Ітеративний, еволюційний та гнучкий процес

Ітеративний, еволюційний та гнучкий процес

Зміст

Ітеративний, еволюційний та гнучкий процес

Ітеративна розробка – це основа підходу до створення програмних систем.

Швидке моделювання дозволяє застосувати мову UML найбільш ефективно.

Уніфікований процес – це приклад одного із найпопулярніших ітеративних методів розробки на базі ООАП.

Ітеративний, еволюційний та гнучкий процес

Ітеративна і еволюційна розробка суттєво відрізняється від послідовного, чи каскадного (waterfall), життєвого циклу і передбачає раннє програмування та тестування частин системи в багаторазово повторюваних циклах роботи над проектом.
При використанні такого підходу розробка зазвичай починається ще до детального визначення всіх вимог. При цьому для прояснення та покращення специфікації активно застосовується зворотній зв'язок із зацікавленими особами.

В процесі ітеративної розробки прояснення вимог і проектних рішень в багатьох випадках забезпечується зворотнім зв'язком.
На відміну від цього при використанні каскадного процесу до програмування суттєві зусилля витрачаються на попереднє визначення теоретичних вимог і відпрацювання потенційних проектних рішень.

Ітеративний, еволюційний та гнучкий процес

Аналіз успішних / невдалих проектів свідчить про те, що невдалі проекти, які виконувались в рамках каскадного процесу розробки, складають більшу частину всіх невдалих проектів.

Проведені дослідження дозволяють зробити висновок про те, що ітеративні методи підвищують ймовірність успішного завершення проектів та їх ефективність, а також дозволяють отримати більш якісне програмне забезпечення.

Ітеративний, еволюційний та гнучкий процес

Зміст

Уніфікований процес

Процес розробки програмного забезпечення включає:
побудову;
розгортання;
і можливо підтримку системи.

Уніфікований процес UP (Unified Process) – це ітеративний процес розробки об'єктно-орієнтованих систем, що широко використовується. Зокрема, широку популярність набув уніфікований процес RUP (Rutional Unified Process), який забезпечує більш детальну проробку всіх етапів, визначених в процесі UP.

Уніфікований процес

Процес UP є надзвичайно гнучким та відкритим. Він дозволяє використовувати різні прийоми з інших ітеративних методів, таких як: екстремальне програмування (Extreme Programming - XP), Scrum та інші.

Наприклад, розробку на основі тестування, рефакторінг та безперервну інтеграцію метода XP, проведення семінарів та щоденних обговорень процесу Scrum можна використовувати і в межах проекту на основі UP.

В уніфікованому процесі знайшов відображення і отримав детальний опис найбільш успішний досвід розробки систем, зокрема ітеративний життєвий цикл і оцінка ризиків.

Уніфікований процес

Висновок:
Уніфікований процес UP – це ітеративний процес. Принципи ООА/П добре описуються в контексті ітеративної розробки, яка зарекомендувала себе найкращим чином при створенні багатьох систем.

Процес UP є дуге гнучким і може включати прийоми з інших методів розробки (таких як XP, Scrum).

Ітеративний, еволюційний та гнучкий процес

Зміст

Ітеративна та еволюційна розробка

Розробка виконується у вигляді декількох короткострокових міні-проектів фіксованої тривалості (наприклад три тижні), які називаються ітераціями. Кожна ітерація включає свої етапи аналізу вимог, проектування, реалізації і завершується тестуванням, інтеграцією і створенням робочої частини системи.

Ітеративний життєвий цикл грунтується на постійному розширенні і доповненню системи в процесі декількох ітерацій з періодичним зворотнім зв'язком та адаптацією модулів, що додаються до існуючого ядра. Система постійно розширюється крок за кроком, тому такий підхід іноді називають ітеративною і інкрементальною розробкою.

Оскільки зворотній зв'язок і адаптація призводять до розвитку отриманих раніше специфікацій і проектних рішень, цей підхід іноді називають ітеративною і еволюційною розробкою.

Перші ідеї ітеративного процеса називались "проектуванням по спіралі і еволюційною розробкою".

Ітеративна та еволюційна розробка

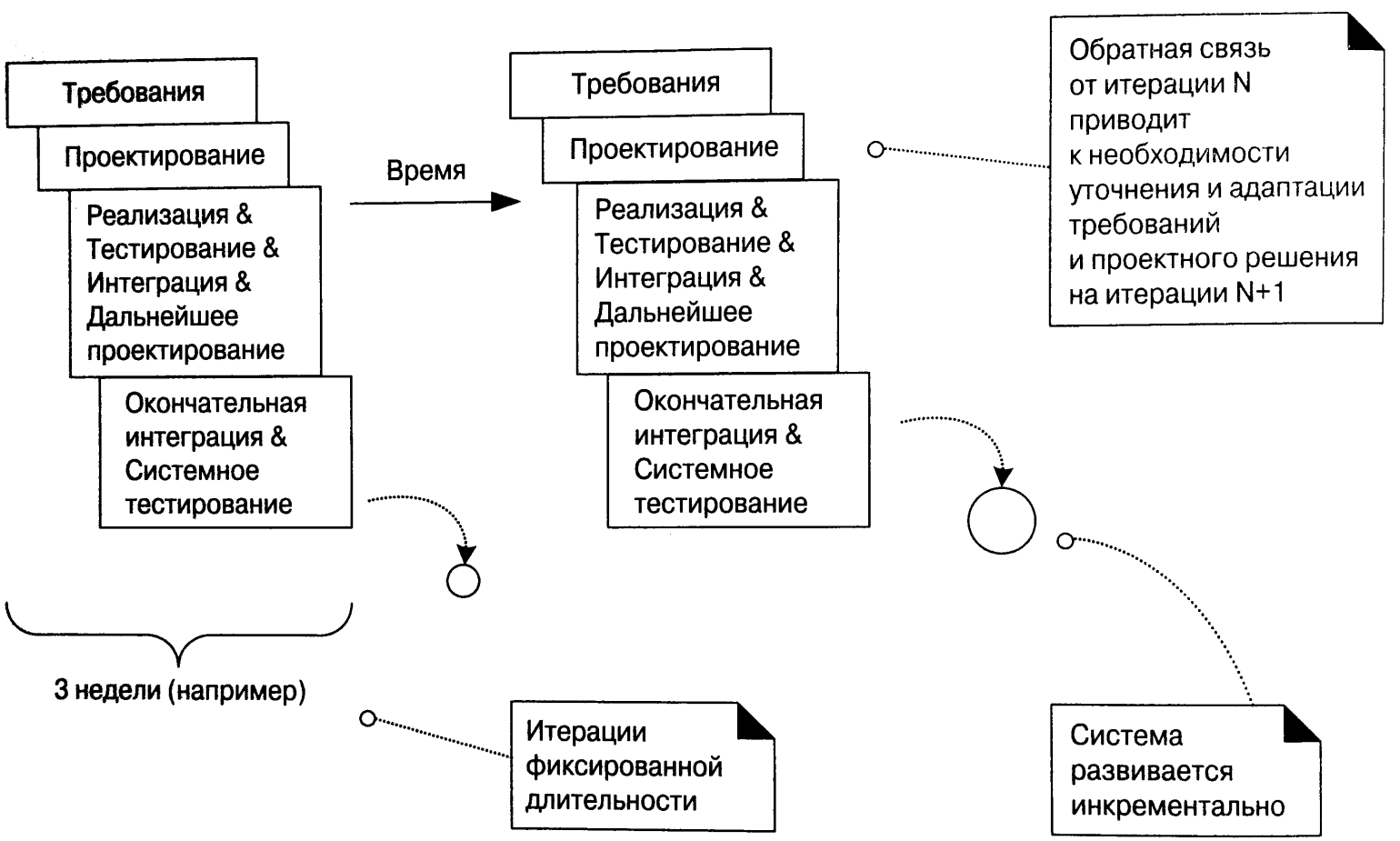


Рис. 2.1. Итеративная и эволюционная разработка

Пр.: Рис. 2.1.

Управління змінами в процесі ітеративної розробки

“Допускати зміни” – девіз ітеративного процесу розробки.
Замість того щоб боротися за створення кінцевого і незмінного варіанту моделі (як правило, без успіху) та намагатися коректно сформулювати, зафіксувати, підписати та “заморозити” набір вимог і модель до початку її програмної реалізації (як в каскадному процесі), набагато краще усвідомити важливість (і неминучість) постійної модифікації і адаптації моделі в процесі розробки системи і змиритися з необхідністю регулярного внесення змін.
На кожній ітерації розробки розглядається невелика підмножина вимог, для задоволення яких швидко розробляється, реалізується і тестується невелика частина системи.
Завдяки ранньому зворотньому зв’язку кінцевий користувач отримує можливість швидко побачити частину системи і зрозуміти точніше, що йому потрібно (виникають нові вимоги). Це не є сигналом тривоги. Це нормальний шлях розвитку і руху до досконалості.
При такому способі розробки досягається компроміс між стійкістю і мінливістю.

Управління змінами в процесі ітеративної розробки

Крім прояснення системних вимог, ранній зворотній зв'язок та тестування кожної невеликої частини системи дозволяють оцінити переваги та недоліки обраної стратегії, краще оцінити можливі ризики і критичні моменти на початковій стадії проекту, ніж на кінцевому етапі реалізації.

На початкових ітераціях відхилення від правильного шляху щодо розвитку системи (з точки зору кінцевого формулювання вимог і проектних рішень) набагато істотніше, ніж на останніх ітераціях.

Переваги ітеративної розробки

Меньший ризик невдалого завершення проекту чи отримання неякісного програмного коду, більша продуктивність.

Своєчасне (ранішнє) усвідомлення можливих технічних ризиків, осмислення вимог, задач проекту і зручність використання системи.

Швидкий та помітний прогрес.

Ранній зворотній зв'язок, можливість врахування побажань користувачів і адаптації системи. Як результат система більш точно задовольняє реальні потреби керівників проекту і користувачів.

Керована складність. Команда розробників не перевантажена зайвою роботою на етапі аналізу і проектування і не "паралізована" занадто складними і довготривалими задачами.

Отриманий при реалізації кожної ітерації досвід можливо методично використовувати для покращення самого процесу розробки.

Ітеративний, еволюційний та гнучкий процес

Зміст

Тривалість ітерації

Рекомендується встановлювати тривалість кожної ітерації в межах від двох до шести тижнів.
Якщо тривалість меньша двох тижнів, то розробникам не залишається часу для виконання істотної частини робіт і отримання зворотнього зв'язку.
Якщо тривалість довша ніж 6 тижнів, то поставлені завдання можуть виявитися досить складними. Крім цього відкладається також і момент отримання зворотнього зв'язку.

Дуже важливо, щоб тривалість ітерацій була фіксованою.
Якщо за вказаний період виконати всі завдання (розробити, реалізувати намічену частину системи, провести її інтеграцію, тестування і адаптацію) нереально, потрібно скорочувати список вимог для даної ітерації і переносити їх реалізацію на наступний цикл, але ні в якому разі не переміщувати дату завершення ітерації.

Ітеративний, еволюційний та гнучкий процес

Зміст

Необхідність зворотнього зв’язку

Для уточнення вимог. На ранніх стадіях процесу розробки, коли розробники аналізують специфікації і користуються демонстраційними матеріалами, отриманими від замовників.

Зворотній зв’язок від фахівців з тестування і розробників, яка забезпечує можливість уточнення проектних рішень чи моделей.

Зоротній зв’язок від менеджерів, які на основі вже реалізованих властивостях системи уточнюють графік робіт і оцінку ходу виконання проекту.

Зворотній зв’язок від користувачів та інших зацікавлених осіб, які можуть змінити пріоритет реалізації різних функцій системи на подальших ітераціях.

Ітеративний, еволюційний та гнучкий процес

Зміст

Ітеративне планування на основі ризиків з урахуванням вимог

Процес UP підтримує ітеративне планування, яке управляється як ризиками, так і вимогами користувачів.

Це значить, що цілі, які вибираються на ранніх ітераціях, повинні забезпечити:
ідентифікацію та врахування найбільших ризиків;
реалізацію найбільш важливих для користувача властивостей системи.

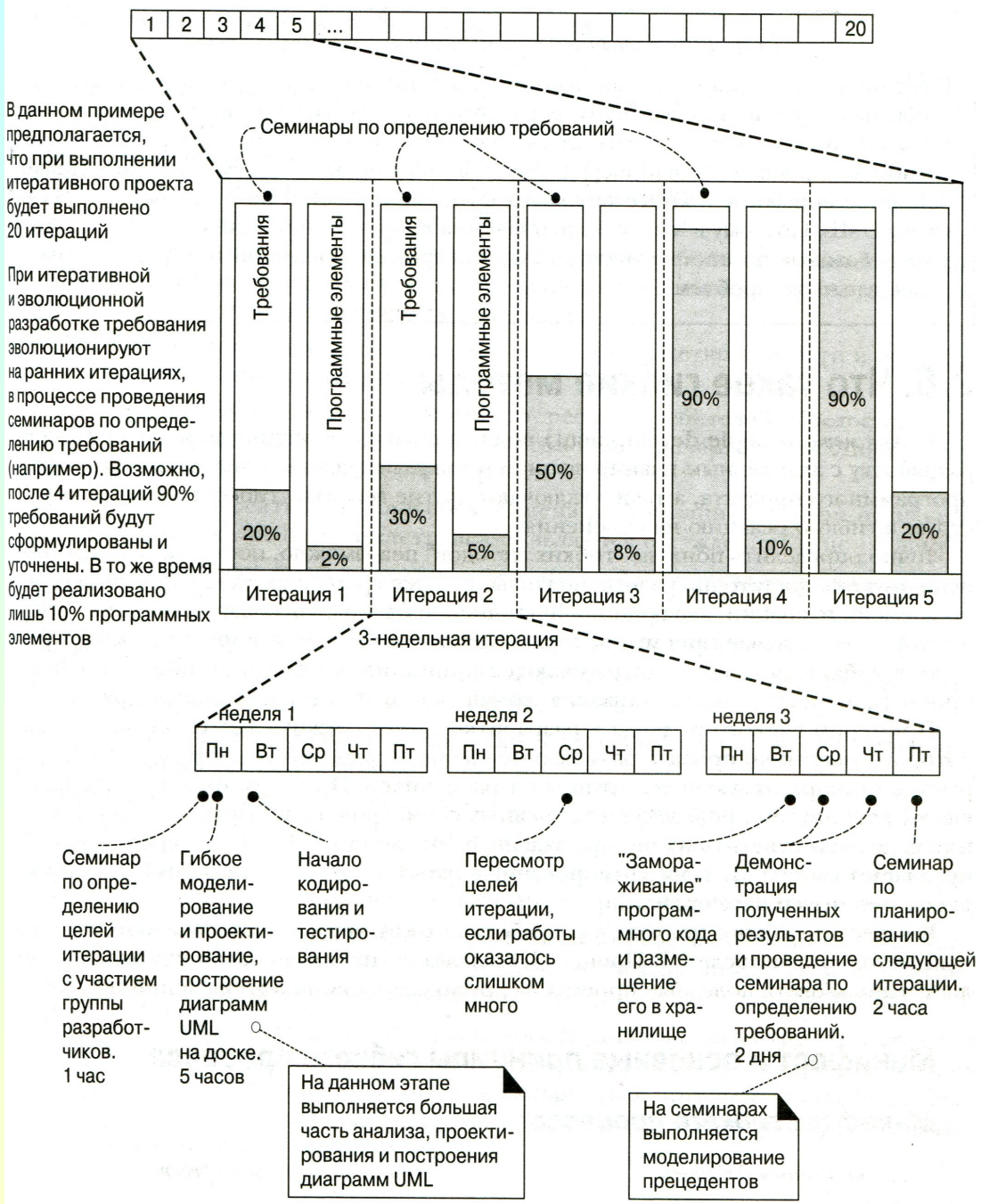


Рис. 2.4. Эволюционный анализ и проектирование – это основные виды деятельности на ранних итерациях

Рис. 2.4.

Ітеративний, еволюційний та гнучкий процес

Зміст

Гнучкі методи

Гнучкі методи (agile development) підтримують ітеративну і еволюційну розробку з адаптивним плануванням і інкрементальним отриманням кінцевого програмного продукту.

Прикладом гнучкого підходу до розробки може служити метод Scrum, в рамках якого учасники проекту розміщуються в загальній робочій кімнаті, а також формують самоорганізовані групи розробників. При цьому їх координація здійснюється шляхом проведення щоденних семінарів, на яких кожний із учасників повинен відповісти на 4 заданих йому питання.

Метод XP пропонує виконувати програмування в парах, а також вести розробку на основі ризиків (test-driven development).

Основні принципи гнучкого процесу

Найвищий пріоритет – задовольнити потребу користувачів шляхом раннього і неперервного надання працездатного програмного забезпечення, яке реалізує відповідні функції.

Необхідність змін допускається навіть на пізних етапах виконання проекту. Гнучкі процеси дозволяють врахувати зміну вимог і надати користувачам ще більші переваги.

Як можна частіше потрібно представляти працездатне програмне забезпечення (кожні декілька тижнів або декілька місяців) і намагатися скоротити ці часові інтервали.

На протязі всього проекту фахівці в предметній галузі і розробники повинні щоденно працювати разом.

Основу проекту повинні складати інтереси споживачів. Надайте їм середовище для роботи, враховуйте їх потреби і довіряйте їм. Тільки в цьому випадку можна успішно завершити проект.

Основні принципи гнучкого процесу

Найбільш ефективним способом обміну інформацією всередині окремих груп розробників і між групами є обговорення “обличчям до обличчя”.

Працездатне програмне забезпечення – це основна міра оцінки розвитку проекту.

Гнучкі методи сприяють стійкому процесу розробки.

Спонсори, розробники і користувачі повинні підтримувати постійний темп розвитку проекту незалежно один від одного.

Постійне прагнення до технічного удосконалення та покращення проектних рішень підвищують гнучкість.

Забезпечення простоти – це мистецтво максимізації кількості роботи, яку не потрібно робити. Це дуже важливо!

Основні принципи гнучкого процесу

Найкращу архітектуру і проектні рішення, а також вимоги можуть розробити тільки самоорганізовані групи розробників.

Розробники повинні регулярно аналізувати стан проекту і думати, як можна підвищити його ефективність. Потім в організацію робіт можна внести відповідні зміни.

Гнучке моделювання

Прийоми гнучкого моделювання:
Використання гнучкого методу зовсім не означає, що моделювання взагалі не виконується.

Моделювання і моделі (мовою UML) в основному використовуються для покращення розуміння проблеми, яка вирішується і взаємодії розробників, а не документування.

Не потрібно будувати модель чи використовувати мову UML для опису всіх чи більшої частини проектних рішень. Рішення простих та очевидних проблем краще відкласти на стадію програмування та тестування.
Виконуйте моделювання і використовуйте UML для невеликої частини проектних рішень, які є не очевидними чи тяжкими для розуміння.

Гнучке моделювання

Використовуйте прості інструменти, які підвищують криативність, не потребують надмірних зусиль (розумових, часових та інших) і дозволяють швидко вносити відповідні зміни. Крім цього важливо розгорнути великий візуальний простір.

Не слід виконувати моделювання на одинці. Краще розділитися на пари або трійки і малювати діаграми на білій дошці. При цьому не слід забувати про основну мету моделювання: вивчити, розібратися в існуючій проблемі, а потім поділитися отриманими знаннями з іншими розробниками. Організуйте роботу так, щоб всі могли прийняти участь в процесі моделювання.

Створюйте моделі паралельно. Наприклад: діаграму взаємодії та статичну діаграму класів.

Гнучке моделювання

При розробці моделей на дошці використовуйте просту систему позначень, яка краще всього підходить до конкретної ситуації.

Враховуйте той факт, що всі моделі скоріш за все виявляться неточними, а програмний код і проектні рішення - зовсім іншими.

Розробники повинні виконувати моделювання і об'єктно-орієнтоване проектування для підвищення власного розуміння проблеми, а не для того, щоб передати побудовані діаграми іншим розробникам.

Інші важливі принципи уніфікованого процесу

Оцінка ризиків та ключових моментів проекту на різних ітераціях.

Постійний відгук користувачів, зворотній зв'язок і модифікація вимог.

Побудова загальної базової архітектури на ранніх ітераціях.

Постійний контроль якості, раннє і часте тестування в реальних умовах.

Використання прецедентів.

Візуалізація програмної моделі (за допомогою мови UML).

Уважне управління вимогами.

Обробка запитів на внесення змін і управління конфігурацією.

Ітеративний, еволюційний та гнучкий процес

Зміст

Фази уніфікованого процесу

Початок – визначення початкового бачення проблеми, прецедентів, оцінка складності задачі.

Розвиток – формування більш повного бачення проблеми, ітеративна реалізація базової архітектури, створення найбільш критичних компонентів (вирішення високих ризиків), ідентифікація основних вимог, отримання більш реалістичних оцінок.

Конструювання – ітеративна реалізація меньш критичних і простих елементів, які залишилися не реалізованими, підготовка до розгортання.

Передача – бета-тестування і розгортання.

Фази уніфікованого процесу

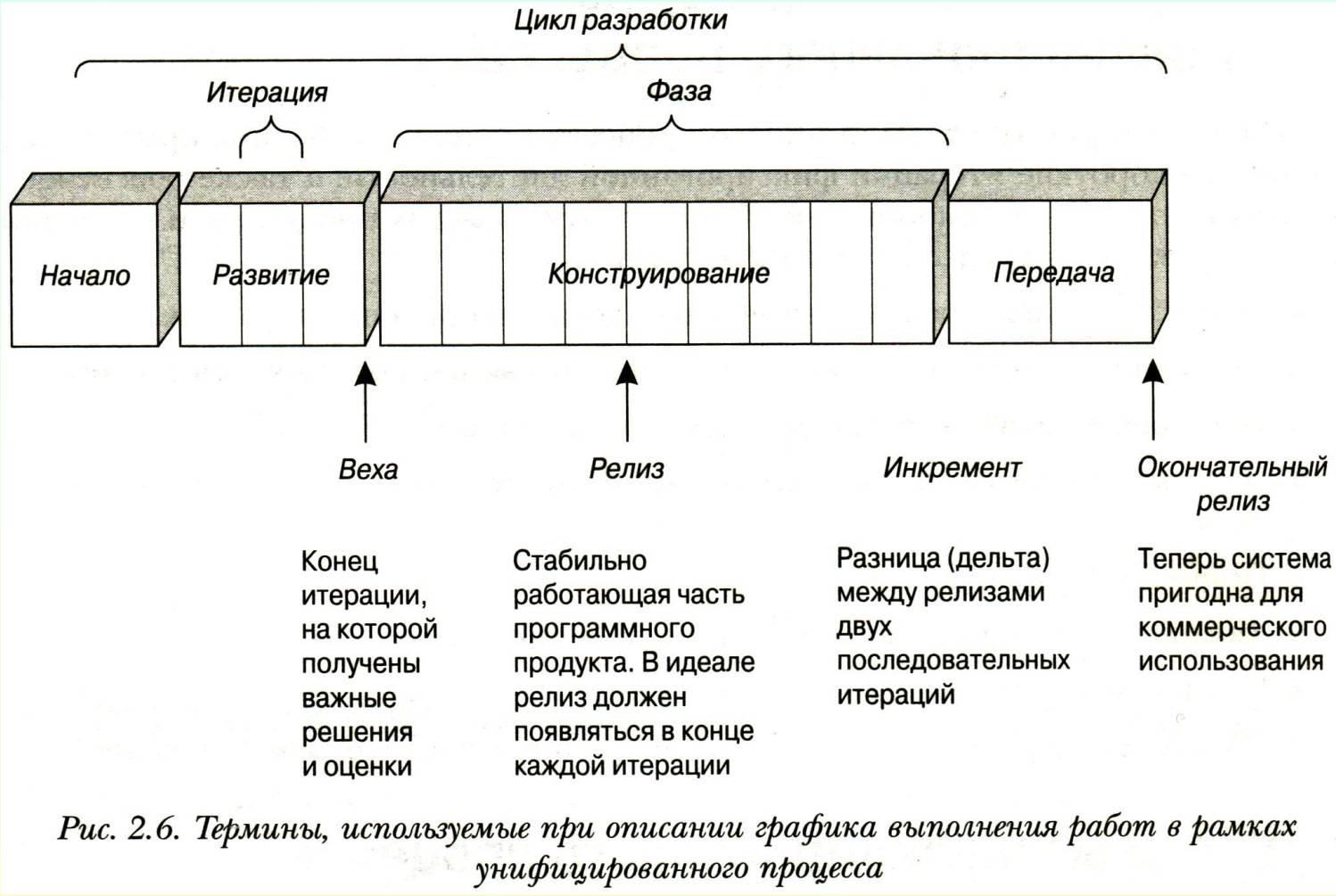


Рис. 2.6. Термины, используемые при описании графика выполнения работ в рамках унифицированного процесса

Рис. 2.6.

Ітеративний, еволюційний та гнучкий процес

Зміст

Дисципліни уніфікованого процесу

Бізнес-моделювання – розробка моделі предметної галузі, яка є візуальним представленням найбільш важливих сутностей із предметної галузі і їх взаємозв'язків.

Вимоги – створюється модель прецедентів і додаткова специфікація. У цих двох артифактах відображаються функціональні і нефункціональні вимоги.

Проектування – створюється модель проектування, яка відображає програмні об'єкти.

Реалізація – програмування і побудова системи.

Тестування.

Розгортання.

Конфігурація і управління змінами.

Управління проектом.

Середовище – встановлення необхідних засобів і налаштування процесу діяльності в рамках проекту.

Дисципліни уніфікованого процесу

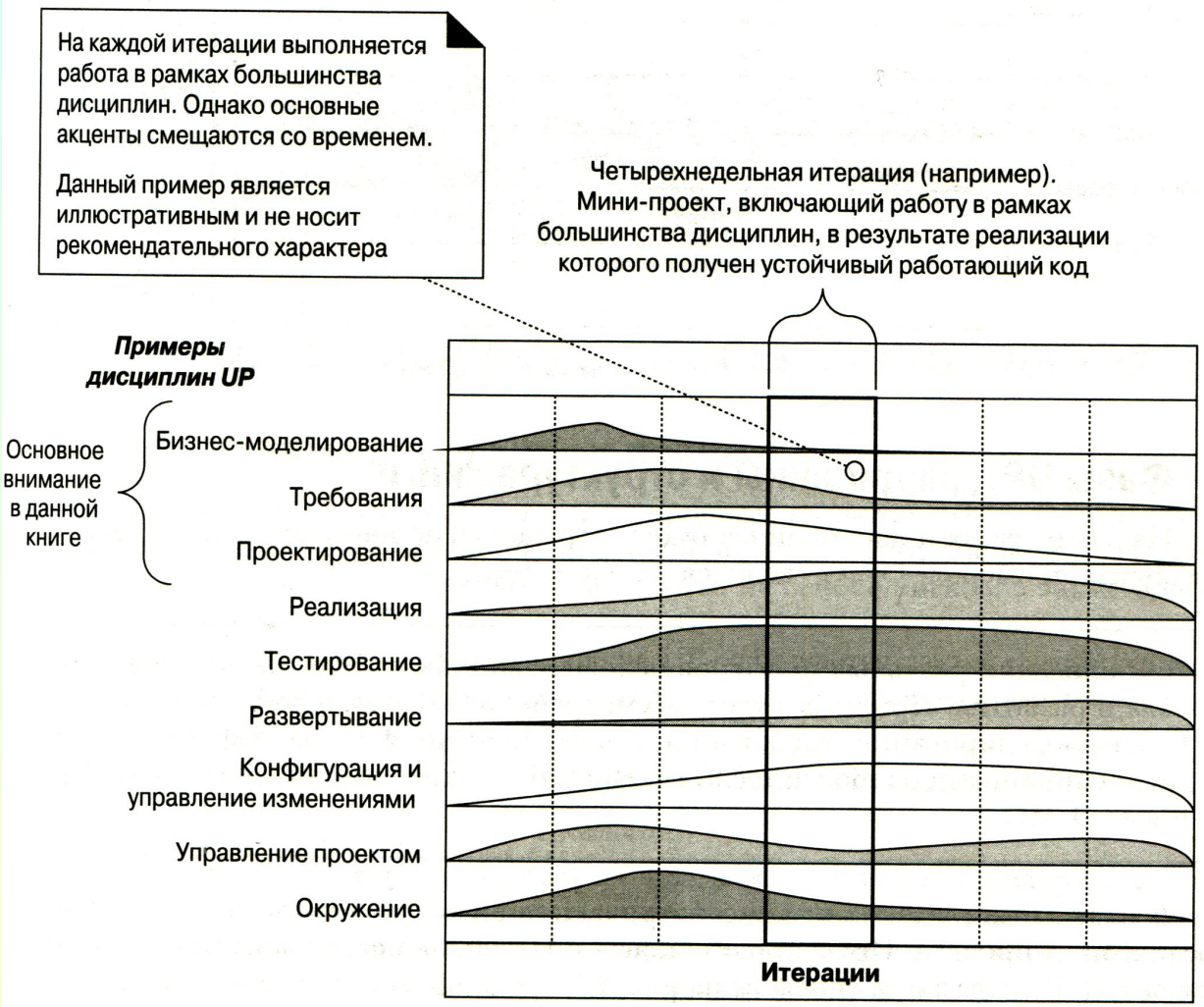


Рис. 2.7. Дисциплины унифицированного процесса

Рис. 2.7.

Дисципліни і фази

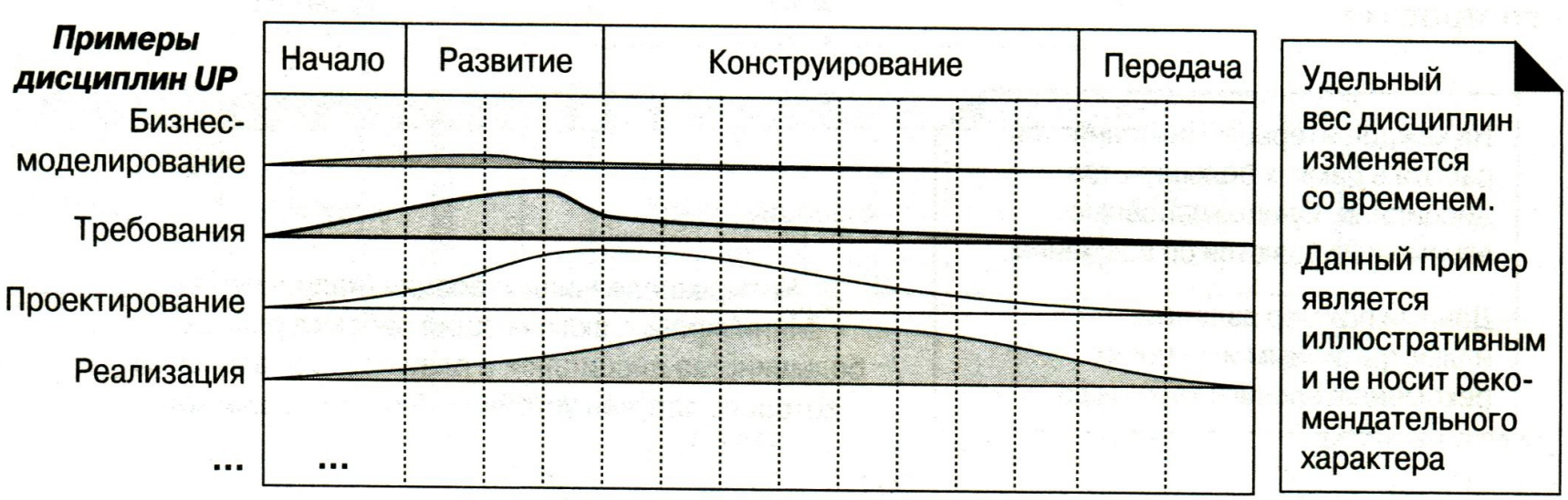


Рис. 2.8. Дисциплины и фазы

Рис. 2.8.

Ітеративний, еволюційний та гнучкий процес

Зміст

Налаштування процесу і вибір інструментів

Майже всі артефакти уніфікованого процесу є необов’язковими (за винятком, можливо, коду)

Таблица 2.1. Примерный набор инструментов унифицированного процесса (н — начало, р — развитие)

Дисциплина	Приемы	Артефакт / Итерация→	Начало I1	Развитие E1..En	Конструирование C1..Cn	Передача T1..Tn
Бизнес-моделирование	Гибкое моделирование, регулярные семинары	Модель предметной области		н		
Требования	Регулярные семинары	Модель прецедентов	н	р		
		Видение системы	н	р		
		Дополнительная спецификация	н	р		
		Словарь терминов	н	р		
Проектирование	Гибкое моделирование, разработка на основе рисков	Модель проектирования		н	р	
		Описание архитектуры		н		
		Модель данных		н	р	
Реализация	Разработка на основе рисков, парное программирование, непрерывная интеграция	...				
Управление проектом	Гибкое управление, ежедневные семинары в стиле Scrum	...				

Таблиця 2.1.

Ітеративний, еволюційний та гнучкий процес

Зміст

Заключна частина

Rational Unified Process (RUP) - методологія розробки програмного забезпечення, створена компанією Rational Software. Rational Software — компания-разработчик программного обеспечения.
До 2003 года Rational была независимой компанией, в 2003 году её поглотила корпорация IBM.

В основі RUP лежать наступні принципи:

Рання ідентифікація і безперервне (до закінчення проекту) усунення основних ризиків.

Концентрація на виконанні вимог замовника до програми (аналіз і побудова моделі прецедентів (варіантів використання)).

Очікування змін у вимогах, проектних рішеннях і реалізації в процесі розробки.

Компонентна архітектура, реалізована і тестована на ранніх стадіях проекту.

Постійне забезпечення якості на всіх етапах розробки проекту (продукту).

Робота над проектом в згуртованій команді, ключова роль в якій належить архітекторам.

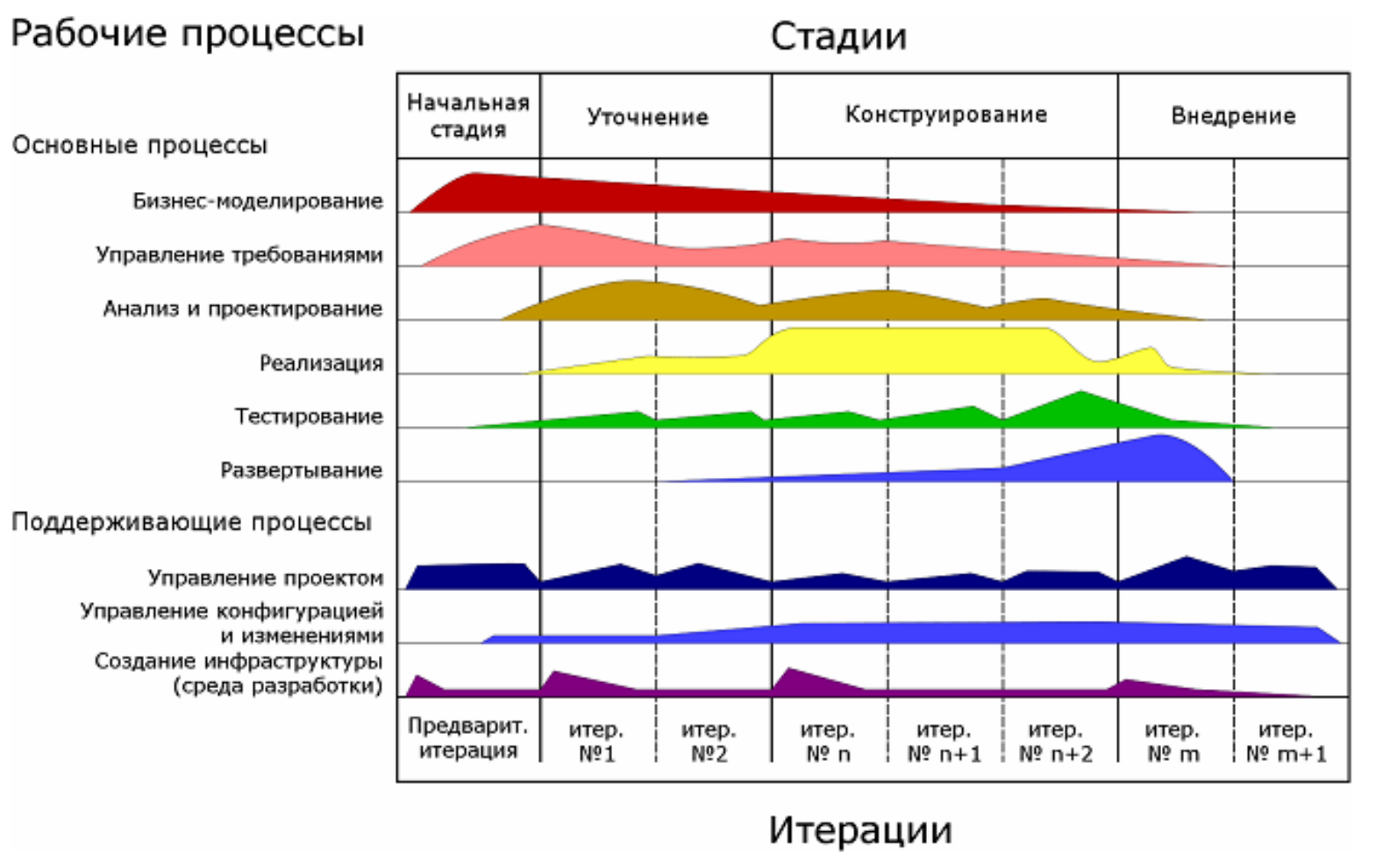
Заключна частина

RUP використовує ітеративну модель розробки.

В кінці кожної ітерації (в ідеалі триваючої від 2 до 6 тижнів) проектна команда повинна досягти запланованих на дану ітерацію цілей, створити або допрацювати проектні артефакти і отримати проміжну, але функціональну версію кінцевого продукту.

Ітеративна розробка дозволяє швидко реагувати на мінливі вимоги, виявляти і усувати ризики на ранніх стадіях проекту, а також ефективно контролювати якість створюваного продукту.

Дисципліни і фази



Повний життєвий цикл розробки продукту складається з чотирьох фаз,кожна з яких включає в себе одну або декілька ітерацій

Заключна частина

1. Початкова стадія (Inception)

У фазі початковій стадії:
Формуються бачення і межі проекту.
Створюється економічне обґрунтування (business case).
Визначаються основні вимоги, обмеження і ключова функціональність продукту.
Створюється базова версія моделі прецедентів.
Оцінюються ризики.

При завершенні початкової фази оцінюється досягнення етапу життєвого циклу мети (англ. Lifecycle Objective Milestone), яке передбачає угоду зацікавлених сторін про продовження проекту.

Заключна частина

2. Уточнення (Elaboration)

У фазі «Уточнення» проводиться аналіз предметної галузі та побудова виконавчої архітектури. Це включає в себе:

Документування вимог (включаючи детальний опис для більшості прецедентів).

Спроектовану, реалізовану і відтестовану виконавчу архітектуру.

Оновлене економічне обґрунтування і більш точні оцінки термінів і вартості.

Знижені основні ризики.

Успішне виконання фази розробки означає досягнення етапу життєвого циклу архітектури (англ. Lifecycle Architecture Milestone).

Заключна частина

3. Побудова (Construction)

У фазі «Побудова» відбувається реалізація більшої частини функціональності продукту. Фаза Побудова завершується першим зовнішнім релізом системи і віхою початкової функціональної готовності (Initial Operational Capability).

Заключна частина

4. Впровадження (Transition)

У фазі «Впровадження» створюється фінальна версія продукту і передається від розробника до замовника. Це включає в себе програму бета-тестування, навчання користувачів, а також визначення якості продукту. У разі, якщо якість не відповідає очікуванням користувачів або критеріям, встановленим у фазі Початок, фаза Впровадження повторюється знову. Виконання всіх цілей означає досягнення віхи готового продукту (Product Release) та завершення повного циклу розробки.

Ітеративний, еволюційний та гнучкий процес

Дякую за увагу