**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ І ЗВ'ЯЗКУ**

Факультет інформаційних технологій та кібербезпеки

Кафедра інженерії програмного забезпечення

**Реферат**

з дисципліни «Архітектура та проектування програмного забезпечення»

       Виконав: студент 4 курсу, групи ІПЗ 4.03

Чергенцов Микола Сергійович

Одеса – 2023р.

**1. Що таке архітектура програмного забезпечення?**

**Архітектура програмного забезпечення (англ. software architecture)** — це структура програми чи обчислювальної системи, що включає програмні компоненти, видимі зовні властивості цих компонентів, і навіть відносини з-поміж них.

Процес проектування архітектури програмного забезпечення включає збір вимог клієнтів, їх аналіз і створення проекту для компонента програмного забезпечення у відповідність до вимог. Успішна розробка ПЗ повинна забезпечувати баланс неминучих компромісів внаслідок вимог, що суперечать; відповідати принципам проектування та рекомендованим методам, виробленим з часом; та доповнювати сучасне обладнання, мережі та системи управління. Надійна архітектура програмного забезпечення потребує значного досвіду в теоретичних та практичних питаннях, а також уяви, необхідної для перетворення бізнес-сценаріїв та вимог, які можуть здаватися неясними, у надійні та практичні робочі проекти.

Архітектура програмного забезпечення включає визначення структурованого рішення, що відповідає всім технічним і робочим вимогам, одночасно оптимізуючи загальні атрибути якості, такі як продуктивність, безпека і керованість. Сюди входить серія рішень, що ґрунтуються на широкому діапазоні факторів, і кожне з цих рішень може значно впливати на якість, продуктивність, підтримуваність та загальний успіх програмного забезпечення.

Сучасне програмне забезпечення рідко буває автономним. Як мінімум, у більшості випадків воно буде взаємодіяти з джерелом даних, наприклад, корпоративною базою даних, що надає інформацію, з якою працюють користувачі програмного забезпечення. Зазвичай сучасне програмне забезпечення також має взаємодіяти з іншими службами та мережевими функціями для перевірки автентичності, отримання та публікації інформації та надання інтегрованих середовищ роботи користувачів. Без відповідної архітектури може бути складно, якщо взагалі можливо, здійснити розгортання, експлуатацію, обслуговування та успішну інтеграцію з іншими системами; крім того, вимоги користувачів не будуть дотримані.

Архітектуру програмного забезпечення можна розглядати як зіставлення між метою компонента ПЗ та відомостями про реалізацію в коді. Правильне розуміння архітектури забезпечить оптимальний баланс вимог та результатів. Програмне забезпечення з добре продуманою архітектурою виконуватиме зазначені завдання з параметрами вихідних вимог, одночасно забезпечуючи максимально високу продуктивність, безпеку, надійність та багато інших факторів.

На найвищому рівні проект архітектури має надавати структуру системи, але приховувати деталі реалізації; охоплювати всі випадки застосування та сценарії; намагатися враховувати вимоги всіх заінтересованих осіб; і задовольняти настільки, наскільки можливо, всім функціональним вимогам і вимогам до якості.

**1.1 Мови опису архітектури**

Мови опису архітектури (ADLS) використовуються для опису архітектури програмного забезпечення. Різними організаціями було розроблено кілька різних ADLS, у тому числі AADL (стандарт SAE), Wright (розроблений в університеті Carnegie Mellon), Acme (розроблений в університеті Carnegie Mellon), xADL (розроблений у UCI), Darwin (розроблений у Imperial College у Лондоні) ), DAOP-ADL (розроблений в Університеті Малаги), а також ByADL (Університет L'Aquila, Італія). Загальними елементами всім цих мов є поняття компонента, коннектора і конфігурації.

**1.2 Види (views)**

Архітектура ПЗ зазвичай містить кілька видів, які аналогічні різним типам креслень у будівництві будівель. В онтології, встановленої ANSI / IEEE 1471-2000, види є екземплярами погляду, де думка існує для опису архітектури з точки зору заданої множини зацікавлених осіб.

Приклади видів:

Функціональний/логічний вигляд

Вид код/модуль

Вид розробки (development)/структурний

Вигляд паралельності виконання/процес/потік

Фізичний вид/вид розгортання

Вигляд з погляду дій користувача

Вид з погляду даних

Хоча було розроблено кілька мов для опису архітектури програмного забезпечення, зараз немає згоди з приводу того, який набір видів має бути прийнятий як зразок. Як стандарт «для моделювання програмних систем (і не тільки)» було створено мову UML.

**1.3 Базові фреймворки для архітектури (software architecture frameworks)**

Існують такі фреймворки, що належать до галузі архітектури:

RM-ODP (Reference Model of Open Distributed Processing)

Service-Oriented Modeling Framework (SOMF)

Такі приклади архітектур як фреймворк Захмана (Zachman Framework), DODAF та TOGAF відносяться до галузі архітектури підприємства (enterprise architectures).

**2. У чому важливість архітектури програмного забезпечення?**

Вимоги до сучасного програмного забезпечення стають дедалі складнішими, оскільки користувачі очікують дедалі більше програм. Можливостей простих автономних настільних додатків більше обмаль для більшості комерційних і ділових ситуацій. У світі розвиненого зв'язку програми повинні взаємодіяти з іншими додатками та службами, а також працювати в різних середовищах, наприклад, у хмарі, і на портативних пристроях. На зміну поширеним у минулому монолітних архітектур прийшло компонентне сервіс-орієнтоване програмне забезпечення, що використовує платформи, операційні системи, хост-додатки та мережі для реалізації функцій, про які було не відомо лише кілька років тому.

Ці труднощі впливають як на архітектуру, а й у розгортання, обслуговування і адміністрування програмного забезпечення. Сукупна вартість володіння (TCO) програмного забезпечення тепер переважно складається з витрат, що виникають після розгортання. Додаток з добре продуманою архітектурою забезпечить мінімальну сукупну вартість володіння завдяки зниженню витрат і часу, необхідних на розгортання додатка, забезпечення його роботи, оновлення для задоволення мінливих вимог та усунення проблем. Адміністрація та підтримка користувачів також буде спрощено.

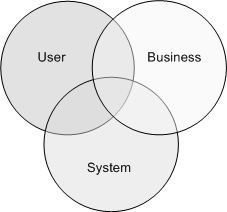
Успішне програмне забезпечення також має відповідати декільком важливим критеріям. Воно має забезпечувати безпеку, щоб додаток та його дані були захищені від атак зловмисників та випадкових помилок. Воно має бути стійким та надійним для мінімізації збоїв та відповідних витрат. Воно має працювати з необхідними параметрами відповідно до вимог користувачів, таких як максимальний час відгуку або певне робоче навантаження. Воно має бути простим у підтримці для зниження витрат на адміністрування та підтримку та достатньо розширюваним для включення неминучих змін та оновлень, які згодом будуть потрібні.

З усіма цими чинниками пов'язані деякі компроміси. Наприклад, реалізація найбезпечніших механізмів з використанням складного шифрування вплине на продуктивність. Реалізація множини параметрів конфігурації та оновлення може ускладнити розгортання та адміністрування. Крім того, що складніша архітектура, то дорожча її реалізація. Правильна архітектура має забезпечувати баланс цих чинників з метою отримання оптимального результату певного сценарію.

**3. Чим займаються розробники архітектури програмного забезпечення?**

Архітектура програмного забезпечення починається з набору вимог. Вони можуть бути виражені у формі діаграм, блок-схем процесу, моделей або документованих списків завдань експлуатації, які мають виконувати програмне забезпечення. Зазвичай клієнт або партнер також висловлює менш точні вимоги, такі як зовнішній вигляд або спосіб роботи певних інтерфейсів для часто зустрічаються завдань. Вимоги також повинні включати інформацію про існуюче програмне забезпечення, системи, обладнання та мережі, з якими взаємодіятиме нове програмне забезпечення; та інші фактори, такі як план розгортання та обслуговування, та, звичайно ж, доступний бюджет проекту.

Розробник архітектури програмного забезпечення повинен враховувати потреби клієнта. Однак загальний термін «клієнт» зазвичай складається з трьох сфер відповідальності, що суперечать: бізнес-вимоги, вимоги користувача і системні вимоги. Бізнес-вимоги зазвичай визначають діапазон таких факторів, як бізнес-процеси, фактори продуктивності (такі як безпека, надійність та пропускна спроможність), а також бюджетні обмеження та обмеження на витрати. Вимоги користувача включають дизайн інтерфейсу, виробничі можливості і простоту використання програмного забезпечення. Системні вимоги мають на увазі можливості та обмеження обладнання, мережі та середовища виконання. На рис. 1 показано, як можуть відрізнятися ці різні вимоги, тому розробнику необхідно створити архітектуру, придатну для областей, що перекриваються.



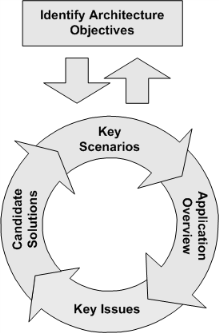
Мал. 1. - конфліктні вимоги типового клієнта

Кожен архітектор програмного забезпечення має власний підхід до збирання та аналізу вимог, а також визначення архітектури. Однак їм часто доводиться відповідати на такі запитання: «Як користувачі працюватимуть із додатком?»; «Як додаток буде розгорнуто у виробничому середовищі та керуватиметься?»; «Які вимоги атрибутів якості для програми, такі як безпека, продуктивність, паралелізм, інтернаціоналізація та конфігурація?»; «Яка має бути архітектура програми для забезпечення гнучкості та зручності в обслуговуванні з часом?» та «Які архітектурні тенденції можуть впливати на додаток зараз і після його розгортання?».

Останнє питання одночасно важливе і цікаве. Хороша архітектура програмного забезпечення не тільки відповідає поточним вимогам клієнтів, але й відповідатиме таким вимогам у найближчому майбутньому. Це впливає на рішення, які приймає розробник архітектури щодо обладнання, компонентів, платформ, середовищ виконання, програмних систем управління та безлічі інших функцій, вбудованих у програмне забезпечення або з якими воно має інтегруватися.

Як і більшість завдань у світі проектування та розробки програмного забезпечення, розробка архітектури – це одночасно запобіжний та ітеративний процес. Багато початкових завдань, такі як аналіз вимог, технічне дослідження та визначення цілей, зазвичай виконуються на початку процесу. Наступний етап – визначення ключових сценаріїв для архітектури. Це основні вимоги, яким має відповідати програмне забезпечення, та обмеження, в рамках яких воно має працювати. На підставі цієї інформації розробник архітектури може створити огляд програми. Цей огляд охоплює такі високорівневі відомості, як тип програми (для Інтернету, телефонів, настільна або хмарна), архітектура розгортання (зазвичай багаторівнева архітектура, компоненти якої зв'язуються через межі обладнання та мережі), відповідні стилі архітектури (наприклад, n-рівнева, клієнт) -сервер або сервіс-орієнтована) та технології реалізації, що найкраще підходять для сценарію.

Після цього розробник може розпочати створення кандидатів архітектури, які задовольняють високорівневим і найважливішим вимогам, визначеним раніше. Після цього проект архітектури перевіряється та тестується у ключових сценаріях, часто у поєднанні з відгуками клієнтів та пробними чи тестовими версіями, для забезпечення отримання оптимального результату. Це малоймовірно за першої ітерації, але при повторенні циклу буде досягнуто відповідність проекту вимогам та ключовим сценаріям. На рис. 2 показано цей поетапний підхід.



Мал. 2. – поетапний процес архітектурного проектування

У міру деталізації проекту та визначення окремих завдань та компонентів архітектор може уточнити та додати подробиці на кожному етапі. Наприклад, після визначення архітектурного стилю та підходу до розгортання архітектор може ухвалити рішення про зв'язок рівнів та компонентів. Сюди може входити вибір протоколу на основі існуючих та майбутніх вимог, а також взяття до уваги оглядів нових технологій та можливостей, визначених у майбутніх стандартах.

Остаточний продукт роботи архітектора - це зазвичай набір схем, моделей і документів, що визначають додаток з кількох точок зору, щоб при об'єднанні вони могли надати розробникам, групам тестування, адміністраторам та управлінню всю необхідну інформацію для реалізації проекту. Ця інформація описуватиме структуру та розміщення компонентів та рівнів програми; спосіб обробки горизонтального перетину ієрархії, наприклад, протоколювання та перевірки; плани тестування та розгортання; та документацію для розробників, адміністраторів та персоналу служби підтримки.

В остаточному проекті також можуть бути зазначені атрибути якості, яким має відповідати програма. Це результат прийнятих рішень та компромісів розробника архітектури щодо консультації з клієнтом. До них відносяться визначення вимог безпеки та план реалізації безпеки, необхідна масштабованість та продуктивність при розгортанні на цільовій платформі, способи реалізації можливості обслуговування та розширюваності та функції забезпечення взаємодії з іншими системами.

**4. Які навички потрібні розробнику програмного забезпечення?**проектування

Очевидно, що розробник архітектури програмного забезпечення повинен мати широкий діапазон як професійних навичок, так і навичок міжособистісного спілкування. Під час етапів аналізу вимоги та перевірки архітектор повинен працювати спільно з клієнтом, консультуватися з партнерами та іншими членами групи та виступати посередником між менеджерами, користувачами та системними адміністраторами. Успішні навички міжособистісного спілкування дозволяють створити кращий вихідний план і більш точний набір вимог, зберігаючи час та зусилля надалі.

Розробник архітектури ПЗ також повинен мати професійні навички для розуміння того, як сучасні програмні системи, платформи та обладнання підтримують вимоги; як фактори мережі та операційної системи можуть впливати на архітектурні рішення; і як тенденції та зміни у цих галузях вплинуть на проект. Після початкового аналізу вимог розробник архітектури також має застосувати свої професійні навички у галузях шаблонів проектування, стандартів зв'язку та обміну повідомленнями, можливостей коду, проблем безпеки та обмежень продуктивності. Все це вимагає глибоких знань технологій, які будуть використовуватись для реалізації підсумкового програмного забезпечення.

Звичайно, для архітектури програмного забезпечення також потрібна уява. Можливість побачити, як системи будуть з'єднуватися і взаємодіяти, як вони будуть секціоновані і розгорнуті, і як вони взаємодіють з користувачами, часто з'являється тільки після того, як розробник архітектури зможе уявити загальне рішення. Для цього потрібен організований підхід і велика увага до деталей, щоб зібрати та зрозуміти всі вимоги та обмеження, а також поступово перетворити все це на динамічний та вичерпний технічний проект. Однак для цього необхідний талант і уява, щоб уявити кінцевий результат і методично добиватися створення ідеального рішення.