*Томашівський Микола Ростиславович*

Abto Software, Developer Engineer, м. Івано-Франківськ

https://orcid.org/0009-0004-9733-461X

*Томашівський Ростислав Петрович*

аспірант, ІФНТУНГ, м. Івано-Франківськ

https://orcid.org/0009-0007-3383-8626

*Рудак Степан Миколайович*, к.т.н.

старший інженер, ПАТ Укрнафта, м. Івано-Франківськ

https://orcid.org/0009-0007-3383-8626

**АРХІТЕКТУРИ АДАПТИВНИХ КОМП’ЮТЕРНИХ СИСТЕМ НА ОСНОВІ МОДЕЛЮВАННЯ ФУНКЦІОНАЛЬНИХ І НЕФУНКЦІОНАЛЬНИХ ВЛАСТВОСТЕЙ ЇХ КОМПОНЕНТІВ**

Основна частина досліджень у сфері робототехніки зосереджена на наданні "підтвердження концепцій" для обґрунтування ідеї дослідника, наприклад, надійного алгоритму планування шляху або системи виявлення зіткнень у реальному часі. Як правило, вони розробляються з нуля або за допомогою зовнішніх бібліотек на основі коду. Проте, коли такі компоненти складаються з інших функціональних модулів, система не демонструє очікуваної поведінки. Отже, підхід, у якому різні функціональні можливості інтегровані, широко відомий як архітектура системи визначає загальну поведінку робота, тому архітектура робота значною мірою впливає на поведінку системи, навіть якщо поведінка окремих функціональних компонентів відома.

Зазвичай програмні архітектури моделюються як набір взаємодіючих компонентів, де деталі реалізації низького рівня приховані, а виражають абстрактні властивості високого рівня. Модель архітектури охоплює кілька рівнів абстракції або точок зору, які задовольняють вимоги різних зацікавлених сторін. Розробник апаратного забезпечення хотів би бачити компоненти, призначені для певного процесора, тоді як архітектор системи був би зацікавлений у топології компонентів.



Рисунок 1 Варіанти моделювання архітектури в робототехніці

Хороша модель архітектури полегшує прийняття рішень і виступає посередником між вимогами та остаточне впровадження. Зокрема, модель архітектури відіграє вирішальну роль у багатьох аспектах життєвого циклу розробки програмного забезпечення, таких як специфікація вимог, проектування системи, впровадження, повторне використання, обслуговування та адаптація під час виконання.

Архітектури, які стосуються аспектів виконання та впровадження, лежать у нижній частині осі абстракції. Мови моделювання, такі як UML, SysML і Marte, моделюють систему, яка ближче до реалізацій програмного забезпечення. Їх семантика здебільшого залежить від реалізації та містить семантичні моделі, такі як шаблони зв’язку, модель обчислення (MoC). Існує ще одна категорія архітектури, яка складається з концептуальних архітектур, які тісно пов’язані з конкретними моделями реалізації. Архітектури, такі як GenoM і ACT-R, надають свої комплекти розробки програмного забезпечення (SDK) для проектування систем, що відповідають цим моделям. Основні переваги таких моделей полягають у тому, що можна застосовувати більш суворі методи перевірки та підтримувати відстежуваність від концепцій предметної області до її реалізації. Однак для перенесення однієї технології реалізації на іншу потрібні значні зусилля. В такій ситуації виникає необхідність розглядати це як деякий спектр моделей на осі абстракції, як показано на рисунку 3.

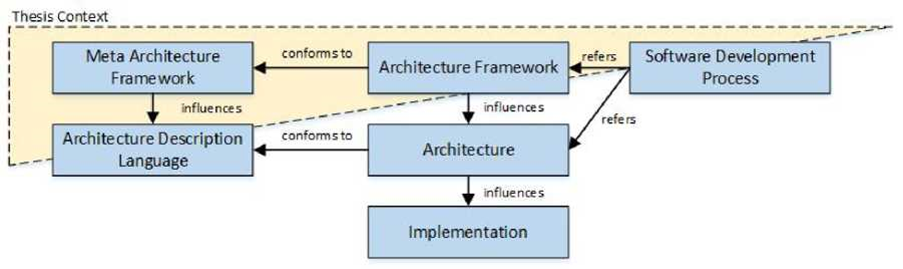


Рисунок 3: Взаємозв'язок між концепцією архітектури та фреймворком

Можна помітити концентрацію архітектур на обох кінцях спектру та значний розрив посередині. Типізований підхід мета-фреймворку, намагається заповнити цю прогалину, діючи як модель архітектора переходу, яка долає розрив між концептуальними архітектурами та архітектурами низького рівня.

**Література**

1. J. R. Anderson, M. Matessa, and C. Lebiere, ACT-R: A theory of higher level cognition and itsrelation to visualattention, Human-Computer Interaction,vol. 12, no. 4, pp. 439 462, 1997.

2. P. Langleyand D. Choi, A unified cognitive architecture for physical agents, in Proceedings of the National Conferenceon Articial Intelligence, vol. 21, no. 2.MIT Press, 2006, p. 1469.

3. D. Garlan, Formal modeling and analysis of software architecture: Components, connectors, andevents, in Formal Methods for Software Architectures. Springer, 2003, pp. 1 24.159.

4. J. Rumbaugh, I. Jacobson, and G. Booch, Uniedmodeling language reference manual, the. Pearson Highe rEducation, 2004.

5. S. Friedenthal, A. Moore and R. Steiner, A practical guide to SysML: the systems modeling language. Elsevier, 2011.