УДК 004.056:65.012.2

О.В. Мороз
(Національний Авіаційний Університет, Україна)

Щодо питання інтеграції управління ризиками в розробку вимог до програмного забезпечення

Представлено підхід до інтегрування управління ризиками в розробку вимог, щоб проблеми, що виникають внаслідок ризиків, можна було мінімізувати і заповнити порожнечу між доменною моделлю і активами проєкту.

Вступ.

Проєкти з розробки програмного забезпечення містять різні ризики, які можуть негативно вплинути на успіх цих проєктів, і тому важливо їх визначити та усунути або зменшити їх наслідки. Належне забезпечення планування і комунікації та стратегії управління ризиками можуть допомогти зменшити ці ризики, але практика ефективного управління ризиками проєкту передбачає застосування методів аналізу та моделювання багатовимірних явищ та процесів в умовах невизначеності. Щодо сучасного стану у сфері управління ризиками розробки програмного забезпечення з аналізу результатів досліджень можна зробити наступні висновки:

1.Не дивлячись на велику кількість досліджень щодо виявлення факторів ризику, до сих пір немає повного розуміння, як ці фактори впливають на проєкти розробки програмного забезпечення. Крім того, окрім найбільш поширених факторів ризику, в кожному проєкті також можуть виникати специфічні фактори ризику. Щоб визначити такі фактори ризику, «… потрібно зрозуміти контекст домену проєкту, цілі, місцеве середовище та інші пов'язані фактори» [1]. Також необхідно врахувати вплив стратегії управління ризиками на появу специфічних факторів ризику в проєкті, що демонструє переваги управління ризиками і мотивує команди розробників до постійного покращення стратегії управління ризиками і використання програмного забезпечення для управління ризиками.

2.Щоб знизити рівень невдач проєктів програмного забезпечення, потрібно спочатку розвинути краще розуміння причин ризиків проєкту та того, як вони можуть вплинути на продуктивність проєкту. Існують різні підходи до моделювання ризиків і прогнозування їх впливу на проєкт, але точність таких методів і підходів, як правило, значно обмежена і основна причина полягає в тому, що їм не вистачає [2]: 1) перевірених інструментів для вимірювання ризику проєкту програмного забезпечення, які враховують різні аспекти ризику, що вважаються важливими для керівників проєктів; 2) теорії для пояснення зв'язку між моделями вимірювання ризику та ефективністю проєкту.

3.Різні типи ризиків розробки програмного забезпечення пов'язані між собою тим, що вони можуть впливати один на одного та на проєкт у цілому. Наприклад, технічні ризики можуть вплинути на розклад і бюджет проєкту, оскільки затримки, спричинені технічними проблемами, можуть призвести до

збільшення вартості проєкту та термінів, або ризики управління проєктом можуть вплинути на розподіл ресурсів і розклад проєкту.

4.Щоб мінімізувати негативний вплив ризиків на успіх проєкту, важливо врахувати сукупний вплив усіх типів ризиків на виконання проєкту. Для цього стратегія управління ризиками повинна дотримуватись певних рекомендацій, але успіх проєктів не залежить лише від систематичного та ретельного застосування стратегій і рекомендацій (моделей, методів, стандартів тощо), заснованих на практичному досвіді. Взагалі, проблема систематичного дослідження впливу загальної системи управління ризиками на розробку програмного забезпечення ще очікує свого вирішення, відтак «.. рівень успіху в розробці проєктів програмного забезпечення все ще низький» [3].

5.Одна з основних проблем у сфері управління ризиками розробки програмного забезпечення полягає в тому, що підходи до управління ризиками зазвичай зосереджуються на обмежених цілях, пов'язаних із розкладом, вартістю та якістю [4] в той час, як вони повинні оцінювати інтегрований вплив ризиків. Це означає, що, окрім вище означених, мають бути враховані й інші цілі, такі як балансування життєвого циклу продукту і технічних навичок в команді, підтримка критичних бізнес-процесів тощо.

Архітектура вимог до проєкту розробки програмного забезпечення.

Сьогодні не визиває сумнівів той очевидний факт, що впровадження заходів з управління ризиками повинно розпочинатися якомога раніше, тобто на етапі розробки вимог. Разом із ретельно розробленою стратегією управління ризиками в організації такий підхід вочевидь сприятиме зменшенню ризиків і уникненню непередбачених проблем під час розробки. «Але ризики аналізуються головним чином з технічної точки зору і відсутні механізми щодо інтеграції управління ризиками в розробку вимог. Отже, існує велика потреба в розумінні інтеграційних процесів в управлінні ризиками за конкретними критеріями відповідно до конкретних цілей.» [4] Іншими словами, утворилася певна порожнеча між процесом управління ризиками і розробкою вимог. Цей висновок знаходить своє підтвердження в аналізі результатів досліджень за останні роки, з яких можна помітити, що увага з боку дослідників, яка стосувалася здебільшого інтегрованого процесу управління ризиками, тепер зосереджена на окремих етапах цього процесу [3]. В такий ситуації створення реальних механізмів інтегрування управління ризиками в розробку вимог могло б відновити status quo?

На наш погляд, існує єдиний шлях до вирішення означеної проблеми – інтеграція моделей ризиків в архітектуру вимог. Як і архітектура системи, що розробляється, архітектура вимог є багаторівневою і має ієрархічну структуру, яка коріниться на моделях цілей [4]. За рівнем моделей цілей слідує рівень моделей ризику. Тісно взаємопов'язані моделі людей і систем утворюють наступний рівень і залежать від моделей ризику, хоча людський фактор є домінуючим з точки зору управління ризиками проєкту. Моделі даних займають найнижчий рівень в цій ієрархії і визначаються переважно моделями

цілей, хоча і залежать також від моделей людей і систем, які впливають на якість даних.

Щодо моделей ризиків, згідно зі стандартом ISO 31000:2018 [5] процес управління ризиками складається з шести фаз, в контексті кожної з яких розробляється певний набір відповідних моделей управління ризиками:

Таким чином, проблема моделювання ризиків має розглядатися в контексті загальної архітектури вимог до проєкту. Таблиця 1 містить короткий опис класів моделей цієї архітектури і для кожного класу – перелік складових моделей і обмежувальну модель, від якої залежать усі інші моделі класу та яка обмежує відповідний домен.

Таблиця 1.

Класи моделей в архітектурі вимог і їх складові моделі

Клас	Опис	Моделі	Обмежувальна модель
Цілі	Описує бізнес-цінність системи та допомагає визначити пріоритетність функцій і вимог на основі їхньої цінності.	• Бізнес-цілі • Цільовий ланцюг • Ключовий показник • Дерево функцій • Матриця відображення вимог	Бізнес-цілі
Ризики	Описує стратегію управління ризиками та допомагає врахувати інтегрований вплив ризиків на успіх проєкту.	• План управління ризиками • Реєстр ризиків • Матриця оцінки ризиків • План дій у надзвичайних ситуаціях • Плани пом’якшення наслідків • Діаграма освоєного обсягу функціоналу • Діаграма витрат на пом'якшення • Звіти про ризики	Реєстр ризиків
Люди	Описує, хто використовує систему, а також їхні бізнес-процеси та цілі.	• Організаційна діаграма • Потік процесу • Випадок використання • Матриця ролей і дозволів	Організаційна діаграма
Системи	Описує, які системи	• Карта екосистеми	Карта

	існують, як виглядає інтерфейс користувача, як системи взаємодіють і як вони поводяться.	• Системний потік • Потік інтерфейсу користувача • Відображення-Дія-Відповідь • Таблиця рішень • Дерево рішень • Таблиця системного інтерфейсу	екосистеми
Дані	Описує зв'язки між об'єктами бізнес-даних з точки зору кінцевого користувача, життєвий цикл даних і те, як ці дані використовуються у звітах для прийняття рішень.	• Діаграма бізнес-даних • Діаграма потоку даних • Словник даних • Таблиця станів • Діаграма станів • Таблиця звітів	Діаграма бізнес-даних

Моделі в архітектурі вимог можуть впливати одна на одну різними способами. Наприклад, одні моделі базуються на інших, якщо між їхніми елементами існують причинно-наслідкові зв'язки, або моделі можуть надавати інформацію для інших моделей, якщо існує певна послідовність дій у процесі процесу. Крім того, у проєкті може бути кілька рівнів зв'язків (каскадних зв'язків), де одна модель пов'язана з іншою моделлю, яка, у свою чергу, пов'язана з іншою моделлю. Все це призводить до необхідності створення онтології архітектури вимог, яка б описувала семантику зв'язків хоча б між основними моделями вимог, тобто тими, які часто зустрічаються в проєктах. Але дуже важко, якщо взагалі можливо, охарактеризувати ці відношення за допомогою формалізму описової (дескриптивної) логіки через їх різноманітність і контекстуальну залежність; можна лише говорити про вплив одних моделей на інші, що і було зроблено в роботі [6]. З цієї причини в роботі [4] запропоновано для опису семантичних відношень між моделями вимог використовувати «онтологічні предикати», які засновані на кантівській трансцендентальній логіці. Ці предикати є онтологічними відношеннями, що витікають з відомих категорій розуму Канта, і, як відзначав сам Кант, їх особливість полягає в тому, що вони не показують властивостей об'єктів, яких вони стосуються, але вказують на те, як ми повинні, керуючись ними, розкривати властивості і зв'язки об'єктів досвіду взагалі, що власне і повинно бути основною задачею онтологічної моделі.

Насамкінець, слід відзначити, що технологія, на основі якої можна ефективно моделювати поведінку об'єктів у багатовимірному середовищі в умовах невизначеності, повинна забезпечувати досягнення кінцевої мети, яка полягає в тому, щоб проблеми, що виникають внаслідок ризиків, можна було мінімізувати і заповнити порожнечу між доменною моделлю і активами проєкту.

Висновок

У цій роботі коротко описано сучасний стан проблеми управління ризиками розробки програмного забезпечення. Представлено огляд дослідницьких підходів в інженерії ризиків, які цікавлять наукову спільноту, і запропоновано підхід до інтегрування управління ризиками в керовану моделями інженерію вимог до програмного проєкту шляхом побудові інтегрованої архітектури вимог до проєкту, що містить набір взаємозв'язаних моделей включно із моделями ризиків. Побудова онтології зв'язків між моделями в цій архітектурі, включно із моделями ризиків, на основі трансцендентальної логіки може забезпечити значно ширший спектр можливостей для опису семантичних відношень між цими моделями, а відтак – і більш точну оцінку впливу їх на процес розробки програмного забезпечення.

Список літератури

1. Shareeful Islam. Software Development Risk Management Model – a goal-driven approach. Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigten Dissertation. 25.03.2011 [Електронний ресурс] – Режим доступу: https://d-nb.info/1011414708/34 – 09.04.2023
2. Wallace L., Keil M., Rai A. How Software Project Risk Affects Project Performance: An Investigation of the Dimensions of Risk and an Exploratory Model. – Decision Sciences. – Volume35, Issue2, 2004. – P. 289-321.
3. Massoa, J., Francisco J. Pino, et al. Risk management in the software life cycle: A systematic literature review. – Computer Standards & Interfaces. – Volume 71, Article 103431, 2020.
4. Moroz O.V., Pysarchuk O.O., Konrad T.I. Transcendental Logic-based Formalism for Semantic Representation of Software Project Requirements Architecture. – Computer and Information Science. – Vol. 15, No. 2, 2022. – P. 15 – 37.
5. Iso 31000 : 2018 Risk Management – Guidelines. 2018. Vernier Geneva: International Organization for Standardization. [Електронний ресурс] – Режим доступу: https://www.iso.org/obp/ui/#iso:std:iso:31000:ed-2:v1:en – 09/042023
6. Beatty J., Chen A. Visual models for software requirements. Redmond, Washington: Microsoft Press, 2015. – 480 p.