https://orcid.org/0000-0002-9639-3544
https://orcid.org/0000-0002-1268-7837
https://orcid.org/0009-0007-1089-9223

УДК 004.457

О.Є. КОВАЛЕНКО[*], В.М. СМОЛІЙ[*], Л. ЛІ[*, **]

ІНЖЕНЕРІЯ ВИМОГ СИСТЕМ ІНТЕРНЕТУ РЕЧЕЙ

[*]Національний університет біоресурсів і природокористування України, м. Київ, Україна
[**]Університет Бенбу, провінція Аньхой, КНР

Анотація. *Інженерія вимог є одним із визначальних процесів інженерії систем. Інтенсивний розвиток та поширення систем інтернету речей (Internet of Things, IoT) потребує розроблення інформаційного та методичного базисів для їх розробки, особливо на етапі визначення вимог. Складність, неоднорідність, конвергентність систем IoT обумовлюють відповідні проблеми на всіх етапах їх проєктування, в тому числі і на етапі визначення вимог. Процес визначення системних вимог перетворює погляд зацікавлених сторін на бажані можливості та технічне уявлення розробника про те, як система може досягти цих можливостей. Системні вимоги описують вимоги, які повинні відповідати цільовій системі, щоб задовольнити потреби зацікавлених сторін, і виражаються у відповідній комбінації добре сформованих текстових заяв і допоміжних моделей або діаграм. Розробка систем IoT пов'язана з вирішенням проблем, обумовлених конвергентним та адаптивним характером таких систем, що визначають їх складність. Урахування адаптивної природи систем IoT на ранніх етапах життєвого циклу розробки має важливе значення для розробки повної та точної специфікації цільової системи. Існує багато аспектів систем IoT: програмне забезпечення, апаратне забезпечення, мережа в контексті середовища. Програмне забезпечення є основою обробки інформації в системах IoT. Програмне забезпечення контролює систему та забезпечує взаємодію між системою та середовищем, обладнанням та мережею. Апаратне забезпечення містить фізичні об'єкти або пристрої, які є частиною системи IoT і мають бути конкретизовані на етапі вимог, щоб забезпечити врахування особливостей взаємодії з апаратним забезпеченням. Мінливість середовища обумовлює невизначеності у роботі. Система IoT має бути розроблена у такий спосіб, щоб враховувати невизначений характер середовища, в якому система працює. В статті проведено аналіз підходів та засобів управління вимогами з урахуванням контексту систем інтернету речей.*
Ключові слова: *інженерія вимог, інтернет речей, контекст системи, SysM.*

Abstract. *Requirements engineering is one of the defining processes of systems engineering. The intensive development and spread of the Internet of Things (IoT) systems requires the development of an information and methodological basis for their development, especially at the stage of requirements definition. The complexity, heterogeneity, and convergence of the IoT systems cause corresponding problems at all stages of their design, including the stage of requirements definition. The process of determining system requirements transforms the stakeholders' view of the desired capabilities into the developer's technical understanding of how the system can achieve these capabilities. System requirements describe the requirements that the target system must meet in order to satisfy the needs of stakeholders and are expressed in an appropriate combination of well-formed textual statements and supporting models or diagrams. The development of the IoT systems is associated with solving problems caused by the convergent and adaptive nature of such systems, which determine their complexity. Taking into account the adaptive nature of the IoT systems at the early stages of the development life cycle is important for developing a complete and accurate specification of the target system. There are many aspects of the IoT systems: software, hardware, and network in the context of the environment. Software is the basis of information processing in the IoT systems. It controls the system and provides interaction between the system and the environment, equipment and network. Hardware includes physical objects or devices that are part of the IoT system and must be specified at the requirements stage to ensure that the features of interaction with the hardware are taken into account. The variability of the environment causes uncertainties in the work. The IoT system must be designed in such a way as to take into account the uncertain nature of the environment in which the system functions. The*

article analyzes approaches and tools for managing requirements, taking into account the context of the Internet of Things systems.
Keywords: *requirements engineering, Internet of Things, system context, SysML.*

DOI: 10.34121/1028-9763-2025-1-64-75

1. Вступ

Визначення вимог є одним із ключових процесів життєвого циклу систем [1]. У процесах розробки систем інженерія вимог виконує завдання визначення всіх матеріальних і нематеріальних аспектів, які мають відношення до системи. Для цього передбачається, якою буде система, коли вона стане реальною. Так можна визначити ті частини реального світу, які потенційно впливатимуть на вимоги системи. Щоб мати можливість правильно і повно визначити вимоги до системи, необхідно якомога точніше визначити зв'язки між окремими матеріальними та нематеріальними аспектами. Частина реальності, яка відповідає вимогам системи, називається контекстом системи.

Контекст системи — це частина системного середовища, яка є релевантною для визначення, а також для розуміння вимог системи, що розробляється [2]. Контекст для системи IoT — це будь-які знання, якими можна поділитися, для представлення ситуацій або умов із різних частин системи. Діапазон контекстів може містити, але не обмежуватися ними, контекст пристрою, контексти мережі, контексти системи та контексти середовища [3]. Безперечно, комунікаційні об'єкти (об'єкти чи речі) у контексті Інтернету речей (IoT) відіграють активну роль у людській діяльності, системах і процесах. Інтернет речей (у перспективі Web of Things) змінює моделі взаємодії з граничними пристроями [4]. Можливості та нові послуги, які вони надають, дозволяють користувачам виконувати автоматичні операції та відстежувати цікаві дані. Незважаючи на те, що багато операцій виконуються пристроями автономно, користувач повинен розуміти надані дані та налаштовувати власні служби відповідно до них.

Інженерія вимог — це систематичний і строгий підхід до розробки специфікації та управління вимогами системи на основі знань, що формуються на основі:

– розуміння відповідних вимог, досягнення консенсусу серед зацікавлених сторін щодо цих вимог, документування їх відповідно до стандартів та систематичне керування ними;

– розуміння та документування бажань і потреб зацікавлених сторін, визначення та управління вимогами для мінімізації ризику впровадження системи, яка не відповідає бажанням і потребам зацікавлених сторін.

Для досягнення цих цілей реалізуються чотири основні процеси:

1) виявлення: використовуються різні методи для отримання вимог від зацікавлених сторін та інших джерел і для більш детального уточнення вимог;

2) документування: виявлені вимоги описуються належним чином. Для документування вимог використовуються різні методи за допомогою природної мови або концептуальних моделей;

3) перевірка та узгодження: щоб гарантувати дотримання попередньо визначених критеріїв якості, задокументовані вимоги мають бути перевірені та узгоджені на ранній стадії;

4) управління вимогами є ортогональним до всіх інших видів діяльності та включає будь-які заходи, необхідні для структурування вимог, підготовки їх, щоб вони могли використовуватися у різних ролях, підтримки узгодженості після змін та забезпечення їх реалізації.

Ці основні дії можна застосовувати для різних рівнів абстракції вимог, як-от вимоги зацікавлених сторін, системні вимоги та вимоги до програмного забезпечення. Їх виконання може відбуватися за різними процесами, наприклад, процесами, рекомендованими в [5].

Управління вимогами (RM) виконується для забезпечення узгодження вимог до системи, підсистеми та системного елемента з іншими представленнями, аналізами та артефактами системи протягом життєвого циклу. Це включає забезпечення розуміння вимог, отримання зобов'язань, керування змінами, підтримку двонаправленої відстежуваності між вимогами та рештою визначення системи, а також узгодження з ресурсами та графіком проєкту. Різні обмеження проєкту впливають на розробку вимог. Наприклад, люди, фактори цільової області або організаційні обмеження (наприклад, просторовий розподіл або тимчасова доступність учасників проєкту) мають великий вплив на вибір відповідних методів.

Практики, орієнтовані на архітектуру, набувають широкого визнання в системній інженерії. Цей процес передбачає фіксацію структури, поведінки, правил і їхніх зв'язків для створення абстрактного представлення системи, яке часто називають моделлю системи. Центральними для правил, які керують системою, є вимоги, що висуваються до неї часто різними зацікавленими сторонами. Ці вимоги допомагають керувати процесом розробки складних та конвергентних систем [6, 7]. У низці статей [8–10] розвивається підхід до розширення ідеї структурованих вимог (вимог, визначених через упорядковану структуру з певними фрагментами вмісту, які потрібно заповнити) засобами SysML за допомогою налаштованих стереотипів, які допомагають забезпечити дотримання структури вимог через атрибути на основі моделі. Цей підхід допомагає перенести моделювання та управління вимогами далі в модельну парадигму від класичних текстових визначень. Крім того, вимоги часто налаштовуються організацією, визначаючи їх за допомогою додаткових атрибутів. Ці додаткові атрибути додаються до структурованих вимог на основі моделі (model-based structured requirement, MBSR), щоб створити чітко визначений стереотип організаційних вимог. Цей підхід може допомогти підтримувати більш формалізоване моделювання вимог, аналіз, управління та комунікацію протягом усього процесу розробки системи.

Системи IoT можуть змінити багато галузей і підвищити ефективність, безпеку та розуміння багатьох аспектів нашого життя. Однак розробка систем IoT може бути складною через складність системи, потребу в обробці в реальному часі та потребу в управлінні величезними обсягами даних.

Управління вимогами є життєво важливим компонентом у розробці систем IoT. Він включає ідентифікацію, збір, визначення, перевірку та підтримку вимог системи протягом усього її існування. Якщо вимоги керуються належним чином, кінцевий продукт може задовольняти вимоги зацікавлених сторін і відповідати призначенню.

Метою статті є аналіз підходів і засобів формалізації представлення та управління вимогами в процесах інженерії систем IoT.

2. Стандартизація представлення вимог до систем

Важливою умовою ефективної та успішної реалізації проєктів програмної та системної інженерії є стандартизація та уніфікація процесів життєвого циклу систем, де одним із ключових етапів є визначення та супроводження вимог. У фундаментальному масиві знань системної інженерії SEBoK (System Engineering Body of Knowledge) [11] першим видом діяльності при визначенні системи є визначення системних вимог. Визначення системних вимог використовує результат діяльності визначення концепції системи для визначення того, що система повинна робити, щоб інтегрований набір потреб був реалізований цільовою системою (System of Interest, SoI). На рис. 1 показано, як надається вхідна інформація для процесів визначення архітектури системи та визначення дизайну системи, а також процесу верифікації системи відповідно до рекомендацій ISO15288 [1].

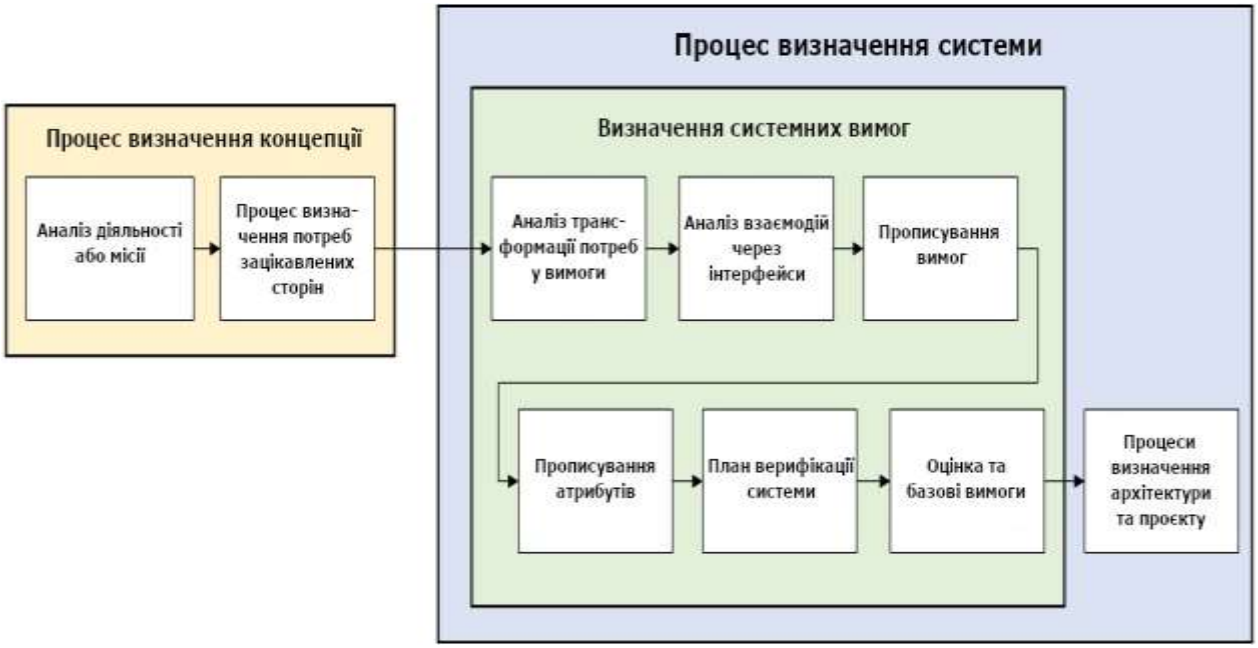

Рисунок 1 — Визначення системних вимог у процесі визначення системи

Кожна вимога представляє інженерне рішення щодо того, що має робити SoI, або якість, яку має мати система, щоб задовольнити потреби, з яких вони були перетворені. Визначення того, що система повинна робити, щоб задовольнити потребу, відбувається за допомогою процесу детального аналізу вимог, який може включати розробку та використання моделей, моделювання та прототипів.

Визначення вимог є результатом «формального перетворення однієї чи кількох потреб або батьківських вимог в узгоджене зобов'язання суб'єкта виконувати певну функцію або мати певну якість у межах визначених обмежень із прийнятним ризиком» [12]. У цьому контексті використання терміна «якість» означає невід'ємну властивість або відмітний атрибут.

Процес визначення системних вимог перетворює погляд зацікавлених сторін на бажані можливості на технічне уявлення розробника про те, як система може досягти цих можливостей. Системні вимоги описують вимоги, яким повинна відповідати цільова система (SoI), щоб задовольнити потреби зацікавлених сторін, і виражаються у відповідній комбінації добре сформованих текстових заяв і допоміжних моделей або діаграм. Вхідними даними в цей процес є концепції життєвого циклу та інтегрований набір потреб, створених під час діяльності з визначення концепції системи. Системні вимоги відіграють важливу роль у системній інженерії, оскільки вони:

– складають основу архітектури системи та проєктування;

– формують основу системної інтеграції та верифікаційної діяльності;

– забезпечують обмін інформацією між різними членами команди проєкту, які взаємодіють протягом усього проєкту.

Результати процесу визначення системних вимог служать вхідними даними для низки інших технічних процесів, які включають визначення проєкту системи, визначення архітектури системи та перевірку системи.

Для забезпечення обміну даними про вимоги до систем застосовуються уніфіковані нотації з можливостями формалізованого представлення вимог, такі як SysML/UML, BPMN, ArchiMate, Flowchart тощо.

Окремо слід звернути увагу на формат обміну вимогами Requirements Interchange Format (ReqIF) [13]. Загальний непатентований формат інформації про вимоги потрібен, щоб

подолати розриви та задовольнити нагальну потребу розробників в обміні інформацією про вимоги між собою, не втрачаючи переваг керування вимогами за межами організацій. За допомогою спеціального формату обміну специфікаціями вимог можна подолати розриви у:

– співпраці між організаціями-партнерами, що покращується завдяки перевагам застосування методів керування вимогами за межами організацій;

– організації-партнери не повинні використовувати той самий інструмент розробки вимог, а постачальникам не потрібно мати кілька інструментів розробки вимог, щоб задовольнити потреби своїх клієнтів щодо сумісності;

– всередині організації можна обмінюватися інформацією про вимоги, навіть якщо для створення вимог використовуються різні інструменти.

Формат обміну вимогами (ReqIF), описаний у специфікації [13], визначає такий відкритий непатентований формат обміну. Обмін інформацією про вимоги здійснюється шляхом передачі документів XML, які відповідають формату ReqIF.

Файл OMG ReqIF складається з XML із кореневим елементом REQ-IF, який містить інформацію про сам файл, а також типи даних і вимоги, що він описує. Контейнери для вимог у ReqIF називаються об'єктами специфікації (SpecObject), що мають визначені користувачем атрибути. Кожен атрибут має тип даних, який є одним із таких: Boolean, Integer, Real, String, Enumeration (зі значеннями, визначеними користувачем) і XHTML, який також призначений для форматованого тексту та вбудованих об'єктів, включаючи зображення. Деякі типи даних можуть бути додатково обмежені, наприклад, діапазон числових значень. Зв'язки між об'єктами представлені як SpecRelations, які також можуть мати атрибути. Ієрархічні дерева створюють структуроване представлення SpecObjects, яке називається Specifications. На один SpecObject дозволяється робити множинні посилання.

3. Формалізація представлення вимог до систем

Розробка систем IoT (Інтернету речей) пов'язана з вирішенням проблем, обумовлених конвергентним та адаптивним характером таких систем, що визначають їх складність. Врахування адаптивної природи систем IoT на ранніх етапах життєвого циклу розробки має важливе значення для розробки повної та точної специфікації цільової системи. IoT — це гетерогенний набір взаємопов'язаних речей (машин, об'єктів, людей, тварин), проміжного програмного забезпечення та програмного забезпечення (розумних) систем. IoT революціонізує сфери свого застосування від споживчих і комерційних до промислових та інфраструктурних [14, 15].

Речі, що утворюють систему IoT, мають спільні цілі й повинні взаємодіяти та співпрацювати, щоб виконувати функції розумної системи. Необхідність спілкуватися з фізичними пристроями та керувати ними надає таким системам нові характеристики. Багато аспектів IoT — програмне забезпечення, апаратне забезпечення, мережа та середовище — необхідно враховувати при визначенні взаємодії системи. Крім того, адаптивний характер систем Інтернету речей створює складності та виклики, які необхідно вирішувати на ранніх стадіях розробки. Виявлення та документування складних взаємодій між різними учасниками або суб'єктами такої системи має вирішальне значення для успішного розвитку системи. Необхідно визначити потенційні виняткові ситуації та дослідити можливу адаптивну поведінку. Відхилення від очікуваної поведінки, які не виявлено під час виявлення вимог, можуть зрештою призвести до неповної або неоднозначної специфікації системи під час аналізу та й до впровадження, якому бракує певних функціональних можливостей або навіть поводиться ненадійним чином. У той час, як складність вимог визнається, досі немає усталених засобів охоплення різних елементів і опису складних взаємодій між різними вимірами таких систем.

Оскільки інженерія вимог ґрунтується на знаннях про цільову систему, то першим кроком формалізації вимог є формалізація контексту цільової системи. Одним із поширених підходів до формалізації знань є використання онтології. Запропонована у [3] модель контекстної онтології надає словники для представлення контекстних знань про ситуації, пов'язані з мережею, і стани систем IoT. Модель розроблена для сприяння обміну контекстами та розуміння таких контекстів обміну між різнорідними вузлами системи IoT, щоб забезпечити оптимальне контекстно-залежне керування основними бездротовими сенсорними мережами (wireless sensor networks, WSN) системи. Кожен вузол містить один екземпляр моделі, яка може використовувати аксіоми виразів для виведення відповідних контекстних знань згідно з інформацією, якою обмінюються на різних рівнях і в різних сферах.

Ця модель розроблена, як ієрархічна структура контекстних класів, де кожен клас характеризує контекстну інформацію однієї або кількох складових частин системи IoT. Нижній рівень моделі — це необроблена інформація, безпосередньо успадкована від компонентів пристрою та системних об'єктів. Верхні рівні — це запропонована контекстна онтологія для моделювання та представлення виведених контекстів для складових частин системи.

Тут необроблена інформація відноситься до будь-якої інформації, зазвичай у числовому форматі, отриманої безпосередньо з апаратних і програмних компонентів одного вузла. Крім того, необробленою інформацією можна обмінюватися між вузлами для визначення глобальних контекстів низького рівня, наприклад, вузли в певній області можуть обмінюватися необробленими показаннями датчиків, щоб отримати екологічний контекст свого оточення.

Зазвичай контексти низького рівня можна отримати шляхом порівняння числових значень необробленої інформації з попередньо визначеними пороговими значеннями. Як простий приклад, «ВИСОКИЙ»/«НИЗЬКИЙ» енергетичний стан вузла — це контекст низького рівня, отриманий шляхом порівняння рівня залишкової енергії на вузлі з пороговим значенням, що представляє половину повної ємності акумулятора. Крім того, ймовірнісні моделі, такі як приховані моделі Маркова (hidden Markov model, HMM) [16, 17], можна застосовувати при отриманні контекстів високого рівня, які не сприймаються безпосередньо, а виводяться з контекстів нижчого рівня. Отже, мають певний рівень невизначеності залежно від точності контекстів нижчого рівня, що використовуються [18]. Наприклад, якщо вважається, що отримані переміщення та розташування певного вузла A переважно (але не на 100%) правдиві, замість того, щоб зробити висновок, що «A покидає мережу», висновок, який враховує рівень невизначеності, такий як «З високою ймовірністю A залишає мережу», може бути більш доречним для опису стану події.

Структура онтологічної моделі контексту систем IoT представлена на рис. 2. Клас Context Resource є кореневою сутністю, яка має два прямих класи-нащадки: класи Local Context і Global Context.

Інтернет речей (IoT) є прикладом системи систем (SoS), яка вимагає як інноваційних, так і еволюційних підходів для врахування всіх аспектів розробки таких систем. Протягом багатьох років було розроблено різні методології, інфраструктури, платформи та інструменти IoT, які потребують упорядкування та систематизації на основі:

1) формулювання базових понять для визначення найбільш підходящого класу продуктів розробки IoT — методологій, фреймворків, платформ та інструментів;

2) перегляду відповідних продуктів за допомогою порівняльного та практичного підходу, заснованого на загальних інженерних функціях SoS із позицій основних характеристик систем IoT, тобто сумісності, масштабованості, інтелектуальності та автономності.

Зменшення плутанини, пов'язаної з методологіями, фреймворками, платформами та інструментами IoT, а також фіксація їх поточного стану дозволить спростити підхід до розробки систем IoT. Короткий огляд модельно-орієнтованих підходів для концептуального

моделювання систем IoT представлено у роботі [19]. На сьогоднішній день розроблено низку заснованих на SysML нотацій для опису вимог та архітектури систем IoT [20, 21].

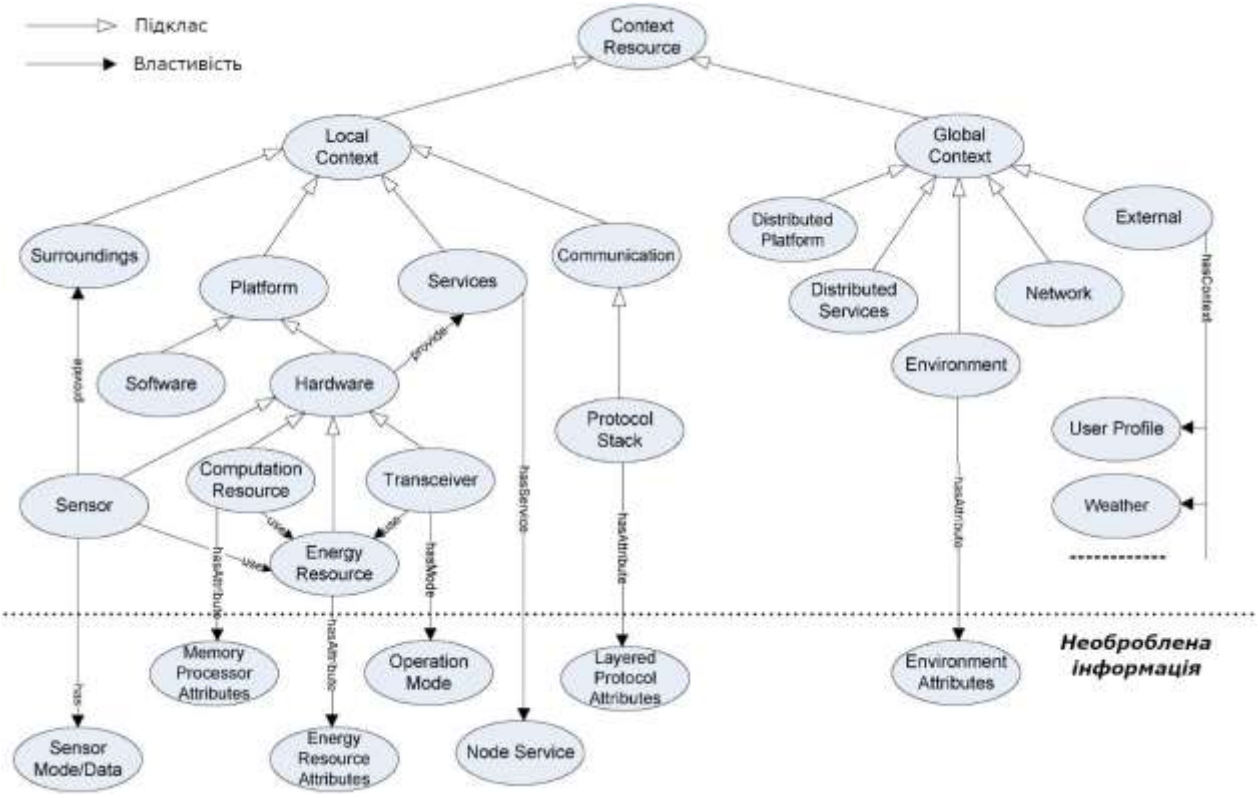

Рисунок 2 — Онтологічна модель контексту систем IoT [3]

4. Приклад опису вимог системи IoT

Процеси діяльності (бізнес-процеси), що підтримуються IoT і обмінюються даними з пристроями Інтернету речей (IoT), набувають все більшого поширення. Впровадження технологій IoT на ранніх етапах процесу розробки діяльності вимагає роботи зі складністю та неоднорідністю таких технологій під час проєктування та аналізу. Для представлення вимог можуть використовуватись фреймворки та онтології IoT, а також розширення BPMN, яке покращує виразність відповідних нотацій моделювання процесів і дозволяє відповідне представлення пристроїв IoT як з архітектурної, так і з поведінкової точки зору. У сфері управління процесами діяльності використання підходів, заснованих на моделюванні, визнано ефективною технологією для аналізу процесів. Імітаційні моделі необхідно параметризувати відповідно до відповідних властивостей досліджуваного процесу. Іноді такі параметри можуть змінюватися протягом терміну реалізації процесу, роблячи імітаційну модель неадекватною щодо фактичної поведінки процесу.

Однією з галузей все ширшого застосування IoT є освіта. Персоналізація освіти є одним із напрямів розвитку на всіх рівнях освітнього сектору. Викладачі часто стикаються з проблемою збору та розуміння індивідуальних потреб здобувачів освіти, не кажучи вже про їх вирішення. Технологія IoT стала чудовим засобом збору даних із аудиторій і лабораторій, а технології великих даних дозволяють обробляти ці дані. Отже, IoT пропонує потенційні рішення деяких ключових проблем, які постануть перед персоналізованою освітою в майбутньому. У статті [22] запропонована система IoT, яка дозволить персоналізувати навчання для великих груп студентів в аудиторіях і лабораторіях.

Основою розробки є ідея побудови системи IoT, яка розгортається у великих лекційних залах і обчислювальних лабораторіях (IoT у цьому випадку відноситься до поєднання фізичних пристроїв, таких як датчики та обчислювальні системи для обробки інформації, створеної локально, але також існуючої як частини інших взаємопов'язаних IT-систем). Впровадження цієї технології дозволяє налаштувати та покращити досвід навчання студентів, даючи можливість оптимізувати навчальні матеріали та різні форми їх доставки. Це дозволяє викладачам зосередитися на індивідуальних навчальних потребах здобувачів освіти і методах навчання. Крім того, існує обґрунтоване очікування зменшення як прямих витрат, так і витрат на персонал шляхом автоматизації звичайних завдань, які безпосередньо не пов'язані з фактичним навчанням студентів, тобто завдань, які вимагають академічних знань. На рис. 3 представлена діаграма вимог до IoT-системи університету.

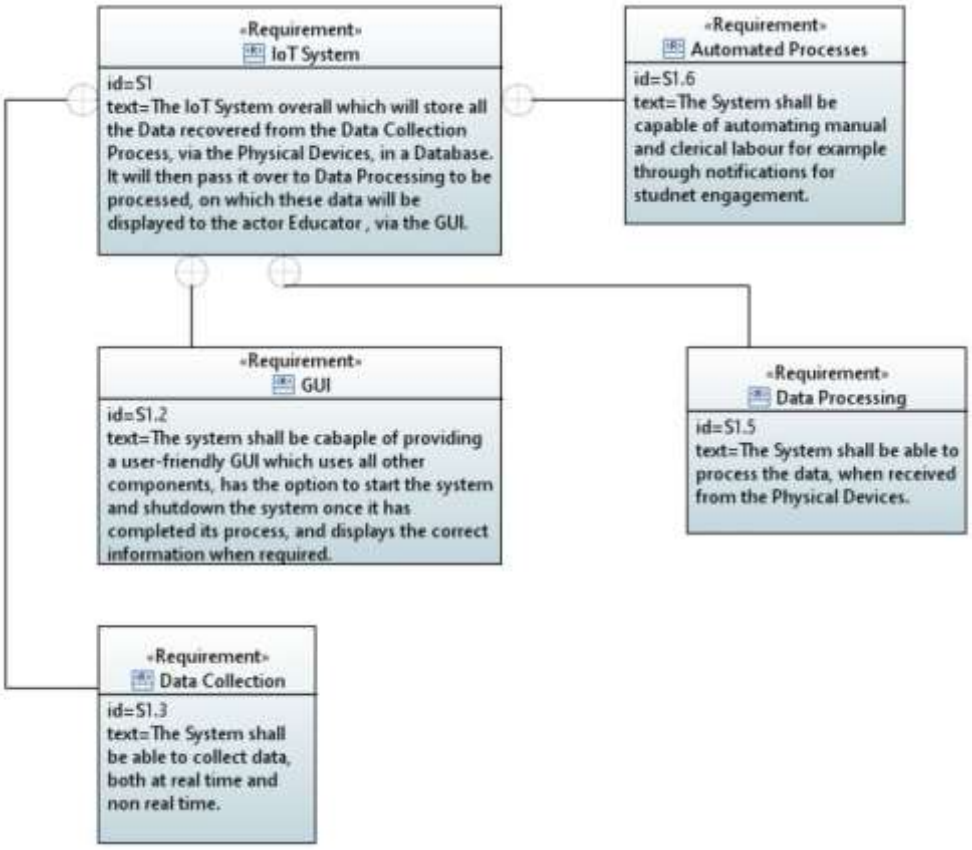

Рисунок 3 — Діаграма вимог системи IoT університету [22]

Існує багато аспектів систем IoT: програмне забезпечення, апаратне забезпечення, мережа в контексті середовища. Програмне забезпечення є основою обробки інформації в системах IoT. Програмне забезпечення контролює систему й забезпечує взаємодію між системою та середовищем, обладнанням та мережею. Апаратне забезпечення містить фізичні об'єкти або пристрої, які є частиною системи IoT і мають бути конкретизовані на етапі вимог, щоб забезпечити врахування особливостей взаємодії з апаратним забезпеченням. Мінливість середовища обумовлює невизначеності у роботі. Система IoT має бути розроблена у такий спосіб, щоб враховувати невизначений характер середовища, в якому система працює. Середовище складається з користувачів, фізичних осіб, а також усіх інтелектуальних систем/об'єктів/пристроїв, які можуть взаємодіяти з системою через інтернет. Мережа — це фундамент, який об'єднує середовище та систему, а також полегшує зв'язок між різними

речами, які взаємодіють із системою, що розробляється. На відміну від традиційних програмних застосунків, системи IoT слід проєктувати та розробляти з урахуванням різних аспектів IoT із ранніх етапів розробки. Виявлення вимог до IoT є проблемою, оскільки типові методи розробки вимог не можуть урахувати особливості функціонування конкретної системи.

5. Засоби інженерії вимог

Переважна більшість сучасних середовищ управління вимогами та розробки інформаційних комп'ютерних систем містять у собі засоби управління вимогами [23, 24]. Існують спеціалізовані засоби для управління вимогами [25, 26], а також технології [27, 28] та мови [29] розробки вимог орієнтовані на використання в системах IoT.

Специфіка розробки вимог для систем IoT описана у статті [30]. Розглянемо деякі засоби управління вимогами.

5.1. IBM Engineering Requirements Management DOORS

DOORS Dynamic (Object-Oriented Requirements System) від компанії IBM [31] — це провідний інструмент керування вимогами, який дозволяє легко фіксувати, відстежувати, аналізувати зміни в інформації та керувати ними. Контроль вимог є ключем до зниження витрат, підвищення ефективності та покращення якості ваших продуктів. Використовуючи сімейство продуктів DOORS, можна оптимізувати передачу вимог, співпрацю і перевірку в усій організації та в ланцюжках постачання.

Серцем сімейства є DOORS, програма, яка працює в системах Windows і Linux. Завдяки власній вбудованій базі даних DOORS надає багатий набір функцій, які допомагають фіксувати вимоги та керувати ними. DOORS дозволяє легко брати участь у процесі управління вимогами та робити свій внесок у нього шляхом:

– використання веббраузера для отримання доступу до бази даних вимог через IBM Engineering Requirements Management DOORS-Web Access (DWA);

– керування змінами вимог або за допомогою простої попередньо визначеної системи пропозицій щодо змін, або за допомогою більш ретельного настроюваного робочого процесу контролю змін за допомогою інтеграції з рішеннями для керування змінами;

– обміну вимогами за допомогою формату ReqIF (RIF) для безпосереднього залучення постачальників і партнерів до процесу розробки;

– зв'язування вимог з елементами дизайну, планами тестування, тестовими випадками та іншими вимогами для легкого та ефективного відстеження;

– співпрацювання з користувачами, маркетингом, постачальниками, системними інженерами та аналітиками безпосередньо через обговорення вимог;

– пов'язування вимоги з тестовими випадками за допомогою Test Tracking Toolkit для середовищ ручного тестування;

– використання специфікації Open Services for Lifecycle Collaboration (OSLC) для керування вимогами, управління змінами та управління якістю для інтеграції із системами та інструментами життєвого циклу програмного забезпечення;

– інтеграції з іншими інструментами, зокрема, IBM Engineering Workflow Management, IBM Engineering Test Management, IBM Engineering Requirements Management DOORS Next, IBM Engineering Systems Design Rhapsody®, Jazz Reporting Service і System Architect, а також багатьма сторонніми інструментами, що забезпечують комплексне рішення для відстеження.

5.2. FRET: Formal Requirements Elicitation Tool

FRET [32] — це фреймворк, що поширюється за ліцензією NASA Open Source Agreement (NOSA), для виявлення, специфікації, формалізації та розуміння вимог. Користувачі вводять системні вимоги спеціальною природною мовою. FRET допомагає зрозуміти та переглянути семантику, використовуючи різні форми для кожної вимоги: опис природної мови, формальну математичну логіку та діаграми. Вимоги можна визначити в ієрархічній формі та експортувати в різні форми для використання інструментами аналізу.

На практиці вимоги зазвичай пишуться природною мовою, яка є неоднозначною і, отже, не піддається формальному аналізу. Оскільки формальні математичні позначення не є інтуїтивно зрозумілими, вимоги до FRET вводяться обмеженою природною мовою під назвою FRETISH [33]. FRET допомагає користувачам писати FRETISH-вимоги, надаючи граматичну інформацію та приклади під час редагування, а також англійською мовою, та схематичні пояснення для прояснення тонких семантичних проблем. Для кожної вимоги FRET автоматично створює формули, які можуть використовуватися інструментами аналізу на всіх етапах життєвого циклу програмного забезпечення. Розширена система перевірки гарантує, що згенеровані формули відповідають семантиці мови FRETISH. Крім того, FRET підтримує відображення вимог високого рівня до сигналів або змінних, які з'являються в програмних моделях або коді. Потім FRET експортує код підтвердження, який можна використовувати безпосередньо різними інструментами аналізу.

FRET містить у собі ідеї з кількох існуючих підходів до розробки вимог. Структура вимог FRETISH включає особливості темпоральної логіки [34], системи шаблонів специфікацій (Specication Pattern System, SPS) [35] і легкого підходу до синтаксису вимог (Easy Approach to Requirements Syntax, EARS) [36].

6. Висновки

Інженерія вимог є одним із визначальних процесів інженерії систем. Інтенсивний розвиток та поширення систем Інтернету речей (Internet of Things, IoT) потребує розроблення інформаційного та методичного базисів для їх розробки, особливо на етапі визначення вимог. Складність, неоднорідність, конвергентність систем IoT обумовлюють відповідні проблеми на всіх етапах їх проєктування, в тому числі і на етапі визначення вимог.

Розглянуті у статті аспекти формалізації та уніфікації процесів інженерії вимог у системах IoT дозволяють сформувати технологічний базис підтримки процесів інженерії систем на етапах визначення та супроводження вимог.

СПИСОК ДЖЕРЕЛ

1. ISO/IEC/IEEE 15288:2023 Systems and software engineering — System life cycle processes. URL: https://www.iso.org/standard/81702.html.
2. Pohl Klaus, Rupp Chris. Requirements engineering fundamentals: a study guide for the certified professional for requirements engineering exam, foundation level, IREB compliant. 2nd edition. Santa Barbara, CA: Rocky Nook Inc., 2015. 164 p.
3. Liu Y., Seet B.-C., Al-Anbuky A. An Ontology-Based Context Model for Wireless Sensor Network (WSN) Management in the Internet of Things. *Journal of Sensor and Actuator Net-works*. 2013. N 2 (4). P. 653–674. DOI: https://doi.org/10.3390/jsan2040653.
4. Коваленко О.Є. Інтелектуалізація граничних обчислень інтернету речей. *Математичні машини і системи*. 2024. № 3–4. С. 50–68. DOI: https://doi.org/10.34121/1028-9763-2024-3-4-50-68.
5. ISO/IEC/IEEE 29148:2018 Systems and software engineering — Life cycle processes — Requirements Engineering. URL: https://www.iso.org/standard/72089.html.
6. Kovalenko O. Systems Convergence for Situational Control and Decision Making in Distributed Environments. *2022 IEEE 16th International Conference on Advanced Trends in Radioelectronics,*

Telecommunications and Computer Engineering (TCSET). 2022. P. 344–347. DOI: https://doi.org/10.1109/TCSET55632.2022.9767006.
7. Коваленко О.Є. Системна інженерія та життєвий цикл систем. *Електронне моделювання*. 2018. Т. 40, № 6. С. 61–82. DOI: https://doi.org/10.15407/emodel.40.06.061.
8. Herber D.R., Narsinghani J.B., Eftekhari-Shahroudi K. Model-Based Structured Requirements in SysML. *2022 IEEE International Systems Conference (SysCon)*. Montreal, QC, Canada, 2022. P. 1–8. DOI: https://doi.org/10.1109/SysCon53536.2022.9773813.
9. Wheaton J.S., Herber D.R. Digital Requirements Engineering with an INCOSE-Derived SysML Meta-model. *Proc. of the 2024 Conference on Systems Engineering Research. CSER 2024. Conference on Systems Engineering Research Series* / A. Salado, R. Valerdi, R. Steiner, L. Head (eds.). Springer, Cham, 2024. DOI: https://doi.org/10.1007/978-3-031-62554-1_2.
10. Gildas Premel-Cabic Discovering System Requirements through SysML. 2021. 15 September. URL: https://cpre.ireb.org/en/knowledge-and-resources/re-mag/discovering-system-requirements-through-sysml.
11. SEBoK System Requirements Definition. URL: https://sebokwiki.org/wiki/System_Requirements_Definition.
12. Wheatcraft L.S., Ryan M.J., Katz T.E. INCOSE Needs and Requirements Manual: Needs, Requirements, Verification, Validation Across the Lifecycle / INCOSE (ed.) Wiley. 2024. 528 p.
13. Requirements Interchange Format™ (ReqIF™). URL: https://www.omg.org/reqif/.
14. Atzori L., Iera A., Morabito G. The Internet of Things. *A Survey, Comput. Netw*. 2010. N 54 (15). P. 2787–2805.
15. Reggio G., Leotta M., Cerioli M., Spalazzese R., Alkhabbas F. What are IoT systems for real? An experts' survey on software engineering aspects. *Internet of Things*. 2020. Vol. 12. DOI: http://dx.doi.org/10.1016/j.iot.2020.100313.
16. Pillai R.R., Lohani R.B. Abnormality Detection and Energy Conservation in Wireless Body Area Networks using Hidden Markov Models: A Review. *2020 International Conference on Communication and Signal Processing (ICCSP).* Chennai, India, 2020. P. 0935–0939. DOI: https://doi.org/10.1109/ICCSP48568.2020.9182131.
17. Alnader I., Lasebae A., Raheem R. Using Hidden Markov Chain for Improving the Dependability of Safety-Critical WSNs. A*dvanced Information Networking and Applications. AINA 2023. Lecture Notes in Networks and Systems* / L. Barolli (eds.). Springer, Cham. 2023. Vol. 661. P. 460–472. DOI: https://doi.org/10.1007/978-3-031-29056-5_40.
18. Bettini C., Brdiczka O., Henricksen K., Indulska J., Nicklas D., Ranganathan A., Riboni D. A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*. 2010. Vol. 6, Issue 2. P. 161–180. DOI: https://doi.org/10.1016/j.pmcj.2009.06.002.
19. Kohan S, Johnstone L., Cetinkaya D. (2023) A survey on the model-centered approaches to conceptual modeling of IoT systems. Front. Comput. Sci. 2023. Vol. 5. P. 1035225. DOI: https://doi.org/10.3389/fcomp.2023.1035225.
20. Santos L., Silva E., Batista T., Leite J., Cavalcante E., Oquendo F. Evaluating a SysML-based Graphical Notation for Modeling Internet of Things System Architectures. *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*. New Orleans, LA, USA, 2020. P. 1–6. DOI: https://doi.org/10.1109/WF-IoT48130.2020.9221497.
21. Costa B., Pires P.F., Delicato F.C. Modeling IoT Applications with SysML4IoT. *2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. Limassol, Cyprus, 2016. P. 157–164. DOI: https://doi.org/10.1109/SEAA.2016.19.
22. Meacham S., Stefanidis A., Gritt L., Phalp K.T. Internet of Things for Education: Facilitating Personalised Education from a University's Perspective. 2018. URL: http://eprints.bournemouth.ac.uk/32703/1/2%20BCS%20Inspire_Design%20an%20IoT%20Solution%20for%20Education_camera%20ready%20final%20.pdf.
23. Requirements engineering tools. URL: https://en.wikipedia.org/wiki/Requirements_engineering_tools.
24. Ozkaya M., Kardas G., Kose M.A. An Analysis of the Features of Requirements Engineering Tools. *Systems.* 2023. Vol. 11. P. 576. DOI: http://doi.org/10.3390/systems11120576.
25. Katis A., Mavridou A., Giannakopoulou D., Pressburger T., Schumann J. Capture, Analyze, Diagnose: Realizability Checking Of Requirements in FRET. *Computer Aided Verification. CAV 2022. Lecture Notes in Computer Science* / S. Shoham, Y. Vizel (eds.). Springer, Cham, 2022. Vol. 13372. P. 490–504. DOI: https://doi.org/10.1007/978-3-031-13188-2_24.

26. Overview of DOORS. URL: https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/doors/9.7.2?topic=engineering-requirements-management-doors-overview.
27. Silva Danyllo Valente da, Bruno Pedraça de Souza, Taisa Guidini Gonçalves and Guilherme Horta Travassos A Requirements Engineering Technology for the IoT Software Systems. *J. Softw. Eng. Res.* 2021. Dev. 9. P. 11:1–11:18.
28. Souza S., Rodrigues E., Meireles M., Lauschner T., Carvalho L., Maldonado J.C., Conte T. Techniques for eliciting IoT requirements: Sensorina Map and Mind IoT. *Journal of Systems and Software*. 2025. Vol. 222. P. 112323. DOI: https://doi.org/10.1016/j.jss.2024.112323.
29. Boutot P., Rehenuma M. Tabassum, Abdul Abedin, Sadaf Mustafiz, Requirements development for IoT systems with UCM4IoT. *Journal of Computer Languages*. 2024. Vol. 78. P. 101251. DOI: https://doi.org/10.1016/j.cola.2023.101251.
30. Aguilar-Calderon, Jose-Alfonso, Carolina Tripp-Barba, Aníbal Zaldívar-Colado and Pedro-Alfonso Aguilar-Calderon. Requirements Engineering for Internet of Things (loT) Software Systems Development: A Systematic Mapping Study. *Applied Sciences*. 2022. Vol. 12, N 15. P. 7582. DOI: https://doi.org/10.3390/app12157582.
31. Engineering Requirements Management DOORS. URL: https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/doors/9.7.2.
32. FRET: Formal Requirements Elicitation Tool. URL: https://github.com/NASA-SW-VnV/fret/tree/master.
33. Giannakopoulou D., Pressburger T., Mavridou A., Schumann J. Generation of Formal Requirements from Structured Natural Language. *Requirements Engineering: Foundation for Software Quality. REFSQ* / N. Madhavji, L. Pasquale, A. Ferrari, S. Gnesi (eds.). Lecture Notes in Computer Science(). Springer, Cham. 2020. Vol. 12045. P. 19–35. DOI: https://doi.org/10.1007/978-3-030-44429-7_2.
34. Goranko V., Rumberg A. Temporal Logic. URL: https://plato.stanford.edu/entries/logic-temporal.
35. Dwyer M.B., Avrunin G.S., Corbett J.C. Patterns in property specications for nite-state veri cation. *Proc. of the 21st International Conference on Software Engineering*. *ICSE '99.* New York: NY, USA, 1999. P. 411–420. DOI: https://doi.org/10.1145/302405.302672.
36. Mavin A. Listen, then use EARS. *IEEE Software*. 2012. Vol. 29 (2). P. 17–18. DOI: https://doi.org/10.1109/MS.2012.36.

Стаття надійшла до редакції 12.01.2025