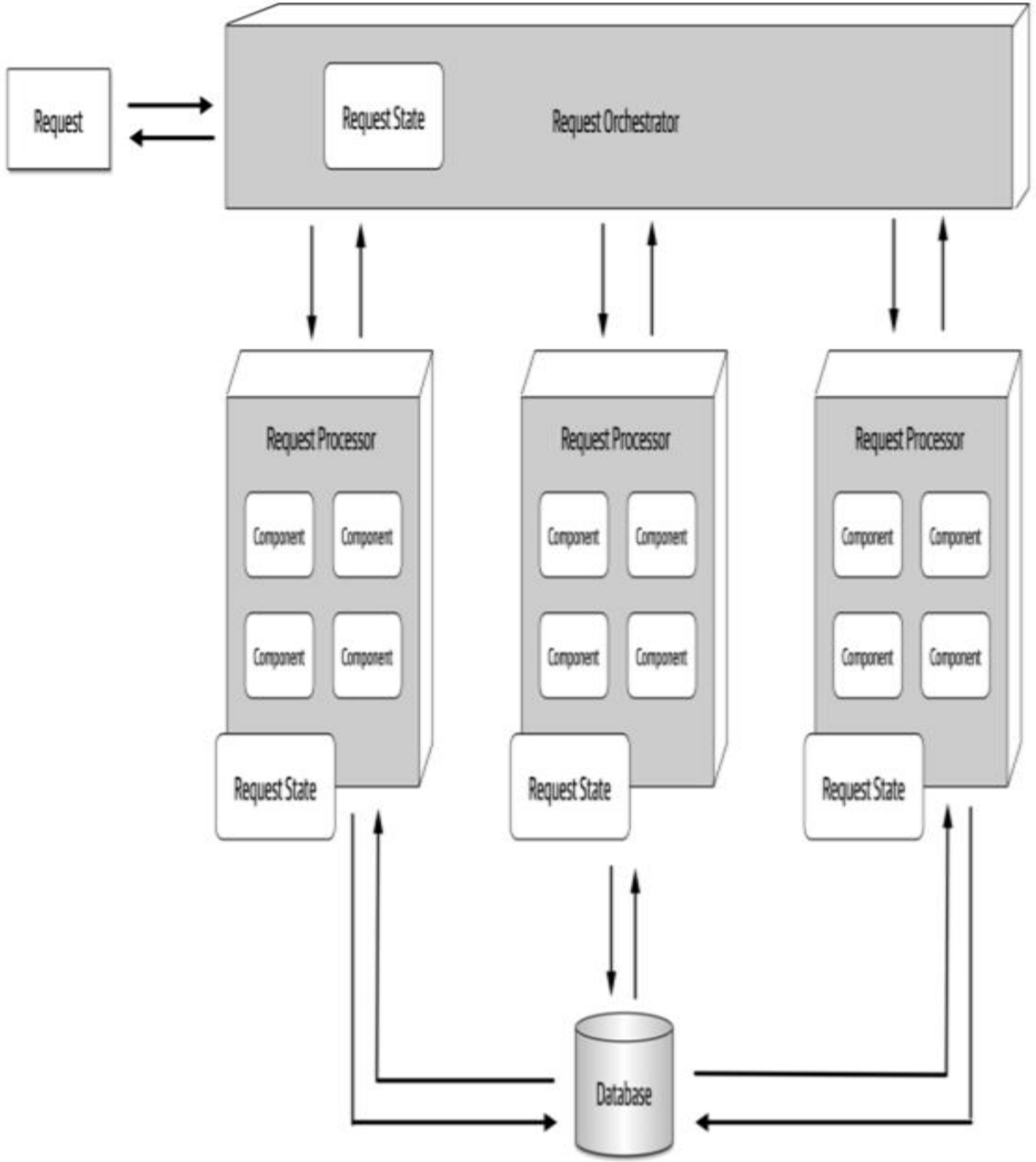
Архітектура, керована подіями

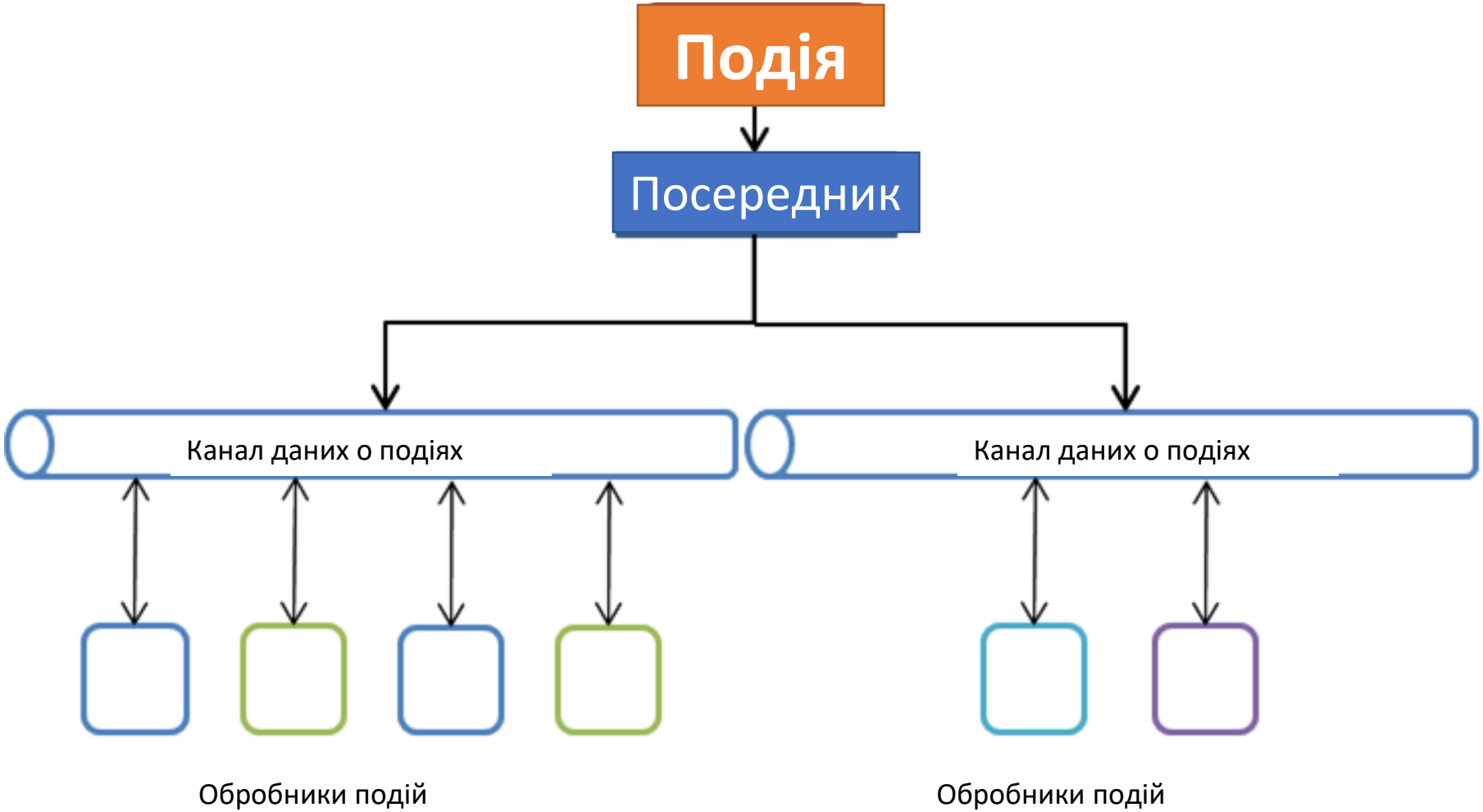


Більшість додатків дотримуються того, що називається моделлю на основі запиту

Хорошим прикладом моделі на основі запиту є запит від клієнта, щоб отримати історію замовлень за останні півроку. Отримання інформації історії замовлень є керованим даними, детермінованим запитом, який зроблений до системи даних у конкретному контексті, а не подія, на яку повинна реагувати система.

Архітектура, керована подіями (EDA) Event-driven

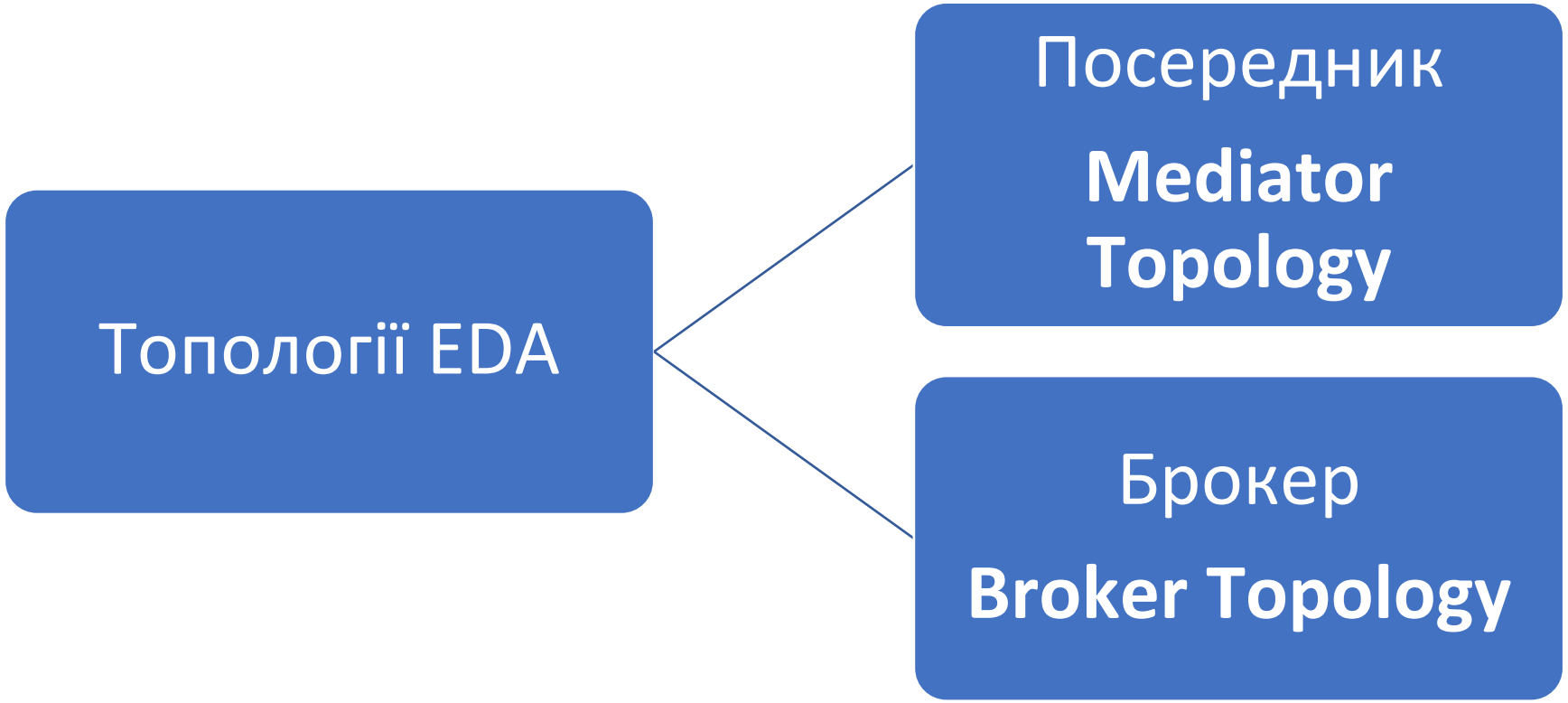
Це популярний адаптивний патерн, широко використовуваний для створення масштабованих систем.



Якщо говорити про програмне забезпечення, то в цій схемі існує два варіанти подій: ініціююча подія і подія, на яку реагують обробники. Обробники є ізольованими незалежними компонентами, що відповідають (в ідеалі) за якусь одну задачу, і містять бізнес-логіку, необхідну для роботи.

Посередник може бути реалізований декількома способами. Самий просто спосіб - це скористатися фреймворками для інтеграції Apache Camel (https://camel.apache.org/), Spring Integration (https://docs.spring.io/spring-integration/reference/html/index.html) або Mule ESB (https://www.mulesoft.com/resources/esb/what-mule-esb). Для великих додатків, яким потрібна більш складні функції управління, ви можете реалізувати посередника, використовуючи концепцію управління бізнес-процесами (наприклад рушій jBPL) https://www.redhat.com/en/technologies/jboss-middleware/process-automation-manager.

Архітектура, керована подіями - це відносно складний патерн. Причиною тому - його розподілена і асинхронна природа. Вам доведеться вирішувати проблеми фрагментації мережі, обробляти помилки в черзі подій і так далі. Плюсами цієї архітектури можуть служити висока продуктивність, легкість розгортки і вражаючі можливості масштабування. Однак можливо ускладнення процесу тестування системи.

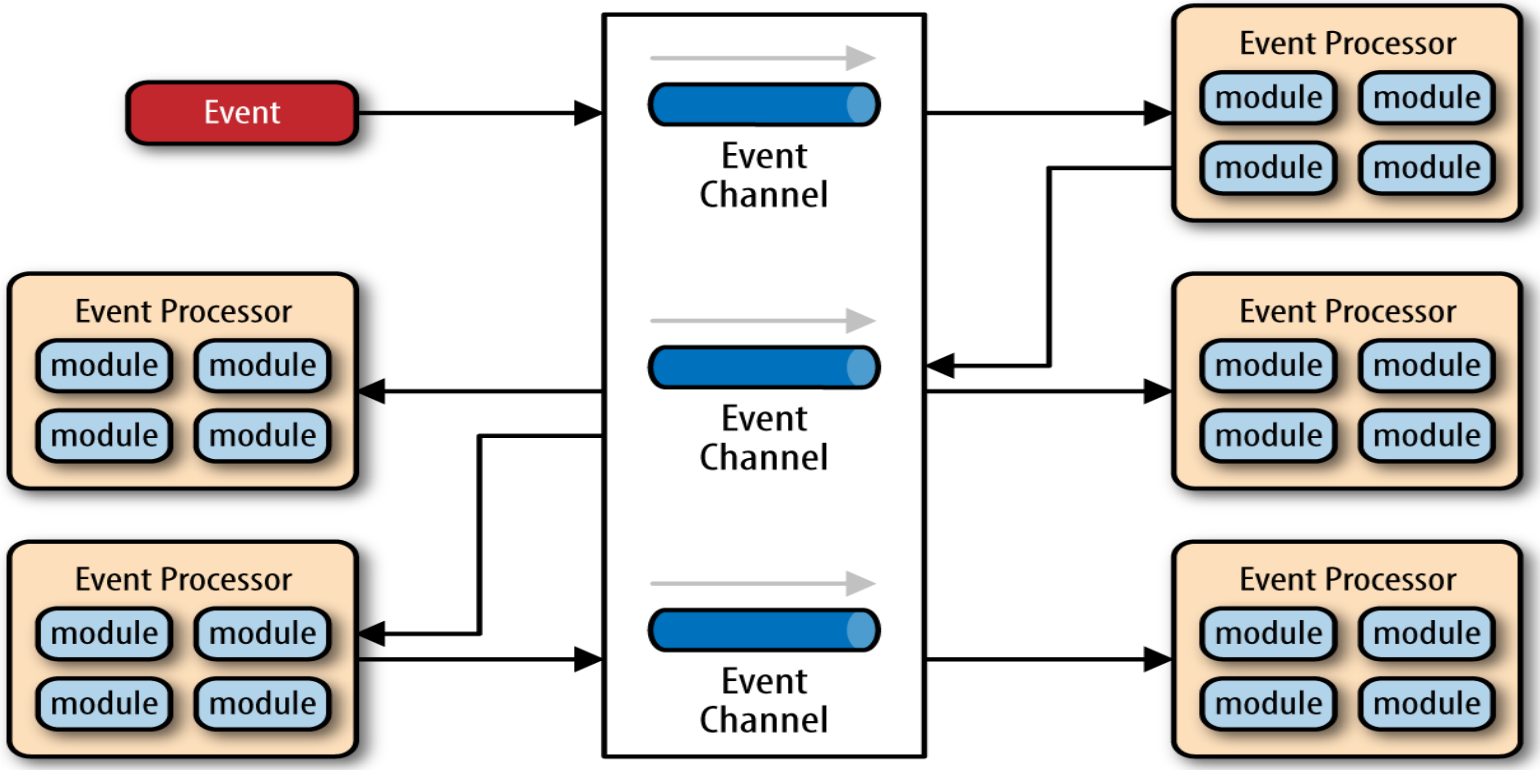


EDA складається з двох основних топологій, посередник і брокер. Топологія посередник зазвичай використовується, коли вам потрібно організувати кілька кроків у події через центрального посередника, в той час як топологія брокера використовується, коли ви хочете організувати ланцюг подій разом без використання центрального посередника. Оскільки характеристики і стратегії здійснення розрізняються між цими двома топологіями, важливо розуміти, кожну з них, щоб знати, яка найкраще підходить для вашої конкретної ситуації.

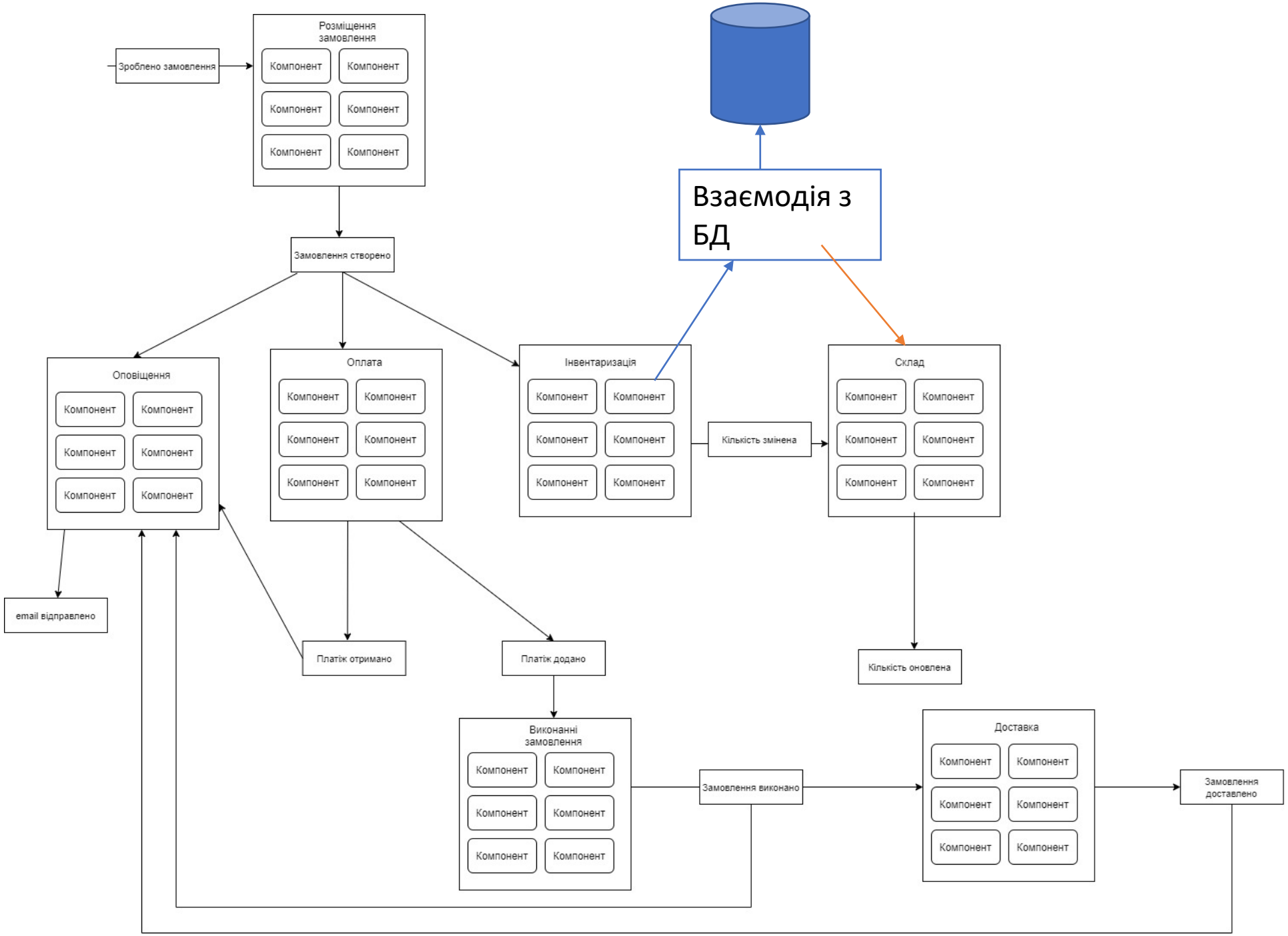
Топологія Брокер

Топологія брокер відрізняється від топології посередника в тому, що немає ніякого центрального посередника подій; а потік повідомлень розподіляється між компонентами процесора подій в каналах через прості повідомлення брокера. Ця топологія корисна, якщо у вас є відносно простий потік обробки подій, і ви не бажаєте (або вам не потрібно) узгодженість подій через єдиний центр.

Є два основних типи компонентів архітектури в рамках топології брокера: компонент брокера і компонент процесора подій. Компонент брокер може бути централізованим або федеративним і містить всі канали подій, які використовуються в потоці подій. Канали подій, що містяться в компоненті брокера можуть бути чергами повідомлень, темами повідомлень, або комбінацією обох.



Як ви можете бачити з діаграми, немає ніякого центрального компонента подій посередника управління і узгодження початкового події; швидше, кожен компонент процесор подій відповідає за обробку події та публікації нової події, вказуючи дію, яку він щойно виконав. Наприклад, процесор подій, який врівноважує портфель акцій може отримати початкову подію під назвою «дроблення акцій». На підставі цієї початкової події, процесор подій може зробити деяку зміну балансу портфеля, а потім опублікувати нову подію для брокера під назвою “ перебалансування портфеля ”, яка потім буде підібрана іншим процесором подій. Зверніть увагу, що бувають випадки, коли подія публікується процесором подій, але не підбирається будь-яким процесором іншої події. Це часто зустрічається, коли додаток еволюціонує або для забезпечення майбутньої функціональності і розширень.

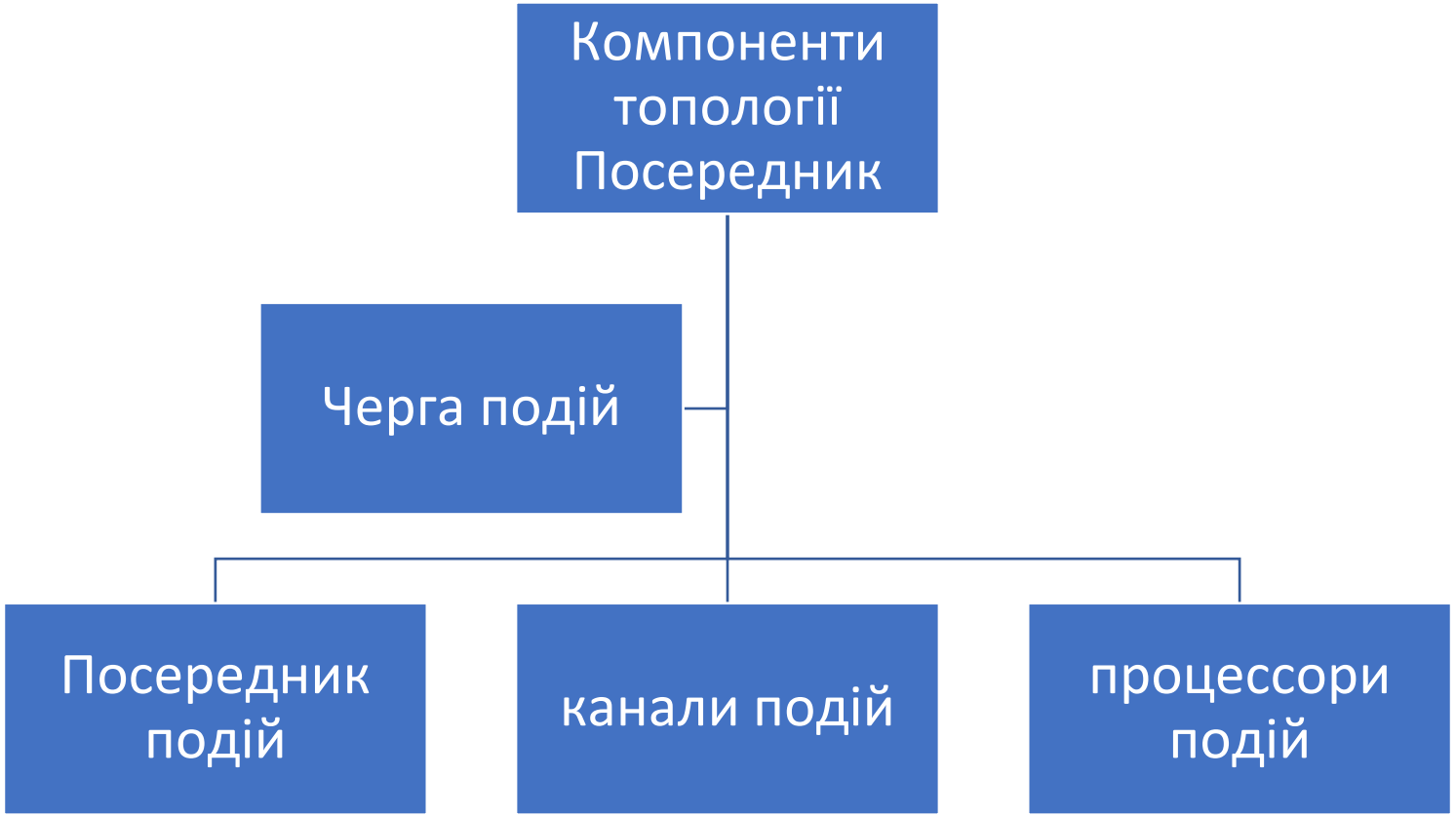

Зроблено замовлення
Розміщення замовлення
Компонент
Замовлення створено
Взаємодія з БД
Оповіщення
Оплата
Інвентаризація
Кількість змінена
Склад
email відправлено
Платіж отримано
Платіж додано
Кількість оновлена
Виконанні замовлення
Замовлення виконано
Доставка
Замовлення доставлено

Плюси/мінуси брокера

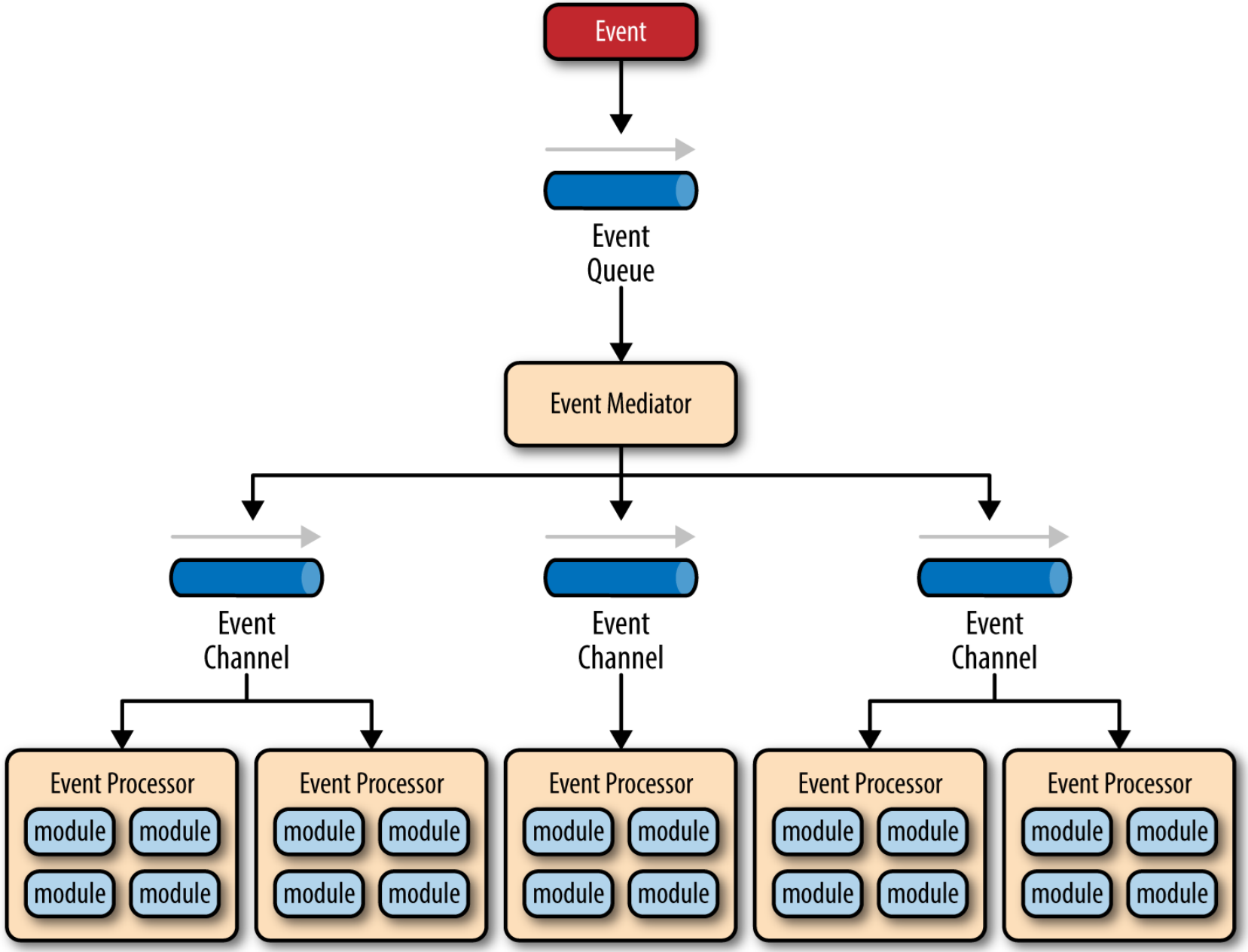
Переваги	Недоліки
Сильно роз'єднані процесори подій	управління робочим процесом
Висока масштабованість	Помилки обробки
Висока гнучкість(High responsiveness)	Складне відновлення (Recoverability)
Висока працездатність	Перезавантаження можливостей (Restart capabilities)
Висока відмовостійкість	Невідповідність даних

Топология Посередник

Топологія посередник використовується для подій, які мають кілька кроків і вимагають деякого рівня узгодженості для обробки події. Наприклад, одна подія для розміщення біржової торгівлі може зажадати, щоб спочатку перевірили угоду, а потім перевірили відповідність цієї біржової торгівлі щодо дотримання різних правил, призначити торгівлю брокеру, порахувати комісію, і, нарешті, місце торгівлі з брокером. Всі ці кроки зажадають деякого рівня узгодженості, щоб визначити порядок кроків і які з них можна робити послідовно а які паралельно.

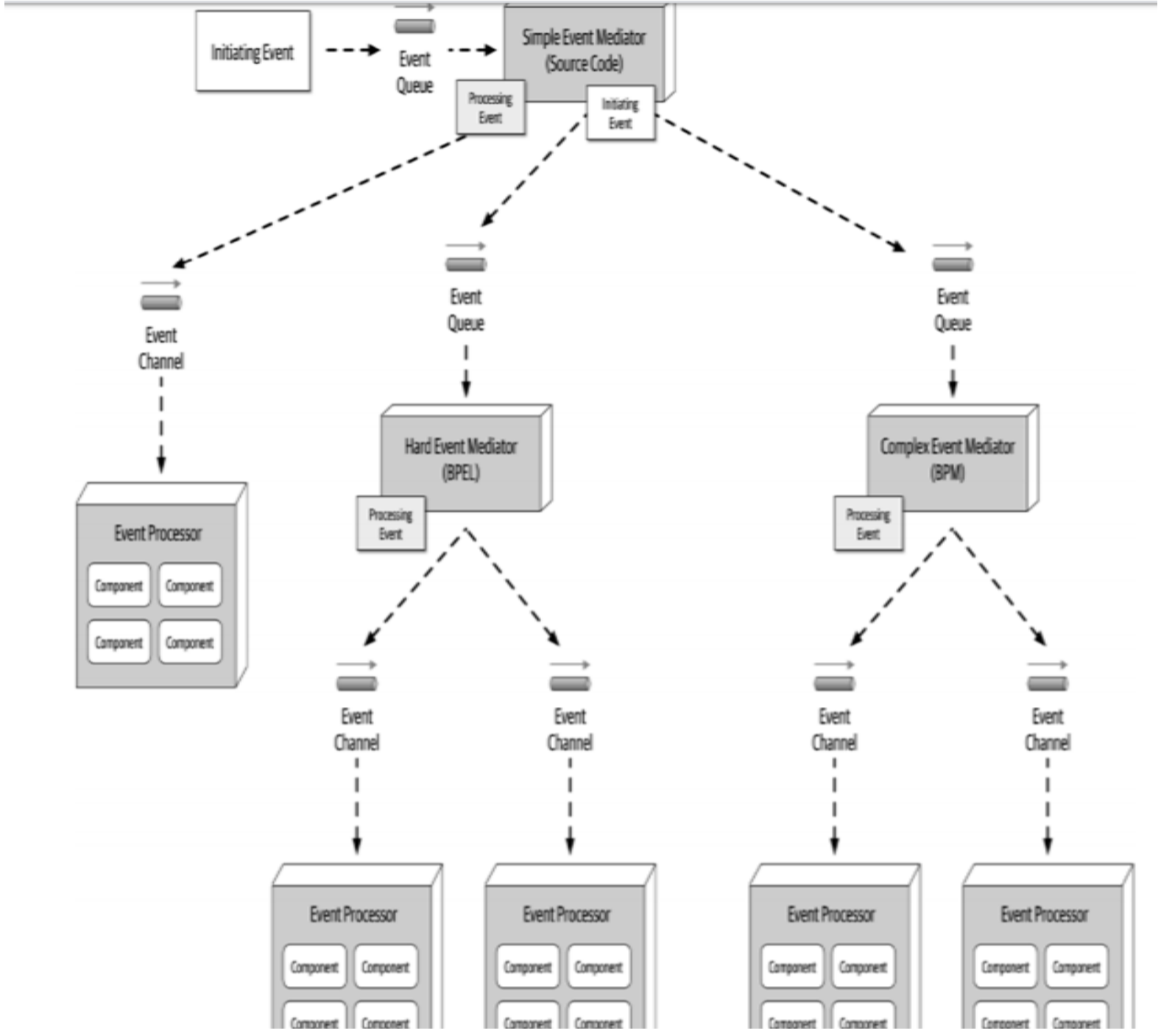


Потік подій починається з клієнта посилає подію в чергу подій, яка використовується для транспортування події до посередника подій. Посередник події отримує вихідну подію і погодить цю подію шляхом відправки додаткових асинхронних подій на канали подій, щоб виконати кожен крок процесу. Процесори подій, які очікують на каналах подій, отримавши подія від посередника подій і виконують певну бізнес логіку для обробки події.


Event
Event Queue
Event Mediator
Event Channel
Event Channel
Event Channel
Event Processor
module
module
module
module
Event Processor
module
module
module
module
Event Processor
module
module
module
module
Event Processor
module
module
module
module
Event Processor
module
module
module
module

Звичайно є від десятків до декількох сотень черг подій. Шаблон не визначає реалізацію компонента «черга подій»; це може бути черга повідомлень, веб-сервіс, або будь-яка їх комбінації.

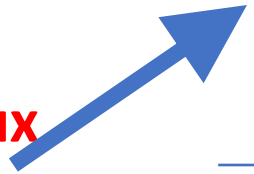
Посередник-подія може бути реалізований різними способами. Як архітектор, ви повинні розуміти, кожен з цих варіантів здійснення, щоб гарантувати, що рішення, яке ви вибираєте для посередника подій відповідає вашим потребам і вимогам.

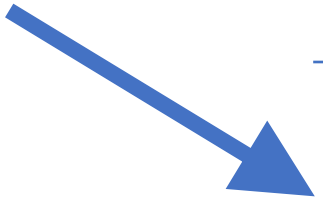


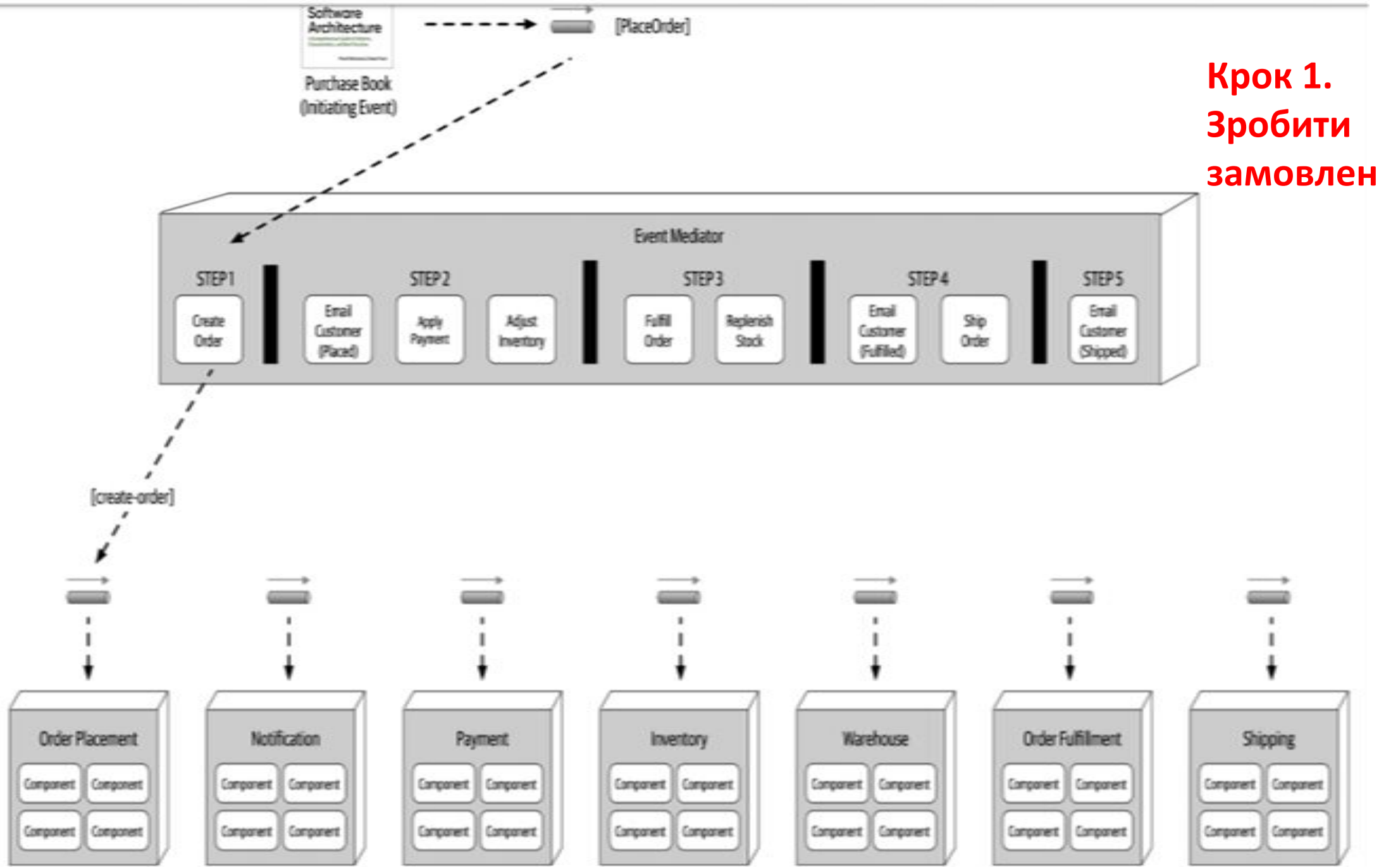
Делегування події на відповідний тип посередника подій

Крок 1. Зробити замовлення	Створення замовлення
Крок 2. Процес замовлення	Email повідомлення замовнику
	Оплата
	Зменшення запасів
Крок 3. Виконання замовлення	Виберіть і упакуйте замовлення
	Замовте більше запасів, якщо це необхідно від постачальника
Крок 4. Відправка замовлення	Email повідомлення замовнику, що замовлення готове до відправлення
	Замовлення відправленне замовнику
Крок 5. Повідомлення замовника	Email повідомлення замовнику, що це замовлення було відправлено

Події в цих кроках проводяться одночасно

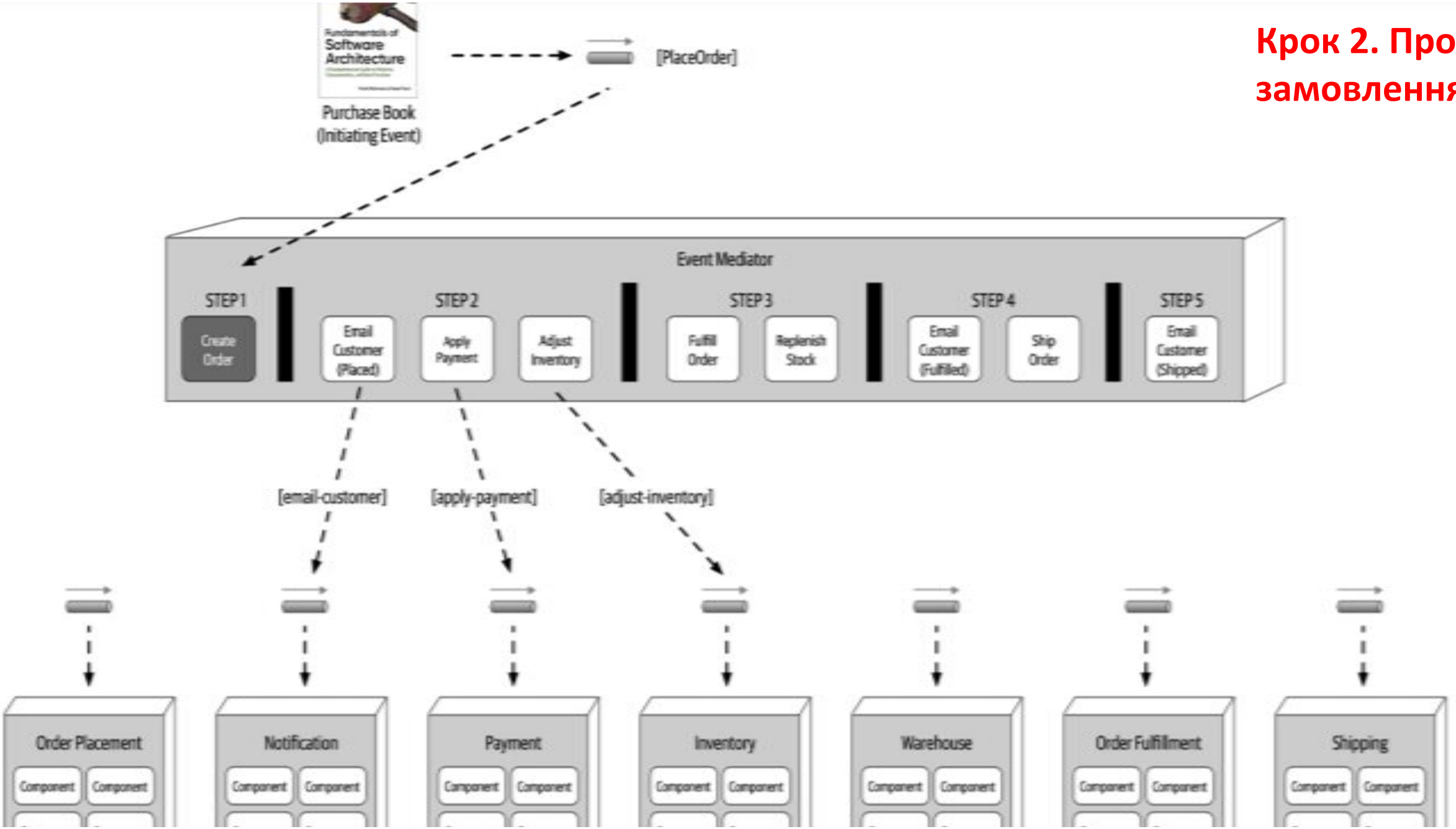




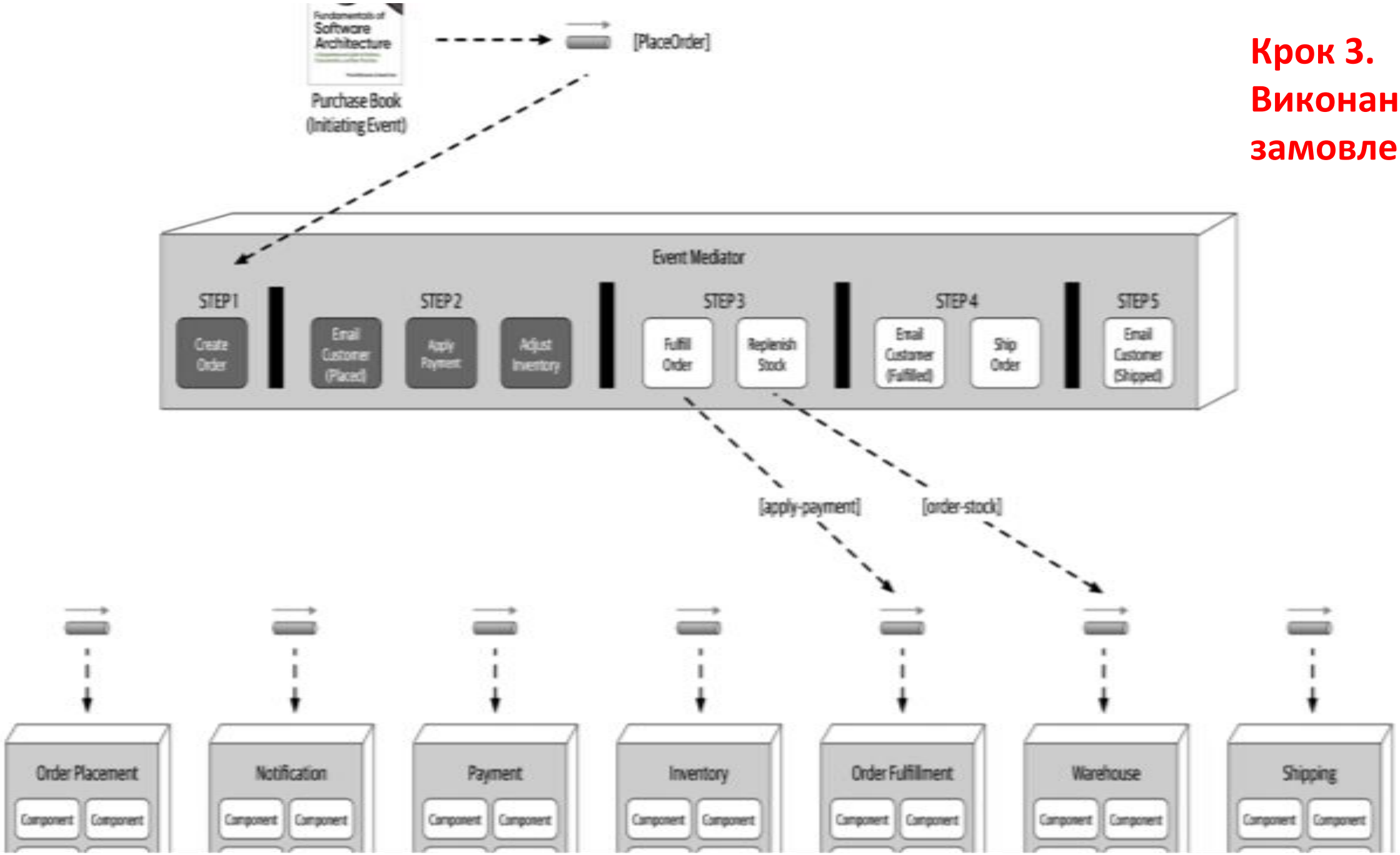


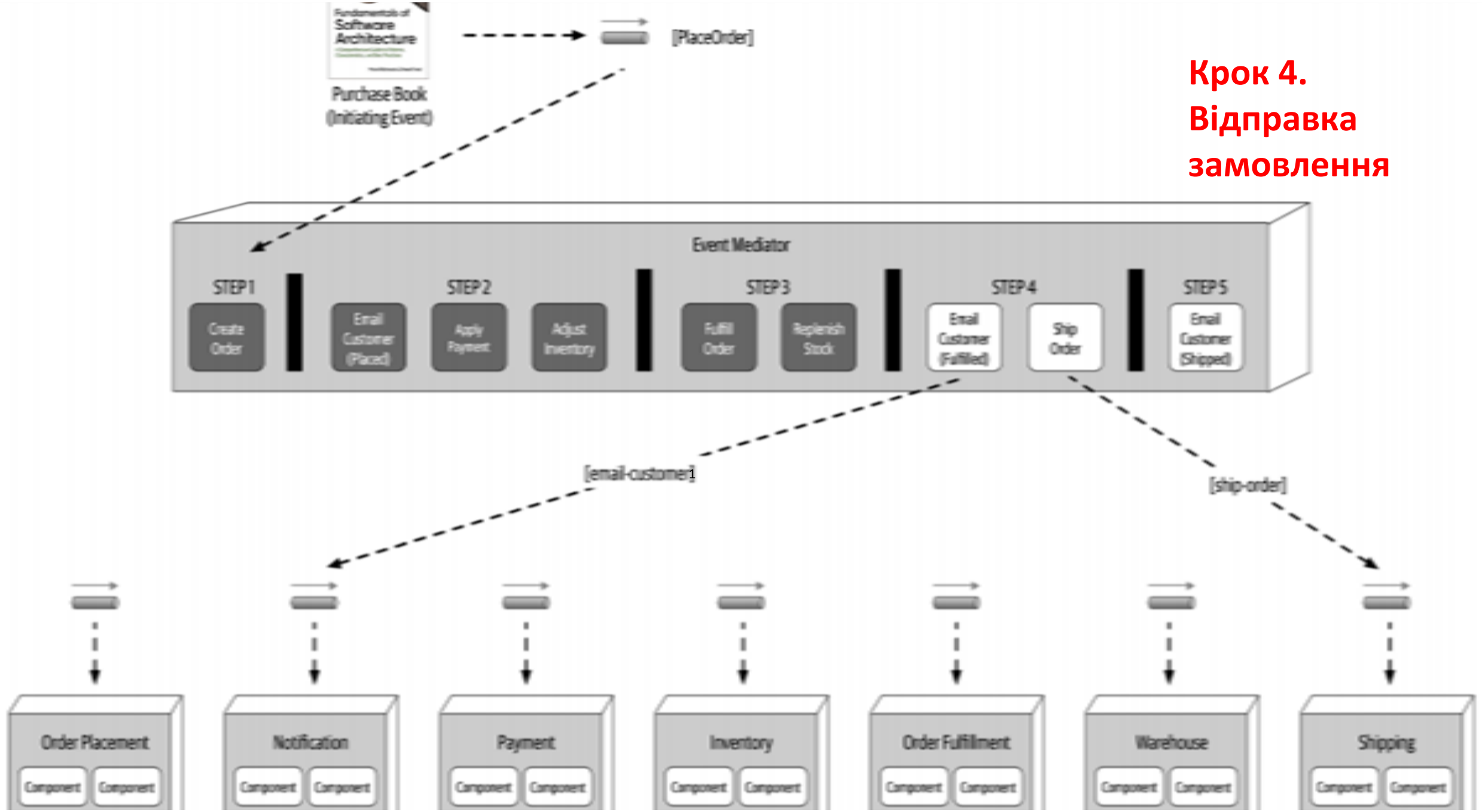
Крок 1. Зробити замовлення

Крок 2. Процес замовлення



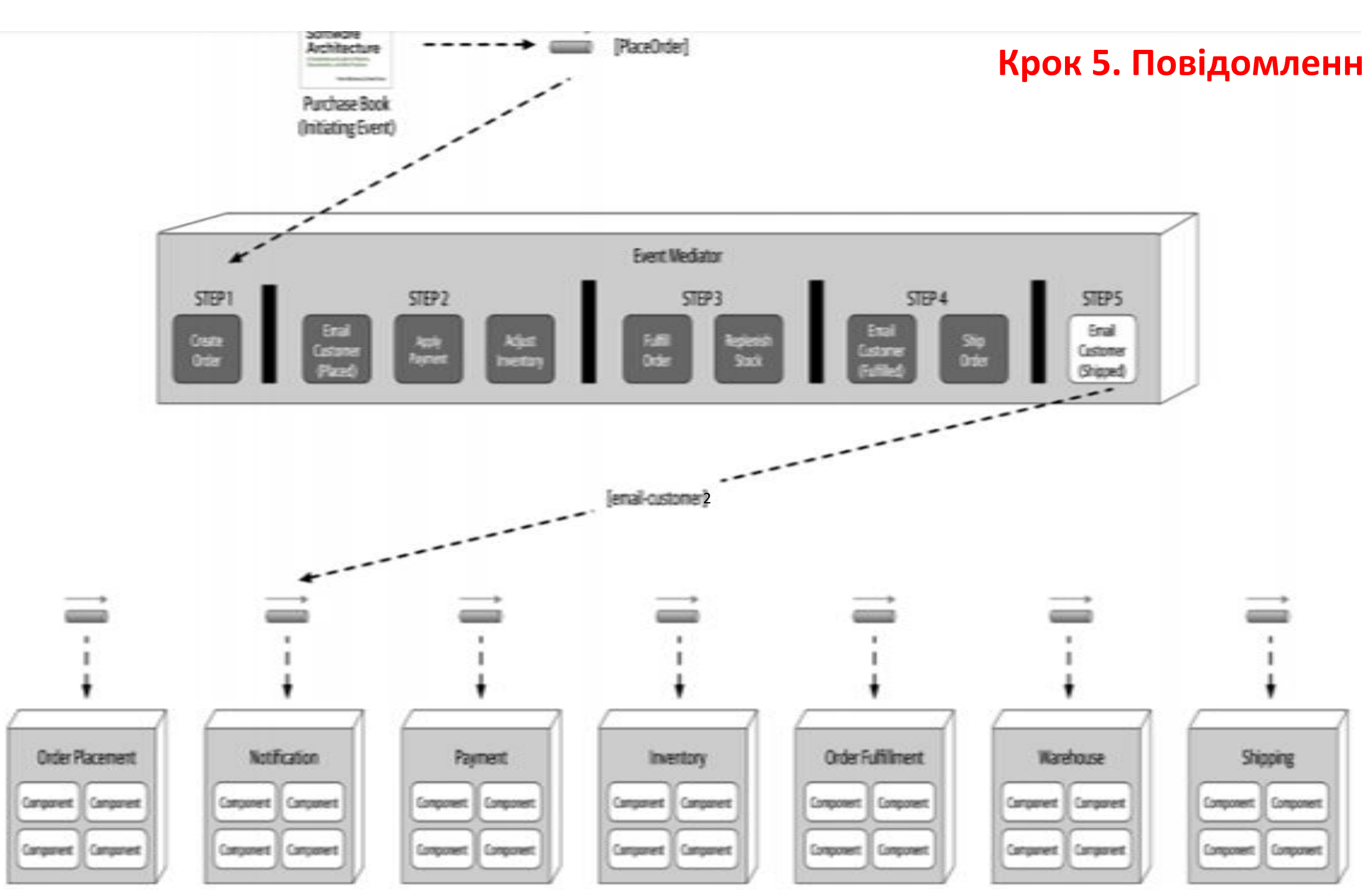
Крок 3. Виконання замовлення





Крок 4. Відправка замовлення

Крок 5. Повідомлення замовника



Плюси/мінуси посередника

Переваги	Недоліки
Контроль робочого процесу	Сильна зчеплиність процесорів подій
Обробка помилок	Менша масштабованість
Відновлення	Зниження продуктивності
Перезавантаження можливостей (Restart capabilities)	Зниження відмовостійкості
Краща узгодженість даних	Моделювання складних робочих процесів

Плюси/мінуси

Керована подіями моделі архітектура є відносно складною для реалізації моделлю, в першу чергу через її асинхронний розподілений характеру. При реалізації цього шаблону, ви повинні вирішувати різні питання, розподіленої архітектури, такі як віддалений доступ процесу, відсутність чуйності і брокер зміни логіки відмови в разі брокера або посередника.

Однією зі складностей даного патерну, є відсутність атомарних операцій для одного бізнес-процесу. Оскільки компоненти процесора подій сильно розв'язані і розподілені, то дуже важко підтримувати транзакційну єдність роботи з ними. З цієї причини при розробці вашого застосування, використовуючи цей шаблон, ви повинні постійно думати про те, які події можуть і не можуть працювати незалежно один від одного і планувати деталізацію ваших процесорів подій відповідно.

Можливо, одним з найбільш складних аспектів подієвого шаблону архітектури є створення, підтримка і управління контрактами компонентів подій процесора. Кожна подія, як правило, має конкретну інформацію, пов'язану з нею(наприклад, значення даних і формат даних, що передаються по відношенню до процесора подій). Це життєво важливо при використанні цього шаблону вибрати стандартний формат даних (наприклад, XML, JSON, об'єкт Java і т.д.) і розробити політику договору управління версіями з самого початку.