СУЧАСНІ ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ

122 «Комп’ютерні науки»
КН-18
2019 / 2020 навчальний рік

Технології # 3.

1. «Жорсткі» методології
2. «Гнучкі» методології
3. CALS ← → CASE
4. Приклади CASE систем.
5. Сучасні системи підтримки розробки ПЗ

https://github.com/eabshkvprof/2020_Mod_Prog_Techn

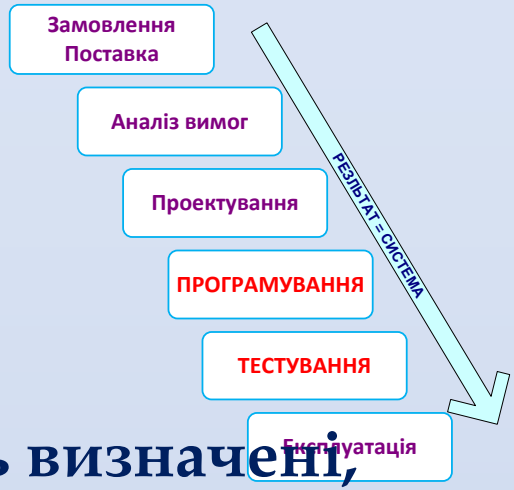
Методології (технології) розробки ПЗ

Методологія розробки програмного забезпечення (Software development methodology) — *сукупність методів*, застосовуваних на різних стадіях життєвого циклу, що мають спільний підхід та дозволяють забезпечити найкращу ефективність процесів створення ПЗ.

Два класи:
Жорсткі – детальне планування процесу створення ПЗ;
Гучки – безперервне взаємодія з замовником, динамічне формування вимог та швидка реакція.

1. Waterfall – каскадна технологія

«Водоспад» – послідовне проходження всіх етапів розробки програмного забезпечення, кожен з яких повинен закінчиться до того, як почнеться наступний.



!!! Вартість і строки розробки заздалегідь визначені, роботи проходять швидко.
!!! Відмінний результат тільки в проектах з заздалегідь чітко заданими вимогами, методиками їх реалізації.
Всі недоліки тільки після закінчення робіт з-за суворої послідовності діяльності. Для початку виправлень доводиться чекати закінчення розробки.
Для невеликих проектів, якщо вимоги відомі, зрозумілі і фіксовані.
Класика жорстокої методології.

2. V – технологія

Прямий нащадок «водоспаду». Стадії розробки ПЗ строго послідовні.

Значна роль приділяється своєчасному тестуванню функціональному, інтеграційному, навантаженню.

Для невеликих і середніх проектів з чітко визначеними вимогами.

3. Iterative – ітеративні методології

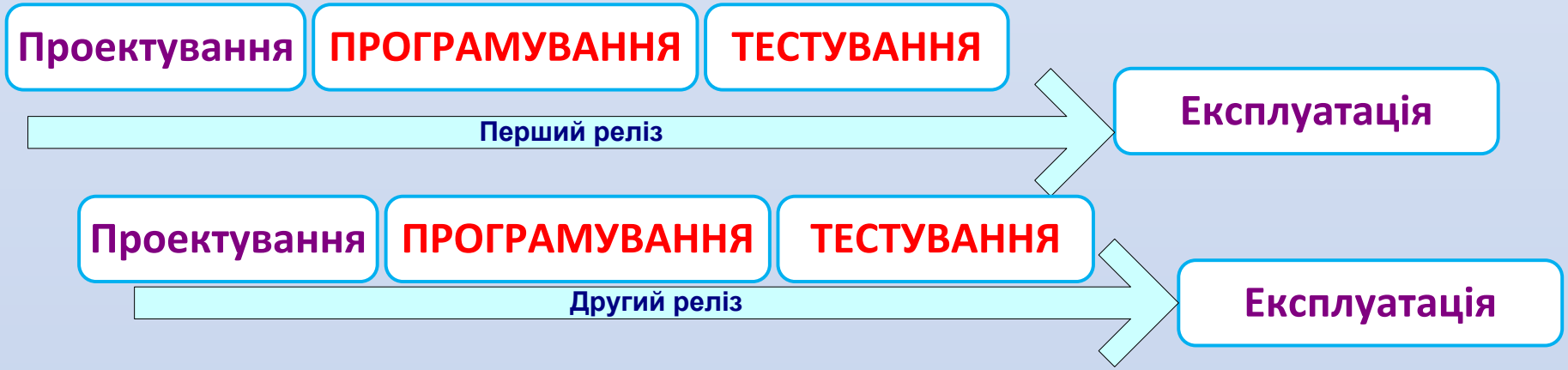
Ітеративні методології:
Розробка виконується у вигляді декількох ітерацій короткострокових міні-проектів фіксованою тривалості.

Кожна ітерація включає свої власні етапи формування вимог, проектування, реалізації (кодування) і завершується тестуванням, інтеграцією і створенням працює версії деякої частини всієї системи.
Система розширюється і доповнюється крок за кроком.

Розробка протікає як послідовність ітерацій, надбудовувати архітектурне ядро до тих пір, поки не будуть досягнуті бажані рівні функціональності, продуктивності і стабільності створюваної системи.

4. Incremental – технологія

Інкрементна технологія (*ітеративна*) - всі вимоги до системи діляться на різні збірки. Система створюється послідовно за допомогою поетапної зборки ПЗ. Використовується декілька циклів розробки. Цикл розділяється на більш дрібні легко створювані модулі.



4. Incremental – технологія

Кожен модуль проходить через фази визначення вимог, проектування, кодування, впровадження та тестування.
Процедура розробки передбачає випуск на першому великому етапі продукту в базовій функціональності, а потім вже послідовне додавання нових функцій, так званих «інкрементів».
Процес триває до тих пір, поки не буде створена повна система.
Використовується -

- **основні вимоги до системи чітко визначені і зрозумілі.**
- **деякі деталі можуть доопрацьовуватися подалі.**
- **потрібно швидкий вивід продукту на ринок.**
- **кілька ризикових цілей.**

5. RAD – «швидка» технологія

RAD (*rapid application development)* - різновид інкрементної технології. Компоненти прототипу ПЗ паралельно розробляються кількома групами (teams). Час на створення кожного компонента жорстко обмежений. Результат - кілька міні-проектів.
Всі компоненти складаються в загальний робочий прототип (версію).

Базові принципи:

- **Циклічність розробки: кожна нова версія продукту ґрунтується на оцінці результату роботи попередньої версії замовником.**
- **Мінімізація часу розробки версії, за рахунок перенесення вже готових модулів і додавання функціональності в нову версію.**
- **Інструментарій має бути націлений на мінімізацію часу розробки.**
- **Створення прототипу для уточнення вимог замовника.**
- **Управління проектом повинне мінімізувати тривалість циклу розробки.**

Допомагає швидко представити замовнику для огляду робочий продукт, який потім можна внести ряд змін.
Рекомендовано використовувати тільки у випадку, коли в команді є висококваліфіковані та "вузькі" фахівці.

6. Spiral – спіральна технологія

Спіральна модель (*ітеративна*)
Головний акцент –
оцінка та керування ризиками + контрольні точки. Ризики:

- **Дефіцит фахівців.**
- **Нереалістичні терміни і бюджет.**
- **Реалізація не відповідає функціональності.**
- **Розробка неправильного користувальницького інтерфейсу.**
- **Непотрібна оптимізація та покращення деталей.**
- **Безперервний потік змін.**
- **Брак інформації про зовнішні компоненти.**
- **Недоліки в роботах, що виконуються зовнішніми (стосовно проекту) ресурсами.**
- **Недостатня продуктивність одержуваної системи.**
- **Розрив у кваліфікації фахівців різних областей.**

6. Spiral – спіральна технологія

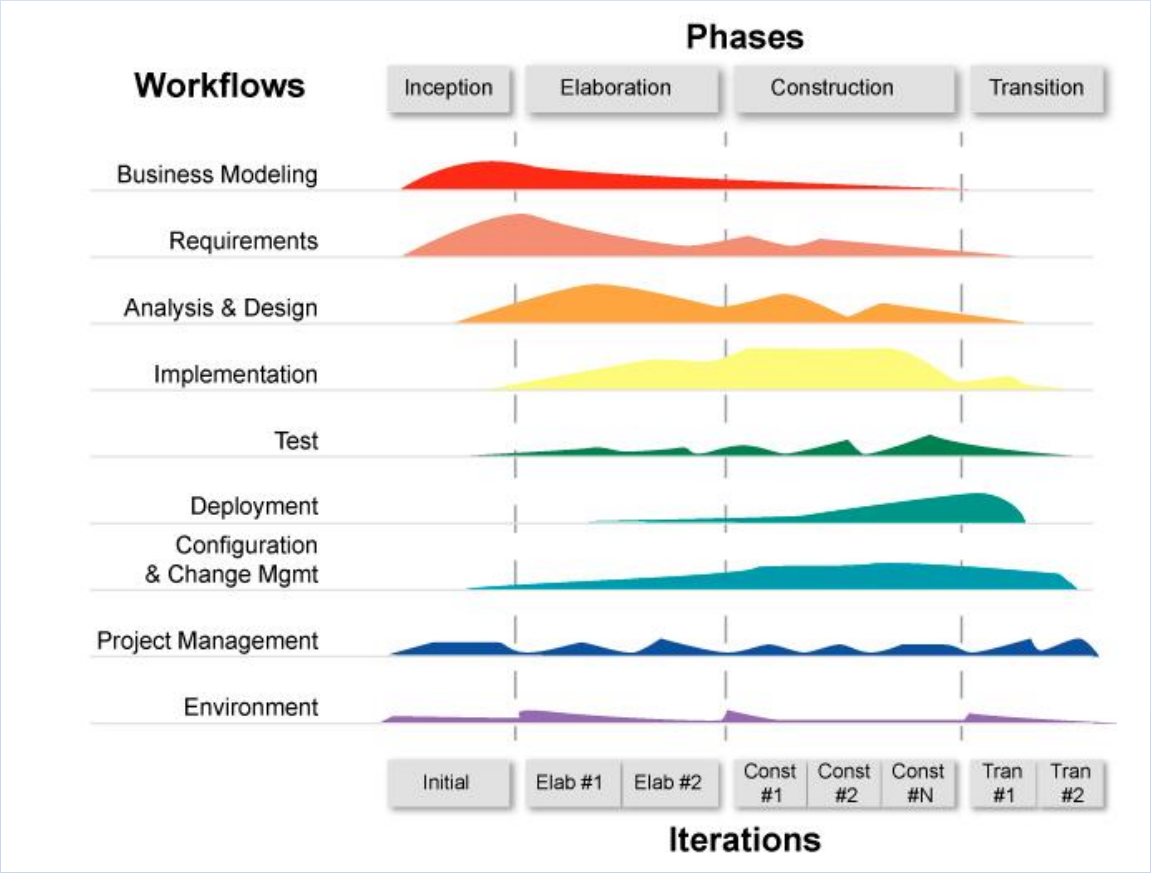
Контрольні точки:

- **Concept of Operations (COO) — концепція (використання) системи;**
- **Life Cycle Objectives (LCO) — цілі і зміст життєвого циклу;**
- **Life Cycle Architecture (LCA) — архітектура життєвого циклу - готовність концептуальної архітектури цільової програмної системи;**
- **Initial Operational Capability (IOC) — перша версія створюваного продукту, придатна для дослідної експлуатації;**
- **Final Operational Capability (FOC) — готовий продукт, розгорнутий (встановлений і налаштований) для реальної експлуатації.**

Застосовується для вирішення критичних бізнес-завдань, коли неуспішність проекту може серйозно зашкодити діяльності компанії.

7. RUP – уніфікований процес

RUP (*Rational Unified Process*) – методологія створення програмного забезпечення, оформлена у вигляді бази знань, що розміщується на Web, та яка забезпечена пошуковою системою.



7. RUP – уніфікований процес

RUP забезпечує строгий підхід до розподілу завдань і відповідальності всередині організації-розробника.

RUP гарантує створення якісного ПЗ точно в строк і в рамках встановленого бюджету, що відповідає потребам кінцевих користувачів.

RUP сприяє підвищенню продуктивності колективної розробки та надає найкраще з накопиченого досвіду по створенню ПЗ, за допомогою посібників, шаблонів і настанов з користування інструментальними засобами для всіх критично важливих робіт, протягом життєвого циклу створення та супроводження програмного забезпечення.

RUP забезпечує всім членам групи доступ до тієї ж самої базі знань, незалежно від ролі розробника.

RUP гарантує, що всі члени групи використовують спільну мову моделювання і процесінгу, мають узгоджене бачення того, як створювати ПЗ.

8. Agile – «гнучка» методологія

AGILE (*Agile software development*) – клас методології розробки ПЗ, що базується на ітеративних технологіях, в якій вимоги та рішення еволюціонують через співпрацю між командами, здатними до самоорганізації. Гнучка розробка — засіб для підвищення продуктивності розробників ПЗ.

! 2001 Agile маніфест (http://agilemanifesto.org/). Ідеї:

- **Особистості та їхні взаємодії важливіші, ніж процеси та інструменти.**
- **Робоче програмне забезпечення важливіше, ніж повна документація.**
- **Співпраця із замовником важливіша, ніж контрактні зобов'язання.**
- **Реакція на зміни важливіша, ніж дотримання плану.**

8. Agile – Принципи:

- Задоволення клієнта за рахунок ранньої та безперебійної поставки версій ПЗ.
- Вітання змін вимог навіть наприкінці розробки;
- Часта поставка робочого програмного забезпечення (кожен місяць, або тиждень, або ще частіше).
- Тісне, щоденне спілкування замовника з розробниками впродовж всього проекту.
- Проектом займаються мотивовані особистості, які забезпечені потрібними умовами роботи, підтримкою і довірою.
- Рекомендований метод передачі інформації — особиста розмова (віч-на-віч).
- Робоче програмне забезпечення — найкращий вимірювач прогресу.

8. Agile – Принципи:

- Спонсори, розробники та користувачі повинні мати можливість підтримувати постійний темп на невизначений термін.
- Простота — мистецтво не робити зайвої роботи;
- Найкращі технічні вимоги, дизайн та архітектура виходять у самоорганізованої команди.
- Постійна адаптація до мінливих обставин.

8. Agile -> Lean методологія

Lean (*Lean Software Development*) – методологія розробки програмного забезпечення, що використовує методи концепції бережливого виробництва.

Виключення втрат. Втратами вважається все, що не додає цінності для споживача: зайва функціональність, паузи в процесі розробки, нечіткі вимоги, бюрократизація, повільне внутрішнє повідомлення.

Акцент на навчанні. Короткі цикли, раннє тестування, частий зворотний зв'язок із замовником.

Гранично відстрочене прийняття рішень. Рішення слід приймати не на основі припущень і прогнозів, а після відкриття істотних фактів.

Гранично швидка доставка замовнику. Короткі ітерації.

8. Agile -> Lean методологія

Lean (*Lean Software Development*) – методологія розробки програмного забезпечення, що використовує методи концепції бережливого виробництва.

Мотивація команди. Не можна розглядати людей виключно як ресурс. Людям потрібно щось більше, ніж просто список завдань.

Інтегрування. Передати цілісну інформацію замовнику. Прагнути до цілісної архітектурі. Рефакторинг.

Цілісне бачення. Стандартизація, встановлення відносин між розробниками. Підтримка розробниками принципів ощадливості.

«Мислити широко, робити швидко, помилятися мало, вчитися стрімко».

9. XP – «екстремальна» технологія

XP (*Extreme Programming*) - різновид Agile.
Мета - поліпшення якості ПЗ та чутливість до змін у вимогах замовників.
Підтримує часті "випуски" програми у коротких циклах розробки для поліпшення продуктивність праці та покращання можливості виконання вимог замовника, що постійно змінюються.
Крім того:

- **парне програмування (2 програміста !!!)**
- **проведення обширної перевірки сирцевого коду,**
- **модульне тестування всього коду (керована тестами розробка),**
- **уникання створення функціональності до того, як вона дійсно необхідна,**
- **простота та ясність коду,**
- **очікування на зміну вимог замовників за часом та коли вимоги до продукту стають ясніші,**
- **часте спілкування із замовником та між самими програмістами.**

10. FDD – динамічна технологія

FDD (*Feature driven development*, керована функціональністю).

Спроба об'єднати найбільш визнані в індустрії розробки ПЗ методики, які беруть за основу важливу для замовника функціональність (властивості) розроблюваного ПЗ.

- Будь-яка функція, яка занадто складна для розробки протягом **ДВОХ тижнів (!)**, розбивається на менші підфункції до тих пір, поки кожна підзадача не може бути названа властивістю (бути реалізована за 2 тижні) - полегшує створення коректне працюючих функцій, розширення і модифікацію системи.
- **Регулярна збірка** гарантує, що завжди є продукт (система), яка може бути представлена замовнику, і допомагає знаходити помилки при об'єднанні частин вихідного коду на ранніх етапах.
- Кожен блок коду закріплений за **конкретним розробником** (власником). Він відповідає за узгодженість, продуктивність і цілісність свого класу.

11. MSF

MSF (Microsoft Solution Framework) – рішення Microsoft в галузі проектування, розробки, впровадження та супроводу ПЗ.

П'ять «білих книг»:

- «Модель процесів MSF»,
- «Модель проектної групи MSF»,
- «Дисципліна управління проектами MSF»,
- «Дисципліна управління ризиками MSF »
- « Дисципліна управління підготовкою MSF ».

Agile + ітеративні підходи

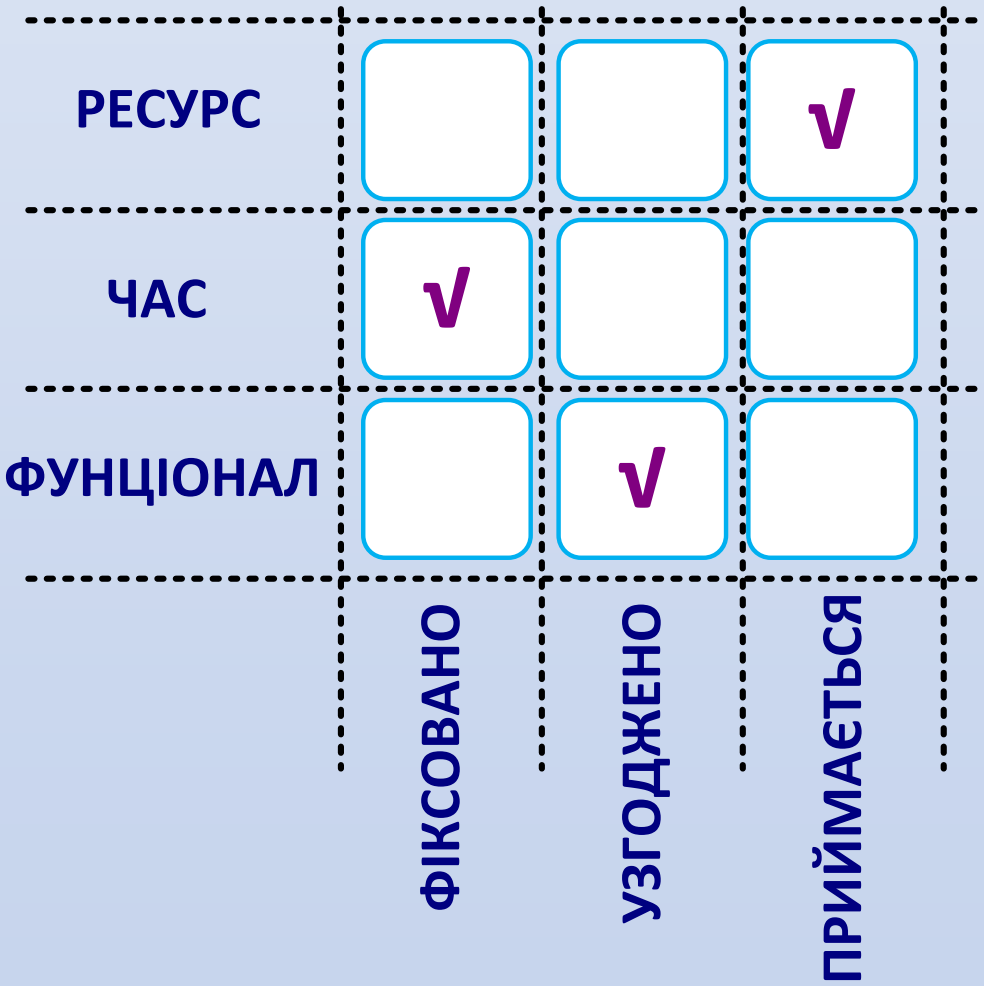
11. MSF

MSF – матриця компромісів. Узгоджується с замовником.

Ресурси:
фінанси, люди

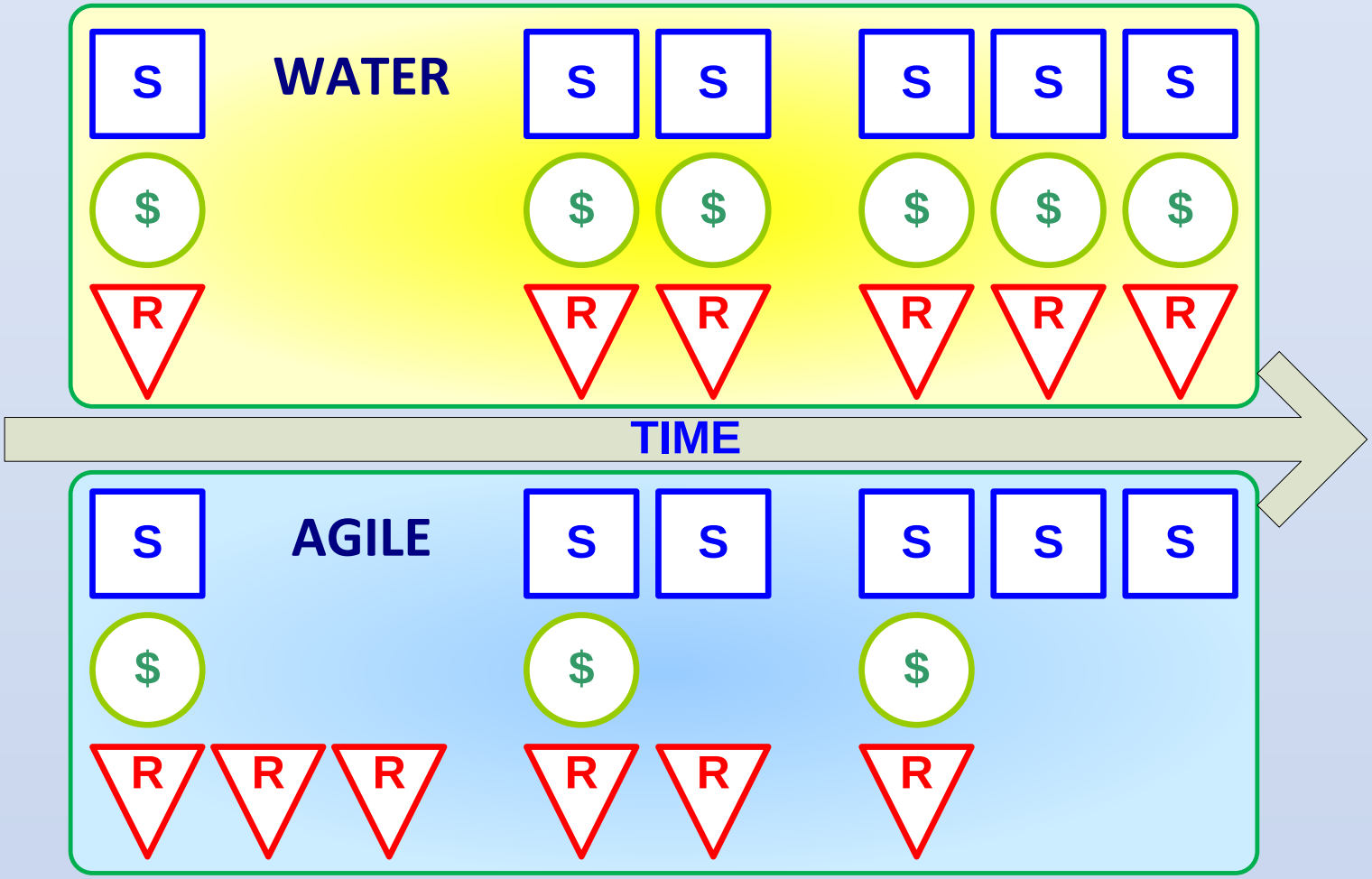
Час:
календарний графік виконання

Функціонал:
сукупність функцій системи



+ Якість (4-й вимір)

Water VS Agile



$ Вартість R Ризики S Продукт

XX. WATER – SCRUM - FALL

Water-Scrum-Fall – гібридна методологія створення програмного продукту.

Ітеративна розробка: програмування & тестування

SCRUM

SCRUM – технологія керування і контроля процесу створення програмного продукту.

Ролі:
Користувач
Клієнт
Експерт

Список вимог

Вимога, поділена на задачі

Задача => 1 доба
Щоденна нарада
Демонстрація

SCRUM

Scrum забезпечує гнучкість і орієнтованість на безперервний розвиток і зміни ПЗ.

- **На етапі планування** спринту власник продукту спілкується зі скрам-командою, визначаючи, на які завдання можна розбити призначені для користувача історії і як їх можна реалізувати.
- **Під час щоденних зборів** учасники скрам-команди обговорюють виконання кожної окремо взятої задачі і визначають можливі шляхи вирішення виниклих проблем.
- **По завершенні спринту** готовий продукт пред'являється замовнику, який може оцінити поточний функціонал і відзначити, що він хотів би змінити.

***** CALS *****

CALS (Continuous Acquisition and Life cycle Support) – концепція безперервної інформаційної підтримки поставок та життєвого циклу продукції.

Базові принципи:

- **безпаперовий обмін даними з використанням електронного цифрового підпису;**
- **аналіз і реінжиніринг бізнес-процесів;**
- **паралельний інжиніринг;**
- **системна організація поствиробничих процесів життєвого циклу виробу - інтегрована логістична підтримка.**

Головна умова ефективності CALS - наявність інтегрованої системи збору та аналізу інформації щодо продукції на всіх етапах її життєвого циклу.

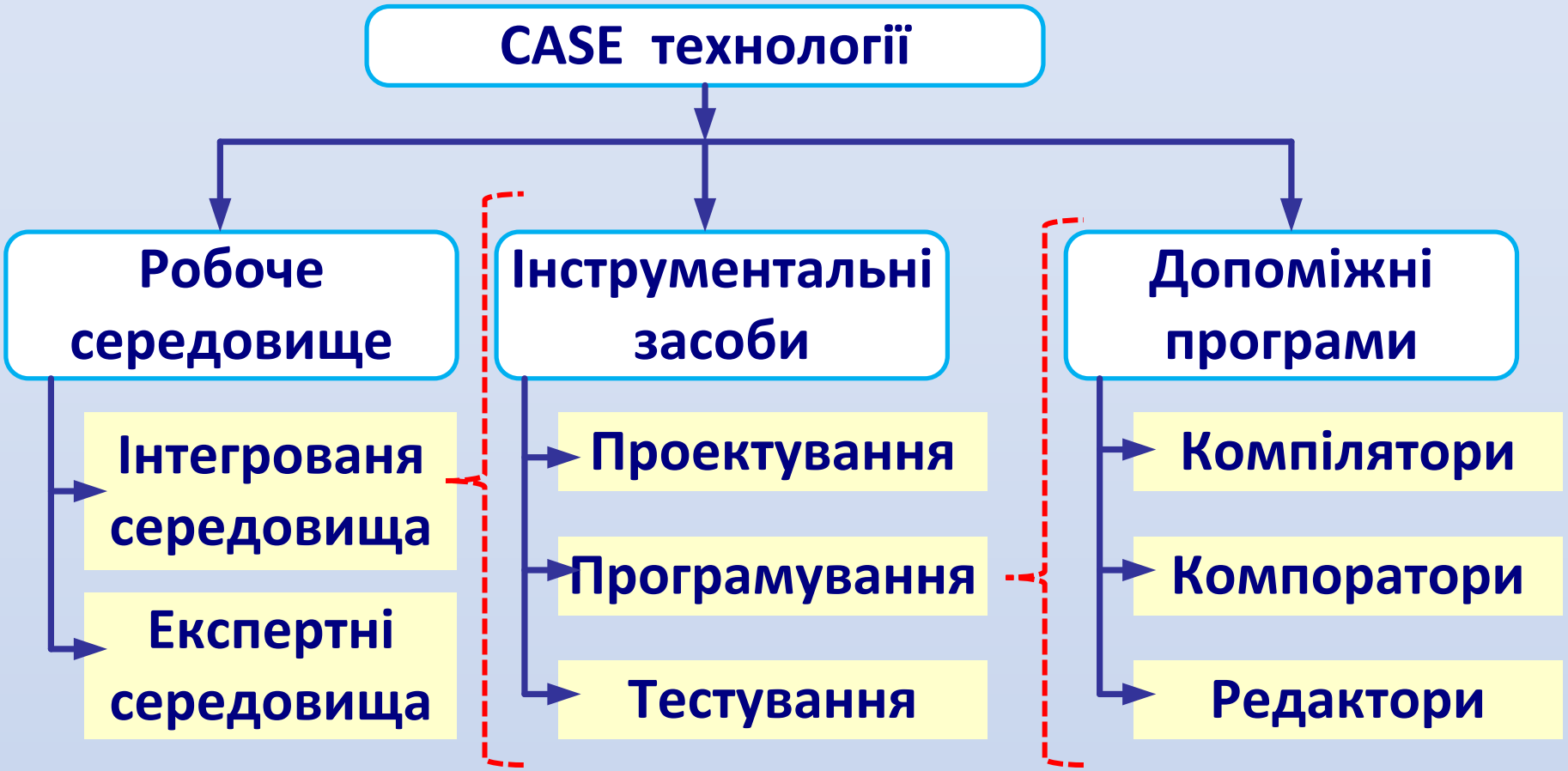
***** CASE *****

Computer Aided Software Engineering (CASE) – концепція автоматизованої розробки програмних систем .

CASE

CASE технологія :
сукупність методів проектування систем, а також набір інструментальних засобів, що дозволяють в наочній формі моделювати предметну область, аналізувати цю модель на всіх стадіях розробки і супроводу ІС і розробляти програми відповідно до інформаційних потреб користувачів.

CASE



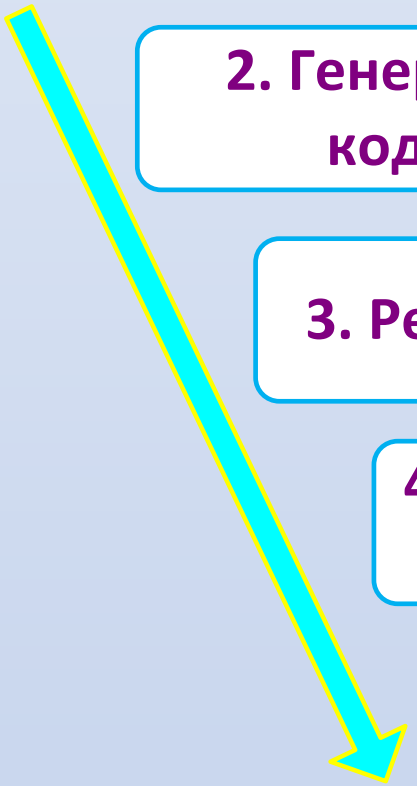
CASE. Класифікація засобів

Робочі середовища розробників (environments) - підтримують всі або більшість процесів розробки ПЗ та включають в себе кілька різних інтегрованих інструментальних засобів.

Інструментальні засоби (workbenches) - підтримують певні процеси розробки ПЗ, такі як *створення специфікації, проектування*. Зазвичай вони представляють собою набір інтегрованих допоміжних програм;

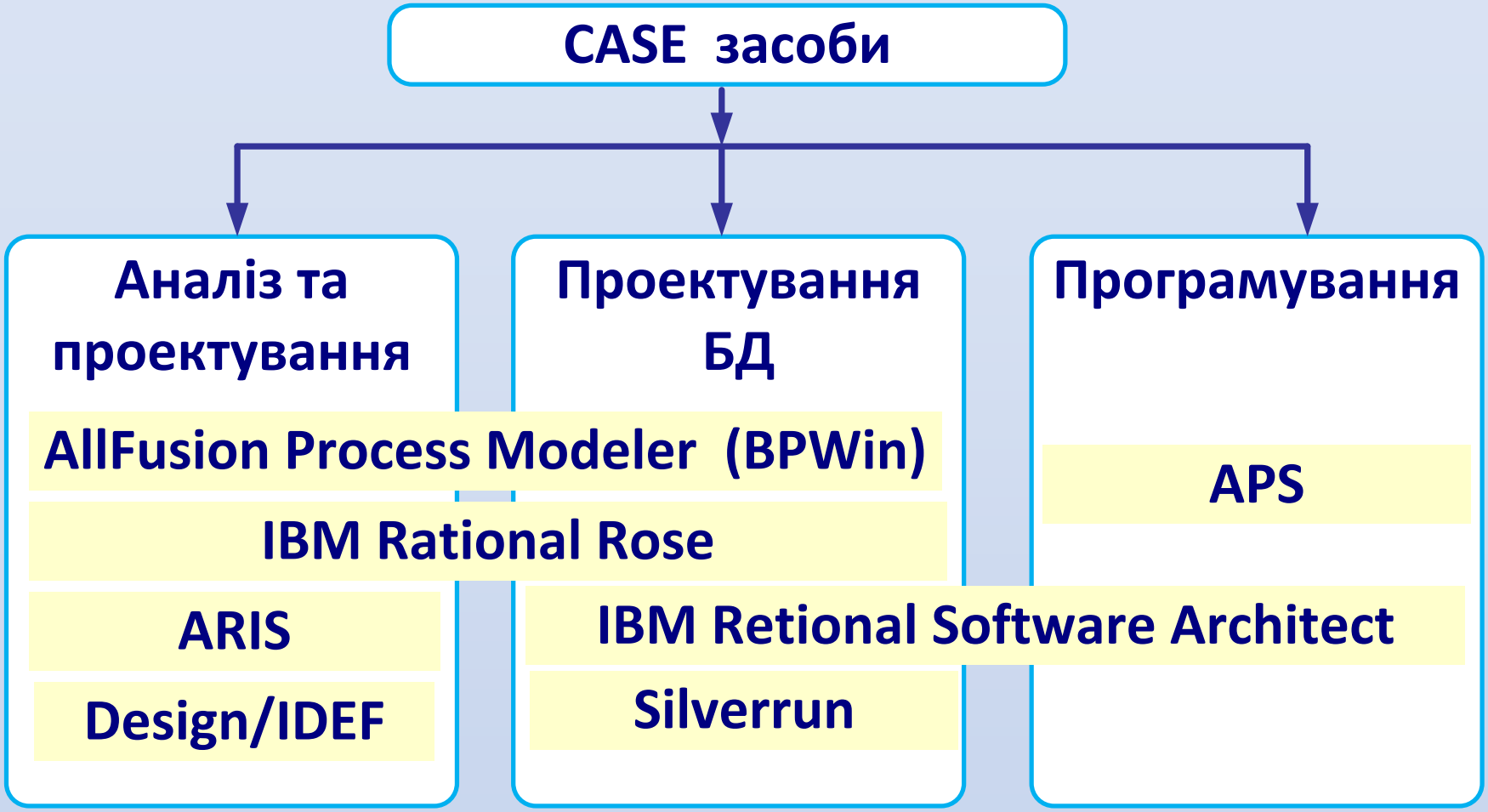
Допоміжні програми (tools) - підтримують окремі процеси розробки ПЗ (перевірка несуперечності архітектури системи, *компіляція програм, порівняння результатів тестів* і т.д.). Можуть входити до складу інструментальних засобів.

CASE. Етапи розвитку



1. Генерація схем БД — Oracle, ERWin
2. Генерація коду — Borland Together Designer
3. Реінженірінг — IBM Rational Rose, Borland Together Developer
4. Синхронізація коду і моделі — IBM Rational Software Arch, Borland Develop Studio
5. Хмарні технології — IBM Urban Cod (Docker), Microsoft Azure Kubernetes

CASE. Приклади



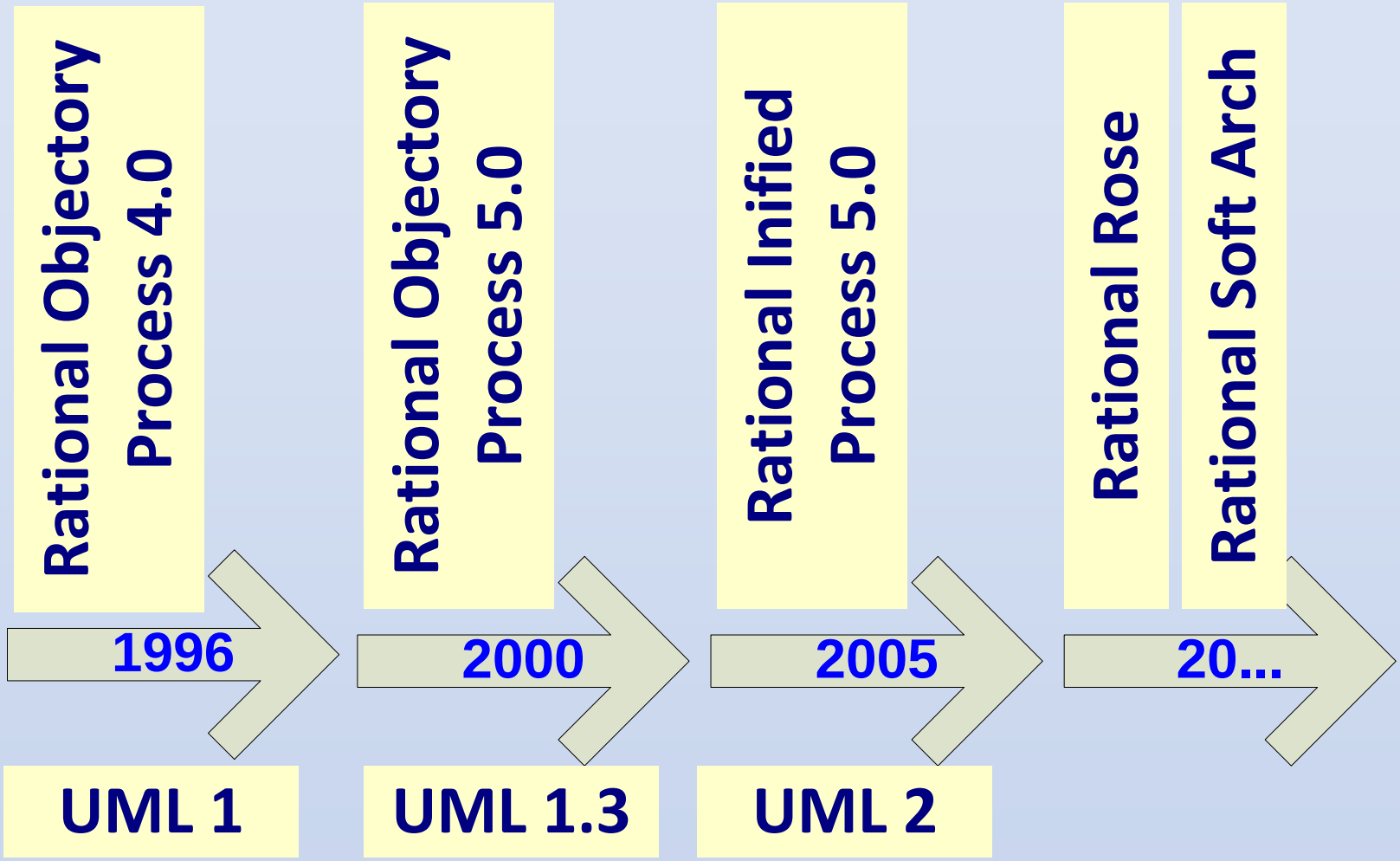
CASE: ERWin

AllFusion Process Modeler (ERWin, BPWin) – система для проектування і документування баз даних. Функціонал:

- **проектування концептуальної моделі даних,**
- **проектування логічної моделі даних,**
- **повна підтримка мови IDEF1X моделювання даних;**
- **повна інтеграція з MySQL, PostgreSQL.**

https://erwin.com/

RUP CASE: IBM


Rational Objectory Process 4.0
1996
UML 1
Rational Objectory Process 5.0
2000
UML 1.3
Rational Inified Process 5.0
2005
UML 2
Rational Rose
Rational Soft Arch
20...

RUP CASE: IBM Rational Rose

Засіб проектування архітектури, аналізу, моделювання та розробки ІС. Функціонал:

- **проектування систем будь-якої складності (від інтерфейсу програми або схеми БД до схеми системи автоматизації підприємств;**
- **повна підтримка мови UML;**
- **надання розгорнутої інформації про проект (сумісно із засобами документування);**
- **кодогенерування (стандартний список модулів включає C++, Visual Basic, XML, Oracle, …);**
- **зворотне проектування наявних систем**
- **інтеграція з Microsoft Visual Studio,**
- **повна підтримка компонентів CORBA.**

https://www.ibm.com/support/knowledgecenter/SS8PJ7_9.6.1/com.ibm.rsa_base.nav.doc/topics/crootintro_rsa_base.html

RUP CASE: IBM Rational Software Arch

Середовище розробки і моделювання на базі UML для проектування архітектури додатків на C ++ та Java (J2EE), а також веб-сервісів.

Функціонал:

- **архітектурний аналіз коду,**
- **підтримка технології MDD (model-driven development),**
- **повна підтримка мови UML для створення стійких додатків і веб-служб.**

https://www.ibm.com/us-en/marketplace/rational-software-architect-designer?mhsrc=ibmsearch_a&mhq=Rational%20Software%20Architect

Cloud - технології

IBM Smart Business Development and Test on the IBM Cloud - рішення для розробки та тестування програмного забезпечення, що включає сервіси повного циклу поставки ПЗ ***Rational Software Delivery Services for Cloud Computing***

- **Самозабезпечення образів, що складаються з базових образів, або з Red Hat Linux®, або з операційними системами SUSE Linux.**
- **Доступ до та використання програмних образів у каталозі.**
- **«Сира» обчислювальна потужність за допомогою віртуальних машин, постійного зберігання та статичних IP-адрес**
- **Підтримка, що складається з доступу до документації та форумів для самообслуговування.**
- **Моніторинг та управління керуючою інфраструктурою 24x7.**

Cloud - технології

Microsoft Azure Dev Spaces дозволяє налагоджувати і тестувати всі компоненти програми на кластерах Azure Kubernetes (AKS) з мінімальним налаштуванням комп'ютера для розробки.

Підтримується Visual Studio, що дозволяє розробляти, тестувати і розгортати функції бібліотеки класу C's в Azure для обробки масивів даних, інтеграції систем, роботи з IoT, створення мікро-сервісів.

Підтримується Visual Studio Code з розширенням Azure Dev Spaces в Linux, macOS і Windows.

https://azure.microsoft.com/ru-ru/products/visual-studio/

Рекомендована ЛІТЕРАТУРА

- **Томашевський О.М., Цегелік Г.Г., Вітер М.Б., Дудук В.Ш. Інформаційні технології та моделювання бізнес-процесів.** Навч. посіб. – К.: «Видавництво «Центр учбової літератури», 2012. – 296 с.
- **Карпенко М.Ю., Манакова Н.О., Гавриленко І.О. Технології створення програмних продуктів та інформаційних систем.** Навч. посіб. - Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім.О.М. Бекетова, 2017. – 93 с.
- **Алексенко О.В. Технології програмування та створення програмних продуктів.** Конспект лекцій. – Суми, Сумський державный університет, 2013. – 133с.

Рекомендована ЛІТЕРАТУРА

- **Иванова Г.С. Технология программирования.** Учебник. – М.: «Кнорус», 2018. – 336 с.

Посилання

- **IBM**

https://www.ibm.com/topics/software-development

- **Microsoft**

https://docs.microsoft.com/ru-ru/azure/?product=featured

Контрольні запитання

- Надайте визначення методології розробки програмних засобів та поясніть сутність та відмінності жорстких та гнучких підходів.
- Поясніть сутність каскадної та V технологій розробки програмних засобів.
- Поясніть сутність ітеративних (інкрементних) технологій розробки програмних засобів. Наведіть приклади.
- Поясніть сутність гнучких технологій розробки програмних засобів (Agile, Lean). Наведіть приклади.
- Наведіть принципи розробки ПЗ за технологією SCRUM.
- Надайте класифікацію CASE засобів та наведіть основні етапи їх розвитку. Наведіть приклади сучасних CASE засобів

The END
Mod 2. Lec 4.