# Фреймворки(каркаси)

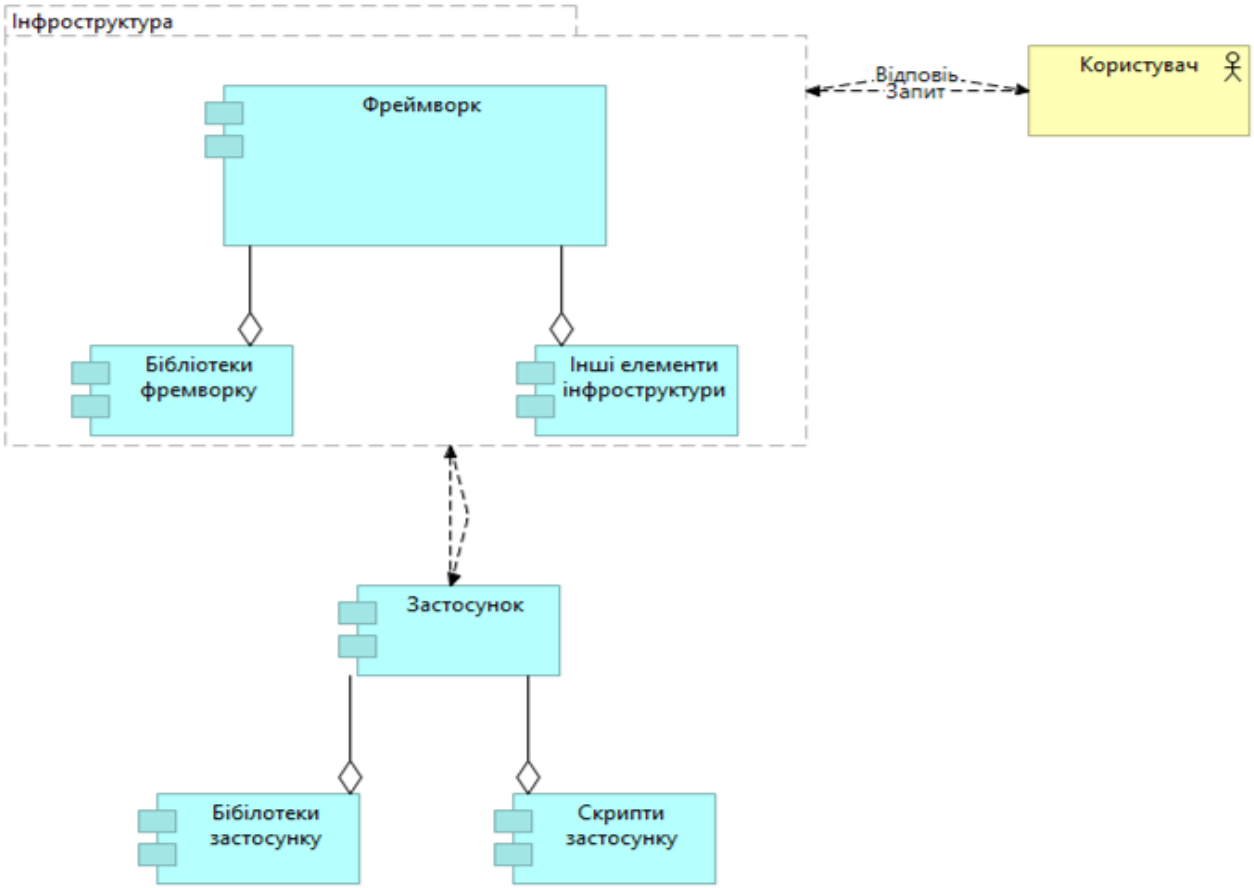
# Поняття фреймворку

**Фреймворк**

- загальноприйняті архітектурно-структурні рішення й підходи до проектування

- ❑ загальне рішення складного завдання

Схема фреймворка, це ідеалістичний підхід до програмування. Де в основі лежить абстракція (душа), а деталі тлінні і повинні їй підкорятися. Це випливає з останнього принципу SOLID - DIP (Dependency inversion principle). Ця архітектура передбачає, що первинний фреймворк. Він "вбирає" в себе користувальницькі скрипти (контролери, моделі та ін) і рулить сам, надаючи свої внутрішні можливості. Інфраструктура фреймворка досить об'ємна.



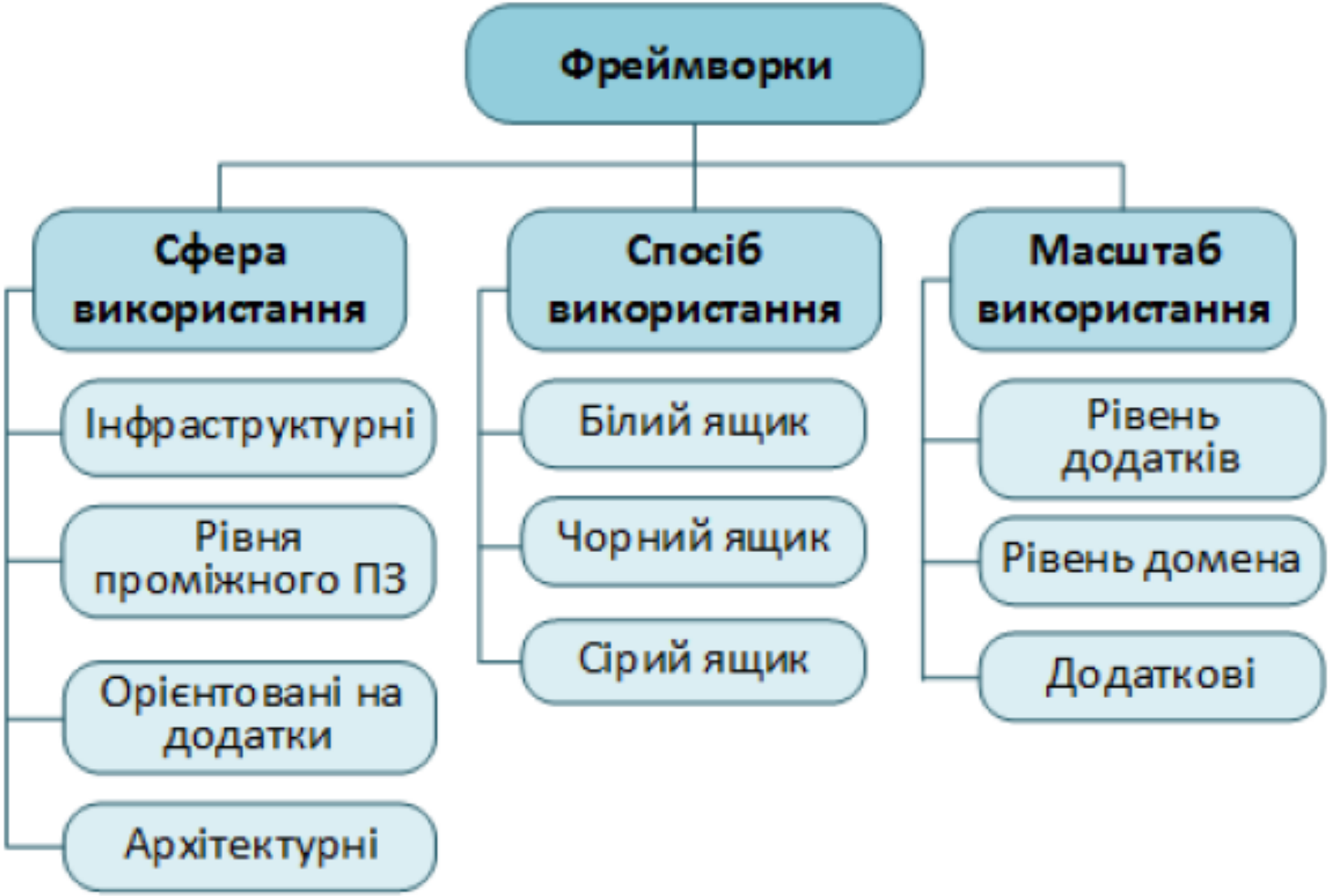
## Фреймворк (плюси)

1. Швидкість і простота розробки додатків (скриптів).
2. Деяка уніфікація.
3. Відповідно простіше супроводженість
4. Накладення обмежень на членів команди. Це важливо, якщо персонал малокваліфікований і не викликає особливої довіри. Тоді їм легше обмежити "зону досяжності".
5. Місцями багостойкість.
6. Кон’юнктура. Саморобку на фреймворку легше продати, так як вважається, що для її супроводу потрібно менш кваліфікований, а значить більш дешевий персонал.

## Фреймворк (мінуси)

1. Висока ресурсомісткість.
2. Наявність неперевизначаємого функціоналу.
3. Наявність "мертвого" функціоналу.
4. Специфічні правила на межі власного синтаксису, що вимагає великої кількості складної документації. Що відволікає від кодування продукту.
5. Залежність. Так як основний принцип архітектури фреймворка DIP, то бібліотеки пишуться практично як плагіни до нього. А значить це дуже сильно знижує їх переносимість.
6. Інкапсуляція. Це мінус в глобальному сенсі. Розробники додатків стають заручниками розробників фреймворка. І не можуть ні вплинути на нього, ні захиститися від багів, можуть допустити другі.

# Класифікація фреймворків



- Інфраструктурні фреймворки (System Infrastructure Frameworks) спрощують процес розробки інфраструктурних елементів, застосовуються усередині організації й не продаються.
- Фреймворки рівня проміжного програмного забезпечення (Middleware Frameworks) застосовуються для вбудовування додатків і компонентів.

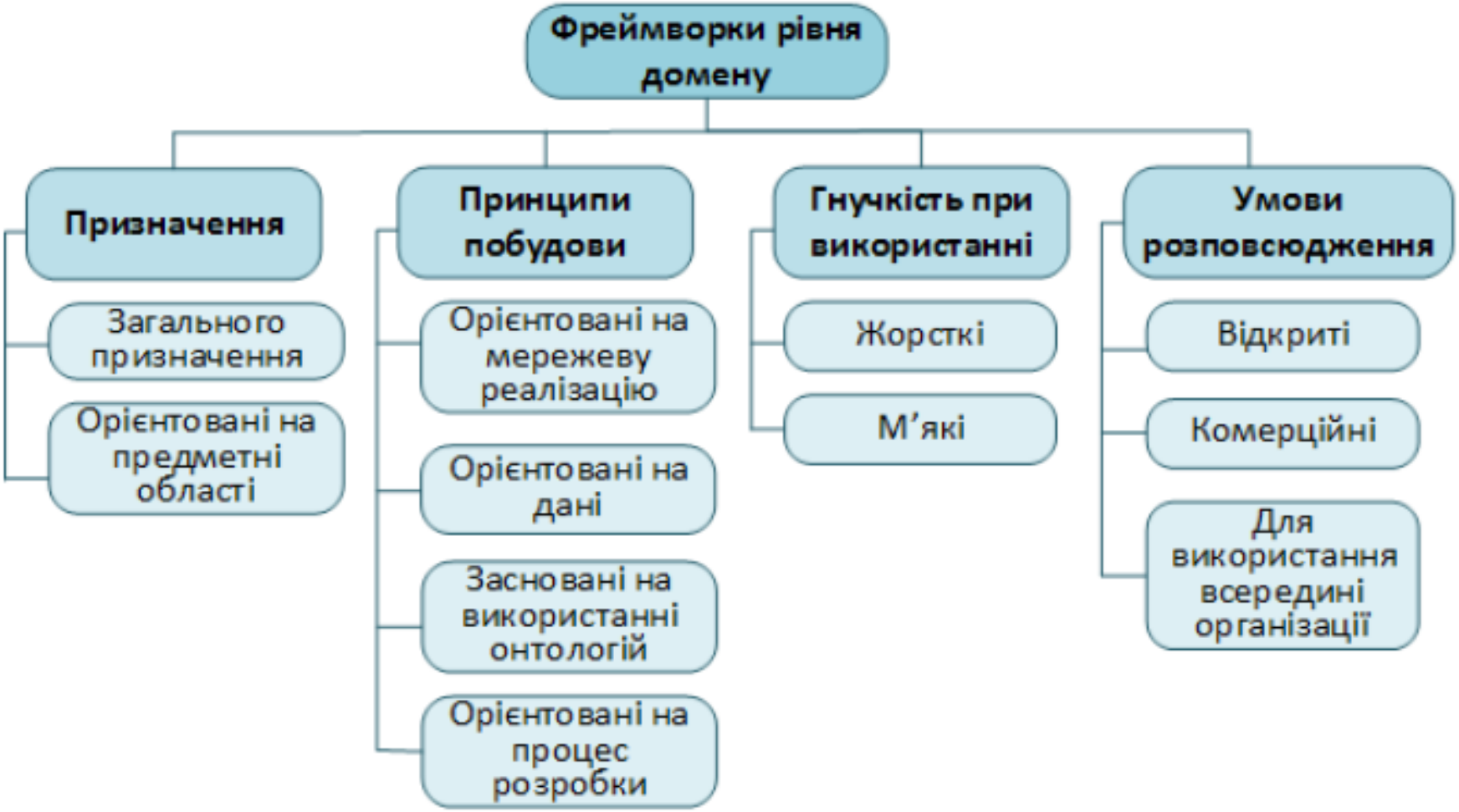
Middleware (зв'язуюче програмне забезпечення) - це програмне забезпечення, яке забезпечує зв'язок інших програмних компонентів. При цьому під «програмними компонентами» можуть розуміється дуже різні речі. Наприклад, middleware можна назвати шари програмного забезпечення для взаємодій між клієнтської частиною і базою даних, мережевих і прикладних програмних компонентів.

- Фреймворки, орієнтовані на додатки, використовуються для підтримки систем, орієнтованих на роботу з кінцевими користувачами в конкретній предметній області.
- У відповідності зі стандартом IEEE 42010 архітектурний фреймворк визначається як «сукупність угод, принципів і практик, які використовуються для опису архітектур і прийнятих відповідно деякому предметному домену й (або) у співтоваристві фахівців (зацікавлених осіб)».

Архітектурний фреймворк містить у собі опис зацікавлених осіб, типові проблеми предметної області, архітектурні точки зору й методи їхньої інтеграції.

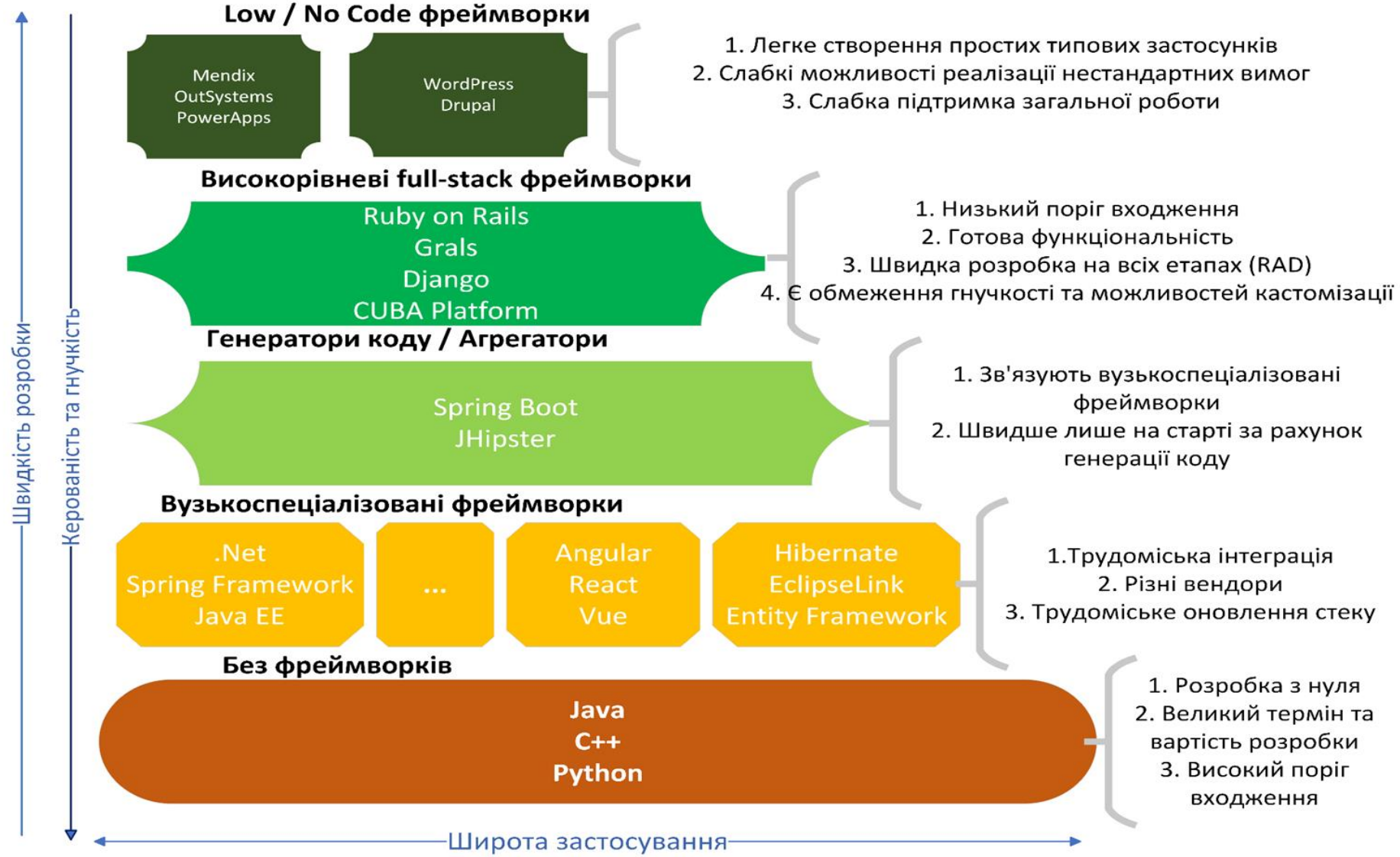
- Фреймворки, використовувані за принципом білого ящика (Architecture-driveng framework), застосовують методи спадкування й динамічного зв’язування для формування основних елементів додатка. Такі фреймворки визначаються через інтерфейси об’єктів, що додають у систему. Для роботи з ними необхідна докладна інформація про класи, розширення яких необхідно.
- Фреймворки, що функціонують за принципом чорного ящика, також називають фреймворками, керованими даними. Основними механізмами формування додатків, у цьому випадку, виступають композиція й параметризація, при цьому функціональність забезпечується додаванням додаткових компонентів. Слід зазначити, що процес використання фреймворків, що працюють за принципом чорного ящика простіше, ніж працюючих за принципом білого ящика, однак їхня розробка складніше.
- На практиці застосовують підхід сірого ящика (grey box), що є комбінацією обох підходів.

- Фреймворки рівня додатків (application frameworks) надають функціонал по реалізації типових додатків (GUI, бази даних і т.д).
- Фреймворки рівня домену (Domain Frameworks) застосовуються для створення додатків у певній предметній області.
- Допоміжні фреймворки (Support Frameworks) застосовуються для рішення приватних завдань.




М'які фреймворки мають на увазі можливість власного настроювання для вирішення конкретного завдання, у той час як тверді – ні.

# Класифікація фреймворків для розробки корпоративних ІС



# Мови програмування високого рівня, або підхід «Без фреймворків»

«Без фреймворків» - це вироджений випадок, який підходить для будь-яких завдань і відрізняється найбільшою керованістю і гнучкістю. Очевидно, що цей спосіб розробки корпоративних додатків найповільніший, адже, по суті, вам доводиться винаходити велосипед. Цей підхід хороший для створення вузькоспеціалізованих фреймворків, які, в свою чергу, підвищують швидкість розробки.

# Вузькоспеціалізовані фреймворки

Це найчисленніший клас. Всі його представники спрощують розробку за рахунок підвищення рівня абстракції і надання зрозумілого API. При цьому кожен такий фреймворк призначений для своєї вузької області. Зв'язати воєдино ORM, Middleware, UI, Messaging і інші фреймворки - завдання нетривіальне. У цього підходу є своя перевага - непоганий баланс між швидкістю розробки і дійсно високою гнучкістю.

# Генератори коду / Агрегатори

- Цей клас дозволяє прискорити розробку на стадії старту проекту. По суті, генератори коду і агрегатори просто вибудовують структуру додатків, заснованих на вузькоспеціалізованих фреймворками. Наприклад, JHipster генерує додаток на базі Spring з призначеним для користувача інтерфейсом на Angular. Однак, на інших стадіях такі фреймворки вже не сильно або взагалі не допоможуть.

# Високорівневі full-stack фреймворки

Цей клас - наступний після Вузькоспеціалізованих фреймворків рівень абстракції. Готова інфраструктура, яку пропонують такі фреймворки, дозволяє створювати корпоративні full-stack системи в рази швидше. Наприклад, Платформа CUBA надає готову архітектуру для створення 3-шарових додатків і включає в себе функції, без яких рідко обходяться корпоративні системи: soft deletion, data-aware компоненти, сховище файлів, а також модулі високого рівня: аутентифікація і авторизація, аудит даних, і т.п.

Цей клас, мабуть, найкраще підходить для розробки систем типу ERP, бізнес-додатків або іншого користувача ПЗ для корпоративного сегмента.

# Low / No Code фреймворки

У таких фреймворків ви взагалі не контролюєте кодову базу і можете писати тільки окремі шматочки логіки в заданих точках розширення. Якщо вам не цікавий хоч трохи відчутний контроль за своїм рішенням у плані продуктивності, деплоймента, інтеграцій та т.п. - тоді це найшвидший спосіб створення додатків. Однак будьте готові до того, що одного разу ви не зможете обійти архітектурні обмеження таких фреймворків.

# Основні архітектурні фреймворки

1. Фреймворк Захмана.
2. TOGAF.
3. DoDAF.
4. FEA.
5. Gärtner.

# Фреймворк Захмана

Для побудови таксономії Захманом запропоновано відповісти на шість питань про

- функціонування організації: що, як, де, хто, чому.

Дані питання ставляться до наступних аспектів системи:

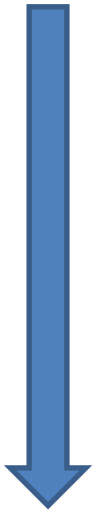
- використовувані дані (що?);
- процеси й функції (як?);
- місця виконання процесів (де?);
- організації й персоналії (хто?);
- керуючої події (коли?);
- мети й обмеження, що визначають роботу системи (чому?).

Відповідати на ці питання необхідно з різним ступенем деталізації. Описано шість рівнів:

- рівень контексту;
- рівень бізнесів-описів;
- системний рівень;
- технологічний рівень;
- технічний рівень;
- рівень реальної системи.

| Мотивація | Люди | Дані | Функції | Місце | Час |
|---|---|---|---|---|---|
| Навіщо? | Хто? | Що? | Як? | Де? | Коли? |

| **КОНТЕКСТ** |
|---|
| **БІЗНЕС** |
| **СИСТЕМА** |
| **ТЕХНОЛОГІЇ** |
| **РЕАЛІЗАЦІЯ** |
| **ВИКОНАННЯ** |



Від загальних вимог предметного середовища до функціонуючої інформаційної системи

- Інформаційна система
  - Мотивація
  - Люди
  - Дані
  - Функції
  - Місце
  - Час

Стосовно до кожного рівня деталізації існує своя зацікавлена особа (stakeholders), тобто точка зору:

- аналітики (рівень контексту);
- топ-менеджери (рівень бізнес- описів);
- архітектори (системний рівень);
- розроблювачі (технологічний рівень);
- адміністратори (технічний рівень);
- користувачі (рівень реальної системи).

За результатами виконаних дій формується матриця розміром 6х6, у кожній комірці якої розташовуються артефакти. Для заповнення комірок уведені наступні правила:

- Стовпчики можна міняти місцями, але не можна додавати й видаляти.
- Кожному стовпчику відповідає власна модель.
- Кожна з моделей має бути унікальна.
- Кожен рівень (рядок) є описом системи з погляду групи користувачів (представляє окремий вид).
- Кожна з комірок унікальна.
- Кожна комірка містить опис аспекту реалізації системи у вигляді моделі й текстового документа.
- Заповнення комірок повинне відбуватись послідовно зверху вниз.

| | Використовувані дані (що?) | Процеси й функції (як?) | Місця виконання процесів (де?) | Організації й персоналії (хто?) | Керуючі події (коли?) | Мета й обмеження | |
|---|---|---|---|---|---|---|---|
| Контекст | Список основних сутностей | Основні бізнес-процеси | Територіальне розміщення організації | Важливі зовнішні організації | Список подій | Бізнес-стратегія | Аналітик |
| Бізнес-модель | Відносини між сутностями | Докладний опис бізнес- процесів | Система логістики | Модель потоків даних | Базовий графік робіт | Дерево цілей, Бізнес-план | Топ-менеджер |
| Системна модель | Концептуальні моделі даних | Архітектура додатків | Архітектура розподіленої системи | Інтерфейси користувача | Модель роботи з подіями | Бізнес-правила | Архітектор |
| Технологічна модель | Фізична модель даних | Програмно-апаратна архітектура | Технологічна архітектура | Архітектура подання | Алгоритми обробки подій | Правила обробки подій | Розробник |
| Детальний опис | Специфікації форматів даних | Виконуваний код | Архітектура мережі | Ролі й права користувачів | Обробка подій за допомогою переривань | Алгоритми роботи системи | Адміністратор |
| Функціонуюча організація | Дані | Реалізована функціональність | Функціонуюча мережна інфраструктура | Організаційна структура організації | Історія функціонування системи | Реалізовані стратегії | Користувач |

Перший рядок матриці визначає контекст всіх інших й являє собою загальний погляд на організацію.

Другий рядок описує функціонування організації в бізнесах-термінах.

Третій рядок описує бізнес- процеси в термінах інформаційних систем.

Четвертий рядок дозволяється розподілити дані й виконувані над ними операції між конкретними апаратними і програмними платформами.

П’ятий рядок описує конкретні моделі устаткування, мережеві топології й програмний код.

Шостий рядок описує готову систему у вигляді інструкції користувачів, довідкових баз даних і т.д.

# The Zachman Framework for Enterprise Architecture™

## *The Enterprise Ontology* ™

Version 3.0

| Classification Names / Audience Perspectives | What | How | Where | Who | When | Why | Classification Names / Model Names |
|---|---|---|---|---|---|---|---|
| | Composite Integrations → | | ← Alignment → | | ← Composite Integrations | | |
| Executive Perspective (Business Context Planners) | Inventory Identification<br>List: Inventory Types | Process Identification<br>List: Process Types | Distribution Identification<br>List: Distribution Types | Responsibility Identification<br>List: Responsibility Types | Timing Identification<br>List: Timing Types | Motivation Identification<br>List: Motivation Types | Scope Contexts (Scope Identification Lists) |
| Business Mgmt Perspective (Business Concept Owners) | Inventory Definition<br>Business Entity<br>Business Relationship | Process Definition<br>Business Transform<br>Business Input/Output | Distribution Definition<br>Business Location<br>Business Connection | Responsibility Definition<br>Business Role<br>Business Work Product | Timing Definition<br>Business Interval<br>Business Moment | Motivation Definition<br>Business End<br>Business Means | Business Concepts (Business Definition Models) |
| Architect Perspective (Business Logic Designers) | Inventory Representation<br>System Entity<br>System Relationship | Process Representation<br>System Transform<br>System Input /Output | Distribution Representation<br>System Location<br>System Connection | Responsibility Representation<br>System Role<br>System Work Product | Timing Representation<br>System Interval<br>System Moment | Motivation Representation<br>System End<br>System Means | System Logic (System Representation Models) |
| Engineer Perspective (Business Physics Builders) | Inventory Specification<br>Technology Entity<br>Technology Relationship | Process Specification<br>Technology Transform<br>Technology Input /Output | Distribution Specification<br>Technology Location<br>Technology Connection | Responsibility Specification<br>Technology Role<br>Technology Work Product | Timing Specification<br>Technology Interval<br>Technology Moment | Motivation Specification<br>Technology End<br>Technology Means | Technology Physics (Technology Specification Models) |
| Technician Perspective (Business Component Implementers) | Inventory Configuration<br>Tool Entity<br>Tool Relationship | Process Configuration<br>Tool Transform<br>Tool Input /Output | Distribution Configuration<br>Tool Location<br>Tool Connection | Responsibility Configuration<br>Tool Role<br>Tool Work Product | Timing Configuration<br>Tool Interval<br>Tool Moment | Motivation Configuration<br>Tool End<br>Tool Means | Tool Components (Tool Configuration Models) |
| Enterprise Perspective (Users) The Enterprise | Inventory Instantiations<br>Operations Entities<br>Operations Relationships | Process Instantiations<br>Operations Transforms<br>Operations In/Outputs | Distribution Instantiations<br>Operations Locations<br>Operations Connections | Responsibility Instantiations<br>Operations Roles<br>Operations Work Products | Timing Instantiations<br>Operations Intervals<br>Operations Moments | Motivation Instantiations<br>Operations Ends<br>Operations Means | Operations Instances (Implementations) The Enterprise |
| | Composite Integrations → | | ← Alignment → | | ← Composite Integrations | | |
| Audience Perspectives / Enterprise Names | Inventory Sets | Process Flows | Distribution Networks | Responsibility Assignments | Timing Cycles | Motivation Intentions | |

*Horizontal integration lines are shown for example purposes only and are not a complete set. Composite, integrative relationships connecting every cell horizontally potentially exist.

# Приклад для освітньої системи

| Власник компанії | |
|---|---|
| Дані<br>ЩО? | Готовий продукт – інформаційна система |
| Функції<br>ЯК? | Пошук замовника; обговорення проекту на концептуальному рівні; підписання договорів з замовником; призначення керівника на проект; продаж, впровадження і підтримка інформаційної системи |
| Дислокація<br>ДЕ? | Відділ продажу; відділ аналізу; відділ розробки; відділ тестування; відділ впровадження і підтримки |
| Люди<br>ХТО? | Зовнішній замовник, користувач системи; керівник проекту; аналітик; розробник; тестировщик; супорт |
| Час<br>КОЛИ? | Часовий період вказаний в контракті |
| Мотивація<br>ЧОМУ? | Отримання прибутку від продажу системи; отримання стабільного прибутку від підтримки системи; продаж готової інформаційної системи стороннім замовникам |

Перший шар

| Керівник проекту | |
|---|---|
| Дані<br>ЩО? | Основні об’єкти і сутності інформаційної системи – концептуальна модель системи |
| Функції<br>ЯК? | Взаємодія з замовником; постановка задачі аналітикам проекту |
| Дислокація<br>ДЕ? | Окреме робоче місце керівника проекту |
| Люди<br>ХТО? | Замовник; аналітики проекту; відділ розробки (керівник відділу розробки) |
| Час<br>КОЛИ? | Календарний план впровадження робіт, створений на основі ТЗ і контракту з замовником |
| Мотивація<br>ЧОМУ? | Збільшення прибутку за рахунок вдалої здачі проекту |

Другий шар

Третій шар

| Бізнес і системні аналітики проекту | |
|---|---|
| Дані ЩО? | Поглиблена модель інформаційної системи; інформаційна система, розбита на окремі моделі (блоки); потреби замовника; вимоги до системи |
| Функції ЯК? | Виявлення вимог замовника; визначення і керування вимог до системи; оформлення бізнес і системних постановок; постановка задачі керівнику відділу розробки; нотатки поправок в бізнес логіку системи на основі зауважень замовника і розробника |
| Дислокація ДЕ? | Відділ аналітики |
| Люди ХТО? | Замовник; керівник проекту; бізнес аналітики проекту; системні аналітики проекту |
| Час КОЛИ? | Часові проміжки, які виділені на розробку окремих модулів (блоків) системи |
| Мотивація ЧОМУ? | Ефективне керування вимог системи (мінімальні поправки); максимально точний і простий опис системи (за рахунок системного підходу) |

Четвертий шар

| Керівник відділу розробки | |
|---|---|
| 1 | 2 |
| Дані ЩО? | Фізичне представлення моделі даних інформаційної системи |
| Функції ЯК? | Розробка фізичної моделі даних на основі системних постановок; управління відділом розробки (постановка задач розробникам, ефективний розподіл навантажень між розробниками); дотримання термінів виконуваних робіт |
| Дислокація ДЕ? | Окреме робоче місце керівника розробки |
| Люди ХТО? | Системні аналітики проекту; керівник відділу розробки |
| Час КОЛИ? | Часові проміжки виконуваних робіт які поставлені аналітиком |
| Мотивація ЧОМУ? | Ефективне керування відділом розробки; максимально просте і прозоре представлення системи, яке легко змінюється |

П’ятий шар

| Розробники | |
|---|---|
| Дані<br>ЩО? | Готові таблиці БД; програмні засоби для розробки системи |
| Функції<br>ЯК? | Виконання завдань, які поставлені керівником; виправлення помилок, які були виявлені на етапі тестування |
| Дислокація<br>ДЕ? | Відділ розробки |
| Люди<br>ХТО? | Керівник розробки; системні аналітики проекту; розробники |
| Час<br>КОЛИ? | Часові проміжки виконання завдань, які були поставлені керівником |
| Мотивація<br>ЧОМУ? | Виконання задач вчасно; виконання завдань з мінімальним набором помилок |

# Фреймворк TOGAF

- Фреймворк TOGAF (The Open GroupArchitecture Framework) являє собою набір засобів для розробки архітектур різного призначення. З його допомогою інформаційна система подається як сукупність модулів.
- В рамках TOGAF дається особливе визначення архітектури – «формальний опис системи, або детальний план системи на рівні компонентів і методології їх реалізації». Загальноприйняте ж визначення архітектури (відповідно до стандарту ANSI / IEEE 1471-2000) визначається як «опис організації системи в термінах компонентів, їх взаємозв’язків між собою і з навколишнім середовищем і принципи управління їх розробкою і розвитком».

# The TOGAF® Library

TOGAF SERIES GUIDES

STANDARDS

BLOGS

CERTIFICATIONS

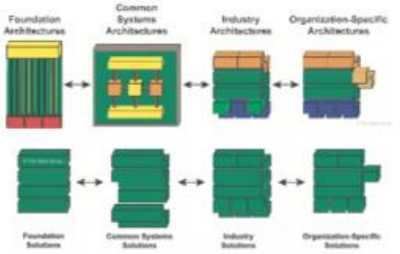
DATA SHEETS

CASE STUDIES

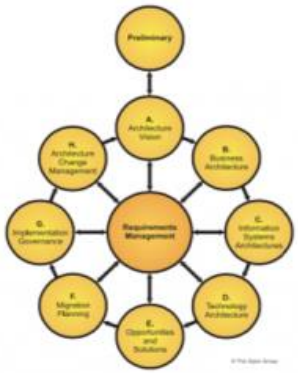
WEBINARS

WHITE PAPERS

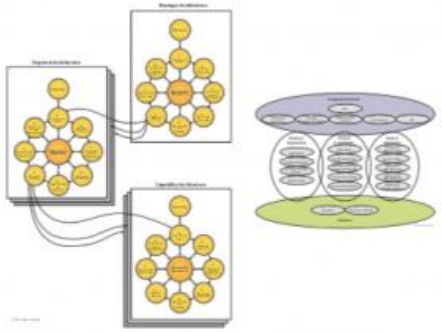
## Enterprise Continuum



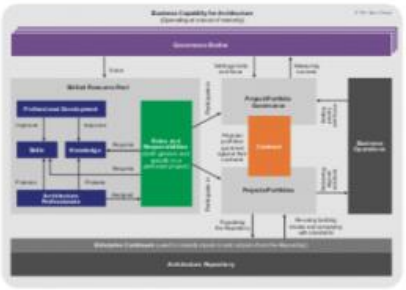

## The TOGAF ADM



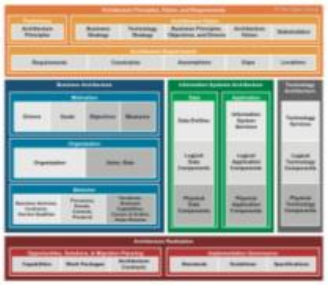

## ADM Guidelines and Techniques



## Architecture Capability Framework



## Content Architecture Framework



The TOGAF® Standard — *Version 9.2*

REFERENCE ARCHITECTURES

POCKET GUIDES

REFERENCE CARDS

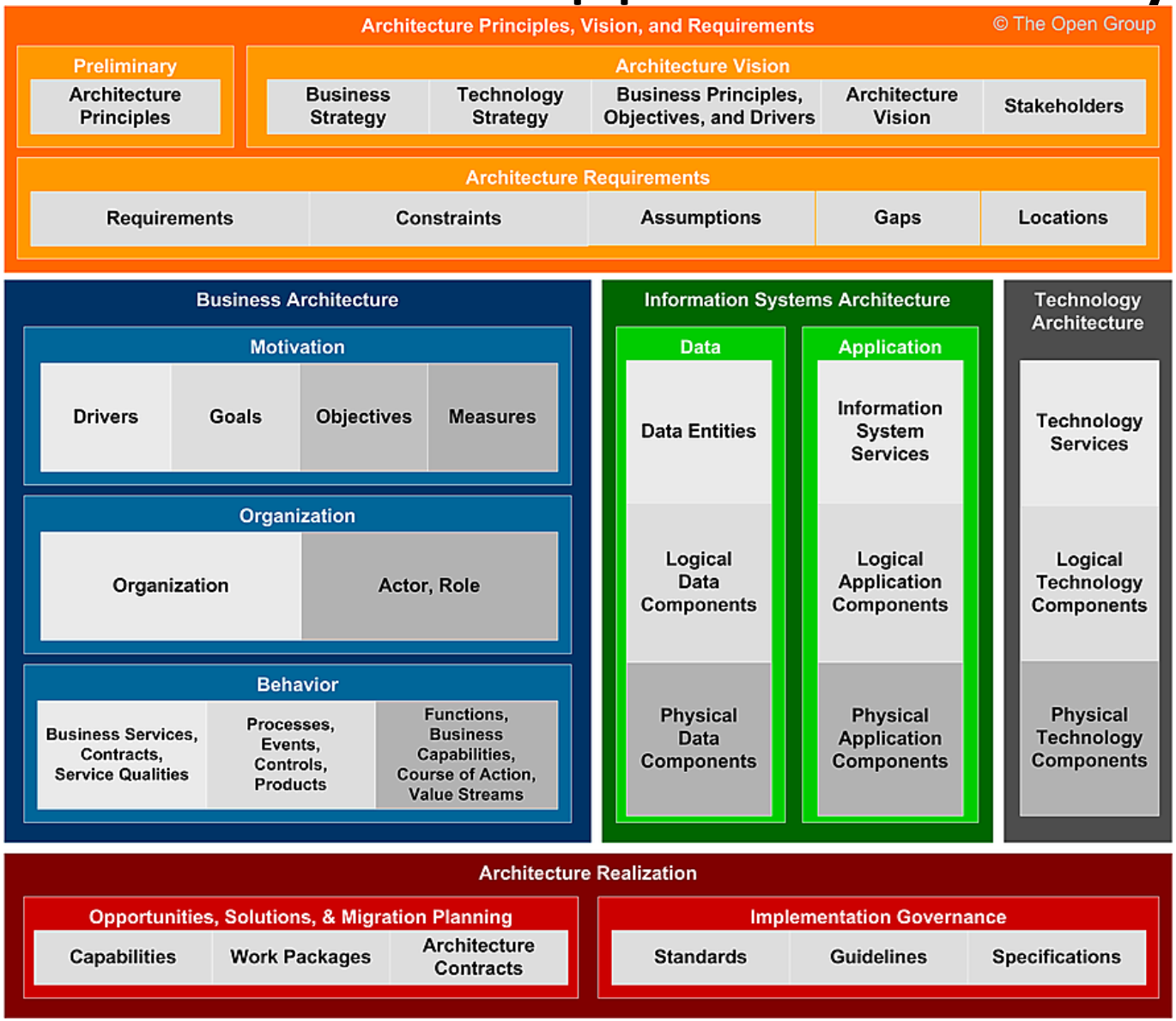
TOGAF складається з чотирьох архітектурних доменів:

описує ключові бізнес-процеси, стратегію розвитку бізнесу і принципи управління

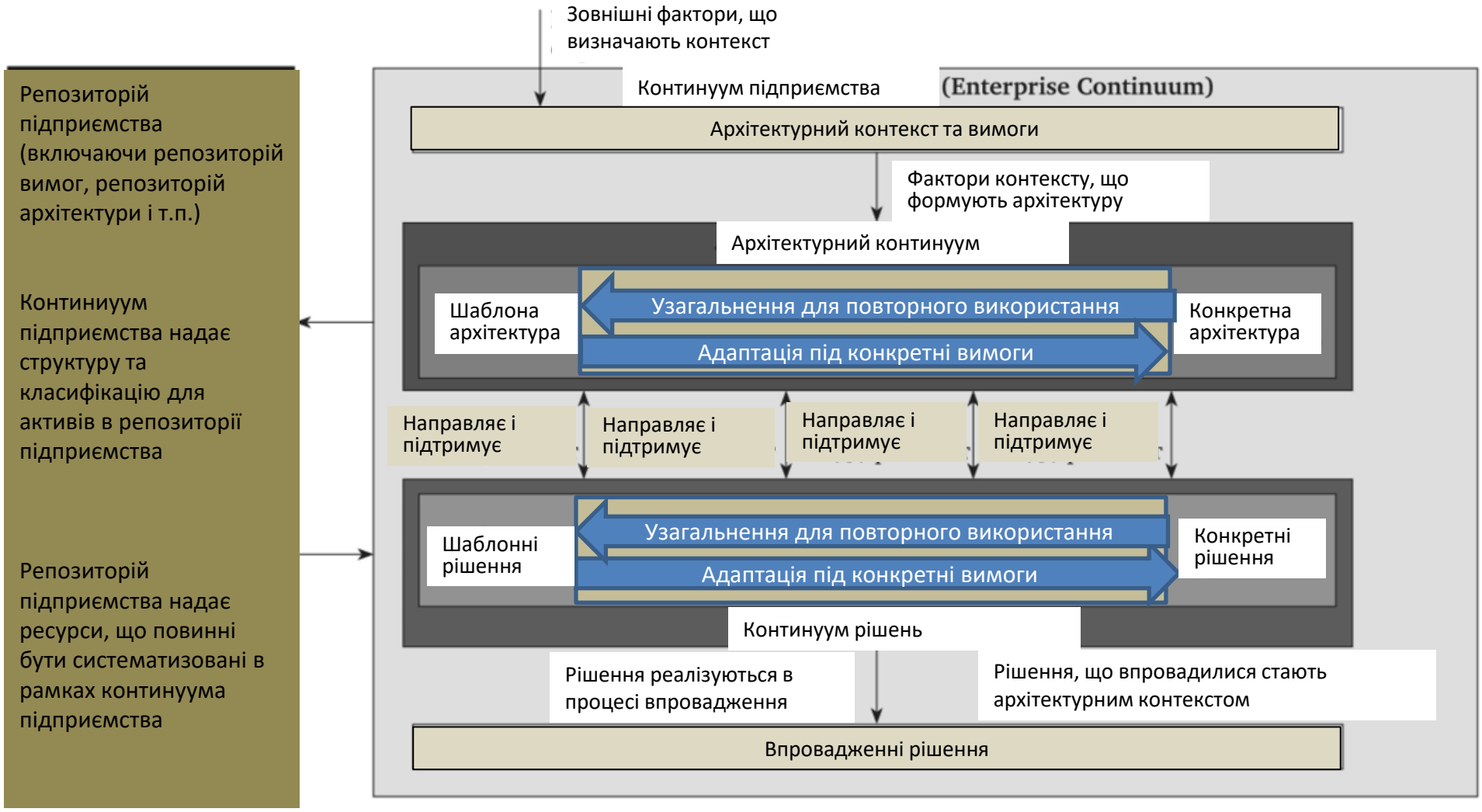
- Бізнес -архітектура;
- архітектура рівня додатків (описує інтерфейси додатків і способи їх застосування в термінах бізнес-сервісів);
- архітектура рівня даних (визначає логічну й фізичну структуру даних в організації);
- технологічна архітектура (визначає програмну, апаратну і мережеву інфраструктури).

- Головними складовими частинами TOGAF є:
  - ADM-методика (Architecture Development Method), що описує процес розробки архітектури; керівництва і методики проектування для ADM;
  - фреймворк архітектурного опису (Architecture Content Framework), що є детально відпрацьованої моделлю результатів розробки;
  - архітектурний континуум організації (Enterprise Continuum), у вигляді репозиторію архітектурних артефактів і реалізацій;
  - еталонні моделі TOGAF (TOGAF Reference Models): TRM (Technical Reference Model) – технічна еталонна модель; III-RM (The Integrated Information Infrastructure Model) – інтегрована модель інформаційної інфраструктури.
  - фреймворк, що описує структуру організації, її персонал, необхідні ролі і рівні відповідальності (Architecture Capability Framework).

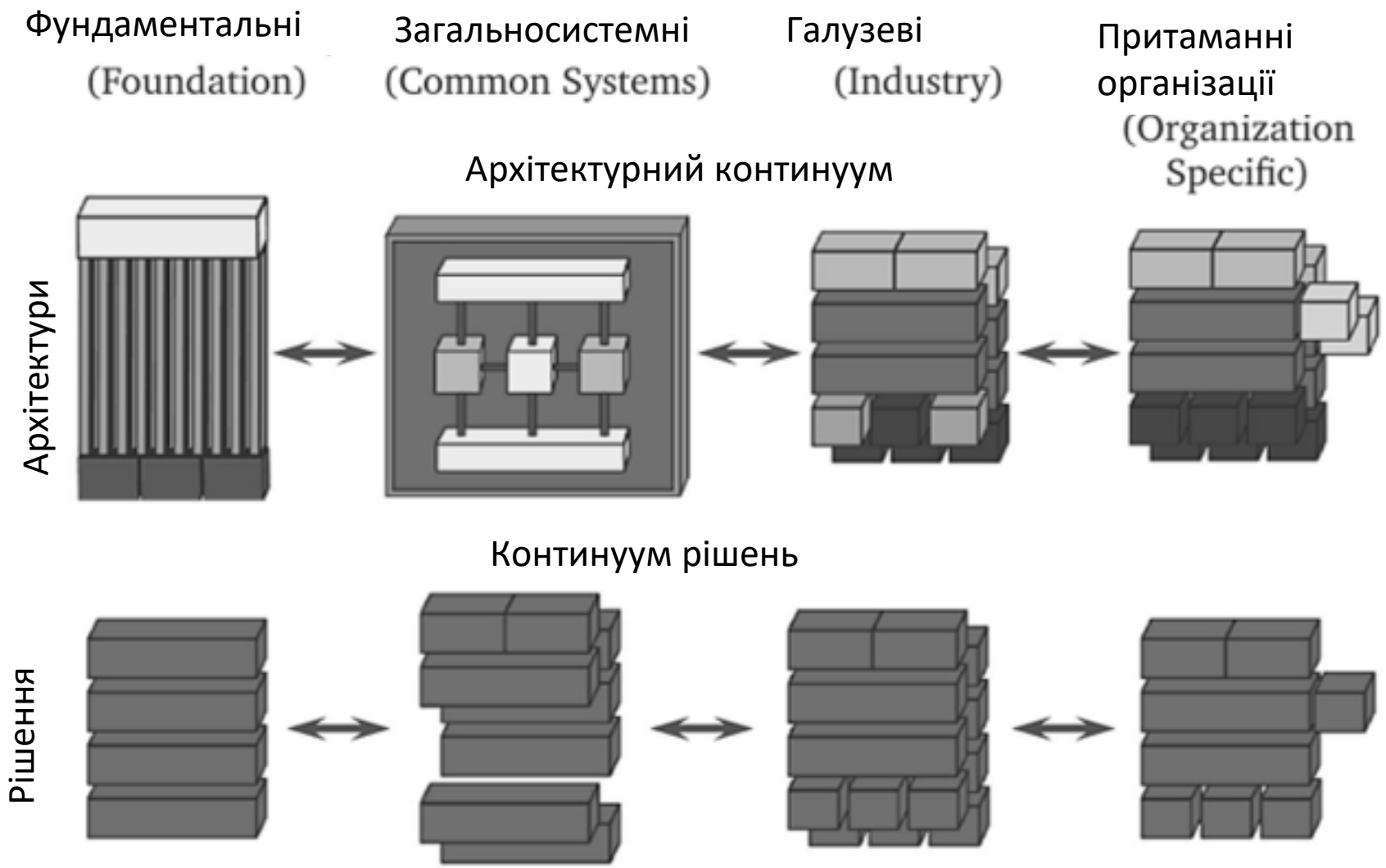
# Метамодель контенту



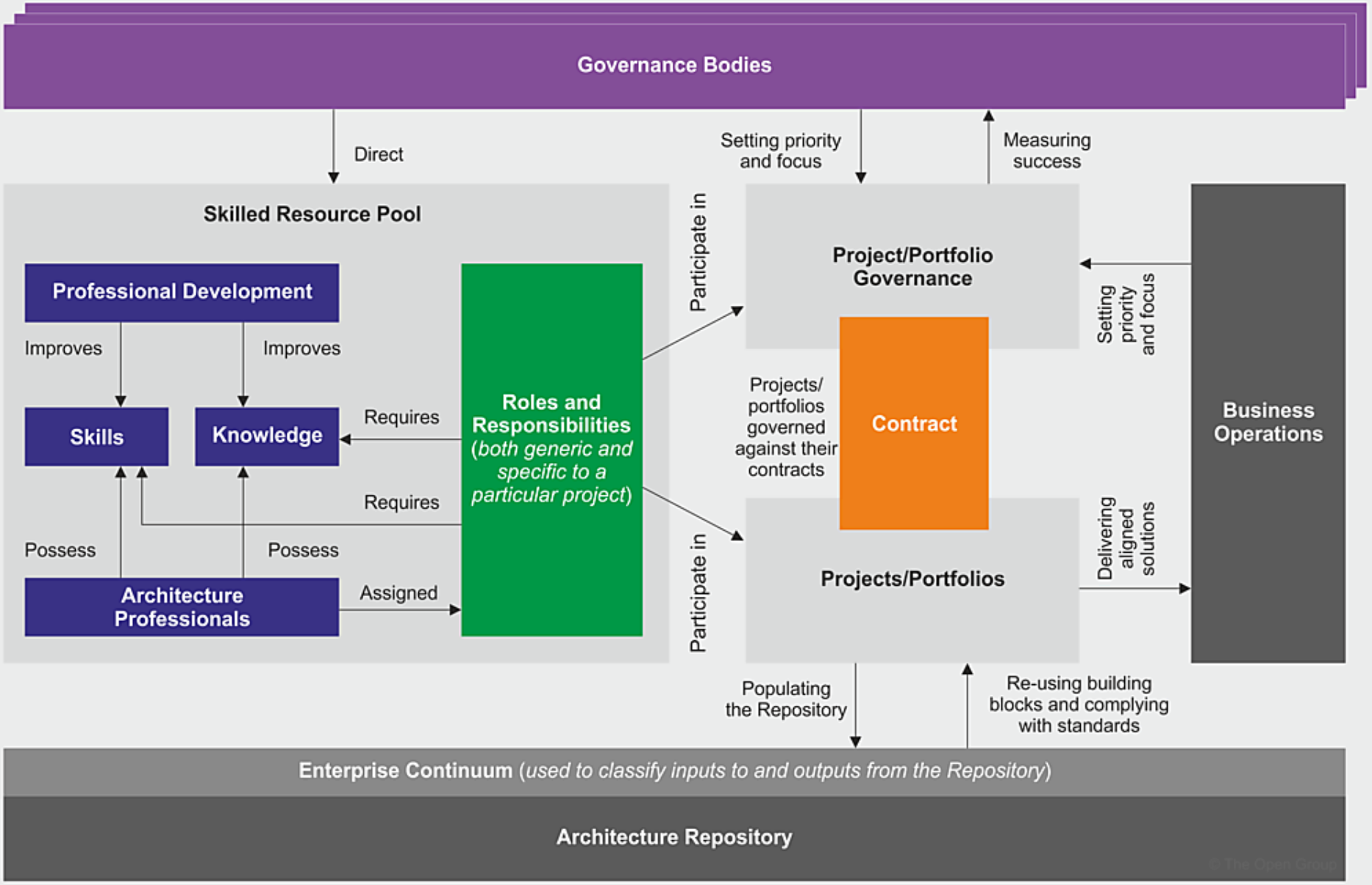
# Континуум підприємства



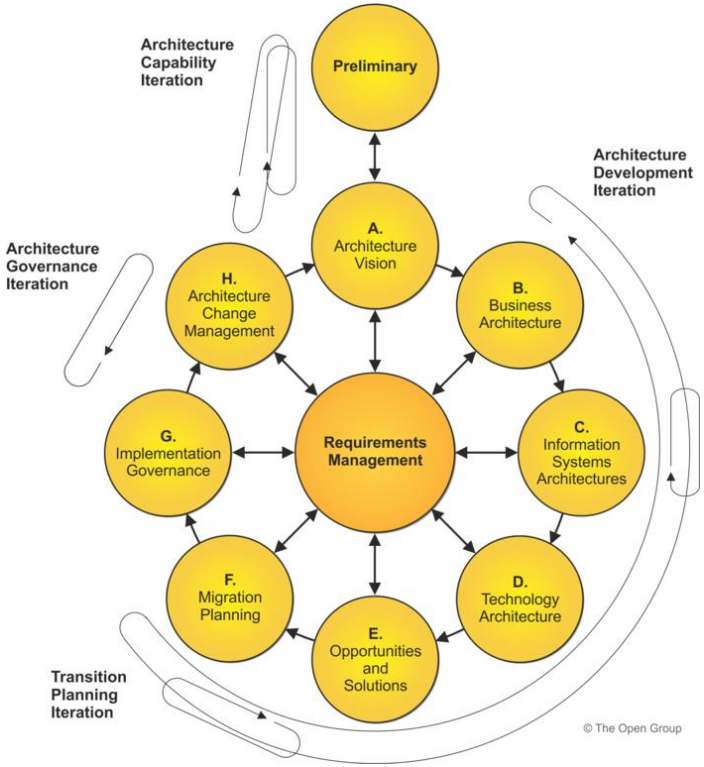
# Рівні контініуму підприємства



# Business Capability for Architecture
(*Operating at a level of maturity*)

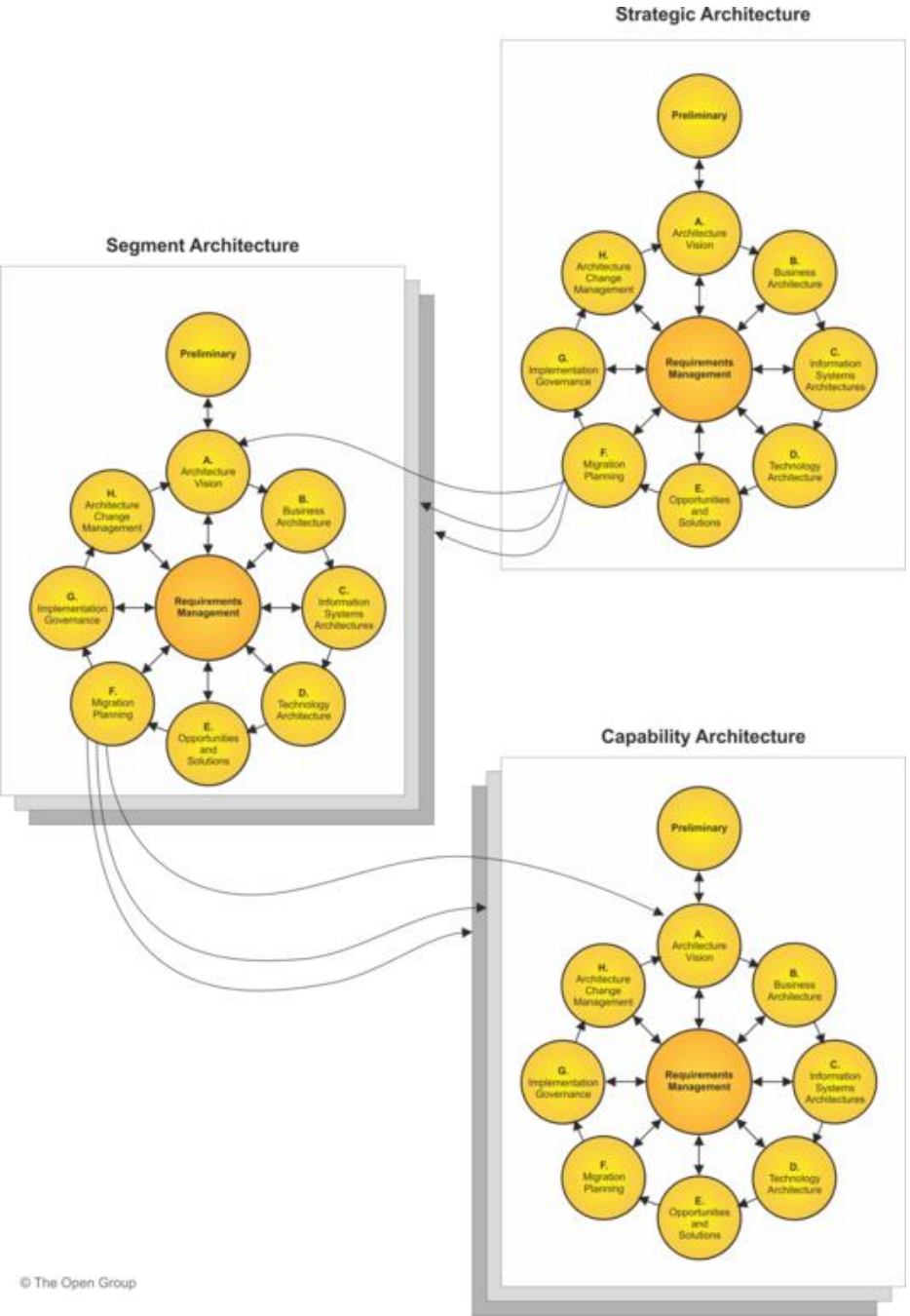


Відповідно до методики ADM архітектурний процес можна розбити на дев’ять фаз. ADM представляє з себе ітераційний процес, який відбувається на двох рівнях. На верхньому рівні кожної ітерації повторюються загальні для кожної з фаз дії. Нижній рівень описує ітерації всередині кожної фази. Рішення приймаються на підставі існуючих вимог бізнесу та існуючих рішень.

| Стадія процесу | Опис |
|---|---|
| Попередня фаза (Preliminary Phase) | Визначення способів управління, меж розробки, принципів реалізації, уточнення моделі щодо специфіки . |
| Фаза A, розробка загального подання (Architecture Vision) | Створення загального уявлення про архітектуру, визначення меж проекту, затвердження плану робіт і завдань для виконавців |
| Фаза B, розробка бізнес-архітектури (Business Architecture) | Виявлення принципів функціонування організації, принципів управління проектом |
| Фаза C, розробка інформаційної архітектури (Information Systems Architecture) | Розробка архітектури даних і додатків |
| Фаза D, розробка технологічної архітектури (Technology Architecture) | Підбір апаратних засобів, засобів мережевої інфраструктури, механизмів їх взаємодії |
| Фаза E, Можливості і рішення (Opportunities and Solutions) | Реалізація розробленої архітектури (купити готове рішення або зробити власне) |
| Фаза F, Планування переходу до нової архітектури (Migration Planning) | Оцінка ризиків, аналіз деталей переходу |
| Фаза G, формування системи керування реалізацією (Implementation Governance) | Моніторинг процесу впровадження і специфікація виникаючих проблем |
| Фаза H, керування зміною архітектури (Architecture Change Management) | Налагодження процесу керування змінами розробленої архітектури |

Слід зазначити, що TOGAF може використовуватися не тільки як єдиний фреймворк при розробці, але і в сукупності з іншими фреймворками. Зокрема, до нього входить спеціальний документ, що визначає відповідності між поняттями TOGAF і фреймворком Захмана.



Розвиток архітектури

Планування переходів

Управління архітектурою

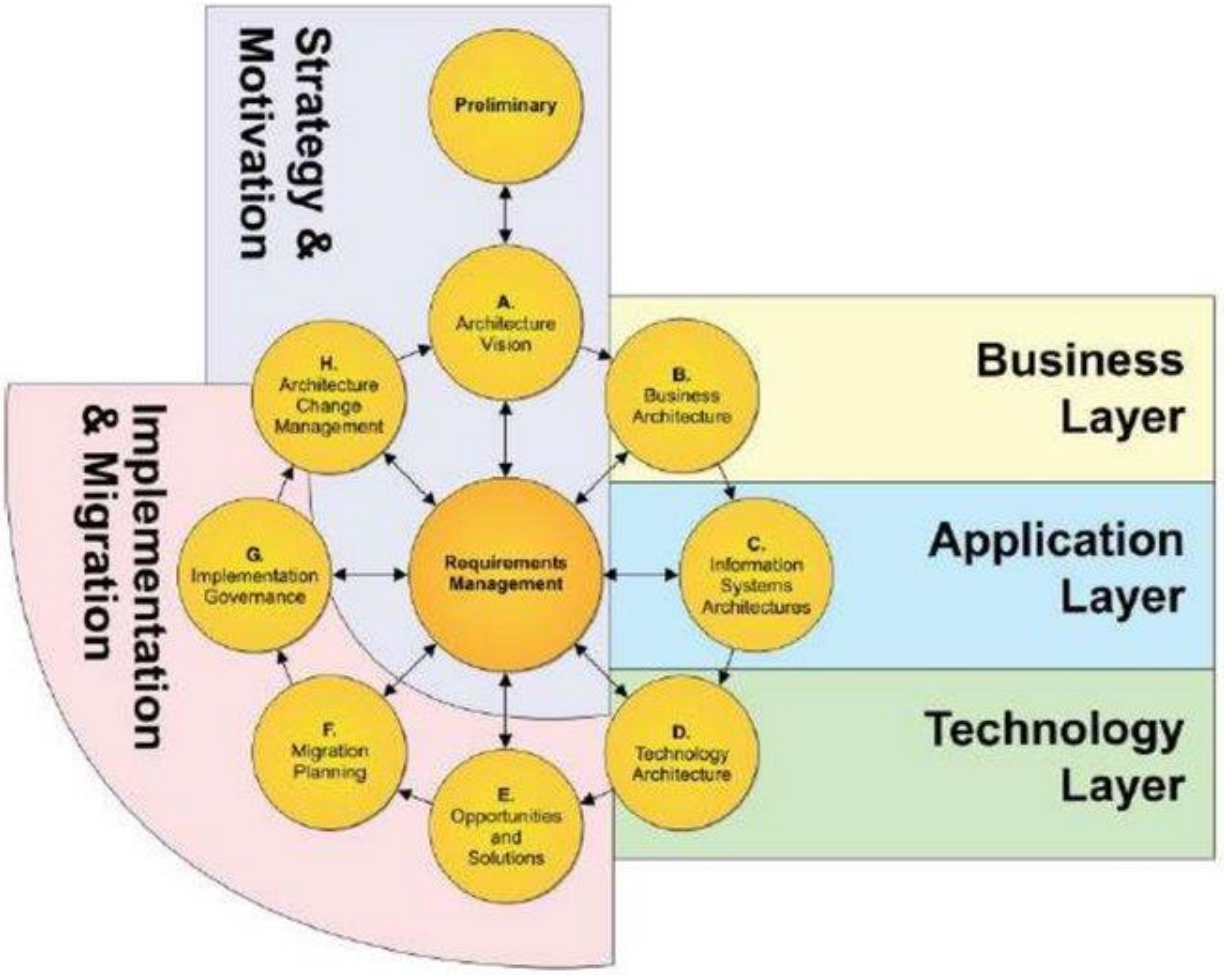
| | | Architecture Development | | | Transition Planning | | Architecture Governance | |
|---|---|---|---|---|---|---|---|---|
| **TOGAF Phase** | | Iteration 1 | Iteration 2 | Iteration *n* | Iteration 1 | Iteration *n* | Iteration 1 | Iteration *n* |
| Preliminary | | Informal | Informal | Informal | | | | Light |
| Architecture Vision | | Informal | Informal | Informal | Informal | Informal | | Light |
| Business Architecture | Baseline | Core | Light | Core | Informal | Informal | | Light |
| | Target | Informal | Core | Core | Informal | Informal | | Light |
| Application Architecture | Baseline | Core | Light | Core | Informal | Informal | | Light |
| | Target | Informal | Core | Core | Informal | Informal | | Light |
| Data Architecture | Baseline | Core | Light | Core | Informal | Informal | | Light |
| | Target | Informal | Core | Core | Informal | Informal | | Light |
| Technology Architecture | Baseline | Core | Light | Core | Informal | Informal | | Light |
| | Target | Informal | Core | Core | Informal | Informal | | Light |
| Opportunities and Solutions | | Light | Light | Light | Core | Core | Informal | Informal |
| Migration Planning | | Light | Light | Light | Core | Core | Informal | Informal |
| Implementation Governance | | | | | Informal | Informal | Core | Core |
| Change Management | | Informal | Informal | Informal | Informal | Informal | Core | Core |

Core: primary focus activity for the iteration — Основний фокус ітерації

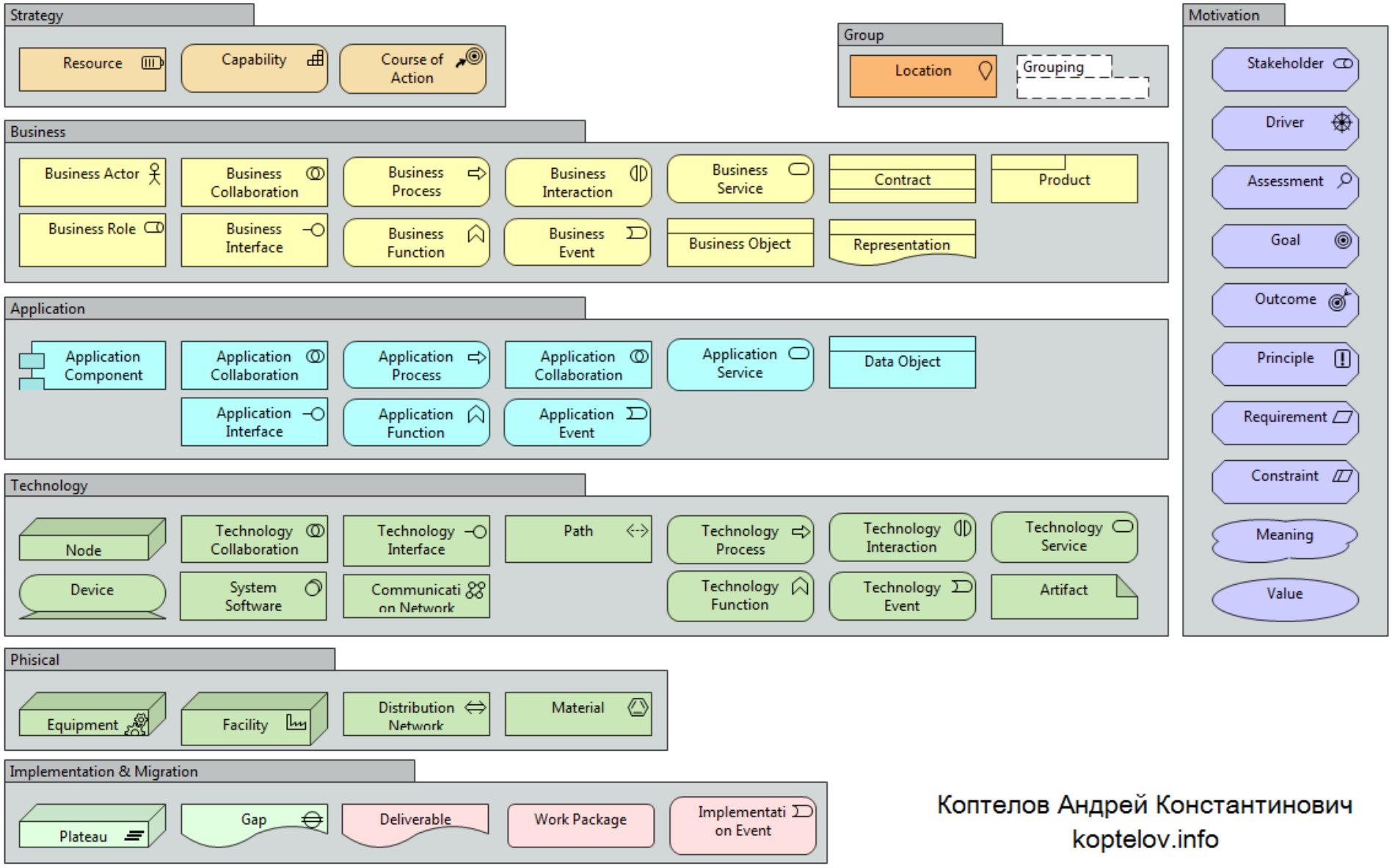
Light: secondary focus activity for the iteration — Вторинний фокус ітерації

Informal: potential activity for the iteration, not formally mentioned in the method — Потенціальний неформальний фокус ітерац

# TOGAF-ARCHIMATE



# Инструментарий Archi 4 - обзор методологии ArchiMate 3.0





Коптелов Андрей Константинович
koptelov.info

Stone Age

Digital Age

**Drawing tool**

- Visio
- Web diagramming

**Modeling tool**

- With model & views
- Element consistency
- Notation validation check
- Model generation
- Report generation
- Project referencing
- etc.

**Modeling with Process Support**

- With TOGAF ADM
- With PMBOK
- With step-by-step instructions, samples
- Generate & archive deliverables

Тут приклад primeri\lek8.archimate