

ПРАКТИЧНЕ ЗАНЯТТЯ № 03

ТЕХНОЛОГІЯ СТВОРЕННЯ ПРОГРАМНИХ ПРОДУКТІВ

Тема: ОБ'ЄКТНО-ОРІЄНТОВАНИЙ АНАЛІЗ ТА ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ:
**ДІАГРАМА СТАНІВ,
ДІАГРАМА ДІЯЛЬНОСТЕЙ (АКТИВНОСТЕЙ)**

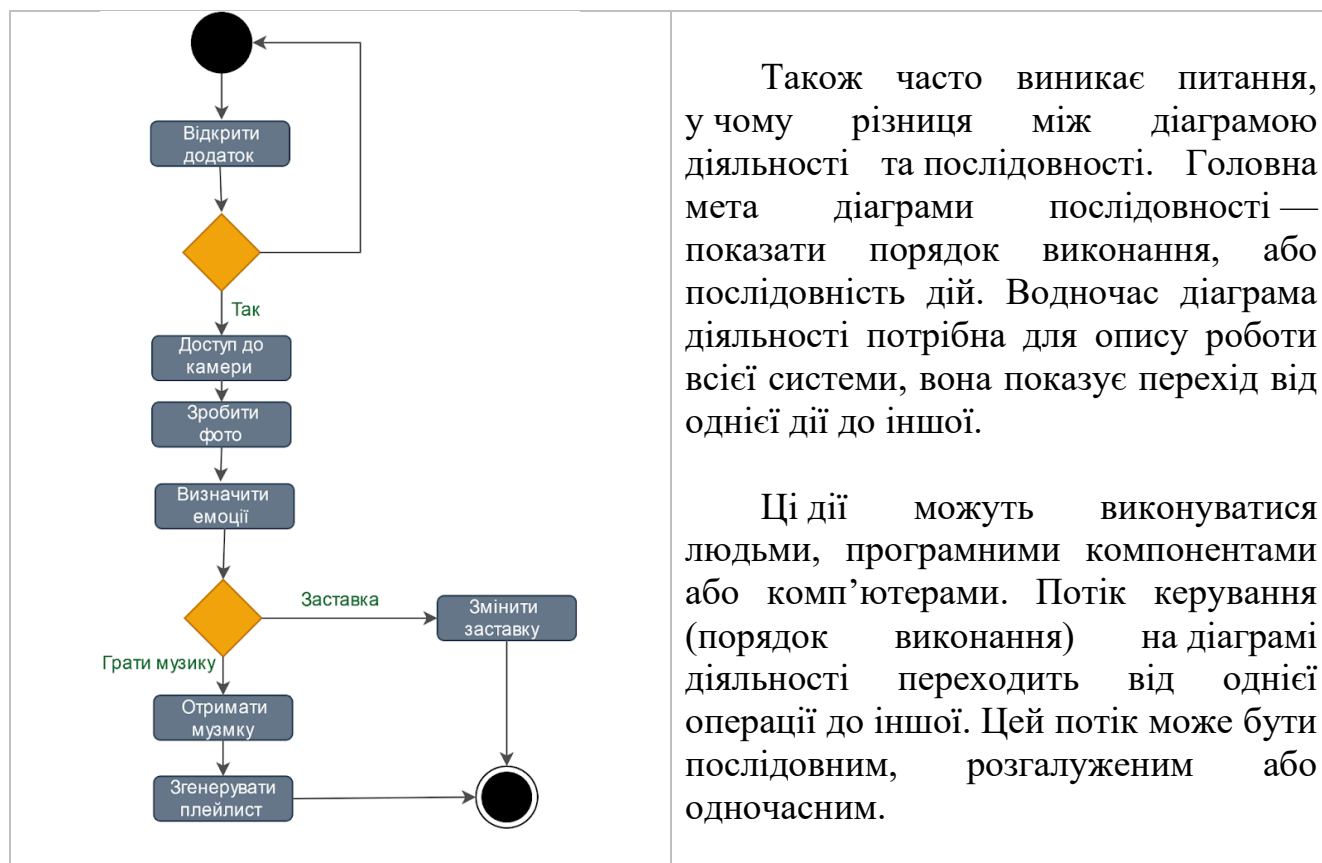
Мета: дослідження предметної області, вивчення процесу побудови діаграми станів та активностей за допомогою інструмента об'єктного моделювання StarUML.

Програмне забезпечення: ОС Windows/Linux, StarUML (<https://staruml.io/>).









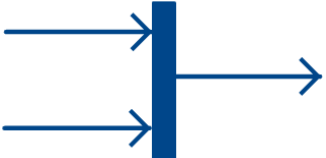



1. Діаграма діяльності

Діаграма діяльності (Activity Diagram) візуалізує процес використання та ілюструє потік повідомлень від однієї дії до іншої. Показує цілісну роботу системи.

Діаграма діяльності вважається розширеним варіантом блок-схем. Проте блок-схема не має стандартизованої нотації та не містить паралелізмів (паралельне виконання дій) та синхронізації.



Таблиця 1. Позначення діаграми діяльності

Символ	Ім'я	Використання
	Початковий вузол	Відправна точка, або початковий стан
	Дія	Представлення діяльності, завдання для виконання
	Потік керування	Спрямований потік, контрольний потік діяльності
	Кінцевий вузол активності	Кінцевий стан, завершення усіх потоків процесу
	Кінцевий вузол потоку	Кінець одного потоку
	Вузол прийняття рішення	Розгалуження з умовою та кількома варіантами дій. Має один вхід декілька виходів
	Вузол злиття	Об'єднання потоків створених вузлом прийняття рішень. Має кілька входів і один вихід
	Вилка	Розподілення потоку на кілька паралельних без прийняття рішення
	Злиття	Об'єднання декількох паралельних потоків
	Надсилення сигналу	Вказує на те що сигнал надсилається на приймальну діяльність
	Отримання сигналу	Вказує на отримання сигналу
	Коментар	Дозволяє робити коментарі до діаграми. З'єднується пунктирною лінією

Потік керування - з'єднує два вузли на діаграмі активності. Вузол розгалуження (вилка) - це вузол керування, який розділяє потік на кілька одночасних. Потрібен для створення декількох одночасних завдань. Вузол злиття об'єднує ці потоки.

Вузол прийняття рішень - це вузол керування, який вибирає між декількома потоками один істинний. Він схожий на оператор if в Java або C#. Умови (охоронні умови) записуються в квадратних дужках поруч з потоком.

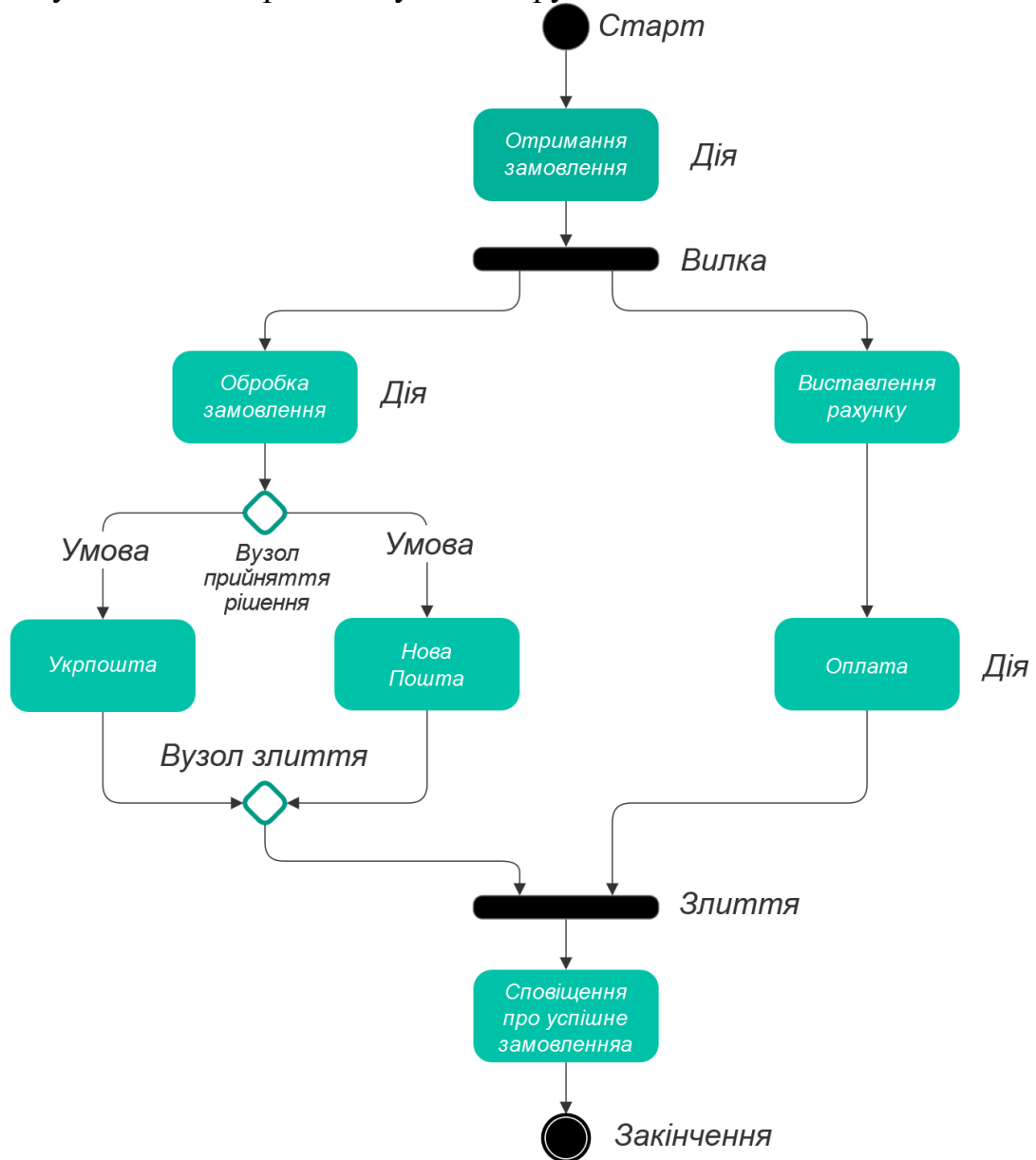


Рис Приклад діаграми діяльності

Доріжки (swimlanes) — це спосіб групування дій, які виконуються одним актором, або декількома акторами в одному потоці. Не додавайте більш як 5 доріжок одночасно. Розташовуйте доріжки у логічному порядку.

Як намалювати діаграму діяльності

1. Уявіть процес, який ви хочете змоделювати, та розбийте його на дії.
2. Подумайте, чи матиме ваш процес учасників, чи буде він мати доріжки.
3. Визначте порядок виконання дій. Чи мають виконуватись якісь умови для виконання певної дії (вузол прийняття рішень). Чи будуть у вас паралельні дії (вузол розгалужень).

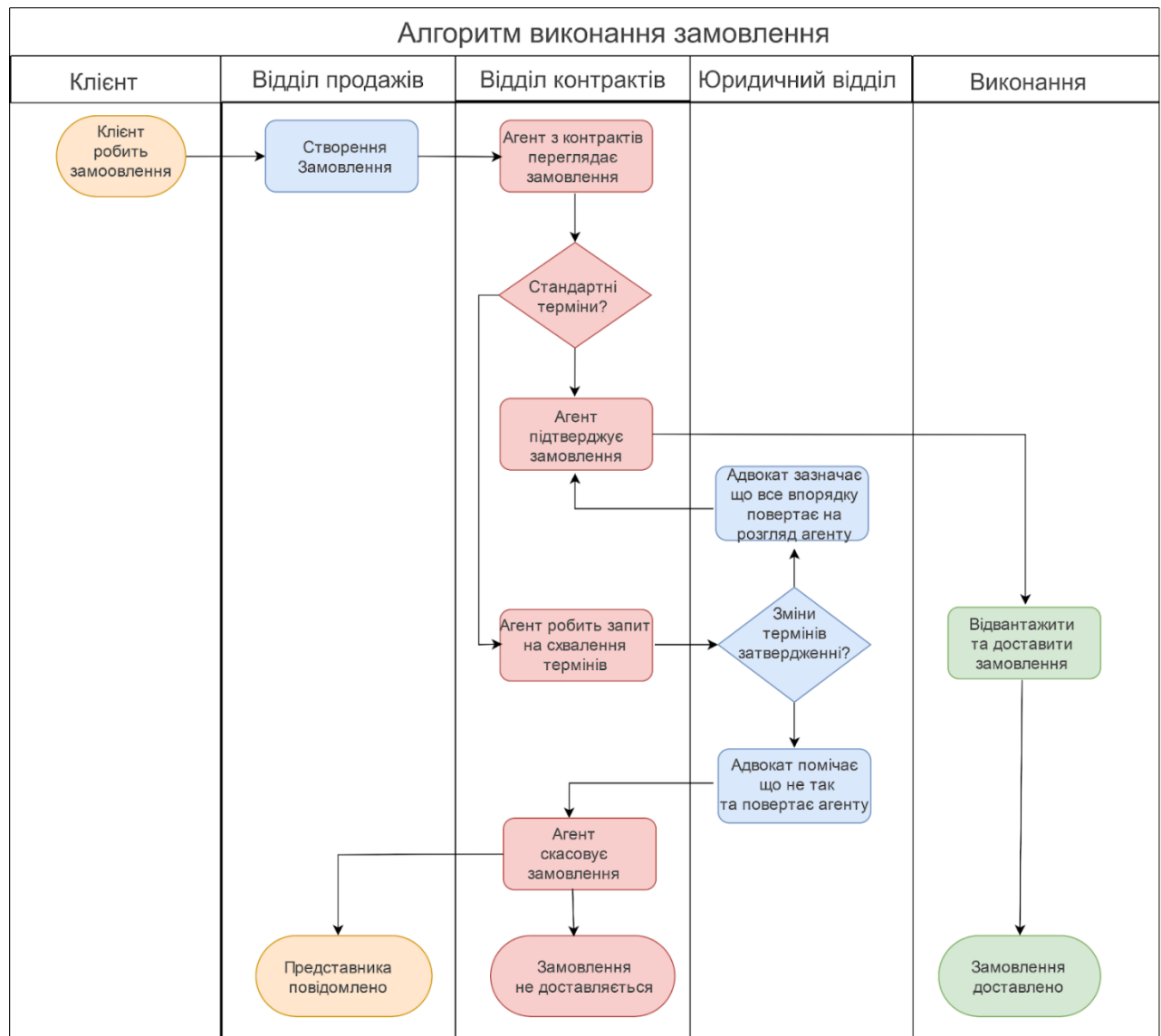


Рис. Приклад діаграми діяльності з доріжками

Приклади діаграм діяльності

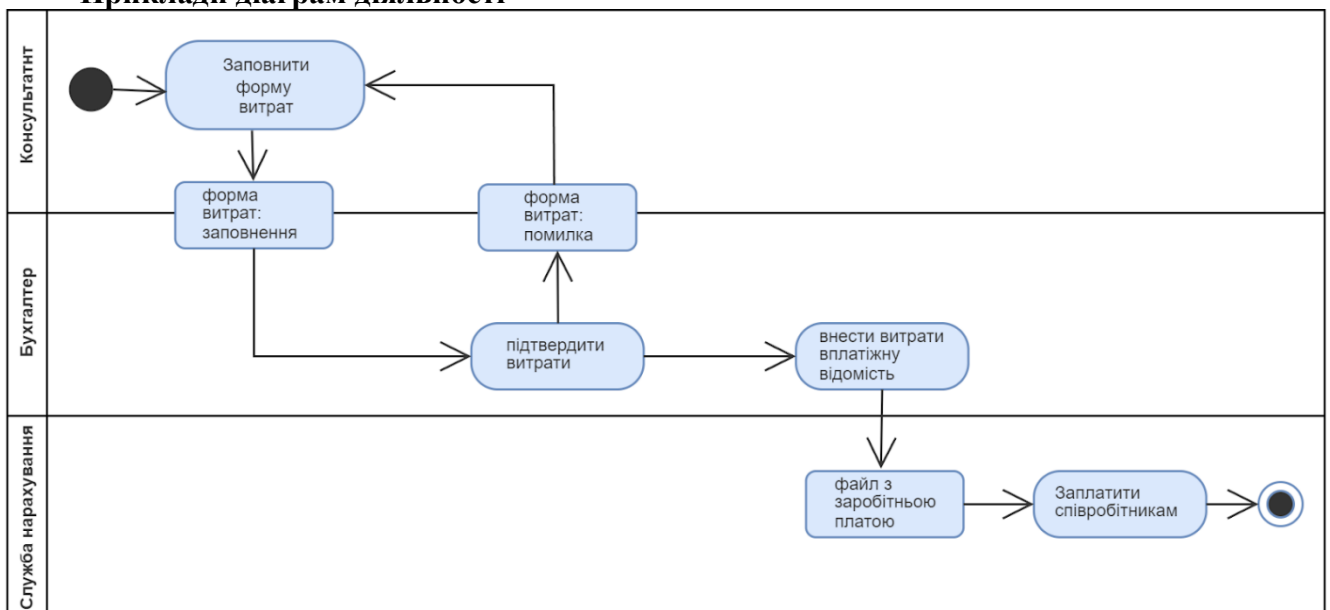


Рис. Діаграма виплати заробітної плати

Знайти більше прикладів можна за наступним посиланнями:

- <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/>
- <https://online.visual-paradigm.com/diagrams/templates/activity-diagram/uml-activity-diagram-example-atm/>

2. Діаграма станів

Діаграма станів моделює поведінку окремого об'єкта, визначаючи послідовність подій, через які об'єкт проходить протягом свого життя у відповідь на події. Наприклад, на наступній діаграмі станів показано стани, через які проходять двері протягом свого життя.

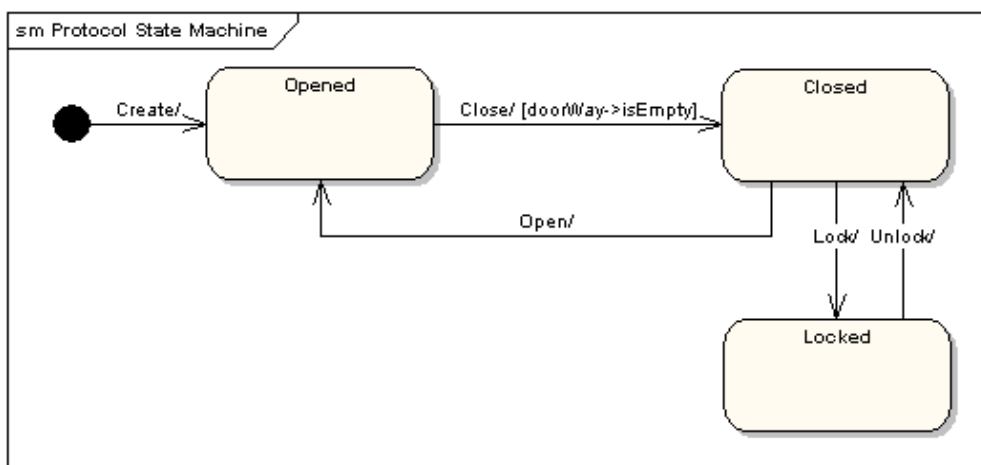


Рис. Діаграма станів

Двері можуть перебувати в одному з трьох станів: "Відчинені", "Зачинені" або "Замкнені". Вони можуть реагувати на події "Відчинити", "Зачинити", "Замкнути" та "Розблокувати". Зверніть увагу, що не всі події є дійсними у всіх станах; наприклад, якщо двері відчинено, ви не можете замкнути їх, поки не закриєте. Також зауважте, що до переходу стану може бути додана умова охорони: якщо двері відчинено, вони можуть реагувати на подію зачинення, тільки якщо виконується умова `doorWay->isEmpty`. Синтаксис і домовленості, що використовуються у діаграмах станів, буде детально розглянуто у наступних розділах.

Стан

Стан позначається прямокутником із закругленими кутами, всередині якого написана назва стану.

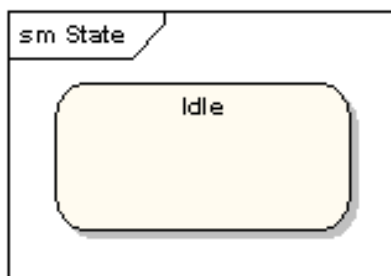


Рис. Позначення стану

Початковий та кінцевий стан

Початковий стан позначається зафарбованим чорним колом і може бути позначений назвою. Кінцевий стан позначається колом з крапкою всередині і також може бути позначений ім'ям.

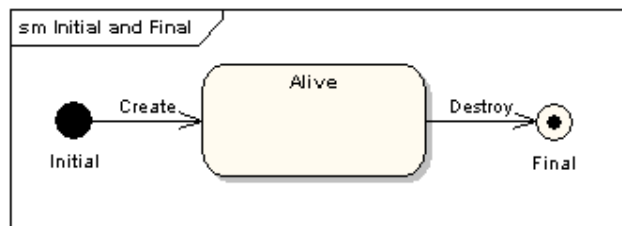


Рис. Початковий та кінцевий стан

Переходи

Переходи з одного стану в інший позначаються лініями зі стрілками. Перехід може мати тригер, запобіжник та ефект, як показано нижче.

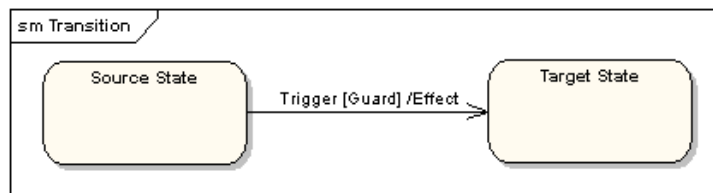


Рис. Позначення переходу

"Тригер" - це причина переходу, яка може бути сигналом, подією, зміною певного стану або плином часу. "Обмежувач" - це умова, яка повинна бути істинною, щоб тригер викликав перехід. "Ефект" - це дія, яка буде викликана безпосередньо на об'єкті, що володіє машиною станів, в результаті переходу.

Дії стану

У наведеному вище прикладі з переходом ефект був пов'язаний з переходом. Якщо цільовий стан має багато переходів, що надходять до нього, і з кожним переходом пов'язаний один і той самий ефект, було б краще пов'язати ефект з цільовим станом, а не з переходами. Це можна зробити, визначивши для стану дію входу. На діаграмі нижче показано стан з дією входу і дією виходу.

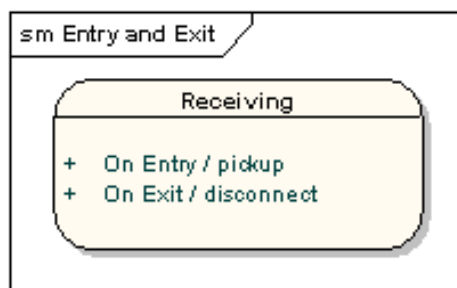


Рис. Позначення дій стану

Також можна визначити дії, які відбуваються на події, або дії, які відбуваються завжди. Можна визначити будь-яку кількість дій кожного типу.

Самопереходи

Стан може мати перехід, який повертається до самого себе, як на наступній діаграмі. Це найбільш корисно, коли з переходом пов'язаний ефект.

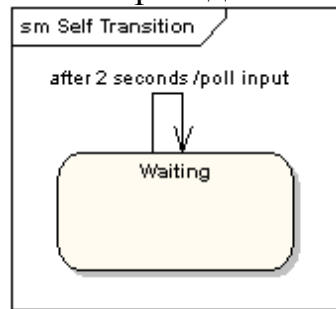


Рис. Нотація самопереходу

Складні стани

Діаграма станів може включати діаграми підмашин, як у прикладі нижче.

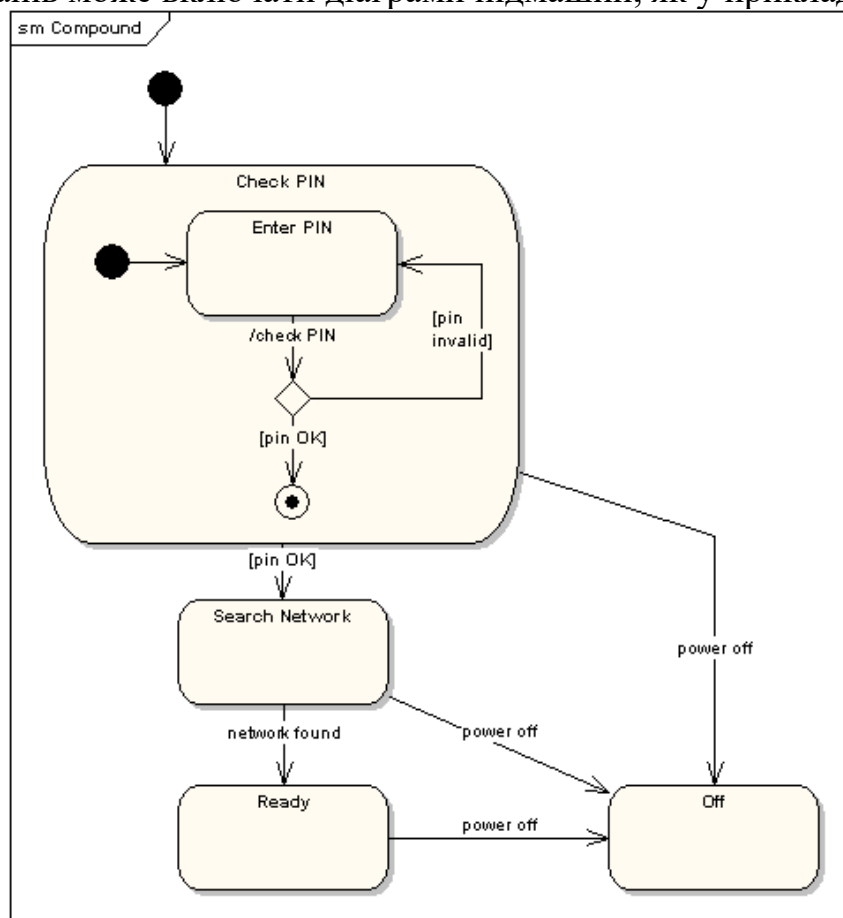


Рис. Позначення складених (вкладених) станів

Альтернативний спосіб показати ту саму інформацію виглядає наступним чином.

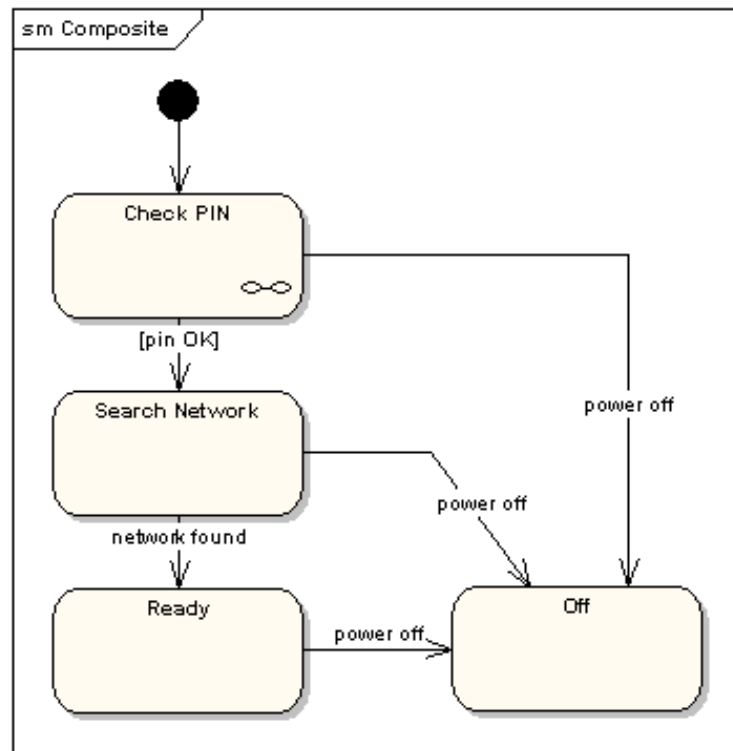


Рис. Альтернативне позначення вкладених станів

Позначення у наведеній вище версії вказує на те, що деталі підмашини перевірки PIN-коду показані на окремій діаграмі.

Точка входу

Іноді вам не потрібно входити у підмашину зі звичайного початкового стану. Наприклад, у наступній підмашині нормальним було б почати зі стану "Ініціалізація", але якщо з якихось причин не потрібно виконувати ініціалізацію, можна почати зі стану "Готовність", перейшовши до названої точки входу.

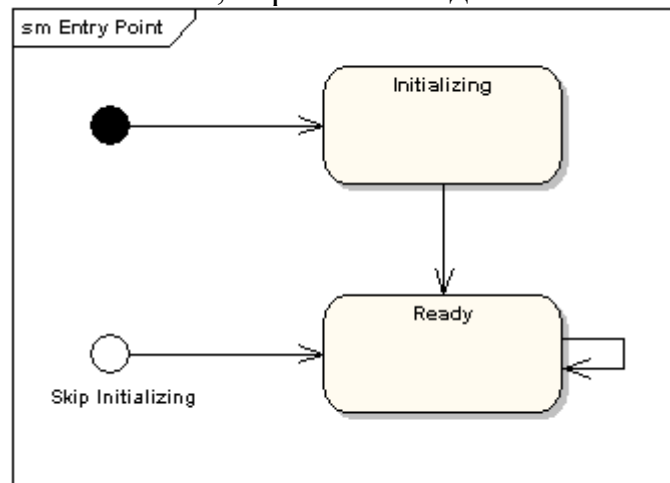


Рис. Нотація точки входу

На наступній діаграмі показано машину станів на один рівень вище.

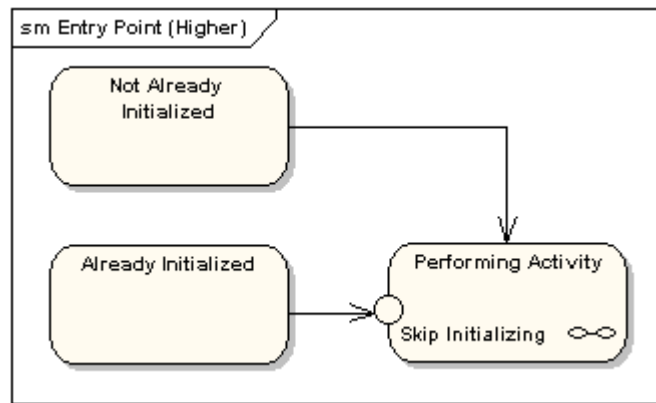


Рис. Нотація точки входу машини станів на рівень вище

Точка виходу

Подібно до точок входу, можна створити іменовані альтернативні точки виходу. На наступній діаграмі наведено приклад, де стан, який виконується після основного стану обробки, залежить від того, який маршрут використовується для переходу зі стану.

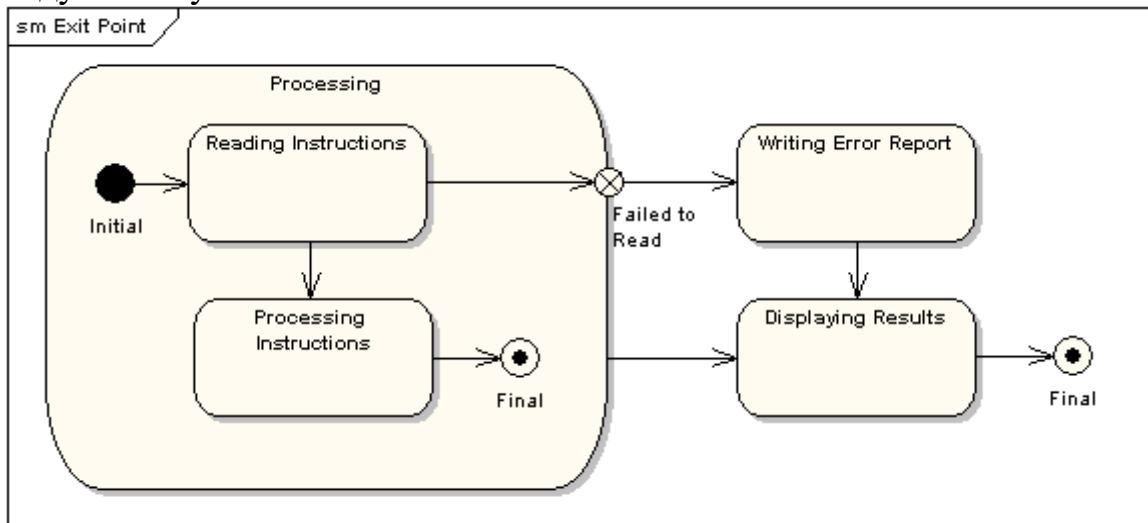


Рис. Нотація точки виходу

Вибір псевдостану

Псевдо-стан вибору показано у вигляді ромба з одним вхідним переходом і двома або більше вихідними переходами. На наступній діаграмі показано, що стан, в який потрапляє система після псевдостану вибору, залежить від формату повідомлення, обраного під час виконання попереднього стану.

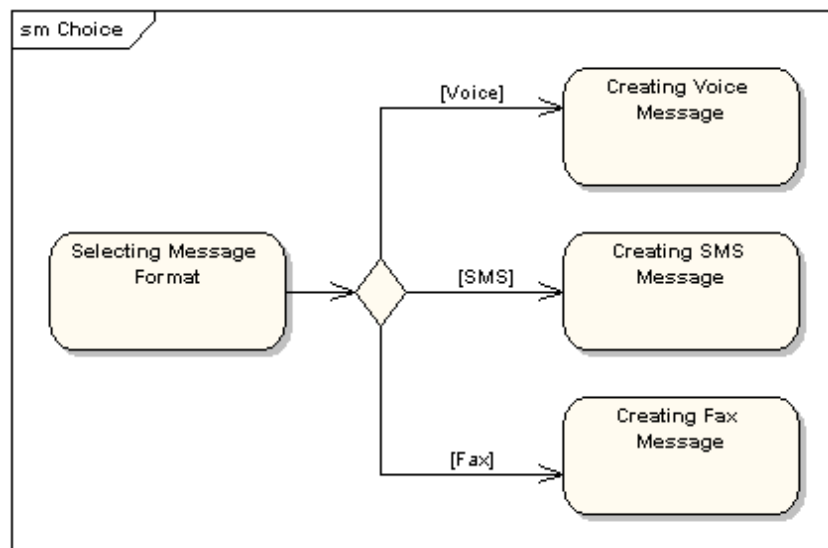


Рис. Нотація псевдостану

Псевдостан Junction

Псевдостани переходів використовуються для з'єднання декількох переходів у ланцюжок. Один перехід може мати один або більше вхідних та один або більше вихідних переходів; до кожного переходу може бути застосовано охорону. Переходи не мають семантики. Перехід, який розділяє вхідний перехід на декілька вихідних переходів, реалізує статичне умовне розгалуження, на відміну від псевдостану вибору, який реалізує динамічне умовне розгалуження.

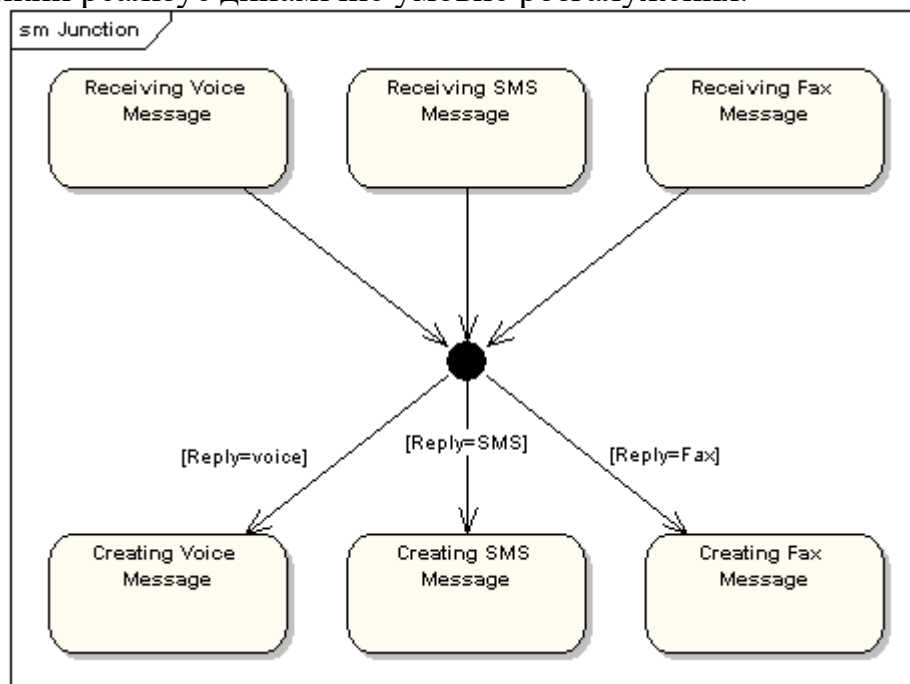


Рис.

Завершення псевдостану

Введення псевдостану завершення вказує на те, що лінія життя автомата станів закінчилася. Псевдостан завершення позначається хрестиком.

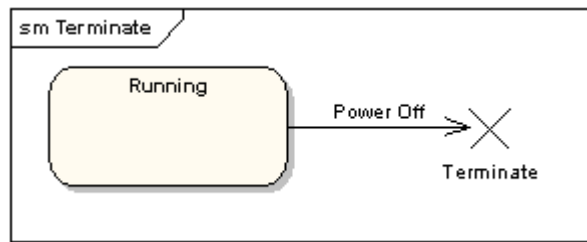


Рис.

Стан-історія

Стан історії використовується для запам'ятовування попереднього стану автомата, коли його було перервано. Наступна діаграма ілюструє використання станів історії. Прикладом є автомат, що належить пральній машині.

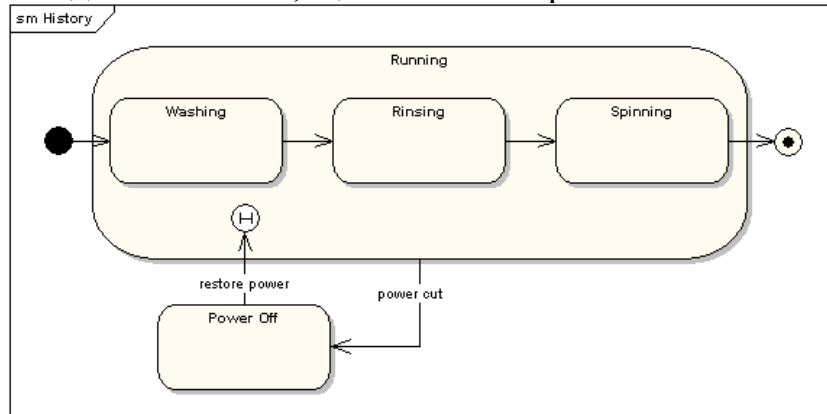


Рис. Стан історія

У цьому стані, коли пральна машина працює, вона переходить від "Прання" через "Полоскання" до "Віджимання". Якщо станеться відключення електроенергії, пральна машина припинить роботу і перейде в стан "Вимкнено". Потім, коли живлення буде відновлено, на символі "Стан історії" з'явиться стан "Робота", що означає, що пральна машина має відновити роботу з того місця, де вона зупинилася востаннє.

Суміжні регіони

Стан може бути розділений на області, що містять підстани, які існують і виконуються одночасно. У прикладі нижче показано, що в межах стану "Застосування гальм" передні та задні гальма працюватимуть одночасно і незалежно. Зверніть увагу на використання псевдостанів `fork` і `join`, замість вибору і злиття псевдостанів. Ці символи використовуються для синхронізації паралельних потоків.

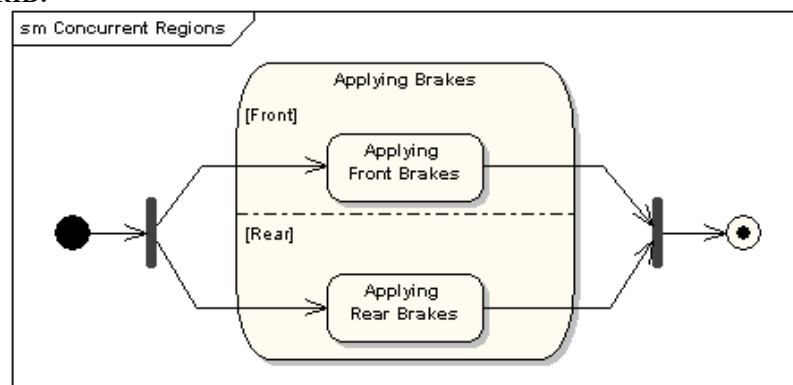


Рис. Нотація суміжних регіонів

Приклада діаграм станів

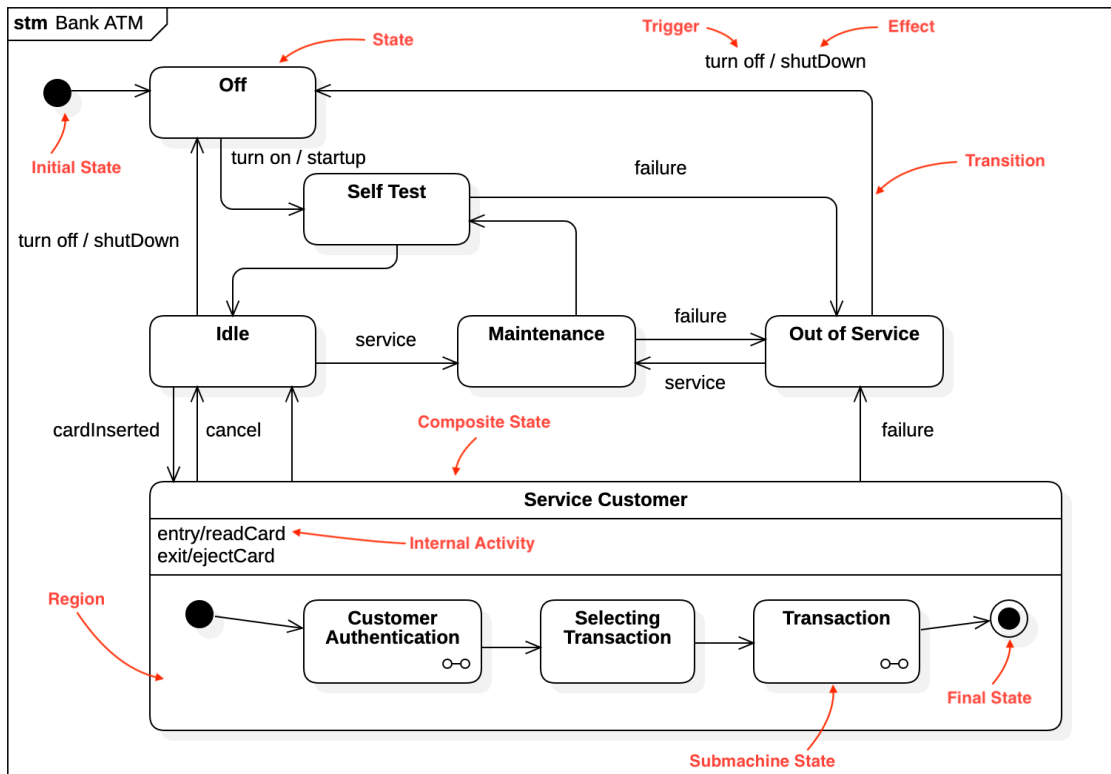


Рис. Приклад 01

Приклада діаграм станів

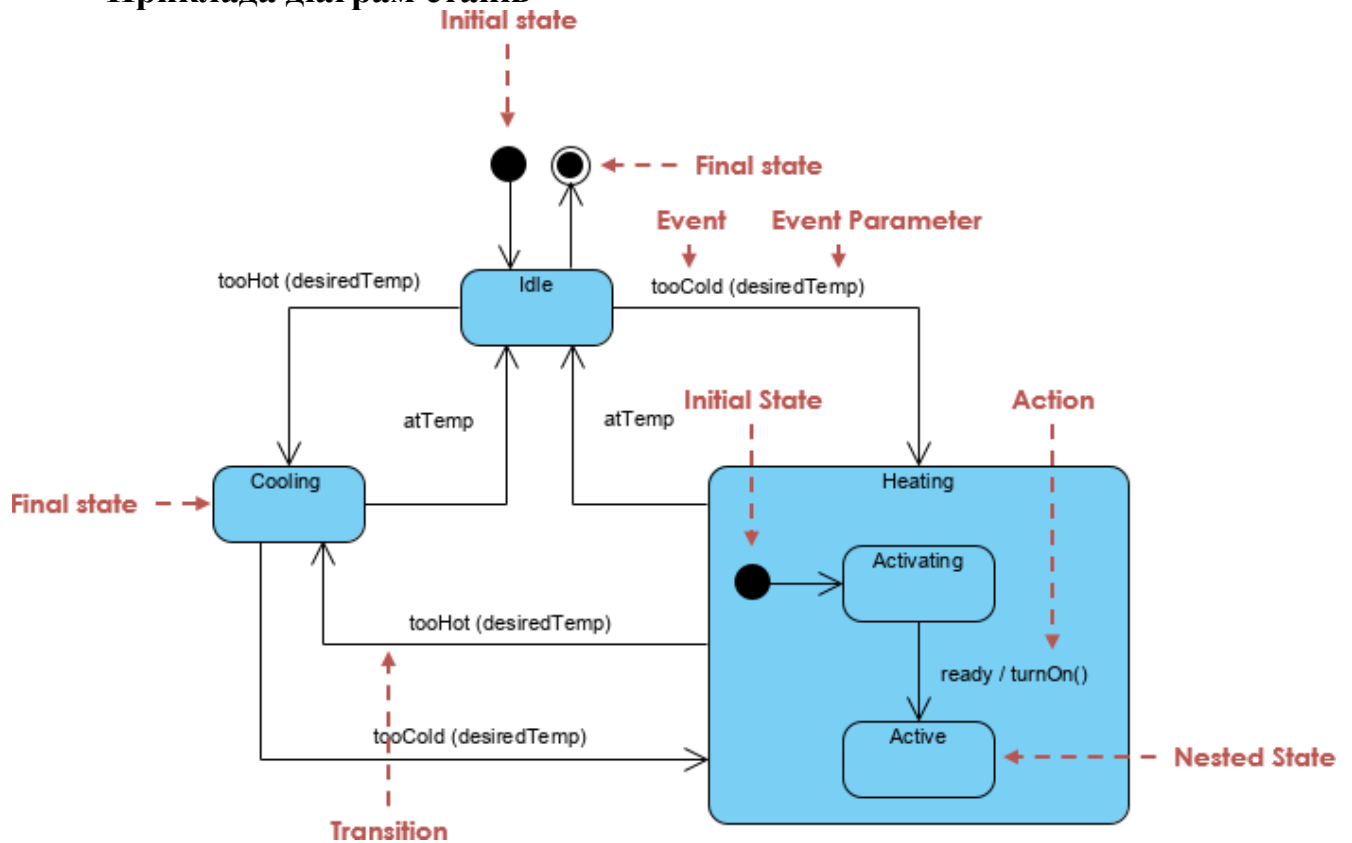


Рис. Приклад 02

Порядок виконання роботи:

1. У відповідності до власного варіанту індивідуального завдання здійснити побудову діаграми станів у середовищі StarUML.
2. У відповідності до власного варіанту індивідуального завдання здійснити побудову діаграми активності у середовищі StarUML.

Контрольні питання:

1. У яких випадках використовується діаграма діяльності?
2. Як позначаються стани дій на діаграмі діяльності?
3. Для чого використовуються стани дій на діаграмі діяльності?
4. Як позначаються переходи на діаграмі діяльності?
5. Для чого використовуються переходи на діаграмі діяльності?
6. Для чого використовуються доріжки на діаграмі діяльності?
7. Наведіть приклади (з графічним позначенням) основних видів станів в діаграмі діяльності
8. Призначення діаграми станів?
9. Що таке автомат? Що таке стан?
10. Які виділяються мітки станів?
11. Як позначаються початковий і кінцевий стани?
12. Що таке перехід, як позначається?
13. Що таке подія? Що задає сторожову умову?
14. Що демонструє діаграма діяльності?
15. Із яких основних елементів складається діаграма діяльності, що вони означають?

Література:

1. Grady Booch, James Rumbaugh, Ivar Jacobson. Unified Modeling Language User Guide. Addison-Wesley, 1999. 496 p. ISBN: 0-201-57168-4.
2. Martin Fowler. UML Distilled: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley, 2004. 208 p. ISBN: 0-321-19368-7.
3. Craig Larman. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Prentice Hall, 2004. 736 p. ISBN: 0-13-148906-2.
4. Grady Booch, James Rumbaugh, Ivar Jacobson. The Unified Modeling Language Reference Manual. Addison-Wesley, 1999. 720 p. ISBN: 0-201-30998-X.
5. Doug Rosenberg, Kendall Scott. Applying Use Case Driven Object Modeling with UML: An Annotated e-Commerce Example. Addison-Wesley, 2001. 512 p. ISBN: 0-201-60489-7.
6. Jean-Marc Nerson. Object-Oriented Software Construction. Prentice Hall, 2002. 704 p. ISBN: 0-13-629155-4.
7. Jim Arlow, Ila Neustadt. UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design. Addison-Wesley, 2005. 624 p. ISBN: 0-321-26779-8.
8. Michael Blaha, James Rumbaugh. Object-Oriented Modeling and Design with UML. Prentice Hall, 2004. 416 p. ISBN: 0-13-196845-0.
9. Alan Dennis, Barbara Haley Wixom, David Tegarden. Systems Analysis and Design: An Object-Oriented Approach with UML. Wiley, 2014. 544 p. ISBN: 978-1-118-56497-6.
10. Craig Larman. UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design. Prentice Hall, 2005. 736 p. ISBN: 0-13-148906-2.
11. Richard F. Schmidt. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. Pearson, 2019. 480 p. ISBN: 978-0131489066.
12. Dan Pilone, Neil Pitman. UML 2.0 in a Nutshell. O'Reilly Media, 2005. 258 p. ISBN: 0-596-00795-7.
13. Doug Rosenberg, Matt Stephens. Use Case Driven Object Modeling with UML: A Practical Approach. Apress, 2007. 648 p. ISBN: 978-1-59059-774-3.