and that ownership is not transferable. Dependency means that any bit of data used to evaluate a dependent quantity is a precedent, and that dependents must be up to date whenever referenced. Interface means that every object implements some set of interfaces, and that these interfaces allow objects to be interchanged. Identity means that all objects have unique identity, that aninterface inherently identifies an object, and that all clients accessing an identical object can expect uniform behaviour. Software modelling helps the engineer to understand the functionality of the system. Models are used for communication among stakeholders. Different models present the system from different perspectives:

- external perspective showing the system's context or environment;
- process models showing the system development process as well

as activities supported by the system;

- behavioural perspective showing the behaviour of the system;
- structural perspective showing the system or data architecture.

Exercise 5. Find in text 1 the English for:

щодо програмного забезпечення; викликати в уяві зображення величезних UML діаграм; застарілий; великомасштабні та громіздкі методи; розпочинатися з нуля; проходити через всю систему; набір правил; право власності, залежність, взаємозв'язок та ідентичність; на відміну від правил об'єктно-орієнтованого проектування; застосовуватися до успадкованих систем так само, як і до нових проектів; виправляти серйозну помилку; порушення правил; створення та знищення; право власності не передається; визначати залежну величину; бути найновішими щоразу, коли до них звертаються; забезпечувати певну сукупність інтерфейсів; дозволяти переставляти об'єкти; мати доступ до ідентичного об'єкту; очікувати однакової поведінки; обмін інформацією між учасниками; презентувати систему з різних ракурсів; поведінка системи.

Exercise 6. Say whether the statements below are true or false. Correct the false ones.

1. When applied to software, the word modelling usually conjures up images of matchbox-sized circuit diagrams.

2. Modelling is simply the practice of creating a large system that has some of the same features of a smaller system.

3. Unfortunately, large-scale and unwieldy methods are the only kind available for modelling software.

4. Internal software does not change quickly and diagrams are usually up to date by the time they are printed.

5. A software model has to permeate the entire system.

6. A software model does not have to start from scratch.

7. The four rules of software modelling are: significance, availability, simplicity and minimization.

8. Unlike the rules of object-oriented design, these four do not apply to legacy systems as they do to new projects.

9. Dependency means that any bit of data used to evaluate a dependent quantity is a precedent, and that dependents must be up to date whenever referenced.

10. Briefly, ownership means that every object has exactly one owner responsible for its creation and destruction.

11. Interface means that every object implements some set of interfaces, and that these interfaces do not allow objects to be complementary to one another.

12. Software modelling helps the engineer to understand the functionality of the system.

13. Identity means that all objects have unique identity, that an interface inherently identifies an object, and that all clients accessing an identical object can expect uniform behaviour.

14. Different models present the system from different perspectives: external perspective, system development perspective, behavioural perspective and structural perspective.

Exercise 7. Form all possible word combinations with the words from both columns. Translate them.

1. to change	a) uniform behaviour
2. to be	b) the system from different perspectives
3. to start from	c) the last serious bug
4. to apply to	d) communication among stakeholders
5. to think back to	e) legacy systems
6. to evaluate	f) some set of interfaces
7. to implement	g) scratch
8. to expect	h) a dependent quantity
9. to be used for	i) quickly
10. to present	j) out of date

Exercise 8. Fill in the blanks with prepositions *to, of, by, from, unlike, with, among, up, for* where necessary.

1. Modelling is simply the practice … creating a small system that has some … the same features … a larger system.

2. When applied …software, the word modelling usually conjures … images … wall-sized UML diagrams.

3. Internal software, however, changes so quickly that such diagrams are usually … … date … the time they are printed.

4. A software model does not have to start … scratch.

5. … the rules … object-oriented design, these four apply … legacy systems as well as … new projects.

6. Briefly, ownership means that every object has exactly one owner responsible … its creation and destruction.

7. Software modelling helps the engineer to understand the functionality … the system.

8. Models are used … communication … stakeholders.

9. A software model has to permeate … the entire system.

10. Different models present the system … different perspectives.

Exercise 9. Fill in the blanks with proper terms *(structural perspective, behavioural perspective, process models, external perspective, modelling, identity, interface, dependency, ownership)* to complete the sentences.

1._____ means that every object implements some set of interfaces, and that these interfaces allow objects to be interchanged.

2. _____is simply the practice of creating a small system that has some of the same features of a larger system.

3._____ shows the system's context or environment.

4._____ means that any bit of data used to evaluate a dependent quantity is a precedent, and that dependents must be up to date whenever referenced. 5._____show the system development process as well as activities supported by the system.

6._____ means that every object has exactly one owner responsible for its creation and destruction, and that ownership is not transferable.

7. _____shows the system or data architecture.

8._____shows the behaviour of the system.

9._____means that all objects have unique identity, that an interface inherently identifies an object, and that all clients accessing an identical object can expect uniform behaviour.

Exercise 10. Answer the questions on text 1.

1. What is modelling?
2. What does the word modelling usually conjure up when applied to software?
3. Why are such diagrams usually out of date by the time they are printed? 4. Does a software model have to start from scratch or permeate the entire system?
5. What four rules apply to all software models?
6. What projects do these four rules apply to?
7. What can violation of these rules lead to?
8. What does ownership mean?
9. What does dependency mean?
10. What does interface mean?
11. What does identity mean?
12. What does software modelling help the engineer in?
13. What perspectives do different models present the system from?
14. What do these perspectives show?

Exercise 11. Put all possible questions to the sentences below.

1. The word modelling usually conjures up images of wall-sized UML diagrams.
2. Internal software changes quickly.
3. Such diagrams are usually out of date by the time they are printed.
4. The only thing that a software model needs is a set of rules.
5. The same four rules apply to all software models.

Exercise 12. Translate into English.

1. Моделювання – це метод створення меншої системи, що має певні характеристики більшої.
2. Щодо програмного забезпечення слово «моделювання» зазвичай викликає в уяві зображення UML діаграм розміром зі стіну.
3. Проте вміст програмного забезпечення змінюється так швидко, що діаграми зазвичай стають застарілими раніш, як їх роздрукують.

4. На щастя, для моделювання програмного забезпечення існують не лише такі великомасштабні та громіздкі методи.

5. Модель програмного забезпечення необов'язково має розпочинатися з нуля й проходити через усю систему.

6. Єдине, чого потребує модель програмного забезпечення – це набір правил.

7. Чотири правила моделювання програмного забезпечення – це право власності, залежність, взаємозв'язок та ідентичність.

8. На відміну від правил об'єктно-орієнтованого проектування, ці чотири правила застосовують до успадкованих систем так само, як і до нових проектів. 9. Згадайте останню серйозну помилку, яку ви виправили.

10. Причиною цього, ймовірно, можна вважати порушення цих правил.

11. Право власності означає, що кожний об'єкт має лише одного власника, і право власності не передається.

12. Залежність означає, що будь-яка кількість даних, які використовуються для визначення залежної величини, є попереднім значенням.

13. Взаємозв'язок означає, що кожний об'єкт забезпечує певну сукупність інтерфейсів, які дозволяють переставляти об'єкти.

14. Ідентичність означає, що всі об'єкти мають унікальну ідентичність, і інтерфейс, що по суті, визначає об'єкт.

15. Моделювання програмного забезпечення допомагає інженерові зрозуміти функціональність системи.

16. Різні моделі представляють систему з різних аспектів.

17. Зовнішній аспект показує контекст або оточення системи.

18. Моделі процесу показують процес розроблення системи, а також функції, які підтримує ця система.

19. Поведінковий аспект показує поведінку системи.

20. Структурний аспект показує архітектуру системи або даних.

Exercise 13. Write a summary of the text "Software Modelling".

Text 2. Mathematical Models

A mathematical model uses mathematical language to describe a system. Mathematical models are used not only in the natural sciences and engineering disciplines (such as physics, biology, meteorology, and electrical engineering) but also in the social sciences (such as economics, psychology, sociology and political science); physicists, engineers, computer

scientists, and economists use mathematical models most extensively. Eykhoff (1974) defined a mathematical model as "a representation of the essential aspects of an existing system (or a system to be constructed) which presents knowledge of that system in usable form". Mathematical models can take many forms, including but not limited to dynamic systems, static models, differential equations, or game theoretic models. These and other types of models can overlap, with a given model involving a variety of abstract structures. Often when engineers analyze a system to be controlled or optimized, they use a mathematical model. In analysis, engineers can build a descriptive model of the system as a hypothesis of how the system could work, or try to estimate how an unforeseeable event could affect the system. Similarly, in control of a system, engineers can try out different control approaches in simulations. A mathematical model usually describes a system by a set of variables and a set of equations that establish relationships between the variables. The values of the variables can be practically anything; real or integer numbers, boolean values or strings, for example. The variables represent some properties of the system, for example, measured system outputs often in the form of signals, timing data, counters, and event occurrence (yes/no). The actual model is the set of functions that describe the relations between the different variables. There are six basic groups of variables: decision variables, input variables, state variables, exogenous variables, random variables, and output variables. Since there can be many variables of each type, the variables are generally represented by vectors. Decision variables are sometimes known as independent variables. Exogenous variables are sometimes known as parameters or constants. The variables are not independent of each other as the state variables are dependent on the decision, input, random, and exogenous variables. Furthermore, the output variables are dependent on the state of the system (represented by the state variables). Objectives and constraints of the system and its users can be represented as functions of the output variables or state variables. The objective functions will depend on the perspective of the model's user. Depending on the context, an objective function is also known as an index of performance, as it is some measure of interest to the user. Although there is no limit to the number of objective functions and constraints a model can have, using or optimizing the model becomes more involved (computationally).

Exercise 14. Make up 10 key questions on the text.

Exercise 15. Find in text 2 the English for:

природничі науки та технічні дисципліни; соціологія та політологія; метеорологія та електротехніка; найбільш широко використовувати математичні моделі; представлення важливих аспектів наявної системи; презентувати знання про систему в практичній формі; набувати багатьох форм; включно з динамічними системами, але не

обмежуючись ними, статичними моделями, диференційними рівняннями або моделями теорії ігор; моделі можуть накладатися одна на одну; включати різноманітні абстрактні структури; описова модель системи; оцінювати, як непередбачувана подія може вплинути

на систему; випробовувати різні концепції керування в моделюванні; описувати систему сукупністю змінних та сукупністю рівнянь; встановлювати зв'язок між змінними; дійсні або цілі числа; булеві величини або рядки; часові характеристики, лічильні функції та настання події; змінні рішення, вхідні змінні, змінні стану, екзогенні (зовнішні) змінні, випадкові змінні та вихідні змінні; цілі та обмеження системи; цільова функція, відома як показник ефективності; показник, що цікавить користувача; моделі використання та оптимізування стають дедалі складнішими.

Exercise 16. Translate the word combinations below into Ukrainian.

Linear vs nonlinear; predictor variables, differential equation, objective functions and constraints; in fairly simple systems; to be often associated with phenomena such as chaos and irreversibility; a common approach to nonlinear problems; aspects such as irreversibility, which are strongly tied to nonlinearity; deterministic vs probabilistic (stochastic) model; every set of variable states; to be uniquely determined by parameters; unique value; probability distribution; lumped vs distributed parameters; consistent state throughout the entire system; varying state within the system.

Text 3. Classification of Mathematical Models

Many mathematical models can be classified in the following ways:

Linear vs nonlinear. Mathematical models are usually composed of variables, which are abstractions of quantities of interest in the described systems, and operators that act on these variables, which can be algebraic operators, functions, differential operators, etc. If all the operators in a mathematical model present linearity, the resulting mathematical model is defined as linear. A model is considered to be nonlinear otherwise.

The question of linearity and nonlinearity is dependent on the context, and linear models may have nonlinear expressions in them. For example, in a static linear model, it is assumed that a relationship is linear in the parameters, but it may be nonlinear in the predictor variables. Similarly, a differential equation is said to be linear if it can be written with linear differential operators, but it can still have nonlinear expressions in it. In a mathematical programming model, if the objective functions and constraints are represented entirely by linear equations, then the model is regarded as a linear one. If one or more of the objective functions or constraints are represented with a nonlinear equation, then the model is known as a nonlinear

one. Nonlinearity, even in fairly simple systems, is often associated with phenomena such as chaos and irreversibility. Although there are exceptions, nonlinear systems and models tend to be more difficult to study than linear ones. A common approach to nonlinear problems is linearization, but this can be problematic if one is trying to study aspects such as irreversibility, which are strongly tied to nonlinearity.

Deterministic vs probabilistic (stochastic). A deterministic model is one in which every set of variable states is uniquely determined by parameters in the model and by sets of previous states of these variables. Therefore, deterministic models perform the same way for a given set of initial conditions. Conversely, in a stochastic model, randomness is present, and variable states are not described by unique values, but rather by probability distributions.

Static vs dynamic. A static model does not account for the element of time, while a dynamic model does. Dynamic models typically are represented with differential equations.

Lumped vs distributed parameters. If the model is homogeneous (consistent state throughout the entire system) the parameters are distributed. If the model is heterogeneous (varying state within the system), then the parameters are lumped. Distributed parameters are typically represented with partial differential equations.

Exercise 17. Find in text 3 the English for:

величини, що нас цікавлять; оператори, що діють на ці змінні; бути лінійним; питання лінійності та нелінійності; бути залежним від контексту; припускати, що; прогностичний параметр; диференціюйне рівняння; цільові функції та обмеження; бути представленим виключно лінійними рівняннями; досить прості системи; бути пов’язаним з такими явищами, як хаос та незворотність; загальний підхід; набір (сукупність) змінних; однозначно визначатися параметрами; зосереджені та розподілені параметри; однорідна модель; неоднорідна модель; стабільний стан; мінливий стан.

Exercise 18. Translate into English.

1. Математичні моделі складаються зі змінних та операторів, що впливають на ці змінні.

2. Змінні величини є абстракціями величин, які нас цікавлять у системах, що ми описуємо.

3. Операторами можуть бути алгебраїчні оператори, функції та диференційні оператори.

4. Якщо всі оператори в математичній моделі є лінійними, то кінцеву математичну модель визначають як лінійну.

5. В іншому випадку модель вважають нелінійною.

6. Питання лінійності та нелінійності залежить від контексту, і лінійні моделі можуть містити нелінійні вирази.

7. Нелінійність часто пов’язують з такими явищами, як хаос та незворотність.

8. Загальним підходом до неліній них задач є лінеаризація.

9. Детермінована модель – це така, в якій кожна сукупність станів змінних однозначно визначається параметрами самої моделі та сукупностями попередніх станів цих змінних.

10. Тому детерміновані моделі діють однаково на даний набір початкових умов.

11. В стохастичній моделі, навпаки, присутня випадковість, і стани змінних описуються не єдиними значеннями, а розподілом імовірності.

12. Статична модель не враховує елемент часу, тоді як динамічна його враховує.

13. Динамічні моделі зазвичай презентують диференційними рівняннями.

14. Якщо модель однорідна (стабільний стан у всій системі), то параметри є розподіленими.

15. Якщо модель є неоднорідною (мінливий стан в системі), то параметри є зосередженими.

Exercise 19. Use the proper tense form of the verbs in brackets (Present, Past or Future Continuous).

1. New types of computers constantly (come) out.
2. I (learn) new software packages at five o’clock yesterday.
3. This time next week, the technicians (troubleshoot) computer problems within a company.
4. Creating new pages for the Web (get) easier all the time.
5. Hardware engineer (design) communications devices for corporate use at the time.
6. This time tomorrow my colleague (work) for a famous hardware manufacturer such as Apple.
7. Computers (get) more powerful over previous generations.
8. When PC specialists (install) computers, they (give) some useful recommendations.
9. I wonder what my friend (do) this time next month.
10. Right now you most likely (use) a GUI interface.
11. The engineer (develop) software for the insurance company all morning yesterday.
12. I expect I still (work) with the same team of specialists.
13. Trends in computer storage constantly (change).

14. Touch screens rapidly (replace) keypads on mobile phones.

15. What you (do) next weekend? – I will (write) users manuals and training materials as usual.

16. Your network administrators (maintain) company's Internet connections while we (install) security services in our main office?

17. Computing equipment (get) smaller and more sophisticated.

18. I (demonstrate) multimedia products for our new customers at 9 o'clock tonight. 19. Our database analyst (train) our employees on the systems all day yesterday.

20. Diskettes (get) rare these days and they are replaced by USB flash memory drives.

21. I (take) part in the conference for network professionals tomorrow morning.

22. What network security analyst (do) when hackers attacked and damaged the computer system?

23. I (discuss) a new Web project with programmers when you see me next.

24. As the specialist (design) for University's website he (use) different sets of programming languages.

25. Look at the time. Your client (come) in a minute and you haven't even started testing a new computer equipment.

26. Digital distribution over a network rapidly (replace) removable storage media such as CDROM's and DVD-ROM's.

27. This time next week he (buy) new hardware.

28. When I first met IT professional he (replace) hardware systems.

29. I am afraid our professor (not deliver) lectures on network design this time next term.

30. Where he (create) multimedia products when you saw him last?

Exercise 20. Complete the sentences below using the appropriate tense forms of the verbs in brackets (Present /Past Indefinite or Present/Past Continuous).

1. My computer exam (take place) next week.

2. When I (arrive) the professor (explain) the multi-leveled architecture of a large software system.

3. Dean (conduct) a survey at the moment investigating how students use free educational content on our University website.

4. He (download) e-mail files when malicious computer viruses (attack) and (disable) the antivirus application.

5. I (wonder) what these programmers (talk) about. – They (discuss) common computer problems and their solutions.

6. Network administrator (maintain) a computer network while other specialist (monitor) its security.

7. Software engineers (not take part) in program design today because the project is not approved.

8. I (read) my electronic course-book when suddenly the power (switch off).

9. What the student (do) to his computer system now? – I (think) he (install) an application program.

10. My colleague still (use) Windows XP when Microsoft (release) a new version with advanced features.

11. The programmer (design) new applications for business operations yesterday morning.

12. My friend usually (learn) foreign languages very quickly and now he (learn) German.

13. I just (update) the database at the time when I suddenly (note) unauthorized access.

14. Using different search engines (become) increasingly popular.

15. Why you (lend) him this book? I still (read) it.

Exercise 21. Choose the right form of the verbs in brackets and translate the sentences. Mind the sequence of tenses.

1. The programming editor said, "I (am writing, was writing, will be writing, write) the source code of an application".

2. The student asked the lecturer what questions he (consider, would consider, was considering) in this lecture.

3. The clerk asked if she (typed, types, is typing, was typing) data into a database.

4. He wondered if employees (was using, use, will use, were using) packaged software.

5. Ann said with regret that she (can't, is able to, couldn't, will be able) install security programs.

6. He told us that he (is going, was going, will go) to take an English exam the following year.

7. They informed that participants of the conference (discussed, will discuss, were discussing) the latest scientific discoveries at that moment.

Exercise 22. Change the sentences into indirect speech.

1. The lecturer said, "Current chip technology is approaching the limits of physics."

2. The teacher asked his students, "Are recent advances in computing communications and software transforming the way people live?"

3. "A number of changes are affecting mainframe technology," he said.

4. “What security programs were you testing when I saw you yesterday morning?” the programmer asked his colleagues.

5. “Are data analytics going to prevent new research opportunities to the computer graphics?” I asked the web designer.

6.“The programmer was not using presentation software for his report yesterday morning”, the lecturer said.

7. “Is the Internet of Things moving to the mainstream activity at the moment?” his friend asked him.

8. “The researchers were doing quantum computing experiments when I saw them last”, the scientist claimed.

9. “In-memory computing is rapidly growing because of its power, versatility and incorporation into several software and hardware products,” explained the hardware specialist.

10. The information manager told me, “Our company is planning to address big data in our integration infrastructure.”

Exercise 23. Translate the word combinations below into Ukrainian.

Standardized general-purpose modelling language; to specify, visualize, modify, construct and document the artifacts; object-oriented software intensive system under development; to visualize a system’s architectural blueprints; actors, business processes, (logical) components, activities, programming language statements, database schemas and reusable software components; best techniques of data modelling (entity relationship diagrams); business modelling (workflows); object modelling and component modelling; throughout the software development life cycle; to synthesize the notations of the object modelling technique (OMT) and object-oriented software engineering (OOSE); common and widely usable modelling language; to model concurrent and distributed systems; a de facto industry standard; to evolve under the auspices of the object Management Group (OMG); to be extensible; mechanisms for customization; to be compatible (with); to recast the methods; to take advantage of the new notations; Rational Unified Process (RUP); Abstraction Method; Dynamic Systems Development Method; to achieve different objectives.

Text 4. Unified Modelling Language

Unified Modelling Language (UML) is a standardized generalpurpose modelling language in the field of software engineering. It is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software intensive system under development. It offers a standard way to visualize a system’s architectural blueprints, including elements such

as actors, business processes, (logical) components, activities, programming language statements, database schemas and reusable software components. UML combines best techniques of data modelling (entity relationship diagrams), business modelling (workflows), object modelling, and component modelling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies. UML has synthesized the notations of the Object-modelling technique (OMT) and Object-oriented software engineering (OOSE) by fusing them into a single, common and widely usable modelling language. It aims to be a standard modelling language which can model concurrent and distributed systems. It is a de facto industry standard, and is evolving under the auspices of the Object Management Group (OMG). UML models may be automatically transformed to other representations (e.g. Java) by means of transformation languages, supported by the OMG. UML is extensible, offering such mechanisms for customization as profiles and stereotypes. UML is not a development method by itself, however, it was designed to be compatible with the leading object-oriented software development methods. Since UML has evolved, some of these methods have been recast to take advantage of the new notations, and new methods have been created based on UML. The best known is IBM Rational Unified Process (RUP). There are many other UML-based methods like Abstraction Method, Dynamic Systems Development Method, and others, designed to provide more specific solutions, or achieve different objectives.

Exercise 24. Answer the questions on text 4.

1. What is UML?
2. What is UML used for?
3. In what way does UML visualize a system’s architectural blueprints?
4. What kinds of modelling techniques does UML combine?
5. What notations has UML synthesized?
6. What kinds of systems can UML model?
7. What mechanisms does UML offer for customization?
8. Is UML a development method by itself?
9. What are the best known objectoriented software development methods?

Exercise 25. Decipher the abbreviations below.

UML, OMT, OOSE, OMG, RUP, ISO.

Exercise 26. Translate the word combinations below into Ukrainian.

To distinguish between the UML model and the set of diagrams of a system; a partial graphical representation of a system's model; to contain a "semantic backplane"; written use cases that drive the model elements and diagrams; different views of a system mode; static (or structural) view and dynamic (or behavioural) view; class diagrams and composite structure diagrams; to emphasize the dynamic behaviour of the system; to show collaboration among objects; sequence diagrams, activity diagrams and state machine diagrams; partially restricted flexibility; structure diagram and behaviour diagram; component diagram and object diagram; profile diagram, deployment diagram and package diagram; use case diagram and interaction diagram; communication diagram and timing diagram; interaction overview diagram.

Text 5. UML Modelling

It is very important to distinguish between the UML model and the set of diagrams of a system. A diagram is a partial graphical representation of a system's model. The model also contains a "semantic backplane" – documentation such as written use cases that drive the model elements and diagrams. UML diagrams represent two different views of a system mode. Static (or structural) view emphasizes the static structure of the system using objects, attributes, operations and relationships. The structural view includes class diagrams and composite structure diagrams. Dynamic (or behavioural) view emphasizes the dynamic behaviour of the system by showing collaborations among objects and changes to the internal states of objects. This view includes sequence diagrams, activity diagrams and state machine diagrams.

Exercise 27. Do the assignments below.

1. Explain the difference between a model and a diagram.
2. Compare two different views of a system mode.
3. Name the categories of UML 2.2 diagrams.
4. Name the diagrams which represent each category.

Exercise 28. Translate the word combinations below into Ukrainian.

Relationship among the classes; to be split up into components; composite structure diagram; collaboration; deployment diagram; execution environment; package diagram; step-by-step workflows of components in a system; to show the overall flow of control; state machine diagram; use case diagram; in terms of actors; to be used extensively; a subset of behaviour diagrams; sequenced messages; interaction overview diagram; life-spans of objects; timing diagram.

Text 6. UML Diagrams

Structure diagrams

Structure diagrams emphasize what things must be in the system being modeled:

Class diagram describes the structure of a system by showing the system's classes, their attributes, and the relationships among the classes. Component diagram depicts how a software system is split up into components and shows the dependencies among these components. Composite structure diagram describes the internal structure of a class and the collaborations that this structure makes possible. Deployment diagram serves to model the hardware used in system implementations, and the execution environments and artifacts deployed on the hardware. Object diagram shows a complete or partial view of the structure of a modeled system at a specific time. Package diagram depicts how a system is split up into logical groupings by showing the dependencies among these groupings. Profile diagram operates at the metamodel level to show stereotypes and profiles. The extension relation indicates what metamodel element a given stereotype is extending. Since structure diagrams represent the structure they are used extensively in documenting the architecture of software systems.

Behaviour diagrams

Behaviour diagrams emphasize what must happen in the system being modeled:

Activity diagram represents the business and operational step-bystep workflows of components in a system. An activity diagram shows the overall flow of control.

State machine diagram is a standardized notation to describe many systems, from computer programs to business processes.

Use case diagram shows the functionality provided by a system in terms of actors, their goals represented as use cases, and any dependencies among those use cases. Since behaviour diagrams illustrate the behaviour of a system, they are used extensively to describe the functionality of software systems.

Interaction diagrams

Interaction diagrams, a subset of behaviour diagrams, emphasize the flow of control and data among the things in the system being modeled.

Communication diagram shows the interactions between objects or parts in terms of sequenced messages. They represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behaviour of a system.

Interaction overview diagram is a type of activity diagram in which the nodes represent interaction diagrams.

Sequence diagram shows how objects communicate with each other in terms of a sequence of messages. Also indicates the life-spans of objects relative to those messages.

Timing diagram is a specific type of interaction diagram, where the focus is on timing constraints.

Exercise 29. Find in text 6 the English for:

структурна діаграма; зв'язки між класами; розподілятися на компоненти; діаграма композитних структур; діаграма розгортання; реалізація системи; повна або часткова картина системи; діаграма пакетів; логічна група; профілограма; поетапний потік бізнес операцій та функціональних операцій компонентів у системі; увесь потік керування; діаграма кінцевого автомату; діаграма прецедентів; підмножина діаграм поведінки; акцентувати увагу на потоці керування і даних між об'єктами у системі, що моделюється; діаграма комунікації; у вигляді послідовності повідомлень; діаграма огляду взаємодії; діаграма послідовностей; довговічність; часова синхронізація; часові обмеження.

Exercise 30. Match the names of the diagrams with their functions.

1. Class diagram components and shows the dependencies among these components.
2. Object diagram implementations, and the execution environments and artifacts deployed on the hardware.
3. Component diagram showing the system's classes, their attributes, and the relationships among the classes.
4. Package diagram in terms of actors.
5. Activity diagram structure of a modeled system at a specific time.
6. Composite structure diagram groupings by showing the dependencies among these groupings.
7. Deployment diagram step-by-step workflows of components in a system and the overall flow of control.
8. Use case diagram systems, from computer programs to business processes.
9. State machine diagram and the collaborations that this structure makes possible.

a) depicts how a software system is split up into

b) serves to model the hardware used in system

c) describes the structure of a system by
d) shows the functionality provided by a system
e) shows a complete or partial view of the
f) depicts how a system is split up into logical
g) represents the business and operational
h) is standardized notation to describe many
i) describes the internal structure of a class

Exercise 31. Speak on the topics:

1. Structure diagrams.
2. Behaviour diagrams.
3. Interaction diagrams.

UNIT 3. HARDWARE

Translate and study the basic vocabulary.

alternating current (AC)	
application	
automation	
Basic Input / Output System (BIOS)	
boot	
boot firmware	
central processing unit (CPU)	
chipset	
Compact Disk Read-Only Memory (CD-ROM)	
Digital Versatile [Video] Disk Read-Only Memory (DVD-ROM)	
direct current (DC)	
disk drive	
expansion card (також expansion board, PC card)	
fan	
firmware	
hard disk	
heat sink	

internal bus	
keyboard	
main memory	
monitor	
motherboard	
mouse	
operating system	
power cord	
power management	
power supply	
Random Access Memory (RAM)	
socket	
storage medium	
switch	
terminal device	
voltage	

Exercise 1. Choose nouns among the following words. Read and translate the word.

Combine, simple, mouse, convert, opportunity, modern, via, and, network, require, item, manipulate, terminal, personal, operator, occur, requirement.

Exercise 2. Give synonyms (a) and antonyms (b) for the following words:

a) expansion card, require, machinery, basic, operation, complex, item, easy, particular, combine, user, firmware, modern;

b) different, advantage, outside, necessary, easily, wide, often, usually, input, progress, modern.

Exercise 3. Give derivatives of the words below and explain their meanings.

Model: automate – automation – automatic – automatically

Automate, compute, combine, machine, store, communicate, refer, direct, process, calculate, apply, access, cool, manage, operate, differ, connect, expand, improve, educate, entertain, electron, significant, history, use.

Exercise 4. Give Ukrainian equivalents for the following word combinations.

A functioning computer system; to combine hardware elements with software elements; mechanical devices, machinery and electronics; to perform physical functions; to require three basic hardware items; to perform all data processing; a terminal device, used like a typewriter; two-way communication between the user and the system; a storage medium; to store programs and data; to cover such parts of the personal computer as the monitor, the motherboard, the CPU, the RAM memory,

the expansion card, the power supply, the CD-ROM drive, the hard disk, the keyboard and the mouse; to be the "body" of the computer; to be directly attached to the motherboard; to include the central processing unit (CPU), the chipset, the Random Access Memory (RAM), the Basic Input Output System (BIOS) and internal buses; to enable a computer to function; to be cooled by a heat sink and fan; to mediate communication between the CPU and the other components of the system; main memory; boot firmware and power management; to handle tasks; operating system drivers; expansion cards; to convert alternating current to direct current; to provide appropriate voltages to different components; to undergo significant improvements; computation, automation, communication, control, entertainment and education.

Text 1. Basic Hardware Elements

A functioning computer system combines hardware elements with software elements. The hardware elements are mechanical devices in the system, machinery and electronics that perform physical functions. Usually, a computer system requires three basic hardware items:

1) a personal computer, which performs all data processing;

2) a terminal device, used like a typewriter for two-way communication between the user and the system;

3) a storage medium for storing programs or data.

The term hardware covers such parts of the personal computer asthe monitor, the motherboard, the CPU, the RAM memory, the expansion card, the power supply, the CD-ROM drive, the hard disk, the keyboard, and the mouse. The motherboard is the "body" of the computer. Components are directly attached to the motherboard and include such elements as: the central processing unit (CPU), the chipset, the Random Access Memory (RAM), the Basic Input Output System (BIOS) and internal buses.

The central processing unit (CPU) performs most of the calculations which enable a computer to function, and is sometimes referred to as the "brain" of the computer. It is usually cooled by a heat sink and fan. The chipset mediates communication between the CPU and the other components of the system, including main memory.

The Random Access Memory (RAM) stores all running processes (applications) and the operating system. The Basic Input Output System (BIOS) includes boot firmware and power management. The BIOS tasks are handled by the operating system drivers.

Internal buses connect the CPU to various internal components and to expansion cards for graphics and sound. As for power supply, it includes a power cord, a switch, and a cooling fan and supplies power to the motherboard and internal disk drives. It converts alternating current to direct current and provides appropriate voltages to different components such as the hard disk, the CD-ROM, the motherboard, the CPU socket, etc. Computer hardware has undergone significant improvements over its history. That is why it has become a platform for uses other than computation, such as automation, communication, control, entertainment, and education. Each field in turn has imposed its own requirements on the hardware, which has evolved in response to the computer uses requirements.

Exercise 5. Find in text 1 the English for:

поєднувати елементи апаратного та програмного забезпечення; механічні пристрої, механічне обладнання та електроніка; потребувати трьох основних апаратних елементів; виконувати оброблення даних; двосторонній зв'язок між користувачем та системою; носій даних; зберігати програми та дані; карта розширення, материнська плата та оперативна пам'ять; джерело живлення та дисковод для компакт-дисків; жорсткий диск, клавіатура та миша; центральний

процесор, мікропроцесорний набір та базова система введення/виведення; охолоджуватися тепловідвідним радіатором та вентилятором; зберігати прикладні програми та операційну систему; включати мікропрограму початкового завантаження та керування електроживленням; перетворювати змінний струм на постійний; забезпечувати потрібну напругу.

Exercise 6. Say whether the statements below are true or false. Correct the false ones.

1. The software elements are mechanical devices in the system, machinery and electronics that perform physical functions.
2. As a rule, computersystem needs two basic hardware items to work perfectly.
3. The term hardware covers such parts of the personal computer as the monitor, the motherboard, the CPU, the RAM memory, the expansion card, the power supply, the CD-ROM drive, the hard disk, the keyboard, and the mouse.
4. Hardware elements are directly attached to the monitor.
5. The Basic Input Output System tasks are handled by the operating system drivers.

6. Internal buses connect the CPU to various internal components and to expansion cards for graphics and sound.
7. Computer hardware has undergone significant improvements over its history.

Exercise 7. Form all possible word combinations with the words from both columns. Translate them.

1) to combine	a) two-way communication
2) to require	b) the motherboard
3) to be used for	c) significant improvements
4) to be directly attached to	d) three basic hardware items
5) to be handled by	e) hardware elements with software elements
6) to include	f) calculations
7) to undergo	g) the operating system drivers
8) to perform	h) "the brain" of the computer
9) to be referred to as	i) alternating current to direct current
10) to convert	j) boot firmware and power management

Exercise 8. Fill in the blanks with prepositions on, of, in, to, with, over, by where necessary.

1. A functioning computer system combines hardware elements … software elements.
2. The hardware elements are mechanical devices … the system, machinery and electronics that perform … physical functions.
3. Power supply supplies power … the motherboard and internal disk drives.
4. The motherboard is the "body" … the computer.
5. Components are directly attached … the motherboard.
6. The BIOS tasks are handled … operating system drivers.
7. Computer hardware has undergone significant improvements … its history.
8. Each field has imposed its requirements … the hardware.
9. Components are attached … the motherboard.

Exercise 9. Fill in the blanks with proper terms (the hardware elements, a power supply, the central processing unit, the BIOS, the motherboard, a personal computer, a terminal device) to complete the sentences.

1. _____are the mechanical devices in the system, the machinery and the electronics that perform physical functions.

2._____ performs all data processing.

3. _____ is used like a typewriter for twoway communication between the user and the system.

4. _____ is the “body” of the computer.

5. _____ performs most of the calculations which enable a computer to function.

6. _____ includes boot firmware and power management.

7. _____ includes power cord, switch, and cooling fan and supplies power at appropriate voltages to the motherboard and internal disk drives.

Exercise 10. Answer the questions on text 1.

1. What combines different elements of the PC?
2. What are the hardware elements?
3. How many basic hardware items does a computer system usually require?
4. What does the term hardware cover?
5. What is the “body” of the PC?
6. What components are directly attached to the motherboard?
7. What function does the central processing unit perform?
8. What is the “brain” of the PC?
9. What is the CPU usually cooled by?
10. What kind of communication does a chipset mediate?
11. What does the RAM store?
12. What does the BIOS include?
13. What are the BIOS tasks handled by?
14. What connects the CPU to various internal components?
15. What does the power supply include?
16. What is the power supply designed for?
17. Why has computer hardware become a platform for different uses?
18. What kinds of uses has hardware become a platform for?

Exercise 11. Make up questions to the italicized parts of the sentences.

1. **Computer hardware** has undergone significant improvements over its history.
2. Computer hardware has become **a platform** for **different uses.**

3. Usually, **a computer system** requires **three** basic hardware items.

4. Components are **directly** attached to **the motherboard**.

5. **The CPU** is usually cooled by **a heat sink and fan**.

6. **The hardware** elements perform physical functions.

7. The CPU performs **most of the calculations.**

8. **The CPU** is sometimes referred to as **the "brain" of the computer.**

Exercise 12. Translate into English.

1. Функціональна комп'ютерна система поєднує елементи апаратного та програмного забезпечення.

2. Як правило, комп'ютерна система потребує трьох основних елементів: персонального комп'ютеру, терміналу та носія даних.

3. Персональний комп'ютер виконує оброблення даних.

4. Термінал використовують як друкарську машинку для двостороннього зв'язку між користувачем та системою.

5. Носій даних використовують для зберігання програм і даних.

6. Термін «апаратне забезпечення» охоплює такі елементи персонального комп'ютера, як: материнська плата, монітор, центральний процесор, оперативна пам'ять, карта розширення, жорсткий диск, блок живлення, дисковод для компакт-дисків, клавіатура та комп'ютерна миша.

7. Материнська плата є головною частиною комп'ютера.

8. Компоненти, що кріпляться безпосередньо до материнської плати, включають центральний процесор, мікропроцесорний набір, оперативну пам'ять, базову систему введення/виведення та внутрішні шини.

9. Центральний процесор виконує більшу частину обчислень, які дозволяють комп'ютеру функціонувати.

10. Центральний процесор зазвичай охолоджується з допомогою теплопровідного радіатора та вентилятора.

11. Мікропроцесорний набір є сполучною ланкою між центральним процесором та іншими компонентами системи, включаючи оперативну пам'ять.

12. В оперативній пам'яті зберігаються всі прикладні програми та операційна система.

13. Базова система введення/виведення включає мікропрограму початкового завантаження та керування електроживленням.

14. Завданнями базової системи введення/виведення займаються драйвери операційної системи.

15. Внутрішні шини з'єднують центральний процесор з різними внутрішніми компонентами та картами розширення для графічного та звукового відображення.

16. Блок живлення включає шнур живлення, перемикач та охолоджувальний вентилятор.

17. Блок живлення забезпечує електроживлення материнської плати та внутрішніх дисководів.

18. Він перетворює змінний струм на постійний, а також подає потрібну напругу до таких компонентів, як: жорсткий диск, CD-ROM, материнська плата, роз'єм центрального процесора.

19. Апаратне забезпечення комп'ютера за свою історію зазнало значних удосконалень.

20. Комп'ютер є основою для виконання не лише обчислень, але й автоматизації, зв'язку, керування, засобом розваг та освіти.

Exercise 13. Retell the text "Computer Hardware".

Exercise 14. Study the ways of constructing Indefinite, Perfect and Continuous Passive verb forms.

You use the Passive Voice when you want to focus on the person or thing affected by the action, rather than on the "doer" of the action (or agent).

Our house **was built** a hundred years ago.

My notebook **has been stolen.**

We use appropriate form of the verb **to be + Past Participle** of the main verb.

Present Indefinite More and more cars **are sold** every year.

Past Indefinite The Internet **was developed** in the 1960s.

Future Indefinite Everyone who applies **will be given** an interview.

Present Continuous The conference **is being held** next weekend.

Past Continuous Every car which left the ferry **was being stopped and searched.**

Present Perfect We **have been invited** to Paul's presentation.

Past Perfect He **had been promoted** three times before becoming a director.

Future Perfect The article **will have been written** by two o'clock tomorrow.

Be going + Passive Infinitive

The new company **is going to be opened** by the President.

Modal verbs + Passive Infinitive

Books **must be returned** within three weeks.

Modal verbs + Perfect

Passive Infinitive

The office **may have been left** open deliberately.

Exercise 15. Change the following sentences into the Passive Voice.

Model: We **bought** this netbook two years ago. – This netbook **was bought** two years ago by us.

1. Smart cards store vital information such as health records, drivers' licenses, bank balances, and so on.

2. Browsers may search, view, and even add and edit data on the World Wide Web.

3. The firm hired an enterprise architect to oversee the development of the new software platform.

4. A malicious program will crash and terminate software with a confusing message.

5. Almost all computers, including hand-held computers, desktop computers, supercomputers, and even video game consoles, use an operating system.

6. The university used encryption on its web site to protect information from unauthorized access.

7. In the future, digital distribution on the Internet will replace all other forms of media distribution including CDs, DVDs, and even radio and television broadcasts.

8. Users of the Internet can send mail messages with vast databases of information to each other.

9. Mobile devices normally provide the programs needed to enable Wi-Fi, Bluetooth, or other wireless connectivity.

10. Years ago, people commonly set up their home networks just to connect a few PCs, share some documents and perhaps a printer.

11. A scanner can take a photograph or magazine article and digitize it.

12. The central processing unit (CPU) performs most of the calculations which enable a computer to function.

13. Windows 8 is now available, but most organizations are still deploying Windows 7.

Exercise 16. Change the following sentences into the Passive Voice and ask questions on them.

1. Small electronic device can control every move of your robotic personal assistant.

2. In the 2000s smartphones, cloud computing, and other innovations revolutionized the way we live and work.

3. In the near future, computers will use nanotechnology to shrink the size of silicon chips, increasing speed and power with parallel processing.

4. The military is currently working on the next generation of future computers.

5. The system analyst was analyzing and interpreting company's needs at that time.

6. Practical computer systems divide software systems into three major classes: system software, programming software, and application software.

7. Cellular phones are now also dialing up the Internet to provide e-mail and answering machine services.

8. School innovators will require some patience as computer skills develop.

9. Our new programmer was developing software for the operating system of a computer yesterday evening.

10. The ARPANET computer network made a large contribution to the evolution of e-mail.

11. The Operating System manages and controls the computer through the use of an interface.

12. Past architecture research often focused on chip microprocessors or stand-alone computers with performance as main optimization goal.

13. Information and communication technology (ICT) is transforming our world, including healthcare, education, science, commerce, government, defense, and entertainment.

14. In the future, PDA will store the entire human knowledge base.

15. Various computer professionals were writing programs in the computer department.

Exercise 17. Change the following sentences into the Active Voice. Add the doer/agent of the action where necessary.

1. Bill Gates was known in computer circles as a founder of Microsoft Corporation and developer of Windows.

2. Mathematical models can be used not only in engineering disciplines but also in the social sciences.

3. Solaris was developed by Sun Microsystems as a more open option of SunOS for its SPARC-based servers and workstations.

4. For hand-held and desktop computers, the user interface is generally considered a part of the operating system.

5. Software programs are normally written and compiled for certain hardware platforms.

6. Mainframes will be completely replaced by the lesser categories of mid-range computers, workstations, and powerful PCs.

7. Multimedia systems are known for their educational and entertainment value.

8. Modern technologies are being researched and developed including biocomputers and quantum computers.

9. Networks with speeds of 100 gigabits and higher are being planned using fiber optics for the Internet 2 system.

10. An alliance between IBM, Motorola, and Apple was formed several years ago to develop and manufacture the PowerPC chip.

11. Microcomputers can be configured to serve multiple users.

12. Today thousands of applications are designed for almost every purpose, from writing letters to playing games.

13. Mobile computing can be employed when we interact with our smartphones.

14. Future Internet appliances will be based on embedded computing systems.

15. The computer system was being upgraded by the administrator all morning yesterday.

Exercise 18. Define the tense and voice used in the following sentences (Present, Past, Future Indefinite / Continuous Active/Passive) and translate them.

1. Programmers are also known as ‘software developers’.

2. The user was advised to reboot the computer after a serious crash in which the computer no longer responded.

3. Embedded systems are typically controlled by inexpensive, specialized processors which can only handle very specific tasks.

4. More and more, software is being distributed over the Internet, as open source, shareware, freeware, or traditional proprietary and upgrade versions.

5. By the early 1990’s, computers were commonly used for writing papers, playing games, financial accounting, and business productivity applications.

6. Each Windows OS is optimized for different users, hardware configurations, and tasks.

7. Since main memory is volatile, all contents are lost when the computer power is turned off.

8. In what way will computers be linked on a network?

9. The notebook was performing slowly when changing programs, so the technician installed more RAM and this solved the problem.

10. The original backbone of the Internet is based on an old military network called ARPANET which was built by ARPA in the late 1960’s.

11. Technology will offer some great capabilities in pulling together the design of any software system.

12. Not surprisingly, Linux’s OS market share is growing very quickly around the world.

13. What operating system are you installing? – Windows Vista.

14. Since nobody likes to wait for a computer, high-quality computers will have fast processors and lots of quick memory.

15. The letter is being sent now.

Exercise 19. Complete these sentences using the appropriate passive form of the verbs in brackets (Present, Past or Future Indefinite / Continuous).

1. Applications (mean) to make users more productive and get work done faster.

2. An embedded OS can (find) inside an increasing number of consumer gadgets including phones (iPhone OS), PDAs (Windows CE), and digital media players.

3. The way that the motherboard (design) and (lay out) dictates how the entire computer is going (organize).

4. The instructions (perform) by a computer while the programs (debug) and (test).

5. Intel (start) by several people who are now legends in the computer world, including Robert Noyce and Gordon Moore.

6. Bluetooth technology (optimize) for networking between common consumer electronics such as mobile phones, mp3 players, and similar devices.

7. We project that mainframe computer applications for businesses essentially (replace), either by client/server applications or new language software.

8. The antivirus programs currently (install) by a specialist.

9. The Macintosh operating systems (design) to (use) on Apple Macintosh computers. 10. The computer contacted the address and the information (make) available to the operator.

11. Most users have no idea that their information (collect) and (store) at the moment on their computer.

12. Ultimately people's power (must exercise) to ensure that computers (use) not only efficiently but in a socially responsible way.

13. Regardless of the networking choice, future Internet appliances (base) on embedded computing systems.

14. When nanotechnology fully (develop), machines (assemble and snap) together using the chemistry of atoms and molecules.

15. New applications program (write) by the program developer yesterday evening.

Exercise 20. Use Indefinite or Continuous Tenses (Present, Past or Future) of the verbs in brackets choosing between the Active and Passive forms.

1. A company or organization (store) its information in electronic documents on one of the Internet computers.

2. The whole exciting Internet world (wait) for you!

3. In some families children (not allow) to surf the net on their own.

4. When the word processor application (crash), the user (have) to abort the program and (lose) all his unsaved changes.

5. As a student of English and Technology, you (hear) people use the words ‘Internet’ and ‘World Wide Web’ almost interchangeably.

6. In the future, all forms of media distribution including CDs, DVDs, and even radio and television broadcasts (replace) by digital distribution on the Internet.

7. Digital cameras (become) more common.

8. The program (design) by the software engineer now.

9. When the network administrator (format) the wrong hard disk drive array, he accidentally (lose) all the company data.

10. The man (not can) spell-check documents in German last Saturday because his copy of Word (upgrade).

11. The computer user (not can) access a computer resource because he (forget) his username and password.

12. You (find out) the answer to almost everything on Google nowadays.

13. When you (surf) the net, you (move) from one document or the web site to another.

14. Computer science experts predict that more traditional desktop computer (replace) by powerful and affordable laptop computers and netbooks.

15. The first version of a software application (not release) to the public because it (contain) serious bugs.

16. In recent years, more and more features (include) in the basic GUI OS, including notepads, sound recorders, and even web browsers and games.

17. Sixty years ago computers also (cost) a lot of money to build and operate and they only (use) by large organizations.

18. Computers all over the world (join) by phone lines, satellite or cable.

19. Students (download) several files from the university’s server yesterday morning.

20. This time next week our department (work) on this project.

21. When computers first (appear), they (take up) whole rooms and (require) specialized training to operate them.

22. The hackers (admit) deleting new programs while they (interview) by security police.

23. PC (protect) from different viruses by special programs.

24. The commercial global Internet system with its World Wide Web applications (begin) in 1993-1994.

25. This time tomorrow the program designer (write) specifications for new computer systems.

Text 2. Hardware Categories

A typical computer consists of two parts: hardware and software. Hardware is any electronic or mechanical part of the computer system that you can see or touch. It is categorized as follows:

Input hardware is used to collect data and input it into the computer system in computer-usable form. The keyboard and mouse are the most common input devices.

Processing hardware retrieves and executes (interprets) instructions (software) provided to the computer. The main components of processing hardware are the central processing unit (CPU), which is the brain of the computer, and main memory, where all instructions and/or data ready for processing are held. Since main memory is volatile, all contents are lost when the computer's power is turned off. Output hardware provides a means for the user to view information produced by the computer system – either in hardcopy form, such as printouts from a printer, or softcopy form, such as a display on a monitor, a TV-like screen.

Communications hardware facilitates connections between computers and computer systems over phone lines and other channels. Examples are modems, cables and fax modems. Software is a set of instructions, called a program, which tells a computer what to do. Software that runs the hardware and allows the computer to manage its resources is system software. Software that performsa general business function is referred to as applications software.

Exercise 21. Fill in the blanks with proper terms from text 2 to complete the sentences.

1._____ refers to any electronic or mechanical part of the computer system that you can see or touch.

2._____ a set of instructions, called a program, which tells a computer what to do.

3._____ runs the hardware and allows the computer to manage its resources.

4._____ performs a general business function.

5._____ retrieves and executes instructions provided to the computer.

6._____facilitates connections between computers and computer systems over phone lines and other channels.

7._____ is used to collect data and input it into the computer system in computer-usable form.

8._____provides a means for the user to view information produced by the computer system.

Exercise 22. Speak on computer hardware adding some information you know from your own experience.

Exercise 23. Read, translate and entitle text 3.

Text 3

Nowadays PCs represent the wide-spread class of digital machines. There are many different individual components which can be mixed and matched in thousands of different configurations. This lets you customize the PC you either buy or build to meet your exact needs. The system case, sometimes called the chassis or enclosure, is the metal and plastic box that houses the main components of the computer. Most people do not consider it a very important part of the computer (perhaps in the same way they would not consider their own skin a very important body organ). While the case is not as critical to the system as some other computer components (like the processor or hard disk), it has several important roles to play in the functioning of a properlydesigned and well-built computer. The case does not appear to perform any function at all, at first glance. However, this definitely is not true; the case is in fact much more than just a box. The motherboard is mounted into the case, and all the other internal components are mounted into either the motherboard or the case itself. The case must provide a solid structural framework for these components to ensure that everything fits together and works well. The system case performs several important functions for your PC, for instance, protection for the computer circuits, cooling and system organization.

Exercise 24. Put key questions to text 3.

Exercise 25. Read and translate text 4.

Text 4. Motherboard

The motherboard is, in many ways, the most important component in your computer (not the processor, even though the processor gets much more attention.). The motherboard and its major components (the chipset, BIOS, cache*, etc.) are the major systems that this brain uses to control the rest of the computer. Having a good understanding of how the motherboard and its contained subsystems work is probably the most critical part of getting a good understanding of how PCs work in general. The motherboard plays an important role in all aspects of the computer system. In one way or another, everything is eventually connected to the motherboard. The way that the motherboard is designed and laid out dictates how the entire computer is going to be organized. The motherboard contains the chipset and BIOS program, which control most of the data flow within the computer. Almost all communication

between the PC and its peripherals, other PCs, and the user goes * cache – надоперативна пам'ять, кеш, надоперативний ЗП through the motherboard. The motherboard dictates directly the choice of processors, memory, system buses for use in the system, and these components affect the system's performance. It also determines what types of peripherals you can use in your PC. The capabilities of the motherboard specify to what extent you will be able to upgrade your machine. For example, there are some motherboards that will accept regular Pentiums of up to 133 MHz speed only, while others will go to 200 MHz. Obviously, the second one will give you more room to upgrade when starting with a P133.

Exercise 26. Write a summary of the text "Motherboard".

Exercise 27. Say whether the statements below are true or false. Correct the false ones.

1. Nowadays PC represents the useful part of design idea of every living-room.
2. The system case is usually one of the most overlooked parts of the PC.
3. The main functions of the system case are to protect the computer circuits, cooling system and repair the motherboard.
4. Any computer system can't work without the motherboard as it is its heart.
5. The chipset and other motherboard circuitry are the liver of the motherboard.
6. The main function of the chipset is to direct traffic and control the flow of information inside the computer.
7. The system buses are the electrical channels through which various parts of the computer communicate.
8. The BIOS is a computer program that helps you to learn hardware.
9. The BIOS gives you the opportunity to set or change many different parameters that control how your computer will function.
10. The system cache is not large, and we can find it between the processor and the system memory.
11. Each time the processor requests great amounts of meals.
12. System case, motherboard and system devices are personal computer components.

Exercise 28. Agree or disagree with the following statements. Give your reasons.

1. Using a computer helps us to develop our language skills.
2. Computers have greatly changed the way we work and study.

Exercise 29. Make a list of advantages and disadvantages of using a personal computer. Discuss it with your group-mates.

Exercise 30. Find some additional information and speak on the topics:

1. The CPU.
2. The main memory (RAM and ROM).
3. Peripherals (input devices, output devices, storage devices).

UNIT 4. OPERATING SYSTEMS

Translate and study the basic vocabulary.

dumb terminal	
embedded (built-in) operating system	
file system	
graphical user interface	
hand-held computer	
hard drive, hard disk	
host	
master system	
move data	
multitasking	
multi-tasking operating system	
multi-user operating system	
networking	
real-time operating system	
scalable processor architecture (SPARC)	
share (resources)	
single-task(ing) operating system	
single-user operating system	
spreadsheet	
universal serial bus (USB)	
USB key, USB drive, flash drive	
video game console	
video/visual display unit (VDU)	
word processing	

Exercise 1. Choose verbs among the following words. Read and translate the word. Operating, manipulate, priority, include, contain, multitasking, system, receive, expensive, offer, software, simplify, responsible, operate, handheld, find, multi-user, tend.

Exercise 2. Give synonyms (a) and antonyms (b) for the following words:
a) manage, make easier, write programs, hand-held computer, however, execute, display, communicate the message, accept, machinery, single-user, design, laptop computer, enable, common, split, locate, develop, host, easy, apply, operate, affect, significant, built-in, memory;
b) easy, significant, input, responsible, centre, good, single-user, all, include, oldest, embedded, effectively, different.

Exercise 3. Write derivatives of the words below and explain their meanings.
Model: simple – simplify – simplification – simplicity – simply
Simple, ease, manage, coordinate, apply, operate, inform, execute, science, effect, store, inform, process, signify, differ, major.

Exercise 4. Give Ukrainian equivalents for the following word combinations.
To be responsible for; the management, coordination and sharing of computer resources; to act as a host for applications that are run on the machine; a master system; to manage the basic operation of the computer; to relieve applications from having to control the hardware; to make it easier to write programs; hand-held computers and video game consoles; embedded operating system; some features; to execute a program; to print or display the result of a program on the printer or the screen; store the output data or programs; to communicate the message from the system to the user; to accept input from the user through the keyboard or mouse; to be designed to manage the computer; single-user, multi-tasking operating system; to take advantage of the computer resources simultaneously; the most common operating systems; built-in networking support; multi-user environment; regardless of the media; SPARC-based servers; a powerful and expensive computer; to be built for serving information to many PCs or dumb terminals; to have the largest share of the Internet market.

Text 1. Operating System Functions and Categories

An operating system is an interface between the hardware and the user. It is responsible for the management, coordination and sharing of computer resources. The operating system acts as a host for applications that are run on the machine. It is the master system of programs that manages the basic operation of the computer. This relieves applications from having to control the hardware and makes it easier to write programs. Almost all computers, including hand-held computers, desktop computers, supercomputers, and even video game consoles, use an operating system. Some of the oldest models may however use an embedded operating system on a compact disk or other data storage devices.
A good operating system should have the following features:
1) help in loading of programs and data from external sources intothe internal memory before they are executed;

2) help programs to perform input/output operations, such as:
a) print or display the result of a program on the printer or the screen;
b) store the output data or programs written on the computer in a storage device;
c) communicate the message from the system to the user through the VDU; and
d) accept input from the user through the keyboard or mouse.
The broad categories of operating systems are:
1) real-time operating systems used to control machinery, scientific instruments and industrial systems;
2) single-user, single-task operating systems designed to manage the computer so that one user can effectively do one thing at a time;
3) single-user, multi-tasking operating systems used by most people on their desktop and laptop computers today. Windows and Mac OS are examples of an operating system that will let a single user have several programs in operation at the same time, and
4) multi-user operating systems allow many different users to take advantage of the computer resources simultaneously.
The most common operating systems include Microsoft Windows, Mac OS, UNIX, Linux and Solaris. Windows lets you display your work in windows. A window is a portion of the video display area dedicated to some specific purpose. With Windows, which supports multitasking and graphical user interface, you can display several windows on a computer screen, each showing a different application, such as word processing or spreadsheet. You can easily switch between the applications and move data between them. Windows 7 is the recent version of Microsoft Windows family. Microsoft Windows has a significant majority of market share in the desktop and notebook computer markets, while servers generally use Linux or other Unix-like systems. Embedded device markets are split among several operating systems. The Macintosh operating system was designed to be used on Apple Macintosh computers. This operating system supports multi-tasking and enables users to read MS-DOS and Windows files. The UNIX operating system is one of the oldest operating systems. It is a multi-tasking operating system that includes built-in networking support. It is a popular operating system in universities, where a multi-user environment is often needed. Unlike other operating systems, Linux and UNIX allow any file system to be used regardless of the media it is stored in, whether it is a hard drive, a disc (CD, DVD...), a USB key, or even contained within a file located on another file system.
Solaris was developed by Sun Microsystems as a more open option of SunOS for its SPARC-based servers and workstations. Sun machines are popular, powerful, and expensive computers built for serving information to many PCs or dumb terminals. Their processors can perform several tasks simultaneously. Many universities and large corporations use Sun machines to serve information on their networks. Currently, Solaris is one of the most popular versions of UNIX and has the largest share of the Internet market.

Exercise 5. Find in text 1 the English for:

інтерфейс між апаратним забезпеченням та користувачем; відповідати за керування, координацію та розподіл ресурсів; головна система; звільняти прикладні програми від необхідності керувати апаратними засобами; полегшувати написання програм; кишенькові комп'ютери та приставки для відеоігор; вмонтована (вбудована) операційна система; пристрої зберігання даних; мати такі властивості; операційна система реального часу; прилади для наукових досліджень та промислові системи; операційна система індивідуального користування; передавати повідомлення від системи до користувача; приймати вхідну інформацію; багатозадачна операційна система; багатокористувацька система; дозволяти різним користувачам одночасно використовувати комп'ютерні ресурси; найпоширеніші операційні системи; частина площі відеодисплея, що має певне призначення; підтримувати багатозадачність та графічний інтерфейс користувача; оброблення текстів та великоформатна (електронна) таблиця; перемикати прикладні програми та пересилати інформацію між ними; остання версія; домінувати на ринку настільних комп'ютерів та ноутбуків; вмонтований пристрій; вмонтована підтримка мережевого режиму; багатокористувацьке середовище; використовуватися незалежно від носія; потужні та дорогі комп'ютери; подавати інформацію багатьом ПК; одночасно виконувати кілька завдань.

Exercise 6. Say whether the statements below are true or false. Correct the false ones.

1. An operating system is an interface between the hardware and the user.
2. It is responsible for the management, coordination and sharing of computer resources.
3. It is the master system of programs that manages the basic operation of the software.
4. The operating system acts as a host for system programs that are run on the machine.
5. The operating system relieves applications from having to control the hardware but makes it more difficult to write programs.
6. Almost all computers, including hand-held computers, desktop computers, supercomputers, and even video game consoles, use an operating system.
7. A good operating system should help in loading of programs and data from external sources into the internal memory before they are executed and help programs to perform input/output operations.
8. Windows and Mac OS are multi-user operating systems that allow many different users to take advantage of the computer resources simultaneously.
9. The most common operating systems include Mac OS, UNIX, Linux and Solaris.
10. Windows lets you display your work in cells.
11. A window is a portion of the video display area dedicated to some specific purpose.
12. WithWindows, which supports multitasking and graphical user interface, you can display several windows on a computer screen, each showing a different application, such as word processing or spreadsheet.
13. Windows 2006 is the recent version of Microsoft Windows family.
14. Microsoft Windows has a significant majority of market share in the servers and notebook computer markets.
15. The Macintosh operating system was designed to be used on Apple Macintosh computers.

16. The Macintosh operating system does not support multi-tasking and enables users to read Windows files.
17. The UNIX operating system is one of the oldest operating systems.
18. UNIX is a single-user operating system that includes built-in networking support. 19. Linux and UNIX allow any file system to be used regardless of the media it is stored in.
20. Solaris was developed by Sun Microsystems as a more open option of SunOS for its SPARC-based servers and workstations.
21. Sun machines are popular, powerful, and expensive computers built for serving information to many mainframes and supercomputers.
22. Sun machines processors can perform several tasks simultaneously.
23. Currently, Solaris is one of the most popular versions of UNIX and has the largest share of the Internet market.

Exercise 7. Form all possible word combinations with the words from both columns. Translate them.

1) to manage	a) computer resources
2) to control	b) several operating systems
3) to store	c) multitasking
4) to take advantage of	d) basic operation of the computer
5) to switch between	e) serving information to PCs
6) to display	f) the output data or programs
7) to support	g) machinery and scientific instruments
8) to be split among	h) the work in windows
9) to be built for	i) several tasks simultaneously
10) to perform	j) the applications

Exercise 8. Fill in the blanks with prepositions to, in, into, on, of, for, from, between, among, through, at, by where necessary.
1. An operating system is an interface … the hardware and the user.
2. It is responsible … the management, coordination and sharing … computer resources.
3. The operating system acts as a host … applications that are run … the machine.
4. It is the master system … programs that manage the basic operation … the computer.
5. This relieves applications … having to control the hardware..
6. A good operating system should help … loading … programs and data … external sources … the internal memory.
7. Operating system should communicate the message … the system … the user … the VDU and accept input … the user … the keyboard or mouse.
8. Single-user, multi-tasking operating systems are used … most people … their desktop and laptop computers today.
9. Windows and Mac OS are examples … an operating system that will let a single user have several programs … operation … the same time.
10. A window is a portion … the video display area dedicated … some specific purpose.

11. You can easily switch …the applications and move data … them.
12. Microsoft Windows has a significant majority … market share … the desktop and notebook computer markets.
13. Embedded device markets are split … several operating systems.
14. Unlike other operating systems, Linux and UNIX allow any file system to be used regardless … the media it is stored … .
15. Solaris was developed … Sun Microsystems as a more open option … SunOS … its SPARCbased servers and workstations.
16. Sun machines are popular, powerful, and expensive computers built … serving information … many PCs or dumb terminals.

Exercise 9. Fill in the blanks with proper terms (operating system, Unix, Windows, multi-user operating system, Macintosh operating system, Sun machines, window, Solaris) to complete the sentences.

1. _____ is a portion of the video display area dedicated to some specific purpose.
2. _____ is an operating system that allows many different users to take advantage of the computer resources simultaneously.
3. _____ are popular, powerful, and expensive computers built for serving information to many PCs or dumb terminals.
4. _____ is one of the most popular versions of UNIX developed by Sun Microsystems as a more open option of SunOS.
5. _____ is one of the oldest multi-tasking operating systems that includes built-in networking support.
6._____ is an operating system that lets you display your work in windows.
7. _____ is an interface between the hardware and the user.
8. _____ is an operating system that was designed to be used on Apple-Macintosh computers.

Exercise 10. Answer the questions on text 1.

1. What is an operating system?
2. What is it responsible for?
3. What are the functions of the operating system?
4. What does the operating system relieve applications from?
5. What kinds of computers use an operating system?
6. What kind of operating system do the old models use?
7. What features should a good operating system have?
8. What are the broad categories of operating systems?
9. What are real-time perating systems used for?
10. What are single-user, single-task operating systems designed for?
11. What operating systems are used by most people on their desktop and laptop computers today?
12. What operating systems allow many different users to take advantage of the computer resources simultaneously?

13. What are the most common operating systems?
14. What does Windows allow the users to do?
15. What operating system has a significant majority of market share in the desktop and notebook computer markets?
16. What operating systems do servers generally use?
17. What files does the Macintosh operating system enable users to read?
18. What kind of system is the UNIX operating system?
19. How do Linux and UNIX differ from other operating systems?
20. What kind of computers was Solaris developed for?
21. What kind of computers are Sun machines?
22. What do many universities and large corporations use Sun machines for?
23. What operating system has the largest share of the Internet market?

Exercise 11. Put all possible questions to the sentences below.

1. An operating system is an interface between the hardware and the user.
2. An operating system is responsible for the management, coordination and sharing of computer resources.
3. The operating system relieves applications from having to control the hardware.
4. Almost all computers, including hand-held computers, desktop computers, supercomputers, and even video game consoles, use an operating system.
5. Some of the oldest models may however use an embedded operating system.
6. A good operating system should help in loading of programs and data from external sources into the internal memory before they are executed.
7. The most common operating systems include Microsoft Windows, Mac OS, UNIX, Linux and Solaris.
8. You can display several windows on a computer screen.
9. The Macintosh operating system was designed to be used on Apple Macintosh computers.
10. This operating system supports multi-tasking and enables users to read MS-DOS and Windows files.
11. Their processors can perform several tasks simultaneously.

Exercise 12. Translate into English.

1. Операційна система – це інтерфейс між апаратним забезпеченням та користувачем.
2. Операційна система відповідає за керування комп'ютерними ресурсами, їх координацію та розподіл.
3. Це головна система серед програм, що керують роботою комп'ютера.
4. Операційна система звільняє прикладне програмне забезпечення від необхідності керувати апаратним забезпеченням та полегшує написання програм.
5. Операційні системи реального часу використовують для керування механічним обладнанням, приладами для наукових досліджень та промисловими системами.
6. Більшість людей використовують у своїх настільних комп'ютерах та ноутбуках багатозадачні операційні системи індивідуального користування.

7. Існують такі класи операційних систем: операційні системи реального часу; однозадачні операційні системи індивідуального користування; багатозадачні операційні системи індивідуального користування та багатокористувацькі операційні системи.
8. Найпоширеніші операційні системи включають Microsoft Windows, Mac OS, UNIX, Linux та Solaris.
9. «Вікно» – це частина площі відеодисплея, призначена для певної мети.
10. Windows підтримує багатозадачність та графічний інтерфейс користувача.
11.3 Windows можна переходити від однієї програми до іншої та пересилати інформацію між ними.
12. Microsoft Windows переважає на ринку настільних комп'ютерів та ноутбуків, тоді як сервери зазвичай використовують Linux чи інші системи, подібні до UNIX.
13. На відміну від інших операційних систем, Linux та UNIX дозволяють використовувати будь-яку файлову систему, незалежно від носія, на якому вона зберігається.
14. Solaris була створена компанією Sun Microsystems як більш відкрита версія SunOS для її серверів та робочих станцій.
15. Машини Sun є популярними, потужними машинами, створеними для надання інформації багатьом ПК та простим терміналам.
16. На сьогодні Solaris є однією з найбільш популярних версій UNIX і домінує на Інтернет ринку.

Exercise 13. Retell the text "Operating Systems".

Text 2. Operating System Interfaces

Operating systems offer a number of services to application programs and users. Applications access these services through application programming interfaces (APIs) or system calls. By invoking these interfaces, the application can request a service from the operating system, pass parameters, and receive the results of the operation. Users may also interact with the operating system with some kind of software user interface (UI) like typing commands by using command line interface (CLI) or using a graphical user interface (GUI, commonly pronounced "gooey"). For handheld and desktop computers, the user interface is generally considered a part of the operating system. On large multi-user systems like UNIX and UNIX-like systems, the user interface is implemented as an application program that runs outside the operating system. The operating system acts as an interface between an application and the hardware. The user interacts with the hardware from "the other side". The operating system is a set of services which simplifies the development of applications. Executing a program involves the creation of a process by the operating system. The kernel creates a process by assigning memory and other resources, establishing a priority for the process (in multi-tasking systems), loading program code into memory, and executing the program. The program then interacts with the user and devices performing its intended function.

Exercise 14. Say whether the statements below are true or false. Correct the false ones.

1. An operating system is an interface between the application software and system software.
2. Operating systems offer a number of services to application programs and users.
3. Applications access these services through APIs or system calls.
4. Users may also interact with the operating system with some kind of software UI like typing commands by using CLI or using a graphical user interface.
5. For hand-held and desktop computers, the user interface is generally considered a part of the hardware.
6. On large multi-user systems like UNIX and UNIX-like systems, the user interface is implemented as an application program that runs outside the operating system.
7. The operating system acts as an interface between an application and the hardware.
8. The operating system is a set of instructions which simplifies the development of applications.
9. Writing a program involves the creation of a process by the operating system.
10. The kernel creates a process by assigning memory and other resources, establishing a priority for the process (in multi-user systems), loading program code into memory, and executing the program.
11. The program interacts with the user and devices performing its intended function.

Text 3. Microsoft Windows

Microsoft Windows is a family of proprietary operating systems that originated as an add-on to the older MS-DOS operating system for the IBM PC. Modern versions are based on the newer Windows NT kernel that was originally intended for OS/2. Windows runs on x86, x86-64 and Itanium processors. Earlier versions also ran on the DEC Alpha, MIPS, Fairchild (later Intergraph) Clipper and PowerPC architectures (some work was done to port it to the SPARC architecture). As of June 2008, Microsoft Windows holds a large amount of the worldwide desktop market share. Windows is also used on servers, supporting applications such as web servers and database servers. In recent years, Microsoft has spent money on significant marketing, research and development to demonstrate that Windows is capable of running any enterprise application, which has resulted in consistent rice/ performance
records and significant acceptance in the enterprise market. The most widely used version of the Microsoft Windows family is Windows XP, released on October 25, 2001. In November 2006, after more than five years of development work, Microsoft released Windows Vista, a new version of Microsoft Windows family which contains a large number of new features and architectural changes. Chief among these are a new user interface and visual style called Windows Aero, a number of new security features such as User Account Control, and a few new multimedia applications such as Windows DVD Maker. A server variant based on the same kernel, Windows Server 2008, was released in early 2008. Windows 7 is the recent version which has recently been developed and released.

Exercise 15. Say whether the statements below are true or false. Correct the false ones.

1. Modern versions of Microsoft Windows are based on Windows NT kernel that was originally intended for OS/1.
2. In recent years, Macintosh has spent money on significant marketing, research and development to demonstrate that Windows is not capable of running any enterprise application.
3. The most popular version of the Microsoft Windows family is Windows XP, released on October 15, 2001.
4. In November 2006 Microsoft released Windows Vista, a new version of Microsoft Windows family which contains a large number of new features and architectural changes.
5. Windows 7 is currently under development.

Exercise 16. Make up questions to the italicized parts of the sentences.

1. Such applications include **some small embedded systems**.
2. Operating system can be categorized **by technology, ownership, licensing, working state, usage, and by many other characteristics**.
3. Modern versions are based **on the newer Windows NT kernel**.
4. In recent years, **Microsoft** has spent significant money on marketing, research and development to demonstrate that Windows is capable of running any enterprise application.
5. The most widely used version of the Microsoft Windows family is **Windows XP**, released on October 25, 2001.

Exercise 17. Fill in the blanks with prepositions.

1. Microsoft Windows is a family … proprietary operating systems that originated as an add-on … the older MS-DOS operating system … the IBM PC.
2. Modern versions are based … the newer Windows NT kernel that was originally intended … OS/2.
3. Windows runs … x86, x86-64 and Itanium processors.
4. Earlier versions also ran … the DEC Alpha, MIPS, Fairchild (later Intergraph) Clipper and PowerPC architectures.
5. … recent years, Microsoft has spent money … significant marketing, research and development to demonstrate that Windows is capable … running any enterprise application, which has resulted … consistent rice/ performance records and significant acceptance … the enterprise market.
6. The most widely used version … the Microsoft Windows family is Windows XP, released … October 25, 2001.
7. … November 2006, … more than five years … development work, Microsoft released Windows Vista, a new version … Microsoft Windows family which contains a large number … new features and architectural changes.

Exercise 18. Change the following words into adjectives. Use the following suffixes -able; -ible; -al; – ant; -ent; – ful;- ic; -ive; -ous; – y.

Purpose, reason, exhaust, success, predominance, skill, power, vary, magnet, create, stick, hierarchy, program, comprehend, observe, emerge, gloom, courage, science, compare, noise, form, apply, event, experiment, base, collect, economy, gloom, construct, transmit, function, redundancy, vision, resistance, depend, select, structure, efficiency.

Exercise 19. Form negative adjectives using the following prefixes un-; in-; ir-; im- ;il-; dis-; non-.

Reasonable, relevant, satisfied, authorized, complete, fortunate, regular, possible, countable, flammable, respectful, forgettable, direct, perfect, reliable, sensitive, legal, existent, equal, responsible, patient, stable, rational, loyal, honest, restrictive, intentional, decent, proper, skilled, capable, legitimate, metallic, productive, correct, logical.

Exercise 20. Give the comparative and the superlative degree of the following adjectives.

Fast, large, intelligent, convenient, complex, simple, small, good, little, much, narrow, long, bad, far, busy, important, regular, reliable, efficient.

Exercise 21. Decide whether the italicized words are adjectives or adverbs. Translate the sentences.

1. Computer hardware has undergone **significant** improvements over its history.
2. Software security practices should contain **significantly fewer** exploitable weaknesses.
3. Malware is a **general** term used by computer professionals to mean various annoying software.
4. For handheld and desktop computers the user interface is **generally** considered a part of the operating system.
5. A **typical** computer includes hardware and software.
6. A database consists of an organized collection of data for one or **more** uses, **typically** in digital form.
7. The professor delivered **more** lectures than his colleague.
8. **Little** attention to documentation and **large** focus on **effective** communication will help produce desired project results.
9. The software should be designed to guard against both **likely** and **unlikely** events. 10. Students would study **better** if they had **better** equipment in the classrooms.

Exercise 22. Choose the right word from those in brackets. Translate the sentences.

1. The software construction is (close, closely) linked to the software configuration management.
2. Multi-user operating systems allow many (different, differently) users to take advantage of the computer resources (simultaneous, simultaneously).
3. (Fortunate, fortunately), such largescale methods are not the only kind available for modelling software.

4. Microsoft Windows has a (significant, significantly) majority of market share in the desktop and notebook computer markets, while servers (general, generally) use Linux or other Unix-like systems.
5. A good operating system helps in loading of programs and data from (external, externally) sources into the (internal, internally) memory.
6. A highlevel design does not (necessary, necessarily) represent the architecture of the software.
7. Antivirus software may include the ability to (periodic, periodically) receive virus definition updates in order to maintain the software effectiveness.
8. (Internal, internally) architecture is concerned with cost, performance, scalability and other operational matters.
9. A management information system generates information (accurate, accurately) and (regular, regularly).
10. Passive matrix displays contain a grid of (horizontal, horizontally) and (vertical, vertically) wires with an LCD element at each intersection.
11. Hypermedia can be considered one (particular, particularly) multimedia application.
12. (Common, commonly) methods of data masking include: encryption, decryption, masking and substitution.

Exercise 23. Translate the word combinations below into Ukrainian.
To be based on his experience in the MULTICS project; a large, complex family of inter-related operating systems; to resemble the original UNIX; to refer to a large set of operating systems; a diverse group of; a trademark of The Open Group; to conform to standards; a wide variety of machine architectures; for serves as well as for workstations; in academic and engineering environments; market share statistics; to make usage under-represented; to acquire multiple OS; to be applied to; to be originally created for.

Text 4. UNIX and UNIX-like Operating Systems

Ken Thompson wrote B, which he used to write UNIX, based on his experience in the MULTICS project. B was replaced by C, and UNIX developed into a large, complex family of interrelated operating systems which have been influential in every modern operating system. The UNIX-like family is a diverse group of operating systems, with several major sub-categories including System V, BSD, and Linux. The name “UNIX” is a trademark of The Open Group which licenses it for use with any operating system that has been shown to conform to their standards. “UNIX-like” is commonly used to refer to a large set of operating systems which resemble the original UNIX. UNIX-like systems run on a wide variety of machine architectures. They are heavily used for servers in business, as well as for workstations in academic and engineering environments. Free software UNIX versions, such as GNU, Linux and BSD, are popular in these areas. Market share statistics for freely available operating systems is usually inaccurate since most free operating systems are not purchased, making usage under-represented. On the other hand, market share statistics based on total downloads of free operating systems is often inflated, as there is no economic disincentive to acquire multiple operating systems so users can download multiple systems, test them, and

decide which they like best. Some UNIX versions like HP's HP-UX and IBM's AIX are designed to run only on that vendor's hardware. Others, such as Solaris, can run on multiple types of hardware, including x86 servers and PCs. Apple's Mac OS X, a hybrid kernel-based BSD version derived from NEXTSTEP, Mach, and FreeBSD, has replaced Apple's earlier (non-UNIX) Mac OS. UNIX interoperability was introduced by establishing the POSIX standard. The POSIX standard can be applied to any operating system, although it was originally created for various UNIX versions.

Exercise 24. Find in text 4 the English for:

група різних операційних систем; взаємопов'язані операційні системи; відповідати стандартам; академічне та технічне середовище; питома вага на ринку; бути неточним; економічні перешкоди; купувати різноманітні операційні системи; нагадувати вихідний UNIX.

Exercise 25. Complete the sentences choosing the proper word combinations. Translate them.

1. Unix developed into a large, complex family of interrelated operating systems which have been influential in … .
a) every complicated operating system
b) every modern operating system
c) every expensive operating system
2. The name "UNIX" is a trademark of … .
a) the institution for protection of the environment
b) the Closed Group
c) the Open Group
3. "UNIX-like" is commonly used to refer to … .
a) a large set of operating systems
b) a large set of applications
c) a large set of interfaces
4. UNIX-like systems run on a wide variety of … .
a) machine architectures
b) machinery
c) machine exhibitions
5. Market share statistics for freely available operating systems is … .
a) usually inaccurate
b) inaccurate to some extent
c) quite inaccurate
6. There is no economic disincentive to acquire multiple operating systems so users can … .
a) test multiple applications and start operation
b) download multiple systems, test them, and decide which they like best
c) check multiple systems, and decide which of them they can sell
7. Some UNIX variants like HP's HP-UX and IBM's AIX are designed

to run only … .
a) on hardware or software
b) on that vendor's software
c) on that vendor's hardware
8. Others, such as Solaris, can run on multiple types of hardware, including … .
a) interface and PCs
b) x86 servers and PCs
c) servers and OS
9. UNIX interoperability was introduced by … .
a) establishing the POSIX standard
b) acquiring the POSIX versions
c) granting a license to the POSIX standard
10. The POSIX standard can be applied to … .
a) any operating system
b) Mac OS
c) HP's HP-UX and IBM's AIX

Exercise 26. Put some key questions on text 4.

Exercise 27. Write out all Non-Finite forms of the verbs from text 4. State their forms and functions.

Exercise 28. Translate the word combinations below into Ukrainian.

A line of proprietary, graphical operating systems; all currently shipped Macintosh computers; the successor to; to be built on the technology; to be first released; a desktop-oriented version; to be usually referred to; the server edition; to be architecturally identical to; its desktop counterpart; to include work group management and administration software tools; to provide simplified access to; a mail transfer agent; a domain name server.

Text 5. Mac OS X

Mac OS X is a line of proprietary, graphical operating systems developed, marketed, and sold by Apple Inc. loaded on all currently shipped Macintosh computers. Mac OS X is the successor to the original Mac OS, which had been Apple's primary operating system since 1984. Unlike its predecessor, Mac OS X is a UNIX operating system built on technology that had been developed at NeXT through the second half of
the 1980s and up until Apple purchased the company in early 1997. The operating system was first released in 1999 as Mac OS X Server 1.0, with a desktop-oriented version Mac OS X v10.0 following in March 2001. Since then, five more distinct "end-user" and "server" editions of Mac OS X have been released, the most recent being Mac OS X v10.5, which was first made available in October 2007. Releases
of Mac OS X are named after big cats; Mac OS X v10.5 is usually referred to by Apple and users as "Leopard". The server edition, Mac OS X Server, is architecturally identical to its

desktop counterpart but usually runs on Apple's line of Macintosh server hardware. Mac OS X Server includes work group management
and administration software tools that provide simplified access to key network services, including a mail transfer agent, a Samba server, an LDAP server, a domain name server, and others.

Exercise 29. Compose five key questions of different kinds (general, special, alternative, disjunctive and subject) on text 5.

Exercise 30. Say whether the statements below are true or false. Correct the false ones.
1. Mac OS has a long history since it has come to the modern word of computer technologies.
2. Mac OS X is a line of proprietary, graphical operating systems developed, marketed, and sold by Cherry Inc., the latest of which is pre-loaded on all currently shipped Macintosh computers.
3. Mac OS X is a UNIX operating system built on technology that had been developed at NeXT through the second half of the 1980s and up until Apple purchased the company in 1997.
4. The operating system was first released in 1999 as Mac OS X Server 1.0, with a desktop-oriented version (Mac OS X v10.0) following in March 2001.
5. The server edition, Mac OS X Server, is identical to its business counterpart but usually runs on Apple's line of Macintosh server hardware.
6. Mac OS X Server includes work group management and administration hardware tools that provide simplified access to key network services.

Exercise 31. Compose a dialogue on "Mac OS X", using the word combinations given below. Mind the grammar form.
A real-time operating system, to include some small embedded systems, to be categorized by a technology, a large amount of the worldwide desktop market share, in the enterprise market, after more than five years of development work, to contain a large number of new features and architectural changes.

Exercise 32. Find some additional information and speak on:
1. The difference between system software and application software.
2. Graphical User Interface.
3. The most popular operating systems and the difference between them.
4. Why is Windows so popular?

UNIT 5. SOFTWARE ARCHITECTURE

Translate and study the basic vocabulary.

aggregate	
composition	
decompose	
decompositional	
eliminate	
emerge	
global control structure	
gross-level component	
interface point	
latency	
maintainability	
mapping	
partitioning	
performance	
platform independence	
refinable	
scaling	
specify	
synchronization	
throughput	
typed object	

Exercise 1. Write derivatives of the words below and explain their meanings.

Model: vary – variant – variety – various – variously Vary, develop, communicate, assign, maintain, inform, represent, function, part, depend, compose, connect, perform, explicit, define, exist, configure.

Exercise 2. Give Ukrainian equivalents for the following word combinations.

Means of communication between modules; representation of shared information; to go beyond the algorithms and data structures, to emerge as a new kind of problem; assignment of functionality to design elements; scaling and performance; partitioning strategy; discrete, nonoverlapping parts; a set of well-formedness constraints that must be satisfied by any architecture; to be decomposed, aggregated, or eliminated

in a concrete architecture; key design issues; life-cycle issues such as maintainability, extent of reuse, and platform independence.

Text 1. Software Architecture Strategies and Concepts

Software architecture is the study of the large-scale structure and performance of software systems. Important aspects of a system's architecture include the division of functions among system modules, the means of communication between modules, and the representation of shared information.

As the size and complexity of software systems increase, the design problem goes beyond the algorithms and data structures of the computation: designing and specifying the overall system structure emerges as a new kind of problem. Structural issues include gross organization and global control structure; protocols for communication, synchronization, and data access; assignment of functionality to design elements; composition of design elements; scaling and performance; and selection among design alternatives. This is the software architecture level of design.

The architecture consists of (a) a partitioning strategy and (b) a coordination strategy. The partitioning strategy leads to dividing the entire system in discrete, non-overlapping parts or components. The coordination strategy leads to explicitly defined interfaces between those parts. Software architecture is represented using the following concepts:

1. Component: An object with independent existence, e.g., a module, process, procedure, or variable.
2. Interface: A typed object that is a logical point of interaction between a component and its environment.
3. Connector: A typed object relating interface points, components, or both.
4. Configuration: A collection of constraints that wire objects into a specific architecture.

5. Mapping: A relation between the vocabularies and the formulas of an abstract and a concrete architecture. The formula mapping is required because the two architectures can be written in different styles.

6. Architectural style: A style consists of a vocabulary of design elements, a set of well-formedness* constraints that must be satisfied by any architecture written in the style, and a semantic interpretation of the connectors.

Components, interfaces, and connectors are treated as first-class objects- i.e., they have a name and they are refinable. Abstract architectural objects can be decomposed, aggregated, or eliminated in a concrete architecture. The semantics of components is not considered part of architecture, but the semantics of connectors is.

Software architecture is an important level of description for software systems. At this level of abstraction key design issues include gross-level decompositional components, protocols of interaction between those components, global system properties (such as throughput and latency), and life-cycle issues (such as maintainability, extent of reuse, and platform independence).

Exercise 3. Find in text 1 the English for:

дослідження великомасштабної структури та функціонування програмних систем; розподіл функцій між системними модулями; обсяг та складність програмних систем; проектування та деталізація загальної структури системи; новий тип завдання; організація на well-formedness – формальна правильність

макрорівні та структура глобального керування; протоколи зв'язку, синхронізації та доступу до даних; склад структурних компонентів; вибір між варіантами проектів; стратегії розподілу та координування; розподіл системи на дискретні частини або елементи, що не перетинаються; чітко визначені інтерфейси між частинами; незалежний об'єкт; модуль, процес, процедура, або змінна; логічна сутність взаємодії між елементом та його середовищем; сукупність обмежень, що пов'язують об'єкти в певну архітектуру; абстрактна та конкретна архітектури; відображення формул; формальні обмеження; ключові питання проектування; декомпозиційні компоненти на макрорівні; глобальні властивості системи; пропускна здатність і час очікування.

Exercise 4. Say whether the statements below are true or false. Correct the false ones.

1. Software architecture is the study of the small-scale and large-scale structures and performance of software systems.

2. Important aspects of a system's architecture include the division of functions among system modules, the means of communication between modules, and the representation of shared information.

3. The size and complexity of software systems decrease.

4. The design problem goes beyond the algorithms and data structures of the computation: designing and specifying the overall system structure emerges as a new kind of problem.

5. The architecture consists of (a) a partitioning strategy and (b) a communication strategy.

6. The partitioning strategy leads to dividing the entire system in discrete, overlapping parts or components.

7. The coordination strategy leads to explicitly defined interfaces between those parts.

8. Interfase is a typed object that is a logical point of interaction between components.

9. The formula mapping is required because the two architectures can be written in similar styles.

10. An architectural style consists of a vocabulary of design elements, a set of well-formedness constraints that must be satisfied by any architecture written in the style, and a semantic interpretation of the connectors.

11. Components, interfaces, and connectors are treated as first-class objects – i.e., they have a name and they are nonrefinable.

12. Abstract architectural objects can be decomposed, aggregated, or eliminated in a concrete architecture.

13. The semantics of components is considered part of architecture, as well as the semantics of connectors.

Exercise 5. Form all possible word combinations with the words from both columns. Translate them.

1) to include	a) a new kind of problem
2) to write	b) a partitioning strategy
3) to go beyond	c) formula mapping
4) to consist of	d) explicitly defined interfaces
5) to emerge as	e) division of functions
6) to lead to	f) part of architecture
7) to require	g) the algorithms and data structures
8) to be treated as	h) constraints

9) to be considered	i) in different styles
10) to satisfy	j) first-class objects

Exercise 6. Fill in the blanks with prepositions in, to, between, among, beyond, of, as, by, at, for where necessary.

1. Software architecture is the study … the large-scale structure and performance … software systems.

2. Important aspects … a system's architecture include the division … functions … system modules, the means … communication … modules, and the representation … shared information.

3. As the size and complexity … software systems increase, the design problem goes … the algorithms and data structures … the computation: designing and specifying the overall system structure emerges … a new kind … problem.

4. Structural issues include gross organization and global control structure; protocols … communication, synchronization, and data access; assignment … functionality … design elements; composition … design elements; scaling and performance; and selection … design alternatives.

5. The partitioning strategy leads …dividing the entire system … discrete parts.

6. Interface is a typed object that is a logical point … interaction … a component and its environment.

7. Architectural style consists … a vocabulary … design elements, a set … well-formedness constraints that must be satisfied … any architecture written … the style, and a semantic interpretation … the connectors.

8. Software architecture is an important level … description … software systems.

9. … the level … abstraction key design issues include gross-level decompositional components and protocols … interaction … those components.

Exercise 7. Fill in the blanks with proper terms (configuration, component, mapping, interface, coordination strategy, connector, partitioning strategy, software architecture) to complete the sentences.

1. _____ is a typed object relating interface points, components, or both.

2. _____is a strategy that leads to dividing the entire system in discrete, non-overlapping parts or components.

3. _____ is an object with independent existence, e.g., a module, process, procedure, or variable.

4. _____ is a collection of constraints that wire objects into a specific architecture.

5. _____is a relation between the vocabularies and the formulas of an abstract and a concrete architecture.

6. _____ is the study of the largescale structure and performance of software systems.

7. _____is a strategy that leads to explicitly defined interfaces between those parts.

8. _____ is a typed object that is a logical point of interaction between a component and its environment.

Exercise 8. Answer the questions on text 1.

1. What is software architecture?
2. What do important aspects of a system's architecture include?
3. Why does the design problem go beyond the algorithms and data structures of the computation?
4. What emerges as a new kind of problem?
5. What do structural issues include?
6. What strategies does the architecture consist of? What do they differ in?
7. What concepts is software architecture represented by?
8. What is a component/interface/ connector?
9. What is configuration/ mapping/an architectural style?
10. What kind of semantics is considered to be a part of architecture?
11. What do the key design issues of software architecture include?

Exercise 9. Put all possible questions to the sentences below.

1. Important aspects of a system's architecture include the division of functions among system modules, the means of communication between modules, and the representation of shared information.
2. As the size and complexity of software systems increase, the design problem goes beyond the algorithms and data structures of the computation.
3. Structural issues include gross organization and global control structure; protocols for communication, synchronization, and data access; assignment of functionality to design elements; composition of design elements; scaling and performance; and selection among design alternatives.

4. The architecture consists of a partitioning strategy and a coordination strategy.

5. The partitioning strategy leads to dividing the entire system in discrete, non-overlapping parts or components.

5. Software architecture is represented using the following concepts: a component, an interface, a connector, a configuration, mapping, an architectural style.

6. Components, interfaces, and connectors are treated as first-class objects.

7. Abstract architectural objects can be decomposed, aggregated, or eliminated in a concrete architecture.

8. The semantics of components is not considered part of architecture, but the semantics of connectors is.

Exercise 10. Translate into English.

1. Архітектура програмного забезпечення – це дослідження великомасштабної структури та функціонування програмних систем.

2. Важливі аспекти архітектури системи включають розподіл функцій між системними модулями, засоби зв'язку між модулями та спосіб представлення розподіленої інформації.

3. У зв'язку зі зростанням обсягу та складності програмних систем завдання проектування виходить за межі алгоритмів та структур даних обчислення – проектування та деталізація загальної структури системи постає як новий тип завдання.

4. Архітектурний рівень проектування програмного забезпечення включає такі структурні питання, як організація на макрорівні та структура глобального керування; протоколи зв'язку, синхронізації та доступу до даних; розподіл функціональності між структурними компонентами та їх склад; масштабування та функціонування; вибір варіанту проекту.

5. Архітектура складається зі стратегій розподілу та координування.

6. В архітектурі програмного забезпечення використовують такі поняття: компонент, інтерфейс, з'єднувальний елемент, конфігурація, відображення та архітектурний стиль.

7. Компонент – це незалежний об'єкт, наприклад, модуль, процес, процедура, або змінна. 8. Інтерфейс – це типізований об'єкт, що є пунктом здійснення взаємодії між компонентом та його середовищем.

9. З'єднувальний елемент є типізованим об'єктом, який пов'язує між собою інтерфейсні вузли, компоненти, або й те, й друге.

10. Конфігурація – це сукупність обмежень, що пов'язують об'єкти в певну архітектуру.

11. Відображення відтворює відношення між словниками та формулами абстрактної та конкретної архітектур.

12. Архітектурний стиль складається зі словника структурних компонентів, низки формальних обмежень, яким має відповідати будь-яка архітектура семантичної інтерпретації з'єднувальних елементів, написана в певному стилі.

13. Компоненти, інтерфейси та з'єднувальні елементи розглядають як об'єкти першого класу, тобто вони мають імена і підлягають деталізації.

14. Абстрактні архітектурні об'єкти можна розкладати, об'єднувати та видаляти в конкретній архітектурі.

15. Архітектура програмного забезпечення є важливим рівнем опису програмних систем.

16. На цьому рівні абстрагування ключові питання проектування включають макрорівневі декомпозиційні компоненти, протоколи взаємодії між цими компонентами, глобальні властивості системи (такі, як: пропускна здатність і час очікування) та питання життєвого циклу (такі, як: зручність супроводу, частотність повторного використання і незалежність від платформи).

Exercise 11. Write a summary of the text "Software Architecture".

Text 2. Essential Characteristics of Software Architecture

Software architecture usually refers to some combination of structural views of a system, with each view being a legitimate abstraction of the system with respect to certain criteria, that facilitate a particular type of planning or usage.

- Software architecture represents the structure of the software. This includes the structural arrangements of software components, and various static and dynamic interrelationships between these components.
- Software architecture is expressed using certain views, each of which serves a specific purpose. Each view is a specific abstraction of the architecture, for a specific purpose.
- Software architecture includes the principles behind design and evolution of the software. The following are some of the essential characteristics of architecture.
- Software architecture should represent a high-level view of the system revealing the structure, but hiding all implementation details. Specifically, it should reveal attributes such as responsibilities (of the constituents of the architecture), distribution, and deployment.

- Architecture should realize all the use case scenarios. While the use case model serves to record the functional requirements as seen by various actors, the architecture should enable the stakeholders of the software to walk through the scenarios of each use case. This guarantees that the structure as represented by the architecture meets the functional requirements.
- It should present other systemic views to all the stakeholders of the software. Examples are – a component view for the development team, a network-centric deployment view for the network and hardware team, and a distribution-centric deployment view for the installation team etc. However, how does the architecture look like? Some of the common representations of software architecture are as follows. As discussed below, none of these representations is complete.
- **High-level design:** A simplistic approach is to represent the architecture as a concise view of a high-level design of the software. However, design is an implementer's view of the software – a view that reveals how to arrive at the structure of the software. So a high-level design does not necessarily represent the architecture of the software.
- **Deployment:** This is the most common form of representations of architecture. In this view, the software is described in terms of how it is deployed across various platforms, and how these parts communicate with each other. Note that deployment view is only one of the possible views of architecture, and does not necessarily reveal the structure of the software. Also note that during the life-cycle of a software, the deployment could change with no or minimal changes to the structure of the software. This is one of the reasons that is driving distributed component based technologies.
- **Generic Technology Architectures:** In this form, software architecture is represented as a two-tier, three-tier or a multi-tier system. Note that architectures such as these, distributed (and layered) architectures such as COM or CORBA, or component based architectures such as MTS or EJB are generic and do not address the needs of the domain in which the software is to operate and evolve. However, these technology architectures provide the basis for developing domain specific application architectures. The architecture should at least present the following views of the software (in the order of the importance).
- **Logical or Conceptual View:** This view of the software represents various abstractions of the system and accounts for various use case scenarios. Using this view, one should be able to walk through these abstractions to realize the use case scenarios. In the case of distributed applications, some of these abstractions may directly map to distributed components.
- **Deployment View:** As pointed earlier, this view depicts how various parts of the software are deployed.

In both views, it is necessary to depict responsibilities of each of the parts of the architecture, static and/or dynamic dependencies between them, the nature of communication between the parts, etc. Additional views such as development views (to show how the application will be developed) and process views (to reveal threading, concurrency etc.) may also be considered if required. In general, software architecture is considered important as it serves as a means of mutual communication between the stakeholders of the software, allows to capture early design decisions, and lets the architecture be reused for similar systems in the same domain.

Exercise 12. Answer the questions on text 2.

1. What does software architecture usually refer to?
2. What does the structure of the software include?
3. How is software architecture expressed?
4. What are some of the essential characteristics of software architecture?
5. What are examples of systemic views presented to all stakeholders of the software?
6. What does a high-level design represent?
7. What does a deployment view show?
8. How is software architecture represented in generic technology architectures?
9. What are examples of generic technology architectures?
10. What does a logical or conceptual view present?
11. What is it necessary to depict in logical and deployment views?
12. What additional views may also be considered?
13. Why is software architecture considered important?

Exercise 13. Choose the right form of the verbs in brackets. Mind the sequence of tenses.

1. He says, “I … just … a class of problems!” (has solved; have solved; had solved).
2. They told us, “We … already …a high-level prototype”. (have built; had built; has built).
3. The students say, “We …just …proven patterns to solve problems”. (had used; were used; have used).
4. He said, “I … … … the system in compliance with the plan by next month”. (will have developed; would have developed).
5. They announce, “We … … our decisions to developers”. (have disseminated; had disseminated; will be disseminated).
6. He told us, “I … … … structural patterns by evening”. (would have identified; will have identified).

7. She said, “Application developers … already …available capabilities”. (has used; had used; have used).

8. We … …that he … just … the problems to be solved. (find out/have identified; found out/had identified; found out/has identified).

9. He asked me what … … me facilitate the standardization of services. (has helped; had helped; will help).

10. I didn’t know you … … your assignment. (have completed; had completed; has completed).

Exercise 14. Use the proper tense form and voice of the verbs in brackets.

1. The complexity of software systems (increase) significantly recently.
2. A new kind of problem (appear) before the designing the overall system structure.
3. By next year two architectures (write) in different styles.
4. Abstract architectural objects just (decompose).
5. He already (represent) the architecture as a concise view of a highlevel design of the software.
6. The deployment view just (reveal) the structure of the software.
7. I (finish) presenting the view of the software by next week.
8. The evolution of software (review) before his arrival.
9. By next week he (capture) design decisions.
10. You already (determine) the components to build the system?
11. He just (realize) the importance of past experiences for software architecture.
12. How long you (solve) this technical problem?
13. He (gain) enough breadth and depth in the relevant domain long before I did it.
14. My friend (not have) the capability to envision a software system before it was developed.
15. I hope you already (understand) the importance of software architects.
16. You (get) enough development and debugging skills by next interview?
17. Unfortunately many projects already (make) mistake of trying to impose a single partition in multiple component domains.
18. His irresponsible actions (lead) to problems in development before we had time to correct the mistakes.
19. The implementation of a complex functional feature (split) between two groups by next meeting.
20. Yesterday I knew that the performance (compromise).

21. Since recently we (use) the resulting models to plan the subsequent development activities.
22. You (determine) the purpose and specifications of software before you started developing a plan for a solution?
23. Software developers (design) a plan by next decade?
24. He (not analyze) yet the software requirements.
25. It just (let) us produce various models.
26. Software designers (evaluate) these models since last month.
27. He said that various alternative solutions and trade-offs (examine) before.
28. More sophisticated methods (apply) by next year.
29. We knew that a set of fundamental design concepts (evolve).
30. I'm sure that a good software architecture already (yield) a good return on investment.

Exercise 15. Put the verbs in brackets into an appropriate form of Perfect or Indefinite Tense.

1. Last time he (divide) the program structure both horizontally and vertically.
2. He (work) on designing modules last week.
3. He never (consider) many aspects in the design.
4. When you (manage) to maintain the software effectiveness?
5. Since last trial the components (test).
6. He already (design) the software with a resilience to low memory conditions.
7. You ever (achieve) these goals?
8. When you (choose) the default values for the parameters?
9. You already (enumerate) all design criteria?
10. Last week he (work) at multiple levels of abstraction.
11. Today I (know) a lot about Human Machine Interface.
12. A month ago he (buy) a new backlit display.
13. He (forget) to ask about data transfer rate when he was in the office.
14. We (study) local area networks since last month.
15. This week they (install) a full keypad.
16. When you (learn) about operating temperature?
17. I don't think he ever (hear) about Ethernet.
18. You already (design) the program?
19. When you last (change) your display?
20. I (learn) about this feature yesterday.

Exercise 16. Change the sentences into indirect speech.

1.The teacher asked, “Did you define the problem?”

2. He enquired,”Will you have finished your report by evening?

3. The professor told us, “We have already performed the major part of our work”.

4. My friend said to me, “I have just separated the interface from implementation”.

5. They said to us, “We have been trying to make it work for a long time”.

6. I asked my groupmate, “Have you already found your mistake?”

7. He said, “I have implemented the design before you did it.”

8. She asked, “How long have you been searching for the decision?”

9. The student said to the teacher,” I will have sent my report by Friday.”

10. He asked, “Did they understand their tasks last time?”

Exercise 17. Translate the word combinations below into Ukrainian.

To give an edge; envisioning a solution; to impart in you the ability to choose among various solutions available; to anticipate and solve technical problems; to gain enough breadth and depth in the relevant domain and technologies; to convince the people above and below them in the hierarchies; to be a vogue in the Software Development industry; to equate one’s success to that of the customer’s; to be a natural progression.

Exercise 17. Read, translate and entitle text 3.

Text 3

Software architecture is the blueprint of a software system. It provides an overview of the composition and functionality of the given software system. Just like a structural architect, a software architect needs to analyze the requirements, determine components that should be used to build the system, and support the project by guiding and solving problems all along the execution cycle. Architecting software is like planning for war. Past experiences are very useful here. Strategizing gives you an edge. Understanding of the domain provides you with capability to analyze the requirements and envisioning a solution. Exposure to various tools and technologies imparts in you the ability to choose among the various solutions available, and anticipate and solve technical problems. A person becomes a software architect when he has gained enough breadth and depth in the relevant domain and technologies, and has the capability to envision a software system before it is developed. Architects are needed for most of the Software Projects, more as the navigator for a sailing ship. Architects design the software system, guide the development team in implementing the system, and anticipate/diagnose problems, find/develop solutions to those problems. These days, most of

the organizations are realizing the importance of software architects, because with an architect you are no longer shooting in the dark.

Architects need to have good written/spoken communication since they have to convince the people above and below them in the hierarchies about their ideas effectively, strong development and debugging skills, domain and technology proficiency and appreciation, and customer's point of view can be very useful for an architect. Due to the maturity of Software Development practice, there are several predefined solutions available for certain problems called Patterns. Design and Architecture patterns are a vogue in the Software Development industry. They are the solutions for a recurring class of problems. Most of the time, they are the best possible solutions to those problems. Knowledge of patterns in their domain/technology areas is an added advantage for an architect. For a person, who is excited by new tools, technologies and domains, who is on the lookout for challenging problems to solve, who is capable of thought leadership, who equates his success to that of the customer's, the role of Software Architect is a natural progression.

Exercise 18. Find in text 3 the English for:

структура програмної системи; загальне уявлення про будову і функціональність програмної системи; інструктування і вирішення проблем впродовж усього циклу виконання; розуміння предметної області; можливе залучення різноманітних засобів і технологій; міцні навички у розробці і налагоджуванні; вправність і уміння добре розібратися як у предметній області, так і в техніці; завдяки розвиненій практиці розроблення програмного забезпечення; стандартні рішення; клас задач, що періодично постають перед розробником; нові інструменти, технології та сфери застосування; шукати складних проблем, що потребують розв'язання; здатний на інтелектуальне лідерство.

Exercise 19. Answer the questions on text 3.

1. What is software architecture?
2. What does software architecture provide?
3. What does a software architect need to do?
4. What aspects should he consider in architecting software?
5. When does a person become a software architect?
6. What is architects' job?
7. What skills do architects need to have?
8. What are patterns?
9. For what kind of person is the role of Software Architect a natural progression?

10. Do you have any experience in architecting software? Speak to your groupmates about it.

Exercise 20. Explain the meaning of the following words in English.

Software architecture, a blueprint, a software system, an overview, a software architect, a problem, experience, to give an edge, a domain, to envision, an exposure, a software project, shooting in the dark, debugging, a skill, technology proficiency, appreciation, maturity, predefined solution, a pattern, a vogue, Software Development industry, a recurring class of problems, to be on the lookout for, a challenging problem, a leadership, a success, a customer, a progression.

Exercise 21. Choose verbs among the following words. Read and translate the word.

Software, include, different, internal, relation, component, architecture, necessitate, domain, transform, exist, porting, availability, with, load, important, independent, specific, architectural, legislate, current, along, scheme, extend, basic, create, therefore, earlier, concerns, partition, transact.

Exercise 22. Give synonyms for the following words.

Require, basic, component, partition, complex, execution, in the end, plan, particular, combine, user, modern, along with, significant, address, the problem, cluster, initial, subsequently, specify, pertinent to.

Text 4. Architectural Structures

The structure of software in a component domain is created by a partition of software into components and their composition into an integrated whole. For every system it is necessary to determine which structures of software affect architecturally significant requirements and to group the requirements in such a way that each group is supported primarily by independent structures that exist in different component domains. One effective way to identify independent (or partly independent) requirements is by different stages of software life cycle with which they are concerned. A typical (though somewhat simplified) set of stages when different structures of software play major roles includes write time, build time, configuration time, upgrade time, start time, run time, and shutdown time. The most important software structure at write time is the structure of modules. Thus write time-related requirements, such as feature addition and evolution, porting, and diversification, are

addressed primarily by appropriate module structures that play a major role at write time. Similarly, start time-related requirements (such as order, presence, independent operation, and

failure modes) are addressed primarily by appropriate executable structures – the startup or shutdown unit or component. In addition, of course, run time-related requirements, such as performance or availability, are addressed by the structures of objects and execution threads – the domain of run-time software components. Many projects make the mistake of trying to impose a single partition

in multiple component domains, such as equating threads with objects, which are equated with modules, which in turn are equated with files. Such an approach never succeeds fully, and adjustments eventually must be made, but the damage of the initial intent is often hard to repair. This invariably leads to problems in development and occasionally in final products. In one case, implementation of a complex functional feature was split between two groups. Two functional clusters were defined, along with the necessary interfaces. Unnecessarily, the modules also ended up in different processes and had to interact at run time using slower interprocess

communication mechanisms. Another example involved a system that was partitioned into a set of distributed processes. The partition was motivated by considerations of required parallelism, availability, and fault tolerance. This partition was subsequently used to allocate additional functionality, which affected resource requirements and timing characteristics, violating the original design. As a cure, non-real-time functionality was allocated to new components. However, because the software architecture was identified with its process structure, these components became independent processes. Consequently, the components had complex interfaces and performance was compromised. Designing a software architecture must start with specific architectural concerns, specify the partition in different component domains, along with a scheme for integration and coordination of the parts, and explain how this specific partition and the corresponding integration of

the software address the specified architectural concerns. Examples of architectural concerns may include timeliness, capacity, availability, effective division of work, conformance to standards, use of existing parts, or controlled propagation of change. To address these concerns, different partitions may exist in different component domains. From the point of view of software reuse, architecture that separates concerns pertinent to different requirement and component domains also results in more reusable components. Therefore it is important to recognize multiple software existence planes with the associated component domains and independent partitions of software and their relations to different requirement domains.

Exercise 23. Put some key questions on text 4.

Exercise 24. Say whether the statements below are true or false. Correct the false ones.

1. A scheme of software in a component domain is created by a partition of software into components and their composition into an integrated whole.

2. One effective way to identify independent requirements is by different stages of software life cycle with which they are concerned.

3. A typical set of stages when different structures of software play major roles includes run time, write time, configuration time, build time, start time, upgrade time and shutdown time.

4. The most important software structure at write time is the structure of diagrams.

5. Many projects make the mistake of trying to impose a larger partition in multiple component domains, such as equating threads with objects.

6. Implementation of a complex functional feature was split between three groups.

7. Two functional clusters were defined, along with the necessary interfaces.

8. The modularities also ended up in different processes and had to interact at run time using slower interprocess communication mechanisms.

9. Because the software architecture was identified with its process structure, these components became independent processes.

10. Examples of component domains may include timeliness, capacity, availability, effective division of work, conformance to standards, use of existing parts, or controlled propagation of change.

11. From the point of view of software architecture, reuse that separates concerns pertinent to different requirement and component domains also results in more reusable components.

12. It is important to recognize multiple software existence planes with the associated component domains and independent partitions of software and their relations to integration of the software.

Exercise 25. Form all possible word combinations with the words from both columns. Translate them.

1) to be created by	a) appropriate module structures
2) to affect	b) considerations of required parallelism
3) to play	c) a partition of software into components
4) to be addressed by	d) functional clusters
5) to make	e) major roles
6) to lead to	f) complex interfaces
7. to define	g) more reusable components
8. to be motivated by	h) the mistake

9. to have	i) problems in development
10. to result in	j) architecturally significant requirements

Exercise 26. Decipher the abbreviations below.

MTS, EJB,COM, CORBA, VDU, SPARC, API, CLI, UI, USB.

Exercise 27. Compose a dialogue on "Architectural structures".

Exercise 28. Find some additional information and speak on:

1. Architectural strategies and concepts.
2. Essential characteristics of software architecture.
3. Structure of software in a component domain.

UNIT 6. SOFTWARE DESIGN

Translate and study the basic vocabulary.

software design	
software solution	
specification	
platform-dependent (platform-specific)	
software architectural design	
software detailed design	

Exercise 1. Choose nouns among the following words. Read and translate the word.
Precisely, define, concept, evaluate, object, subsequent, maintainability, sufficiently, procedure, fulfill, availability, implement, text, enable, item, examine, blueprint, analyze, integration, various, level, observable, interoperability, sophisticated, tolerance, internal, year.

Exercise 2. Give synonyms (a) and antonyms (b) for the following words:
a) software, purpose, architecture, designer, enable, consist, identify, various, precisely, activity, blueprint, trade-off, examine, fulfil, in the end, requirement, basis;
b) hardware, advantage, platform-independent, sufficiently, sophisticated, precisely, finally, top-level design, internal, employ, various.

Exercise 3. Write derivatives of the words below and explain their meanings.
Model: specify – specification – specifier – specific – specifically Specify, solve, process, depend, design, require, describe, produce, develop, precise, add, sufficient, vary, implement, maintain, operate, evaluate, available, refine, elaborate.

Exercise 4. Give Ukrainian equivalents for the following word combinations.

Software design; software solution; to determine the purpose and specifications of software; to employ designers; platform-independent or platform-specific; availability of the technology called for by the design; software requirements; to produce a description of the software internal structure; more precisely; to enable construction; software developing; to plan subsequent development activities; in addition to; a standard listing of software life cycle processes; to consist of two activities; to fit between software requirements analysis and software construction; identifying various components; to describe each component sufficiently.

Text 1. Software Design Activities

Software design is the process of defining the architecture, components, interfaces, and other characteristics of a system or component and planning for a software solution. After the purpose and specifications of software are determined, software developers will design or employ designers to develop a plan for a solution. Software design may be platform-independent or platform-specific, depending on the availability of the technology called for by the design. Viewed as a process, software design is the software engineering life cycle activity in which software requirements are analyzed in order to produce a description of the software internal structure that will serve as the basis for its construction. More precisely, software design (the result) must describe the software architecture and the interfaces between those components. It must also describe the components at a level of detail that enables their construction. Software design plays an important role in developing software: it allows software engineers to produce various models that form a kind of blueprint of the solution to be implemented. We can analyze and evaluate these models to determine whether or not they will allow us to fulfill the various requirements. We can also examine and evaluate various alternative solutions and trade-offs. Finally, we can use the resulting models to plan the subsequent development activities, in addition to using them as input and the starting point of construction and testing.

In a standard listing of software life cycle processes software design consists of two activities that fit between software requirements analysis and software construction:

- Software architectural design (sometimes called top-level design): describing software top-level structure, organization and identifying various components.
- Software detailed design: describing each component sufficiently to allow for its construction.

Exercise 5. Find in text 1 the English for:

процес визначення архітектури, компонентів, інтерфейсів та інших характеристик системи; планування програмного продукту; призначення та технічні вимоги до програмного забезпечення; розробник програмного забезпечення; бути незалежним чи залежним від платформи; технологія, потрібна для проектування; операція з розроблення життєвого циклу програмного забезпечення; описувати компоненти на рівні деталізування, що спрощує їх побудову; відігравати важливу роль в розробленні програмного забезпечення; структура рішення, яке потрібно реалізувати; оцінювати моделі; задовольняти різні вимоги; аналізувати та оцінювати різні варіанти рішень; стандартний перелік процесів життєвого циклу програмного забезпечення; проектування архітектури програмного забезпечення; детальне проектування програмного забезпечення.

Exercise 6. Say whether the statements below are true or false. Correct the false ones.

1. Software design is the process of defining the architecture, components, interfaces, and other characteristics of a system or component and planning for a hardware solution.
2. After the purpose and specifications of software are determined, software architects will design or employ designers to develop a plan for a solution.
3. Software design may be platform-independent or platform-specific, depending on the availability of the technology called for by the design.
4. Viewed as a process, software design is the software engineering life cycle activity in which software requirements are analyzed in order to produce a description of
the software's internal structure that will serve as the basis for its construction.
5. More precisely, software design (the result) must describe the software construction and the interfaces between those components.
6. Software design plays a minor role in developing software.
7. Software design allows software engineers to produce various models that form a kind of blueprint of the solution to be implemented.
8. In a standard listing of software life cycle processes software design consists of
three activities that fit between software requirements analysis and software construction.

Exercise 7. Form all possible word combinations with the words from both columns. Translate them.

1) to define	a) a plan for a solution
2) to employ	b) software requirements
3) to determine	c) components and interfaces

4) to develop	d) various models
5) to be	e) the purpose and specifications of software
6) to analyze	f) subsequent development activities
7) to produce	g) alternative solutions
8) to serve as	h) designers
9) to examine	i) platform-independent
10) to plan	j) the basis for construction

Exercise 8. Fill in the blanks with prepositions in, on, of, for, to, as, between, by where necessary.

1. Software design is the process … defining the architecture, components, interfaces, and other characteristics … a system or component and planning … a software solution.
2. After the purpose and specifications … software are determined, software developers will design or employ designers to develop a plan … a solution.
3. Software design may be platform-independent or platform-specific, depending … the availability … the technology called … … the design.
4. Viewed … a process, software design is the software engineering life cycle activity … which software requirements are analyzed … order to produce a description … the software internal structure that will serve … the basis … its construction.
5. Software design must also describe the components … a level … detail that enables their construction.
6. Software design plays an important role … developing software: it allows software engineers to produce various models that form a kind … blueprint … the solution to be implemented.
7. Finally, we can use the resulting models to plan the subsequent development activities, … addition … using them … input and the starting point … construction and testing.
8. … a standard listing … software life cycle processes software design consists … two activities that fit … software requirements analysis and software construction.

Exercise 9. Fill in the blanks with proper terms (component, software architectural design, software development, software design, software detailed design, software construction, interface, software architecture) to complete the sentences.

1. _____ is a typed object that is a logical point of interaction between a component and its environment.
2. _____ is the study of the large-scale structure and performance of software systems. Important aspects of a system's architecture include the division of functions among

system modules, the means of communication between modules, and the representation of shared information.

3. _____ is the process of defining the architecture, components, interfaces, and other characteristics of a system or component and planning for a software solution.

4. _____is the term that refers to the detailed creation of working, meaningful software through a combination of coding, verification, unit testing, integration testing, and debugging.

5. _____ is describing software's top-level structure, organization and identifying various components.

6. _____ is an object with independent existence, e.g., a module, process, procedure, or variable.

7. _____ is the process of writing and maintaining the source code that may include research, prototyping, modification, reuse, reengineering, maintenance, or any other activities that result in software products.

8. _____ is describing each component sufficiently to allow for its construction.

Exercise 10. Answer the questions on text 1.

1. What is software design?
2. What may software design depend on?
3. What kind of activity is software design?
4. What must software design describe?
5. What is the role of software design in developing software? 6. What can software models be used for?
7. What activities does software design consist of in a standard listing of software life cycle processes?
8. What is the difference between these activities?
9. Can you explain the difference between software architectural design and software detailed design?

Exercise 11. Put all possible questions to the sentences below.

1. After the purpose and specifications of software are determined, software developers will design or employ designers to develop a plan for a solution.
2. Software design may be platform-independent or platform-specific, depending on the availability of the technology called for by the design.
3. Viewed as a process, software design is the software engineering life cycle activity in which software requirements are analyzed in order to produce a description of the software's internal structure that will serve as the basis for its construction.

4. Software design must describe the software architecture and the interfaces between those components.
5. Software design plays an important role in developing software.
6. Software design allows software engineers to produce various models that form a kind of blueprint of the solution to be implemented.
7. We can analyze and evaluate these models to determine whether or not they will allow us to fulfill the various requirements.
8. In a standard listing of software life cycle processes software design consists of two activities that fit between software requirements analysis and software construction.

Exercise 12. Translate into English.

1. Проектування програмного забезпечення – це процес визначення архітектури, компонентів, інтерфейсів та інших атрибутів системи та планування програмного продукту.
2. Після визначення мети та технічних характеристик програмного забезпечення, розробники програмного забезпечення розробляють план створення продукту.
3. Проектування програмного забезпечення може бути незалежним або залежним від платформи, що зумовлено наявністю технології, необхідної для проекту.
4. Якщо проектування програмного забезпечення розглядати як процес, воно є операцією розроблення життєвого циклу програмного забезпечення, де аналізуються вимоги до програмного забезпечення для того, щоб описати його
внутрішню структуру, яка слугуватиме основою для конструювання.
5. Точніше кажучи, проект програмного забезпечення має описувати архітектуру програмного забезпечення та інтерфейси між його компонентами. 6. Проектування програмного забезпечення дозволяє інженерам з розроблення програмного забезпечення створювати різні моделі, що являють собою певну схему рішення, яке необхідно реалізувати.
7. Можна проаналізувати та оцінити ці моделі та визначити, чи вони будуть задовольняти різні вимоги.
8. Можна також дослідити та оцінити різні варіанти рішень, їх переваги та недоліки.
9. Зрештою можна використати кінцеві моделі для планування наступних етапів розроблення, а також щоб розпочати конструювання та тестування.
10. У стандартному переліку процесів життєвого циклу програмного забезпечення розроблення програмного забезпечення розпочинається з аналізу вимог до нього та закінчується конструюванням.
11. Проектування архітектури програмного забезпечення включає опис високорівневої структури програмного забезпечення, організацію та визначення різних компонентів.

12. Детальне проектування програмного забезпечення включає опис кожного компонента, достатній для його побудови.

Exercise 13. Write a summary of the text "Software Design Activities".

Exercise 14. Translate the word combinations below into Ukrainian.

Design concept; to apply sophisticated methods; a set of fundamental design concepts; an observable phenomenon; in order to retain the information which is relevant to a particular purpose; the process of elaboration; decomposing a macroscopic statement of function in a stepwise fashion; complementary concepts; to yield a good return on investment with respect to the desired outcome of the project; in terms of performance; quality, schedule and cost; to imply a hierarchy of control;

horizontal and vertical partitions; information hiding, to specify modules.

Text 2. Design Concepts

Design concepts provide a software designer with a foundation from which more sophisticated methods can be applied. A set of fundamental design concepts has evolved. They are:

1. Abstraction. Abstraction is the process or result of generalization by reducing the information content of a concept or an observable phenomenon, typically in order to retain the information which is relevant to a particular purpose.

2. Refinement. It is the process of elaboration. A hierarchy is developed by decomposing a macroscopic statement of function in a stepwise fashion until programming language statements are reached. In each step, one or several instructions of a given program are decomposed into more detailed instructions. Abstraction and refinement are complementary concepts.

3. Modularity. Software architecture is divided into components called modules.

4. Software Architecture. It refers to the overall structure of the software and the ways in which that structure provides conceptual integrity for a system. A good software architecture will yield a good return on investment with respect to the desired outcome of the project, e.g. in terms of performance, quality, schedule and cost.

5. Control Hierarchy. A program structure that represents the organization of program components and implies a hierarchy of control.

6. Structural Partitioning. The program structure can be divided both horizontally and vertically. Horizontal partitions define separate branches of modular hierarchy for each major program function. Vertical partitioning suggests that control and work should be distributed top down in the program structure.

7. Data Structure. It is a representation of the logical relationship among individual elements of data.

8. Software Procedure. It focuses on the processing of each module individually.

9. Information Hiding. Modules should be specified and designed so that information contained within a module is inaccessible to other modules that have no need for such information.

Exercise 15. Answer the questions on text 2.

1. What do design concepts provide a software designer with?
2. What are the fundamental design concepts?
3. What is abstraction?
4. What is refinement?
5. How is hierarchy developed?
6. What is software architecture divided into?
7. What is software architecture?
8. What will a good software architecture yield?
9. What is control hierarchy?
10. What does the horizontal and vertical partitioning of a structure suggest?
11. What is data structure?
12. What does a software procedure focus on?
13. What does information hiding imply?

Exercise 16. Change the following sentences to the Passive Voice.

1. Recently this firm has designed a new operating system.
2. He said that Sun Microsystems had developed Solaris as a more open option of SunOS.
3. Solaris will have got the largest share of the Internet market by next decade.
4. This new operating system has already offered a number of services to application programs and users.
5. We knew that they had released Windows XP in October 2001.
6. Each field of science will have imposed it's own requirements on the hardware by next year.
7. This kind of hardware has greatly facilitated connections between computers.
8. The teacher told us that the system case had provided a solid structural framework.
9. The student says that he will have presented his project by next month.
10. An operating system has relieved applications from having to control the hardware.
11. The teacher said that they designed the Macintosh operating system to be used on

Apple Macintosh computers.
12. He will have assigned the tasks by next week.
13. Microsoft has spent money to significant marketing research and development.
14. He said that after more than five years of development work, Microsoft released Windows Vista.
15. By next decade they will have replaced an old version by a new one.

Exercise 17. Change the following sentences to the Active Voice.
1. Since the last 25 years numerous distinct activities have been identified by researchers.
2. We found out that detailed design, unit testing, integration testing had been included in construction.
3. Substantial creativity and judgement have been involved in the process of construction.
4. I suppose that the latest enterprise information systems will have been adapted by some client service companies by next decade.
5. Reduced complexity has been recently achieved through emphasizing the creation of the code that is simple and readable.
6. They said that our tasks had already been defined by him.
7. A lot of efforts will have been made to reach an agreement by next time.
8. Significant constraints have been recently introduced in our activity.
9. I knew that some unanticipated actions had been observed by him.
10. Details of the software design will have been fleshed out by next meeting.
11. Many mathematical models have been classified as: linear and nonlinear.
12. The professor explained that the notations of the object-modelling technique and object-oriented software engineering had been synthesized by UML.
13. A standard modelling language will have been created by us by that time.
14. UML has been designed to be compatible with the leading object-oriented software development methods.
15. We learnt that they had been used extensively to describe the functionality of software system.

Exercise 18. Make the following sentences interrogative using the Passive Voice.
Model: Employees have already used new programs. – Have new programs been already used by employees?
1. He had illustrated his example before we asked him about it.
2. We will have designed a new program before they know it.
3. We have identified primary keys very quickly.
4. We have implemented data model by generating SQL.

5. They said that they had used the process model in structured analysis and design methods.
6. He will have carried out these experiments in his laboratory by next month.
7. We have used the model due to its ability to express concurrency.
8. He has used the task model to create high-level system.
9. They will have attached all components by the time you ask.
10. This device has performed all calculations at high speed.
11. They will have introduced significant changes by next week.
12. I knew that he had already determined the types of peripherals.
13. We will have run applications on the machine before they come.
14. He has downloaded the programs quite easily.
15. She has bought a new lightweight notebook to carry on business trip.

Exercise 19. Complete these sentences using the correct passive form of the verbs in brackets (Present, Past or Future Perfect Passive).

1. The teacher said that executive information systems (develop) as mainframe computer-based programs.
2. This computer applications (use) so far to satisfy senior executive's needs.
3. I'm sure that great success (achieve) by the company by next term.
4. Computer security problems always (consider) as a significant factor in the development of computer technology.
5. The lecturer explained us that a web server (hide) in a matchbox so that a few people could give an accurate count of the number they had in their homes.
6. By next decade good prevention measures (introduce) to stop unauthorized users from accessing any part of the computer system.
7. Valuable information and services just (protect) from publication by collective processes and mechanisms.
8. It is impossible to determine whether a disclosure or modifications (authorize) properly without authentication.
9. He informed me that information just (share) among companies.
10. I want to remind you that my computer system (secure) recently.
11. He mentioned that the threats for computer security (classify) into several categories.
12. A software flaw (discover) by specialists recently.
13. In two year's time the code from the exploit program (reuse) in Trojan horses.
14. I'm very sorry to say that his private conversation (eavesdrop).
15. I found out that the program (intend) to act as a system of eavesdropping protocols.

Exercise 20. Use the verbs in brackets in the Active or Passive form of Perfect Tenses.

1. The plan for construction of the system (create) before we knew about it.
2. I hope the architect (make) a right decision by next meeting.
3. We (not find) any rough mistakes in our research.
4. I found out that each interface in the system (mechanize) with more than one of the coordination protocols.
5. It just (ensure) conceptual integrity.
6. The need to retest components (reduce) by next experiment.
7. Up to now these instructions (reflect) choices about particular analysis.
8. He was sure that it (foster) the creation of the simplest solution to the system problem.
9. Lately all solutions (take into account) and a right one (choose).
10. The software (restore) by next week by a good team of specialists.
11. I think they already (define) their tasks.
12. Their achievements (evaluate) by next summit.
13. Some trade-offs already (find).
14. He announced that the implementation of a complex functional feature (split) between three groups.
15. Today I (review) multiple software existence planes.

Exercise 21. Translate into English.

1. Він повідомив, що кілька складних завдань було виконано цим процесором досить швидко.
2. Ця операційна система буде встановлена до вечора.
3. Нещодавно були розроблені нові версії цієї операційної системи.
4. Вони сказали, що результати їхніх досліджень вже оприлюднені.
5. Всі проблемні питання щодо вдосконалення програмної системи будуть вирішені до наступного місяця.
6. Архітектура програмного забезпечення була презентована досить детально.
7. Я дізнався, що в процес проектування були введені нові важливі компоненти.
8. Деякі критерії покращення процесу планування були щойно розглянуті.
9. Він сказав нам, що всі функціональні вимоги були враховані.
10. Значні переваги цього методу щойно знайшли підтвердження.

Text 3. Design Considerations

There are many aspects to consider in the design of a piece of software. The importance of each should reflect the goals the software is trying to achieve. Some of these aspects are:

- **Compatibility**. The software is able to operate with other products that are designed for interoperability with another product. For example, a piece of software may be backward-compatible with an older version of itself.
- **Extensibility**. New capabilities can be added to the software without major changes to the underlying architecture.
- **Fault-tolerance**. The software is resistant to and able to recover from component failure.
- **Maintainability**. The software can be restored to a specified condition within a specified period of time. For example, antivirus software may include the ability to periodically receive virus definition updates in order to maintain the software's effectiveness.
- **Modularity**. The resulting software comprises well defined, independent components. That leads to better maintainability. The components could be then implemented and tested in isolation before being integrated to form a desired software system. This allows division of work in a software development project.
- **Packaging**. Printed material such as the box and manuals should match the style designated for the target market and should enhance usability. All compatibility information should be visible on the outside of the package. All components required for use should be included in the package or specified as a requirement on the outside of the package.
- **Reliability**. The software is able to perform a required function under stated conditions for a specified period of time.
- **Reusability**. The software is able to add further features and modification with slight or no modification.
- **Robustness**. The software is able to operate under stress or tolerate unpredictable or invalid input. For example, it can be designed with a resilience to low memory conditions.
- **Security**. The software is able to withstand hostile acts and influences.
- **Usability**. The software user interface must be usable for its target user/audience. Default values for the parameters must be chosen so that they are a good choice for the majority of the users.

Exercise 22. Match the aspects to be considered in the software design with their explanations.

1. Security	a) The software can be restored to a specified condition within a specified period of time.
2. Reusability	b) New capabilities can be added to the software without major changes to the underlying architecture.
3. Robustness	c) The software is able to add further features and modification with slight or no modification.
4. Usability	d) The software is able to operate with other products that are designed for interoperability with another product.
5. Modularity	e) The software is resistant to and able to recover from component failure.
6. Maintainability	f) The software is able to perform a required function under stated conditions for a specified period of time.
7. Fault tolerance	g) The software is able to withstand hostile acts and influences.
8. Compatibility	h) Printed material such as the box and manuals should match the style designated for the target market and should enhance usability.
9. Packaging	i) The software is able to operate under stress or tolerate unpredictable or invalid input.
10. Extensibility	j) The software user interface must be usable for its target user/audience.
11. Reliability	k) The resulting software comprises well defined, independent components.

Exercise 29. Answer the questions on text 3.

1. What aspects should be considered in the design of a piece of software?
2. What is compatibility?
3. How can extensibility be explained?

4. What does fault tolerance mean?
5. What is maintainability?
6. What is modularity and what does it allow?
7. What components and information should be included in the package?
8. What are reliability and reusability?
9. Why is robustness important?
10. What acts is well designed software able to withstand?
11. What is the software user interface designed for?

Exercise 23. Compose a dialogue on different design aspects.

Text 4. Rules of Design

- Make sure that the problem is well-defined.
  - All design criteria, requirements, and constraints should be enumerated before a design is started.
  - This may require a "spiral model" approach.
- What comes before how.
  - i.e., define the service to be performed at every level of abstraction before deciding which structures should be used to realize the services.
- Separate orthogonal concerns.
  - Do not connect what is independent.
- Design external functionality before internal functionality.
  - First consider the solution as a black-box and decide how it should interact with its environment.
  - Then decide how the black-box can be internally organized. Likely it consists of smaller black-boxes that can be refined in a similar fashion.
- Keep it simple.
  - Fancy designs are buggier than simple ones; they are harder to implement, harder to verify and often less efficient.
  - Problems that appear complex are often just simple problems huddled together.
- Our job as designers is to identify the simpler problems, separate them, and then solve them individually.
- Work at multiple levels of abstraction.
  - Good designers must be able to move between various levels of abstraction quickly and easily.
- Design for extensibility.
  - A good design is "open-ended," i.e., easily extendible.

- A good design solves a class of problems rather than a single instance.
- Do not introduce what is immaterial.
- Do not restrict what is irrelevant.
- Before implementing a design, build a high-level prototype and verify that the design criteria are met.

➢ Details should depend upon abstractions.

- Abstractions should not depend upon details.
- Principle of Dependency Inversion.

➢ The granule of reuse is the same as the granule of release.

- Only components that are released through a tracking system can be effectively reused.

➢ Classes within a released component should share common closure.

- That is, if one needs to be changed, they all are likely to need to be changed.
- i.e., what affects one, affects all.

➢ Classes within a released component should be reused together.

- That is, it is impossible to separate the components from each other in order to reuse less than the total.

➢ The dependency structure for released components must be a DAG.

- There can be no cycles.

➢ Dependencies between released components must run in the direction of stability.

➢ The more stable a released component is, the more it must consist of abstract classes.

- A completely stable component should consist of nothing but abstract classes.

➢ Where possible, use proven patterns to solve design problems.

➢ When crossing between two different paradigms, build an interface layer that separates the two.

- Don't pollute one side with the paradigm of the other.

➢ Software entities (classes, modules, etc) should be open for extension, but closed for modification.

- The Open/Closed principle – Bertrand Meyer.

➢ Derived classes must be usable through the base class interface without the need for the user to know the difference.

- The Liskov Substitution Principle.

➢ Make it work correctly, then make it work fast.

- Implement the design, measure its performance, and if necessary, optimize it.

➢ Maintain consistency between representations.

- e.g., check that the final optimized implementation is equivalent to the high-level design that was verified.

➢ Don't skip the preceding rules!

- Clearly, this is the most frequently violated rule!!!
- Good designs can generally be distilled into a few key principles:
- Separate interface from implementation.
- Determine what is common and what is variable with an interface and an implementation.
- Allow substitution of variable implementations via a common interface.
- i.e., the “open/closed” principle.
- Dividing commonality from variability should be goal-oriented rather than exhaustive.
- Design is not simply the act of drawing a picture using a CASE tool or using graphical UML notation!!!
- Design is a fundamentally creative activity.

Exercise 24. Find in text 4 the English for:

переконатися, що завдання добре визначене; перераховувати критерії проектування, вимоги та обмеження; розглядати рішення як «чорну скриньку»; деталізувати подібним чином; містити більше помилок; прості завдання, об’єднані разом; кілька рівнів абстракції; легко розширюваний; відповідати критеріям проектування; принцип інверсії залежностей; орієнтований ациклічний граф; принцип підстановки «Лісков»; оцінювати функціонування; ігнорувати попередні правила; порушуване правило; система автоматизованого розроблення програм.

Exercise 25. Discuss the rules of design. Which of them are the most important/ more often used/ can be skipped? Can you add any other rules to those listed above?

Exercise 26. Prepare a presentation on one of the topics:

- “Software Design Fundamentals”
- “Software Design Context”
- “Key Issues in Software Design”
- “Concurrency”
- “Design Patterns”
- “Software Design Notations”
- “Software Design Strategies and Methods”
- “Function-Oriented -Structured Design”
- “Object-Oriented Design”
- “Data-Structure-Centered Design”
- “Component-Based Design”

VOCABULARY

A	
ability	можливість, здатність
abort	переривати, аварійно завершувати
abuse case	випадок неправильного / зловмисного використання
accept	приймати
access	(отри)мати доступ
access control	контроль доступу
accomplish	здійснювати; досягати
account	здійснювати облік
account for	враховувати
account for	пояснювати
account name	реєстраційне / облікове ім’я
accounting	бухгалтерський облік; фінансова звітність
accurately	точно, правильно
achieve	досягати
act as smth	виконувати функцію чогось
active matrix display	РК-екран / дисплей з активною матрицею
activities	діяльність
activity	дія, операція, робота, захід
activity	операція
activity diagram	діаграма діяльності (у UML – діаграма, на якій презентовані переходи потоку керування від однієї діяльності до іншої)
actor	актор (в мові UML – людина або пристрій, що взаємодіє з системою; його зображають у вигляді фігурки людини)

address	1) спрямовувати (зусилля); 2) (to) звертатися (до когось); адресувати
address	займатися (проблемою, питанням); братися (за щось)
adjustment	регулювання, налагодження, коригування
adopt	приймати, запроваджувати
advertisement	реклама, оголошення
advertising	реклама, рекламування
adware	безкоштовний рекламний продукт; вірус, що скачує рекламу та спам
affect	впливати
affect	впливати
affect	впливати
affiliate	філіал
affordable	доступний
aggregate	об’єднувати
aggregate level	аґреґований, сукупний, комплексний рівень
aggregation	аґреґування (механізм багаторазового використання одним об’єктом методів іншого об’єкта; цей механізм реалізує розподіл інтерфейсів)
aid	1) допомога; 2) допомагати, сприяти
aim	1) мета, намір, прагнення 2) прагнути
alarm condition	аварійна / небезпечна ситуація
alarm monitor	монітор аварійних сигналів
align	орієнтувати
all along	у/впродовж усього часу
allocation	виділення (ресурсу), розташування, розподіл
allow	дозволяти
allow for	дозволяти, забезпечувати

alphanumeric display	цифрово-, цифро-літерний дисплей
alter	змінювати
alternate	альтернативний, інший
alternating current (AC)	змінний струм
altogether	зовсім, цілком, цілковито
ambiguity	неоднозначність
analytic(al) database	аналітична база даних
animation	анімація
annoying	дратівний, набридливий, надокучливий
anomaly	відхилення від норми, аномалія
anonymity	анонімність
anonymize	робити анонімним, приховувати особисту інформацію
anticipate	передбачати
anticipate	передбачати, передчувати
appealing	привабливий
applet	прикладна міні-програма, утиліта
applicable	придатний, застосовний
application	прикладна програма
application programming interface (API)	інтерфейс прикладного програмування
apply (to)	1) стосуватися (чогось); 2) застосовувати(ся) (у чомусь / до чогось)
appreciation	розуміння, уміння добре розібратися (у чомусь)
approach	підхід
appropriate	відповідний; доречний; придатний
appropriate	відповідний, належний
approximately	приблизно

arbitrary	довільний
architectural risk analysis	аналіз архітектурного ризику
architecture pattern	архітектурний шаблон
archive	архівувати, поміщувати до архіву
arise	виникати, з’являтися, поставати
arrangement	організація
array of pixels	масив елементів зображення
artifact	артефакт, продукт розроблення, робочий продукт
as well as	так само, як; а також
asset	ресурс, цифровий об’єкт
assign	призначати, розподіляти
assignment	призначення, присвоєння, розподіл
associate	пов’язувати
associated	пов’язаний (із чимось), відповідний (до чогось)
Association for Computing Machinery	Асоціація з обчислювальної техніки
assume	припускати, вважати
assure	гарантувати; забезпечувати
attach	приєднувати, прикріплювати
attack amplifier	підсилювач атак
attack pattern	схема атаки
attempt	пробувати, робити спробу, намагатися
attribute	атрибут
auditing	ревізія / перевірка системи, сленг. аудитування
auditory	слуховий
auspice	сприяння
authentication	автентифікація, підтвердження автентичності, іден-

	тифікація
authentication	автентифікація, перевірка (справжності)
authorization	авторизація
automation	автоматика
availability	1) експлуатаційна готовність; 2) працездатність, неперервність
available	наявний, підхожий
B	
backdoor	лазівка, таємний вхід, чорний вхід, шлях обходу системи захист
background	1) походження, біографічні дані людини; 2) кваліфікація, освіта, (фахова) підготовка
backlit display	екран з підсвічуванням
backplane	з'єднувальна плата
backup	резервне копіювання
backup media	резервні носії
backward-compatible	зворотно сумісний, сумісний «назад» (такий, що не виключає використання попередніх версій чи модифікацій)
balanced serial interface	зрівноважений послідовний інтерфейс
bar	прямокутник (на блок-схемі)
bar chart	стовпчикова діаграма, гістограма
Basic Input / Output System (BIOS)	базова система введення/виведення
be on the lookout (for)	бути насторожі (щодо чогось), пильнувати, шукати (щось)
be regarded as	розглядатися як
be subject to smth	перебувати під впливом чогось, бути об'єктом чогось

be uniquely determined	однозначно визначатися
behaviour	поведінка, режим роботи
behaviour diagram	діаграма поведінки
behavioural perspective	поведінковий аспект
behavioural sciences	біхевіоризм
benefit	1) користь, вигода, прибуток; 2) мати прибуток
benefits	пільги та допомога, соцпакет
bibliographic	бібліографічний
bill	банкнота, купюра, рахунок
binary	двійковий файл
Bitmap	растр, растрове (бітове) зображення (графічне зображення у вигляді масиву точок на екрані)
bitmapped graphics	растрова графіка
black-box	«чорна скринька», пристрій або програма з невідомою внутрішньою структурою
blueprint	ескіз, креслення, проект
blueprint	схема, будова, структура
boolean	булівський, булів
boot	початкове завантаження (комп'ютера), самозавантаження
boot	завантажувати, виконувати початкове завантаження
boot firmware	мікропрограма початкового завантаження
bootable media	завантажувальний носій
bot (скор. від robot)	мережевий агент-робот (програма, що автономно вирішує завдання)
botnet	ботнет (мережа комп'ютерів, уражених програмою, яка підтримує зв'язок з її розробниками для того, щоб надсилати пошту без запиту, атакувати вебсайти)

boundary	межа
brain	мозок, розум
branch office	відділення, філія
breach	порушення
break (p.broke, pp. broken) into a system	«зламувати» систему (незаконно входити до системи)
breeding ground	родючий ґрунт
briefly	коротко, стисло
broad	широкий
broadly	у загальних рисах
buffer overflow	переповнення буферу
buggy	що містить значну кількість помилок, проф. «глючний»
buggy code	код, що містить помилки
build time	час компонування / побудови поточного варіанту програми
bundle	об'єднувати (у пакет)
bus topology	шинна топологія (топологія ЛОМ, у якій усі абоненти лінійно під'єднані до однієї магістралі (шини) пересилання даних)
business	(комерційна) організація
business activity	бізнес-операція
business modelling	бізнес-моделювання
business process	бізнес-процес, технологічний / виробничий процес
business workflow	потік бізнес-операцій
bypass	1) обхід; 2) обходити
C	
call for	вимагати, потребувати
call for	вимагати

capacity	продуктивність, пропускна здатність
caption	субтитр
capture	збирати (дані)
capture	фіксування зображення
capture	збирати (інформацію)
carefully	уважно, обережно
carelessness	неуважність, необережність, легковажність, недбальство, недотримання правил безпеки
cast (p., pp. cast)	тут: представляти
catch-all phrase	всеосяжна фраза
categorize	класифікувати
causal	причиновий
cause	1) причина; 2) спричиняти, викликати
CD/DVD-ROM drive	дисковод для компакт(DVD)-дисків
central processing unit (CPU)	центральний процесор (ЦП)
challenging	складний
change	1) зміна; 2) змінювати(ся)
chaotic	невпорядкований, хаотичний
char (character)	знак
chief executive officer	керівник, головна посадова особа, генеральний директор (компанії, підприємства)
chief executive officer	керівник, головна посадова особа, генеральний директор (компанії, підприємства)
child diagram	породжена діаграма, діаграма другого рівня
chipset	мікропроцесорний набір, чіпсет
choice	вибір

chunk (of data)	порція (даних)
circumvention	обхід
claim	стверджувати, заявляти
class diagram	діаграма класів (діаграма UML, яка презентує статичний погляд на систему з погляду класів і відношень між ними)
class-action law suit	колективний судовий позов
clearly	очевидно, зрозуміло
clip-art	політипаж, графічний фрагмент
closely	1) близько, тісно; 2) дуже
closure	закриття, припинення
cluster	сукупність, група, кластер
coarse	великомодульний
coding	програмування, кодування
cognitive psychology	когнітивна психологія
coin	монета
collaborate	співпрацювати
collaboration	співпраця, спільна праця
collage	колаж, комбінування різних елементів
collapse	1) руйнування; 2) вихід з ладу
collection	сукупність, набір
collection	сукупність
combination	поєднання
combination	поєднання
come to terms (with)	домовлятися (з кимось); приймати (чиїсь) умови
command line interface (CLI)	інтерфейс командного рядка, командний інтерфейс

Common Object Request Broker Architecture (CORBA)	загальна архітектура брокера / посередника запитів до об’єктів,стандарт CORBA (технологія побудови розподілених об’єктних програм, запропонована фірмою IBM)
commonality	спільність, уніфікованість
commonly	зазвичай, звичайно, переважно
communicate	передавати
communication	спілкування, зв’язок
communication diagram	діаграма комунікації (починаючи з UML 2.0; у UML 1x – це діаграма кооперації)
communication method	метод переда(ва)ння інформації
communications technologies	комунікаційні технології
Compact Disk Read-Only Memory (CD-ROM)	CD-ROM на компакт-диску
compared to	порівняно (з чимось)
compatibility	сумісність
compatible	сумісний, сполучний, схожий
complementary	доповняльний, додатковий
complete	закінчувати, завершувати
complexity	складність
complicated	складний
component diagram	діаграма компонентів
component failure	відмова елемента
Component Object Model (COM)	модель компонентних об’єктів Microsoft
component view	компонентне представлення

composite	комплекс, сукупність
composite structure diagram	діаграма композитних структур
composition	1) склад; 2) побудова, формування, утворення
comprise	містити в собі
compromise	1) порушувати (конфіденційність), розголошувати (таємну інформацію); 2) зламувати (систему)
computer crime	комп'ютерна злочинність, використання комп'ютера для скоєння правопорушень
computer security	комп'ютерна безпека
computer technology	обчислювальна техніка
computer-aided design	автоматизоване проектування
Computer-Aided Software Engineering (CASE)	система автоматизованого розроблення програм, CASE-технологія
computer-based training	комп'ютеризоване навчання
computing system	обчислювальна система
concept	поняття, концепція
conceptual view	концептуальне представлення
concern	проблема, питання, справа
concern	1) занепокоєння, турбота; 2) зацікавленість, інтерес; 3) участь
concerned with	пов'язаний (з чимось)
concise	короткий, стислий, лаконічний
conclude	робити висновок
concurrency	паралелізм
concurrency	1) паралелізм; 2) взаємна сумісність (властивість об'єктів в об'єктно-орієнтованому програмуванні)

concurrency	паралелізм
concurrently	одночасно, паралельно
confidentiality	конфіденційність, секретність
confidentiality	конфіденційність
configuration item	елемент конфігурації
configuration time	час конфігурування
conformance (to)	відповідність (до чогось)
conglomeration	конгломерат, суміш
conjure up	викликати в уяві
consecutive	послідовний
consent	згода, дозвіл
consequently	тому, в результаті (того, що)
consider	розглядати
consideration	міркування, підстава
consideration	міркування
consistency	логічність, послідовність, зв’язність
consistent	однаковий, єдиний, стабільний
constant	постійна величина, константа
constantly	постійно
constraint	обмеження
constraint	обмеження
content	зміст, інформаційне наповнення
content	зміст, інформаційне наповнення, контент
content security threat	загроза безпеці контенту
contribute	сприяти
control design	розрахунок системи автоматичного регулювання
control flow	потік керування
control structure	керівна конструкція

controller	контролер, пристрій керування
conversely	навпаки
conversely	навпаки
convert	перетворювати
convey	передавати, висловлювати
convey	передавати (думку), висловлювати (думку)
convince	переконувати
convinced	впевнений, переконаний
cool	охолоджувати
core	основний, центральний
corruption	1) зміна, викривлення (інформації, тексту); 2) пошкодження
count	підрахунок
counters	лічильні функції
cover	охоплювати
cover	охоплювати, містити в собі
covert	таємний, невидимий, секретний
craft	виготовляти, створювати
craft-like	професійний, майстерний
crash	1) аварійна відмова, збій; 2) зазнати аварії / збою
create	створювати
creation	створення
creativity	1) творчі здібності, здатність створювати; 2) креативність, потенціал інформації
credentials	мандат (обліковий запис з параметрами доступу користувача, сформований після його успішної автентифікації)

crippled	(не)придатний; бракований; зіпсований
criteria (pl. від criterion)	критерії
critical	важливий; необхідний
cross	мішаний, гібридний
cross	перетинати
cryptosystem	криптосистема
cure	ліки, засіб вирішення проблеми
currently	тепер, зараз, нині
customer	клієнт, замовник
customer database	клієнтська база даних
customization	пристосування під вимоги замовника
customize	налаштовувати, адаптувати
cybercriminal	кіберзлочинець, комп'ютерний злодій
D	
data architecture	архітектура (структура) даних
Data erasure	стирання даних, руйнування інформації
Data Loss Prevention (DLP)	попередження втрати даних
data structure	структура даних
data transfer rate	швидкість пересилання даних
data warehouse	сховище даних
database	база даних
database architecture	архітектура, конфігурація бази даних
database index	індекс (показник) бази даних
database management system (DBMS)	система керування базами даних (СКБД)
database schema	схема даних, логічна структура даних (зовнішній

	опис або діаграма заданої у СКБД структури запису; термін був запроваджений у 1971 р. для дворівневого підходу до опису структури БД)
Data-stealing malware	шкідливі програми, що крадуть дані / інформацію
day-to-day	повсякденний
de facto	фактично, де-факто
deadlock	взаємне блокування, проф. клінч
debate	дебати, дискусія
debugging	налагодження
deceive	обманювати
deception	обман, брехня
decision support system (DSS)	система підтримки рішень
decision variable	змінна рішення
decline	зниження, занепад
decompose	розкладати на складові, піддавати декомпозиції
decompositional	декомпозиційний
decrypt	розшифровувати
decryption	декодування, дешифрування
dedicate	призначати
default value	значення за замовчуванням
defeat	перемагати, ліквідовувати
defective	недосконалий, дефектний
define	визначати
definition	визначення, дефініція; тлумачення
deliberately	навмисно, свідомо
delivery vehicle	засіб доставлення

denial of service attack	атака типу «відмова обслуговування» (дія, що спричиняє відмову обслуговування законних користувачів)
denote	означати
denote	1) означати; 2) позначати
deny (access/resources)	відмовляти (у доступі/ресурсах); не давати доступитися до інформації; заперечувати
Department	відділ
depend (on)	залежати (від)
dependency	залежність
dependent	залежна величина
dependent quantity	залежна величина
depict	описувати, зображувати
deploy	розгортати, розташовувати
deploy	розміщувати
deployment	розміщення
deployment diagram	діаграма розгортання (діаграма UML, яка окреслює фізичну конфігурацію системи в термінах вузлів і з'єднань між ними, наприклад, з допомогою обчислювальної мережі)
deployment view	представлення розміщення (представлення системної архітектури, що виділяє вузли, котрі формують апаратну топологію системи)
deposit	1)депозит, рахунок у банку; 2) депозитний внесок
derived	похідний
design	призначати
design	1) проектувати; 2) призначати

design compartmentalization	роздробленість проектування
design pattern	конструктивний шаблон
desktop publishing	верстка друкованих видань на комп’ютері
destruction	руйнування, знищення
detailed	детальний, досконалий
detectable	такий, що може бути виявлений, виявний
detection	виявлення
determine	визначати
deterministic	детермінувальний; визначальний
development environment	середовище розроблення
development team	група розробників
device	пристрій, механізм
device-independent protocol	апаратно-незалежний протокол
devote	присвячувати
diagram	діаграма, схема
dialer	пристрій набору номерів
dial-up modem	модем комутованої лінії пересилання
differential equation	диференційне рівняння
Digital Versatile [Video] Disk Read-Only Memory (DVD-ROM)	DVD-ROM на компакт-диску
digitized	(п)оцифрований
dimensions	розміри
direct access attack	атака прямого доступу

direct current (DC)	постійний струм
directed acyclic graph (DAG)	орієнтований ациклічний граф
directly	безпосередньо, прямо, точно
disastrous	катастрофічний, згубний
disclosure	розкриття, викриття
disguise	маскувати
dishonest	нечесний, непорядний
disinfection	знезараження, дезінфекція
disk drive	дисковод
disk encryption hardware	апаратні засоби шифрування диску
disk encryption software	програмні засоби шифрування диску
dispense	видавати
distil	очищувати
distinct	ясний, виразний; чіткий
distinct from	відмінний (від чогось)
distinction	різниця, відмінність
distinguish	відрізняти
distorted	викривлений
distract	заважати, відволікати
distribute	розподіляти
distributed database	розподілена база даних
distributed parameter	розподілений параметр
distributed system	розподілена система
diversification	введення різноманітності, диверсифікація
divest	позбавляти (прав, повноважень, власності)

DNS – 1) Domain Name System	служба імен доменів;
DNS server	сервер служби імен доменів
document-text	документно-текстовий
dollar amount	сума в доларах
domain	(предметна) галузь, контекст (середовище, у якому повинна працювати програма)
Domain Name Server	сервер доменних імен
domain specific	що визначається галуззю застосування
dominate	переважати, панувати, домінувати
dongle	1) (електронний) захисний ключ-заглушка (під'єднується до порту введення–виведення); 2) електронний пристрій захисту (вбудований у комп'ютер)
download	завантажувати, скачувати
draw (p. drew, pp. drawn) (from)	добувати, діставати, брати (з чогось)
drill-down capabilities	можливості деталізування
drive	запускати, керувати
drive (p. drove, pp. driven)	рушати, приводити в рух, слугувати рушієм
drive-by download process	процес автоматичного завантаження непотрібної програми в комп'ютер
driver	драйвер, рушійна сила
driving record	особова картка водія
dropper	скидач
dumb terminal	простий (неінтелектуальний, «німий») термінал

E	
eavesdropping	підслуховування, перехоплення
edge	перевага
education	виховання, освіта, навчання
effort	зусилля; обсяг робіт, робота
elaboration	детальне розроблення, уточнення
electrical engineering	електротехніка
electromagnetic interference (EMI)	електромагнітні завади (наведення)
electronic data interchange	електронний обмін інформацією
electronics	електроніка, електронні схеми
eliminate	видаляти; усувати
elusive	1) невиразний, непевний; 2) важкодосяжний
embarrassing	такий, що заплутує (когось); ускладнює (щось)
embedded (built-in) operating system	вмонтована / вбудована операційна система
emerge	виникати, поставати
emerge	виникати, зароджуватись
emphasis	акцент, наголос
emphasis	акцент, наголос, особлива увага
emphasize	надавати особливого значення, акцентувати увагу
emphasize	надавати особливого значення (чомусь), підкреслювати (щось), робити акцент, акцентувати увагу, наголошувати (на чомусь)
employ	наймати (на роботу)
enable	робити можливим, уможливлювати, дозволяти
enable	давати можливість, уможливлювати, дозволяти

enable	дозволяти
enable	1) активувати, вмикати; 2) розблокувати; 3) уможливлювати, дозволяти
enclosure	1) корпус; 2) огорожа, загорожа; 3) обгороджування; 4) вкладення (вміст конверта)
encrypt	шифрувати, кодувати
encryption	кодування, шифрування
end-user database	база даних кінцевих користувачів
enforce	зміцнювати, посилювати
engineering approach	технічний підхід
engineering database	конструкторська база даних
engineering discipline	інженерна / технічна дисципліна
engineering drawing	технічне креслення
enhance	збільшувати, поліпшувати
Enhance	поліпшувати, збільшувати
enhance	поліпшувати, удосконалювати
ensure	забезпечувати
ensure	гарантувати, забезпечувати
Enterprise Java Beans (EJB)	специфікація EJB (на серверній частині стандартизує доступ до баз даних та до систем оброблення транзакцій, що важливо для корпоративних прикладних програм, оскільки забезпечує їх перенесення на інші платформи)
enterprise-wide	(загально)корпоративний, у масштабі підприємства
entertainment	розвага, забава
entire	цілий, повний, увесь
entitle (to)	давати право (на щось)
entity	об’єкт

entity	(логічний) об’єкт
entity relationship diagram	діаграма відношень логічних об’єктів-сутностей, ER-діаграма
entity-relation diagram (ERD)	діаграма відношень логічних об’єктів-сутностей, ER-діаграма
entry	елемент, позиція
environment	оточення, середовище
envision	уявляти, передбачати
equate	прирівнювати, ототожнювати
equate	пов’язувати
equation	рівняння
ergonomics	ергономіка, вивчення трудових процесів і умов праці
essential characteristics	істотні, важливі характеристики
essentially	по суті, посутньо, істотно, надзвичайно
establish	встановлювати
estimate	оцінювати, підраховувати
Ethernet	мережа (протокол, стандарт, технологія)
Ethernet small computer systems interface (SCSI)	інтерфейс малих обчислювальних систем, інтерфейс SCSI
evaluate	оцінювати, визначати
event flag	прапорець події
event handler	обробник подій
event occurrence	настання події
eventually	врешті решт, зрештою, з часом
evolution	1) розвиток; 2) розроблення
evolve	розвиватися
evolve	розвиватися, еволюціонувати

exactly	точно
examine	досліджувати, аналізувати
excited	захоплений
exclusive lock	блокування із забезпеченням взаємовиключального доступу (до набору даних)
executable	що виконується
executable software	виконуване програмне забезпечення
execution environment	1) середовище виконання; 2) елементи процесора
execution thread	потік виконання; потік завдань, що виконуються
Executive information system (EIS)	інформаційна система для керівників
exhaust	вичерпувати
exhaustive	вичерпний, повний, виснажливий
exogenous variable	екзогенна (зовнішня) змінна
expansion card (також expansion board, PC card)	карта / плата розширення
expect	очікувати
expensive	дорогий
experience	випробувати (на собі), відчувати; тут: споживати
expert system	експертна система
expertise	знання, кваліфікація, досвід
explicitly	ясно, точно; відкрито, недвозначно
explicitly	1) ясно, чітко; 2) явно
explicitly	очевидно, експіцитно, експліковано, зовні
explode	тут: розбиватися (на складові)
exploit	програма атаки / зламу
exponential decline	експонентне зниження / зменшення

expose	1) робити видимим, показувати; виставляти, показувати 2) піддавати дії
exposure (to)	можливе залучення (до чогось)
extend	розширювати, збільшувати, нарощувати
extensibility	розширюваність, можливість розширення (нарощення)
Extensible Markup Language (XML)	розширювана мова гіпертексту
extension	розширення
extensively	значно, дуже, широко
extent	ступінь, міра, обсяг, величина
extent	ступінь, міра
external perspective	зовнішній аспект
extract	вибирати, отримувати
extreme programming	екстремальне програмування
F	
facet	аспект, сторона
facilitate	полегшувати
facilities	1) засоби; 2) можливості
facilities	можливості
failure mode	стан відмови
faint	слабкий
fairly	досить, доволі
fake	підроблювати, дурити
fan	вентилятор
fancy	незвичайний
fault	пошкодження, дефект, збій
fault tolerance	відмовостійкість
fault-tolerance	відмовостійкість

feature	властивість, характеристика, особливість
ferret out	розшукувати, знаходити
fidelity	точність
field	галузь, сфера діяльності
file system	файлова система
File Transfer Protocol (FTP)	протокол передання файлів, протокол FTP
financial database	база даних для оброблення фінансової інформації
financial dealings	фінансові оборудки
fine-grained	дрібномодульний
firmware	вмонтоване ПЗ, програмно-апаратні засоби
fit between	розташовуватися між, бути на перетині
fix	1) визначати; 2) зафіксовувати, закріплювати
fix a bug	виправити помилку (усунути збій)
flat panel display (FPD)	плоскопанельний дисплей
flaw	дефект
flesh out	конкретизувати
flexibility	гнучкість
flood	1) повінь, потік непотрібної інформації, розм. «флуд»; 2) наповнювати, затоплювати
flow of control	потік керування
flush	1) очищувати (наприклад, вміст буферів оперативної пам’яті від старих даних); 2) скидати (вміст кеша або буфера на диск)
foreign key	зовнішній ключ (у базах даних)
formatting	подання (інформації) у форматі
fortunately	на щастя
foundation	фундамент, підвалина, основа

fractal	фрактал (геометрична форма, яка може бути розбита на окремі частини, котрі є приблизними зменшеними копіями цілого; фрактали описують такі об’єкти реального світу, як: гори, контури берегів, хмари тощо)
fragility	вразливість
frame	кадр
framework	структура
fraud	шахрайство
freehand drawing	малювання рукою
frequently	часто
fulfill	виконувати, задовольняти (вимоги)
full keypad	повна допоміжна клавіатура
fundamentals	основи
funds	гроші, кошти, сума
furthermore	до того ж, крім того, більш того
fuse	комбінувати, поєднувати
G	
gain	здобувати, діставати, отримувати
game theory	теорія ігор
gap	прогалина (у чомусь), брак (чогось)
gas plasma display	газорозрядний дисплей, плазмовий дисплей
gas plasma technology	газорозрядна технологія
generalization	узагальнення
general-purpose	універсальний, загального призначення
generic	узагальнений
generic	типовий, стандартний, звичайний
glitch	програмна помилка, розм. «глюк»
global control structure	структура глобального керування

go about	займатися (чимось), братися / взятися (до чогось)
go beyond	виходити за межі
govern	керувати
granularity	рівень модульності (системи), «гранулярність»
granule	гранула, частинка
graphic display	графічний дисплей
graphical user interface	графічний інтерфейс користувача
graphics	графіка
grid	решітка, сітка
gross	великий, макроскопічний
gross-level component	макрорівневий компонент
groundwork	основа, база, фундамент
H	
hacking	хакерство
hand-held computer	кишеньковий комп'ютер
handle	1) керувати; 2) мати справу, займатися (проблемою)
hand-produced material	матеріал, що виробляється ручним способом
hard disk	жорсткий диск, вінчестер
hard drive capacity	обсяг жорсткого диску
hard drive, hard disk	жорсткий диск, вінчестер
hardware	апаратне, технічне забезпечення або оснащення (на відміну від програмного); елементи комп'ютерів; сл. «залізо»
hardware team	група розробників апаратного забезпечення
hardware-based security	апаратний захист; захист, що забезпечують апаратні засоби

harmful	шкідливий
harsh	жорсткий, суворий
heat sink	тепловідвідний радіатор (що його застосовують для запобігання перегріву потужних інтегральних схем)
hence	тому, звідси
heterogeneous	гетерогенний, неоднорідний
hide (p. hid, pp. hidden)	ховати
hierarchically	ієрархічно
highlight	підкреслювати, зосереджувати увагу (на чомусь), наголошувати (щось)
highlight	підкреслювати, виділяти, акцентувати увагу
hinge on sth	залежати (від чогось), розм. упиратися (у щось)
historical data	ретроспективні дані (у базах даних – сукупність даних, отриманих за тривалий період, час)
hit (p., pp. hit)	завдавати шкоди, уражати
homogeneous	однорідний, гомогенний
hook (into)	підмикатися (до чогось)
host	головна система, хост-система
host	містити
hostile	ворожий
huddle (together)	збирати(ся) (разом)
human error	суб’єктивна помилка, помилка людини
Human–computer interaction (HCI)	взаємодія людини з машиною
human-machine interaction engineering	технологія взаємодії людини з машиною

Human-Machine Interface (HMI)	інтерфейс «людина – машина», людино-машинний інтерфейс
hybrid database	гібридна база даних, база даних зі змішаною («гібридною») структурою
hypermedia	гіпермедіа, гіперсередовище (розширений, порівняно з гіпертекстом, метод організації мультимедіа-інформації, у якому, крім тексту, підтримуються перехресні посилання з іншими типами даних (відео, графіка, звук)
hypermedia database	гіпермедійна база даних
hypertext	гіпертекст (1) технологія, що забезпечує пошук заданих тем у текстових масивах; пошук забезпечується включенням у тексти спеціальних покажчиків, що звуться гіпертекстовими посиланнями (hyperlinks) 2) текст, що має такі гіперпосилання)
hypothesis	гіпотеза
I	
I/O parallel peripheral bus	паралельна периферійна шина введення-виведення
I/O port	порт введення–виведення
ID (identifier)	ідентифікатор
identical	однаковий, тотожний, ідентичний

identical	однаковий, тотожний, ідентичний
identify	встановлювати, виявляти, визначати
identity	ідентичність
identity	1) тотожність, ідентичність; 2) справжність, істинність; 3) індивідуальність, особа
if/whether	чи
image	зображення
imitate	імітувати
immaterial	нематеріальний, неістотний
immediately	негайно, невідкладно
immovable	нерухомий, незмінний
impart	наділяти, надавати
impedance mismatch	розузгодженість інтерфейсів (розбіжність між інтерфейсами об'єкта бази даних)
implement	реалізувати, здійснювати, забезпечувати
implement	реалізовувати
implementation details	деталі реалізації
implication	підтекст, зміст; те, що мають на увазі
implicitly	неочевидно, імпліцитно, всередині
imply	означати, передбачати, мати на увазі
impose (a constraint)	накладати (обмеження)
improvement	поліпшення, в/удосконалення
in a fashion	певним чином, певною мірою
in diverse ways	різними способами, по-різному
in loose terms	у загальних рисах; невизначено
in terms of	в показниках, в одиницях, в перерахунку (на щось)
in terms of	через, у вигляді

in turn	почергово, послідовно, своєю чергою
inch	дюйм
incorporate	об’єднувати, містити у своєму складі
independent operation	автономне (незалежне) функціонування
index of performance	показник ефективності / продуктивності
indirect attack	непряма / опосередкована атака
industrial control panel	панель керування, панель приладів
industry	індустрія, галузь
Infectious	інфікований, заразний
inference rules	правила виведення (висновків в експертних системах)
infiltrate	просочуватися, проникати
information and communication technology (ICT)	інформаційно-комунікаційні технології
information engineering	інформаційна інженерія, інфотехніка
information hiding	приховування інформації
information system (IS)	інформаційна система
inherently	за своєю сутністю, у своїй основі; від природи
inheritance	1) успадкування; 2) спадщина; 3) спадок
inject	впорскувати, вводити, впускати
innocuous	безпечний, нешкідливий
input variable	вхідна змінна
input/output (I/O) interface	інтерфейс введення-виведення
installation team	група встановлення
instead (of)	замість
instrument	прилад
integer	ціле число

integer number	ціле число
integrate	об’єднувати, інтегрувати
integration	інтеграція, поєднання
integration testing	тестування взаємодії компонентів системи
integrity	цілісність (даних)
intend	призначати, планувати, мати намір
intend	призначати
intent	намір
intent	намір, мета
interaction	взаємодія
interaction diagram	діаграма взаємодії (загальна назва діаграм UML, на яких презентований динамічний погляд на систему з погляду об’єктів і повідомлень, якими вони обмінюються; практично використовується або діаграма кооперації, або діаграма послідовності)
interaction overview diagram	діаграма огляду взаємодії
interactive aspects	інтерактивні аспекти
interactivity	інтерактивність
interchange	1) обмінювати(ся); 2) переставляти, міняти місцями
interest rate	ставка банківського відсотка
interface	інтерфейс, взаємодія, взаємозв’язок
interface point	інтерфейсний вузол
intermediate	проміжний
internal	внутрішній
internal bus	внутрішня шина

International Organization for Standardization (ISO)	Міжнародна організація зі стандартизації
Internet Relay Chat	система діалогового спілкування Інтернетом
interoperability	функціональна сумісність, можливість взаємодії (програмних та апаратних виробів різних постав-ників)
interplay	взаємодія
interpret	тлумачити, пояснювати, інтерпретувати
interprocess communication mechanism	механізм взаємодії між процесами
interrelate	1) бути взаємопов'язаним; 2) взаємодіяти
interrupt service routine	підпрограма оброблення переривання
intersection	перетин, точка, лінія перетину
intruder	особа, що незаконно вдерлася; зламник
Intrusion Detection System (IDS)	система виявлення (мережевих) атак
intrusive	набридливий, надокучливий
invalid	неправильний, недійсний, помилковий
invariably	неодмінно
inventory	1) інвентаризація; переоблік; 2) наявні товари; товарно-матеріальні запаси
inventory	інвентаризація, облік товару
inventory	інвентаризація, облік товару
inversion	інверсія, зворотне перетворення
investigation	розслідування
invoice	рахунок-фактура

invoke	викликати, запускати, активізувати програму
involve	включати в себе, передбачати
involve	передбачати, бути пов’язаним
involved	складний, заплутаний (механізм)
irreversibility	незворотність
isolation	ізольованість
issue	питання, проблема
issue	проблема, питання
iterative	ітеративний; той, що повторюється, повторюваний
J	
judgement	судження, думка, оцінка
judgement	1) судження, думка, погляд, оцінка; 2) здоровий глузд, розсудливість
K	
keep track of	відслідковувати
kernel	ядро (частина операційної системи, що виконує найбільш важливі завдання)
key drive	флеш-пам’ять
key entry	клавішне введення
keyboard	клавіатура
keylogger	логер клавіатури (програма чи апаратний пристрій, що реєструє кожне натискання клавіш на клавіатурі комп’ютера)
keystroke	натискання клавіши чи кнопки
knob	куляста ручка, кнопка
L	

lack	бракувати, не вистачати
land information system	ландшафтна інформаційна система
large-scale computerized system	велика обчислювальна система
latency	час очікування, затримка
launch	запускати, починати
lay (p., pp. laid)	класти
layered	багаторівневий, багатошаровий
leakage protection	захист від втрати
legacy system	успадкована система (система, що не задовольняє вимог, але використовується через складність її заміни)
legitimate	розумний, прийнятний, обґрунтований
legitimately	законно, обґрунтовано
level of detail	рівень деталізації
lever	важіль керування, регулювання, ручка, руків'я
life-span	довговічність
linear	лінійний
linear	лінійний
linearity	лінійність
linearization	лінеаризація
link	з'єднувати, зв'язувати, сполучати
liquid crystal solution	рідкокристалічний розчин
liquid-crystal display (LCD)	рідкокристалічний дисплей / РК-дисплей
listing	перелік
live performance	вистава наживо
live presentation	презентація наживо

load	завантажувати
local area network (LAN)	локальна (обчислювальна) мережа, ЛОМ
locally	локально, у певних межах або масштабах; тут: у незначному колі
lock	замикати, блокувати
lock (out)	блокувати, відмикати
log	журнал реєстрації
log in	входити в систему, реєструватися
log in	реєструватися в системі / мережі
log out	виходити із системи / мережі
logical grouping	логічна група
logical view	логічне представлення
long-range	тривалий
loop diagram	контурна схема
loop gain	коефіцієнт підсилення замкненого ланцюга
low(high)-level design	низько-, високорівневий проект
lumped parameter	зосереджений параметр
M	
machinery	устаткування (механічне), обладнання
main memory	оперативна (головна) пам’ять
maintain control	підтримувати контроль
maintainability	1) відновлюваність; 2) зручність супроводу (програмного забезпечення)
maintenance	супровід, підтримка, експлуатація, технічне обслуговування
maintenance	технічне обслуговування
major	основний, головний

majority	більшість
make sure	переконатися, пересвідчитися
malfunction	1) несправна робота, неправильне функціонування; 2) неправильно працювати, не спрацьовувати
malware	шкідливі програми
manage	керувати
management	управління, керування
management information system	управлінська / адміністративна інформаційна система
management reporting system	система управлінської звітності
manipulate	маніпулювати, керувати, поводитися (з машиною)
manipulate	маніпулювати
manual	посібник; довідник, покажчик; підручник
manufacturing database	технологічна база даних
manufacturing plant	виробництво, підприємство
map	відображати(ся)
map point	план / схема пунктів / координат
mapping	відображення
market research	вивчення кон'юнктури, стану ринку
market share	питома вага даного товару на ринку
master	головний, провідний
master system	головна, центральна система
mastermind a ring	керувати злочинним угрупуванням
match	відповідність, збіг
maturity	розвиненість
meaningful	значущий
measure	1) міра; 2) вимірювати, оцінювати, визначати

measure	1) показник, критерій; 2) вимірювати
media	1) будь-яка форма інформації (аудіо-, відеоінформація, анімація тощо) 2) аудіовізуальне середовище
media control tools	засоби керування аудіовізуальним середовищем
media file	мультимедійний файл, медіа файл
media player	медіаплеєр
mediate	бути посередником, бути сполучною ланкою
mediating construct	проміжний структурний компонент
medium (pl. media)	1) засіб 2) середовище 3) носій (інформації)
meet the requirements	відповідати вимогам
metadata	мета дані (дані з описом інших даних)
metamodel	метамодель, модель високого рівня
meteorology	метеорологія
Microsoft Transaction Server (MTS)	сервер транзакцій корпорації Microsoft
minimize	мінімізувати, зменшувати
miscreant	злодій, негідник
misuse	неправильне застосування / використання
mitigation	пом’якшення (згубних наслідків)
modularity	модульність
monetize	перетворювати на гроші
monitor	монітор
monochrome	монохромний, одноколірний, однотонний
motherboard	материнська плата
mouse	миша
move data	пересилати дані / інформацію
msec (millisecond)	мілісекунда
multidisciplinary	політематичний; багатопрофільний

multimedia	мультимедійні засоби, мультимедіа
multimedia application	1) мультимедійна програма; 2) застосування мультимедійних засобів
multiple	багатократний
multitasking	багатозадачність
multi-tasking operating system	багатозадачна операційна система
multi-user operating system	багатокористувацька операційна система
N	
N	
named constant	іменована константа
National Electronics Manufacturers Association (NEMA)	Національ- на асоціація виробників електроустаткування, асоціація NEMA
natural science	природознавство; одна з природничих наук (фізика, хімія)
navigational control	навігаційний контроль
navigator	штурман, навігатор
networkable	такий, що під’єднується до комп’ютерної мережі
network-centric deployment view	мережецентричне представлення розміщення
networking	1) робота / спілкування в мережі; мережевий режим
node	вузол
non-linear	нелінійний
nonrepudiation	неможливість заперечення авторства
notation	система числення, система позначень, запис

notation	система позначень
Notation	нотація, система позначень, набір символів і правил для запису синтаксису
numerous	численний
object database model	модель об’єктної бази даних
object diagram	діаграма об’єктів
Object Management Group (OMG)	група керування об’єктами
object oriented paradigm	об’єктно-орієнтована парадигма
object-enhanced	об’єктно-розширений
objective	мета
objective	мета
objective	мета
objective function	цільова функція
Object-modellingtechnique (OMT)	метод об’єктного моделювання
object-oriented design	об’єктно-орієнтоване проектування
object-oriented programming	об’єктно-орієнтоване програмування
Object-oriented software engineering (OOSE)	об’єктно-орієнтована програмотехніка
obscure	приховувати, робити непомітним
observable	спостережуваний
occasionally	іноді, час від часу
occur	траплятися, відбуватися

offer	пропонувати
offer	пропонувати
office	управління уповноваженого з питань інформації
office activity	офісна операція, вид офісної діяльності
office information system	офісна інформаційна система
off-shoring	практика переміщення робочої бази компанії в іншу країну
on-the-fly encryption	оперативне шифрування, шифрування в реальному часі
open-ended	з можливістю розширення, розширюваний
operate	керувати, управляти
operating humidity	робоча вологість
operating system	операційна система
operating system call	виклик операційної системи
operating temperature	робоча температура
operational	операційний, оперативний
operational	експлуатаційний
operational data	1) робочі дані; 2) дані щодо функціонування системи, експлуатаційні дані
operational database	операційна база даних
operational decision	оперативне рішення
operational security	безпека в експлуатації
operational workflow	потік функціональних операцій
operator interface terminal	термінал операторського інтерфейсу, модуль зв’язку оператора з об’єктом
optimize	оптимізувати
option	варіант, версія, опція
orderline	кількість найменувань у замовленні

organize	організовувати, створювати, впорядковувати
out of date	застарілий
outbreak	атака
output variable	вихідна змінна
outsource	1) віддавати роботу стороннім, переводити виробництво в інший регіон; 2) залучати зовнішній ресурс
overall	повний, (у)весь, загальний
overhead	1) службові, протокольні сигнали або дані; 2) витрати обчислювальних ресурсів; 3) накладні витрати
overlap	перекривати(ся), частково збігатися
overlap	перекривати(ся), частково збігатися
override	1) перевизначення; 2) обхід; 3) скасування (команди)
overtake	1) наздогнати, надолужити; 2) випередити
overview	загальне уявлення, загальна картина
overwriting	перезапис
overwriting	перезапис
owner	власник
ownership	право власності
P	
package diagram	діаграма пакетів
packaging	пакування
packet queue	черга пакетів
page layout program	програма компонування сторінок
paradigm	1) парадигма; 2) зразок, еталон
parent diagram	породжувальна діаграма, діаграма першого рівня
partial	частковий
particular	конкретний, даний

particular	конкретний
particularly	дуже, надзвичайно, особливо
particularly	1) особливо; 2) зокрема
partitioning	розподіл, декомпозиція
party	сторона, суб’єкт, учасник
passive matrix display	РК-екран / дисплей з пасивною матрицею
PAT = PaT= P&T	1) (picture and text) – «малюнок і текст», режим або функція PAT 2) program association table – таблиця з перерахуванням програм потоку та їх ідентифікація
pattern	шаблон
payload	інформаційне наповнення
payment	сплата, платіж
penetrate	проникати
penetration testing	випробування на можливість проникнення до системи, тест на захист від несанкціонованого доступу
perceivable	відчутний
perform	виконувати
performance	1) робота, функціонування; експлуатаційні власти-вості; 2) ККД
performance specifications	1) експлуатаційні / робочі характеристики; 2) вимоги до характеристик
performer	виконавець
perhaps	можливо
permeate	проходити крізь, проникати
persist	залишатися, зберігатися, продовжувати існувати
personal identification number (PIN)	особистий ідентифікаційний номер, ПІН-код
personnel database	база даних персоналу, кадрів

perspective	ракурс, точка зору, аспект
perspective	бачення, концепція, точка зору
pertinent to sth	такий, що стосується чогось, відповідний
phishing	фішинг (різновид Інтернет-шахрайства – випитування конфіденційної інформації з допомогою запитів, що мають вигляд офіційних листів)
phrase	висловлювати
physically	фізично
pie chart	секторна, кругова діаграма
plain text	незашифрований / незакодований текст
plant	1) встановлювати, розміщувати; 2) ховати
platform independence	незалежність від платформи
platform-dependent (platform-specific)	залежний від платформи
plausible	переконливий, виправданий
pleasing	приємний
point	пункт
point data	координати точок, дані про координати
poisoning	тут: зміна, псування, викривлення
polarizing material	поляризаційний матеріал
policies	заходи
polish	шліфувати; детально опрацьовувати; вдосконалювати
political science	політологія
pollute	забруднювати
population trends	тенденції зміни структури та кількості населення
pop-up ad	спливаюче вікно (наприклад, в інтернет-рекламі)
port	переносити (напр., програму з однієї машини на іншу)
port	переносити, адаптувати

port	порт
porting	портування (у програмуванні портування розуміють як адаптацію програми або її частини до роботи в іншому середовищі, відмінному від того, для якого вона була написана)
post-relational database model	пост-реляційна модель бази даних
power	сила, потужність
power cord	шнур живлення
power management	керування електроживленням
power plant	1) силова установка; 2) електростанція
power supply	блок живлення
powerful	потужний
Power-over-Ethernet (PoE) equipment	устаткування / обладнання живлення через Ethernet
practice	1) технологія, практика; 2) метод, спосіб
practices	інструкції
prank	жарти, пустощі, витівки
precaution	запобіжний захід
precede	передувати
precedent	прецедент; тут: попереднє значення
preceding	попередній
precisely	точно
pre-date	передувати, відбуватися, статися (перед чимось)
predefined	наперед визначений, стандартний
predictor variable	предикторна змінна, прогностичний параметр
preexisting	який існував раніше; що існує

premium-rate telephone number	телефонний номер привілейованого тарифу, номер з premium-тарифом
prerequisite	1) передумова; 2) необхідний як передумова
presenter	ведучий
preventio n	запобігання
previously	раніше
primarily	головним чином
primarily	головним чином; передусім
primary key	первинний ключ (у базах даних)
print	друкувати
prior	попередній
priority	пріоритет, порядок черговості
privacy	конфіденційність / приватність персональної інформації
privilege escalation	розширення привілеїв
privilege level	рівень привілеїв, рівень доступу
privileged operation	привілейована операція
probabilistic (stochastic) model	і/ймовірнісна (стохастична) модель
probability distribution	розподіл імовірностей
probably	ймовірно
proceed	1) відбуватися; 2) продовжувати(ся)
process model	модель процесу
processing device	пристрій оброблення інформації
produce	створювати, виробляти
proficiency	досвідченість, уміння, вправність, професійність
profile	профіль, сукупність параметрів користувача
profile diagram	профілограма

Profit	1) прибуток, зиск, вигода; 2) давати прибуток
programmable function key	програмована функціональна клавіша
progression	просування, рух уперед, прогрес
proof	непроникний, непробивний, міцний
propagation	поширення, передавання
property	властивість
property	властивість
proprietary information	службова інформація; інформація, що є власністю фірми, організації; конфіденційна інформація (фірми, організації)
prosecution	судове переслідування
protection failure	відмова захисту
protection of information	захист інформації
prototyping	макетування, розроблення прототипу
prove	1) доводити, засвідчувати, підтверджувати; 2) засвідчувати, підтверджувати документами
proven	доведений, випробуваний, перевірений
provide	забезпечувати, надавати
provide	забезпечувати
proximity	близькість; схожість
proxy	сервер-посередник, proxy-сервер
purpose	мета, намір
Q	
quantity	величина, параметр
query language	мова запитів (мова керування даними)
query result	результат запиту

R	
radio frequency interference (RFI)	радіозавади, радіочастотні наведення
Random Access Memory (RAM)	оперативна пам’ять, оперативний запам’ятовувальний пристрій (ОЗП)
random variable	випадкова змінна
randomness	випадковість, і/ймовірність
rate	1) темп, швидкість, частота; 2) коефіцієнт; 3) інтенсивність
rating	1) параметр, розрахункова величина; 2) потужність, номінальна характеристика
Rational Unified Process (RUP)	раціональний уніфікований процес (розроблення)
reach	1) розмах; 2) сфера (впливу); 3) простір, протяжність; 4) радіус дії
react with	вступати в реакцію (з чимось)
read back	1) зчитувати щойно записану інформацію; 2) повторно зчитувати
read lock	блокування зчитування
real number	дійсне число
real-time clock	годинник реального часу
real-time operating system	операційна система реального часу
reasonable	прийнятний, підхожий, адекватний
reasoning	мислення, міркування
recast	змінювати, переробляти
receiver	приймач

recent	останній, найновіший
recent	останній, новітній
reclassify	рекласифікувати, змінювати класифікацію, переводити до іншої категорії, перегруповати
recognition	1) визнання; 2) у/впізнавання, розпізнавання
recorded presentation	записана презентація
recover	відновлювати(ся)
recurring	такий, що повторюється, періодичний,
reduce	зменшувати, знижувати, послаблювати
refer	називати, позначати
refer (to)	1) стосуватися; 2) називати
reference	1) давати посилання, посилатися (на щось); 2) подавати у вигляді таблиць
refill	1) поповнення; 2) поповнювати
refinable	що підлягає деталізації
refinement	вдосконалення, підвищення якості
regardless of	не звертаючи уваги (на що), попри (те, що)
register with an Internet site	реєструватися на інтернет-сайті
reinforce, enforce	підсилювати, посилювати, зміцнювати, збільшувати
reject	не приймати, відкидати
relation	матем. відношення
relation(ship)	(взаємо)зв’язок, залежність, співвідношення
Relational Database Management System (RDBMS)	система керування реляційною базою даних
relational database model	реляційна модель бази даних

relative	відносний, порівняльний, відповідний
relative timing	відносна синхронність, узгодженість у часі
relatively	відносно, порівняно
release	1) вивільнення, розблокування; 2) вивільняти(ся)
release	1) публікація; 2) надання
relevant	такий, що стосується (даного питання, справи), релевантний
relieve	позбавляти, звільняти
remain	залишатися
remain undetected	залишатися невиявленим
remediation	виправлення, тут: ліквідування наслідків
remote	віддалений, дистанційний
remote access	віддалений доступ
render	(у сполученні з прикметником) спричиняти певний стан, робити
repel	відбивати
represent	представляти, презентувати
representative	представник
request	робити запит
requirement	вимога
reservation	резервування, бронювання
reside	перебувати, міститися
reside	тут: полягати
resilience	стійкість
resistant	стійкий
resistant	стійкий, міцний
resolution	роздільна здатність
respectively	відповідно

response time	час відгуку (час, потрібний комп'ютеру для відповіді на запит)
responsibility	відповідальність
responsible	відповідальний
responsible	відповідальний
responsible	відповідальний
restrict	обмежувати
restricted	обмежений
result in	мати результатом (щось), закінчуватися (чимось)
retain	утримувати, зберігати
retire	тут: видаляти
retrieve	вибирати (інформацію з пам'яті)
reusability	можливість повторного використання
reveal	показувати, розкривати
revenue	дохід, прибуток, виручка
review	перевіряти; оглядати; рецензувати
risk-based security testing	тестування захищеності, що базується на оцінюванні ризиків
robustness	міцність
rogue	некерований
rootkit	руткіт (програмні засоби, що приховують наслідки зламу та ховають засоби, які використовують зловмисники, від антивірусного програмного забезпечення)
routine	підпрограма
routinely	щодня, регулярно, як заведено
rule	правило
run	виконувати

run time	час виконання (програми), час прогону (програми)
S	
sales income	прибуток від реалізації
sales promotion	просування товару, стимулювання збуту
sales records	торговельна статистика, реєстрація обсягів продажу
scalability	розширюваність, масштабування
scalable processor architecture (SPARC)	архітектура процесора, яку можна нарощувати
scale	шкала, масштаб
scale	масштабувати
scaling	масштабування, масштабне перетворення
scenario	сценарій, план дій
schedule	1) графік, розклад; план робіт; 2) планувати
schedule	графік
schedule	складати розклад, планувати
scholar	вчений
screenshot	моментальний знімок екрану
script	сценарій, скрипт
search engine optimization/optimizer (SEO)	оптимізація / оптимізатор пошукових систем
secure	1) безпечний; 2) охороняти, захищати, забезпечувати, гаран- тувати
security	безпека
security solution	рішення про забезпечення захисту
security token	маркер доступу, токен (фізичний пристрій, який надається авторизованому користувачеві з ме-

	тою сприяння автентифікації)
select	вибирати, добирати; комплектувати
self-paced	такий, що дозволяє самостійно обирати швидкість вивчення матеріалу (про учбовий курс)
semantic backplane	семантичний задній план (в UML об'єднує модель і наповнює її змістом)
semantics	семантика
semaphore	семафор
senior executive	керівник вищого рангу
sense	відчуття
sensitive	1) чутливий; 2) конфіденційний
sensitive information	таємна / конфіденційна інформація
separately	окремо
sequence	послідовність
sequence diagram	діаграма послідовностей
sequenced	1) послідовний; 2) впорядкований
serve to	подавати
server cluster	серверний кластер
set	сукупність, набір
severely	дуже, значно, досить
severity	серйозність, критичність
share (resources)	розподіляти, спільно використовувати (ресурси)
shared lock	блокування із забезпеченням спільного доступу (до набору даних)
shared memory	розподілена пам'ять
sheet	лист (тут: поляризаційного матеріалу)
shielding	екранування (спосіб захисту передавального середовища від електромагнітних завад)

shipping	поставка, відправлення, відвантаження
shoot(p., pp. shot)	стріляти
short-range	короткочасний
shutdown	вимкення, зупинка
shutdown time	час зупинки
significant	значний, важливий, істотний
similar	подібним чином
similar	подібний
similarly	подібним чином
similarly	так само, у той самий спосіб, аналогічно
simplistic	спрощений
simulation	моделювання
simultaneously	одночасно
single-task(ing) operating system	однозадачна операційна система
single-user operating system	операційна система індивідуального користування
skills data	дані про кваліфікацію
skip	пропускати, стрибати, перестрибувати
small-scale	невеликий, незначний
social engineering	«соціальна інженерія», соціотехніка (тактика обдурювання користувачів мережі чи адміністратора, що її використовують зловмисники з метою отримання паролів, необхідних для проникнення у захищену систему)
social sciences	суспільні науки
societal awareness	соціальна (суспільна) поінформованість, «соціетальна» (широка суспільна), обізнаність

socio-technical system	соціальнотехнологічна система
socket	розетка, роз’єм, рознім
software	програмне забезпечення (ПЗ)
software architect	розробник структури ПЗ, фахівець з архітектури систем ПЗ
software architectural design	архітектурне проектування ПЗ, проектування архітектури ПЗ
software architecture	архітектура програмного забезпечення
software configuration	конфігурація програмних засобів
software construction	конструювання ПЗ
software design	проектування ПЗ
software design	1) проектування ПЗ; 2) проект ПЗ
software detailed design	детальне проектування ПЗ
software development life cycle	життєвий цикл розроблення ПЗ
software driver	програмний драйвер
software engineer	спеціаліст з розроблення програмного забезпечення
software engineering	інженерія ПЗ
software intensive system	система з громіздким ПЗ
software interface specification	специфікація, детальний опис програмного інтерфейсу
software life cycle	життєвий цикл ПЗ
software model	модель програмного забезпечення
software modelling	моделювання програмного забезпечення
software procedure	програмна процедура
software project	проект ПЗ
software release	версія програмного забезпечення

software security	захист / безпека програмного забезпечення
software solution	1) програмний продукт; 2) програмне рішення
software system	система ПЗ, програмний комплекс
somewhat	певною мірою, почасти, дещо
sophisticated	складний, витончений
sound alarm	звукове попередження про небезпеку
source file	вихідний файл
spammer	спамер (той, хто розсилає спам)
spatial database	розосереджена база даних
spatial information system	просторова інформаційна система
speaker	гучномовець
specialized	спеціалізований
specific	конкретний; специфічний; особливий
specifically	а саме, зокрема, конкретно
specification	1) специфікація; 2) часто pl технічні вимоги
specification	1) специфікація; 2) уточнення, конкретизування
specification	специфікація
specify	визначати, задавати
specify	уточнювати, деталізувати, конкретизувати
split	ділити на частини
split up	розподіляти(ся)
spoof	підробляти, підміняти (наприклад, адресу електронної пошти), імітувати
spreadsheet	великоформатна (електронна) таблиця
spyware	програмне забезпечення, призначене для шпигування за діями користувача

staged-delivery model	каскадна модель
stakeholder	учасник
stakeholder	прогр. учасник
star topology	топологія «зірка» (топологія ЛОМ, у якій пристрої і комп'ютери з'єднані радіальними лініями з центральним вузлом)
start from scratch	розпочинати з нуля (з чистого аркуша)
start time	час запускання, початковий момент
startup	(за)пуск
state	стан
state machine diagram	діаграма кінцевого автомату
State model	модель станів
state transition	перехід станів
state variable	змінна стану
statement	оператор (мови програмування)
static model	статична модель
steal(p. stole, pp. stolen)	красти
stealthy	непомітний, таємний, прихований
step-by-step	поступовий, ступінчастий, поетапний
stepwise	поетапний
stereotype	стереотип (у мові UML – створення нових елементів моделі шляхом розширення функціональності базових елементів)
still image	статичне зображення
storage medium	носій даних
store	запам'ятовувати, зберігати
storehouse of information	енциклопедія

string	рядок
structural architect	архітектор-будівельник
structural arrangement	структура
structural perspective	структурний аспект
structural view	структурне представлення
structure diagram	структурна діаграма
Structured Query Language (SQL)	мова структурованих запитів
Structured Query Language (SQL)	мова структурованих запитів, мова SQL
stylus	стилус (перо для введення даних у планшетних ноутбуках і кишенькових ПК)
subject matter	тематика, зміст
subpixel	субелемент ,зображення, субпіксель
subscription	передоплата, плата наперед
subsequently	згодом, пізніше, потім
subset	підмножина
substantial	1) істотний, важливий, значний; великий; 2) основний, головний
succeed	досягати мети, мати успіх
successfully	успішно
successive	послідовний
sufficient	достатній
sufficiently	достатньо, достатньою мірою
suited	придатний, підхожий
summarize	1) підсумовувати; 2) зводити, узагальнювати
supply	постачати, подавати
support	підтримувати, підтримка

support	підтримувати
surreptitious	таємний
surround	оточувати
switch	перемикач
switch (between)	перемикати
switch on/off	вмикати/вимикати
synchronization	синхронізація
synthesize	синтезувати
synthesized	синтезований
system call	системний виклик
System Development Life Cycle (SDLC)	життєвий цикл розроблення системи
system implementation	реалізація системи
system mode	системний режим
systemic view	системне представлення
T	
tag	маркувати, розставляти теги, супроводжувати дані тегами
take advantage of	скористатися, використовувати
take on	набувати нового значення
tamper	1) зламувати 2) псувати 3) фальшувати, підробляти
tampering	1) псування; 2) фальшування, підроблення
tangible	відчутний
target audience	цільова аудиторія
target system	цільова система
target user	цільовий користувач
task model	модель завдання
technique	техніка, метод

temporal relationships	часові співвідношення
tempt	спокушати, зваблювати
tend to	мати тенденцію, схильність (до чогось)
terminal device	термінал, термінальний пристрій
test case	набір тестових даних, контрольний приклад (документально оформлений посібник, який визначає, як має / може бути протестована функція або комбінація функцій)
textual	текстовий; що стосується тексту
theft	крадіжка
therefore	тому, отже
thought leadership	інтелектуальне лідерство
Thread	потік, тред
threading	організація потокового оброблення (повідомлень або даних)
threat	загроза, небезпека
throughput	продуктивність; пропускна здатність
thus	так, таким чином
thwart	заважати, перешкоджати
tick	імпульс
timeliness	своєчасність (реагування, подання інформації в комп. системі)
timely	своєчасно
timing characteristic	часова характеристика
timing data	часові показники / характеристики
timing diagram	часова діаграма
tolerate	витримувати
toll	плата (за послуги)

toll call	міжміська телефонна розмова
tool	засіб, інструмент, сервісна програма
top down	згори донизу
touch point of sth	тут: точка дотику (із чимось); питання, що стосується чогось
touch screen	сенсорний екран
trace	слід, ознака
tracking system	система стеження
trade-offs	плюси та мінуси, баланс переваг і рівень недоліків
transaction	транзакція (самостійне або завершене повідомлення про якусь подію або стан, зафіксоване на якомусь носії інформації і призначене для ініціювання операції СКБД)
transaction processing system	система оброблення транзакцій
transferable	який може передаватися
transformation	перетворення, трансформація
transmit	передавати
transparent encryption	«прозоре» (непомітне) шифрування (не залежить від характеристик системи і не впливає на її нормальне функціонування)
trap	вміщувати
treat	розглядати, трактувати, інтерпретувати
trigger	пусковий сигнал
trustworthiness of data	достовірність даних
tune	регулювати, налагоджувати
two-tier, three-tier, multi-tier	дворівневий, трирівневий, багаторівневий

typed object	типізований об’єкт
typewriter	друкарська машинка
typically	зазвичай, звичайно
typically	зазвичай
U	
ultimate	остаточний, кінцевий
unanticipated	непередбачуваний
unauthorized activity	неправомірна, несанкціонована діяльність, несанкціоновані дії
unauthorized personnel	сторонній персонал, сторонні особи
unavoidably	неминуче
uncomplicated	неускладнений
undergo(p. underwent, pp. undergone)	зазнавати
underground	нелегальний, секретний, підпільний
underlying	такий, що лежить в основі, основний, головний, базовий
unforeseeable event	непередбачувана подія
Unified Modelling Language (UML)	уніфікована мова моделювання
Unified Modelling Language (UML)	мова UML, уніфікована мова моделювання)
uniform	сталий, рівний, однаковий
unique	унікальний, незвичайний
unique value	єдине значення
unit testing	тестування компонентів системи
universal serial bus (USB)	універсальна послідовна шина

universal serial bus (USB)	універсальна послідовна шина, шина USB
unlike	на відміну від
unlock	1) розмикати, розблоковувати; 2) розблокування
unwieldy	громіздкий
up to date	сучасний, оновлений
upgrade	модернізувати, покращувати
upgrade time	час модернізування, оновлення
upload	пересилати (інформацію в комп'ютер вищого рівня, наприклад, з локального комп'ютера – на сервер)
usability	1) зручність користування; 2) практичність
usable	придатний для використання, практичний, зручний
USB key, USB drive, flash drive	флеш-накопичувач / флеш-пам'ять
use case	прецедент (у мові UML – з допомогою прецедентів моделюють діалог між актором і системою; набір усіх прецедентів системи визначає її функціональність; на діаграмах прецедент зображають у вигляді еліпса)
use case	варіант використання
use case	варіант використання
use case diagram	діаграма прецедентів (у мові UML – графічне зображення акторів, прецедентів та їх взаємодій у системі; розрізняють головну діаграму прецедентів і додаткові діаграми)
user database	база даних користувачів (абонентів)
User interface	інтерфейс користувача (програми)
user interface (UI)	користувацький інтерфейс, інтерфейс користувача
user-interface	користувацький інтерфейс

V	
variable	змінна
variety	1) різноманітність; 2) відмінність, розбіжність
vary	міняти(ся), змінювати(ся), відрізнятися, варіюватися
vector graphics	векторна графіка
vendor	постачальник, продавець
verification	верифікація
verify	контролювати, перевіряти
verify	перевіряти, контролювати
video game console	приставка для відеоігор
video/visual display unit (VDU)	монітор, дисплей
videoconferencing	відеоконференц-зв'язок
view	розглядати
view	1) вид 2) погляд, аспект, точка зору
view in person	переглядати особисто
viewer	глядач
violate	порушувати
violation	порушення
virtual item	віртуальний елемент
vision-impaired	з вадами зору
visualize	наочно презентувати, візуалізувати
vogue	популярність, широке застосування
voice mail	голосова пошта; автовідповідач компанії
voltage	напруга
Vs	проти, відносно, замість
Vulnerability	слабке місце, вразливість, чутливість (щодо чогось)
W	

wakeup	активізація
wall-sized	розміром зі стіну
waterfall model	водоспадна модель
weak	слабкий
web browser	браузер, програма веб-перегляду, навігатор
web crawler	пошуковий агент, «павук»
web server	веб-сервер
web-enabled	веб-орієнтований
well-documented	переконливо підтверджений документальними доказами
whenever	кожного разу, коли; щоразу, коли б не
while	в той час як; тоді як
wire	дріт, провід
with respect to	відносно
withstand	протистояти, витримувати
word processing	оброблення текстів
work site	робоче місце, об’єкт (виконання робіт)
work system	тут: виробнича система
workflow	послідовність операцій
workflow	трудовий процес
World (WWW)	«Всесвітнє павутиння», глобальна гіпертекстова система Internet
worm	черв’як (програма, що самостійно поширює свої копії мережею)
write lock	блокування запису
write time	час запису
X	
XQuery	перехресна мова запитів

Y	
yield	давати результат
Z	
ZIP	найбільш поширений стандарт ущільнення; формат архівів на FTP (File Transfer Protocol)-серверах

LITERATURE

1. Пройдаков Е.М., Теплицький Л.А. Англо-український тлумачний словний з обчислювальної техніки, Інтернету і програмування. Вид. К.: Видавн. дім "Софт-Прес", 2005. 552 с.
2. Саммервил И. Инженерия программного обеспечения. М.: Вильямс, 2002. 620 с.
3. Сидоров М.О. Вступ до програмної інженерії: конспект лекцій. К.: НАУ, 2009.130с.
4. Glendinning E.H., McEwan J. Oxford English for Information Technology. Oxford Press, 2003. 222 p.
5. Sidorov M.O. Software Engineering. Lecture Course. Kyiv: NAU, 2007.139 p.
6. Wiegers K. Creating a software engineering culture. Dorset House Publ. New York, 2003. 358 p.
7. Wilkinson G.G., Winterflood A.R. Fundamentals of Information Technology. Chichester: John Wiley and Sons, 1987. 363 p.

http://www.wikipedia.com