DOI https://doi.org/10.30525/978-9934-588-47-1.7

МОДЕЛІ І МЕТОДИ ПРЕДСТАВЛЕННЯ АРХІТЕКТУРНИХ РІШЕНЬ КРИТИЧНОЇ ІТ-ІНФРАСТРУКТУРИ

Дорогий Я. Ю.

ВСТУП

У науковій літературі зустрічаються різні формальні і неформальні описи розподілених систем (РС)[1,2,3]. Характерною властивістю РС є наявність взаємодіючих компонентів системи. У цій роботі основна увага приділяється моделюванню та верифікації моделей особливого класу розподілених систем – критичним ІТ-інфраструктурам, тому далі під компонентами РС розуміються взаємодіючі процеси, критичні та некритичні. Процеси виконуються на одному або декількох обчислювальних пристроях. У загальному випадку обчислювальні пристрої можуть бути фізично віддалені один від одного. У цій роботі затримка, що виникає під час передачі даних, не враховується, і взаємодія процесів вважається миттєвою.

Далі під розподіленою системою будемо мати на увазі критичну ІТ-інфраструктуру.

Для аналізу поведінки розподіленої системи часто як модель РС використовуються транзиційні системи (Transition System, ТС)[4]. Ключовими складниками системи переходів є поняття стану і переходу. Стан ТС є абстракцією стану розподіленої системи. У стані ТС відображені значення структур цих процесів РС, необхідних для аналізу її поведінки. Зі стану моделі може здійснюватися перехід в одне або кілька наступних станів. Перехід відповідає локальній або комунікаційній події РС, що моделюється.

---

[1] Tel G. Introduction to Distributed Algorithms. Cambridge: Cambridge University Press, 2000. P. 610.

[2] Tanenbaum A., van Steen M. Distributed Systems: Principles and Paradigms. Upper Saddle River: Create Space Independent Publishing Platform, 2016. P. 704.

[3] van Steen M., Tanenbaum A.S. A brief introduction to distributed systems. *Computing*. 2016. № 98 (10). С.967-1009.

[4] Теленик С.Ф., інш. Методы исследования свойств высокопроизводительных инфраструктур. Обзор. *Фундаментальные и прикладные проблемы информатики и информационных технологий*. 2015. № 1. С. 3-13.

ТС можуть супроводжуватися розміткою станів та/або переходів. У цьому разі говорять про розмічені транзиційні системи (Labelled Transition System, РТС)[5]. У роботі використовуються РТС з розміткою станів значеннями булевих змінних і розміткою переходів назвами комунікаційних дій. Під час побудови моделі будуються РТС процесів РС. Модель всієї РС утворюється в результаті паралельної композиції РТС процесів.

1. Розмічені транзиційні системи

Нехай задана множина можливих міток дій ***A.*** Дії відповідають класам еквівалентності подій ***E.*** Прикладами дій можуть бути: відсилання / отримання повідомлень процесом або компонентом, зміна стану процесу або компоненту тощо. Далі мається на увазі, що всі дії РТС належать множині ***A***. В множині також знаходиться спеціальна неспостережна дія τ.

Означення 1.1. РТС називається шістка $(S, S^0, A, R, \Sigma, L)$, в якій S – скінченна множина станів, S^0 – підмножина S початкових станів системи, $A \in \boldsymbol{A}$ – скінченна множина міток дій, R – відношення переходів на множині $S \times A \times S$, Σ – множина змінних системи, L – функція розмітки станів $L : S \to 2^{\Sigma}$.

Далі будемо використовувати запис $s \xrightarrow{a} t$, що означає $(s, a, t) \in R$ для $s, t \in S, a \in A$. За допомогою відношення R задаються можливі варіанти дискретних переходів з одного стану до іншого. Переходи супроводжуються мітками дій.

Для кожного стану $s \in S$ за допомогою функції L задаються значення змінних з множини Σ в стані s.

За допомогою символу $\mathscr{M}$ будемо позначати множину всіх РТС.

Означення 1.2. Скінченним шляхом π в РТС $(S, S^0, A, R, \Sigma, L)$ називається така скінченна послідовність станів та дій $s_1, a_1, \ldots, a_{n-1}, s_n$, які чергуються, що $s_i \in S, a_j \in A$ для всіх $1 \le i \le n, 1 \le j < n$ та $(s_i, a_i, s_{i+1}) \in R$ для всіх $1 \le i < n$. Нескінченним шляхом називається така нескінченна послідовність станів та дій $s_1, a_1, \ldots, a_{n-1}, s_n, \ldots$, які чергуються, що $s_i \in S, a_j \in A$ для $i \le 1, j \ge 1$ та $(s_i, a_i, s_{i+1}) \in R$ для всіх $i \ge 1$.

[5] Крывый С.Л., Максимец А.Н. Верификация программ: состояние, проблемы, результаты. *Кибернетика и системный анализ*. 2013. Т. 49. С. 3–14.

Далі скінченний шлях $\pi = s_1, a_1, ..., a_{n-1}, s_n$ будемо позначати як $s_1 \overset{a_1,...,a_{n-1}}{\rightarrow} s_n$, а нескінченний шлях $\pi = s_1, a_1, ..., a_{n-1}, s_n, ...$, як $s_1 \overset{a_1,...,a_{k-1}}{\rightarrow} s_k \overset{a_k}{\rightarrow} ...$ Для скінченного або нескінченного шляху π запис $\mathrm{suf}(\pi, k)$ означає суфікс шляху π, який починається з стану s_k.

Означення 1.3. Для шляхів на множині станів S визначимо таку функцію станів, які зустрічаються на шляху, $\mathrm{PrS} : S^* \cup S^\omega \to 2^S$. Стан $s \in \mathrm{PrS}(\pi)$ для шляху $\pi = s_1 \overset{a_1,...,a_{k-1}}{\rightarrow} s_k \overset{a_k}{\rightarrow} ...$ тоді і тільки тоді, коли знайдеться таке $i \geq 1$, що $s = s_i$.

Означення 1.4. Нехай задана РТС $M = (S, S^0, A, R, \sum, L)$ та шляхи $\alpha' = s_1 \overset{a_1,...,a_{m-1}}{\rightarrow} s_m$, $\alpha'' = t_1 \overset{b_1,...,b_{n-1}}{\rightarrow} t_n$ і при цьому $s_m = t_1$. Тоді конкатенацією шляхів α' і α'' назвемо шлях $\alpha = s_1 \overset{a_1,...,a_{m-1}}{\rightarrow} t_1 \overset{b_1,...,b_{n-1}}{\rightarrow} t_n$. Далі будемо позначати конкатенацію шляхів α' і α'' як $\alpha = \alpha' \circ \alpha''$.

Означення 1.5. Для РТС $M = (S, S^0, A, R, \sum, L)$ стан $t \in S$ називається досяжним із стану $s \in S$, якщо знайдеться деякий скінченний шлях $\alpha = s \overset{a_1}{\rightarrow} s_1 ... s_k \overset{b}{\rightarrow} t$. Стан t називається досяжним, якщо він є досяжним з деякого початкового стану РТС.

Означення 1.6. Нехай задана РТС $M = (S, S^0, A, R, \sum, L)$ та функція перейменування дій $\mathrm{ren} : A \to A'$. Тоді результатом застосування функції ren до РТС M (позначається як $\overset{\mathrm{ren}}{M}$) є така РТС $M_l = (S_l, S_l^0, A_l, R_l, \sum_l, L_l)$, що:

– вірні рівності $S_l = S, S_l^0 = S^0, A_l = A, R_l = R, \sum_l = \sum, L_l = L)$,

– перехід $(s, a, t) \in R$ виконується тоді і тільки тоді, коли $(s, \mathrm{ren}(a), t) \in R_r$.

В подальшому будемо використовувати спеціальну функцію ren^i дописування індексу i діям РТС, яка відображає кожну дію a на дію a_i.

Означення 1.7. Нехай задана РТС $M = (S, S^0, A, R, \sum, L)$ та функція початку переходу $\mathrm{bg} : R \to S$. Тоді результатом застосування функції bg до $r_i \in R$ (позначається як $\mathrm{bg}(r_i)$) є стан початку переходу r_i, тобто $\mathrm{bg}(r_i) \in s_i, s_i \in S$.

Означення 1.8. Нехай задана РТС $M = (S, S^0, A, R, \sum, L)$ та функція кінця переходу $\text{end} : R \to S$. Тоді результатом застосування функції end до $r_i \in R$ (позначається як $\text{end}(r_i)$) є стан кінця переходу r_i, тобто $\text{end}(r_i) \in s_i, s_i \in S$.

1.1. Асинхронна паралельна композиція РТС

Розглянемо деревоподібний хвилевий алгоритм як приклад РТС. В алгоритму приймають участь процеси трьох типів: кореневий процес-ініціатор (прототип Root); процес, який відповідає листовій вершині комунікаційного дерева (прототип Inner); процес, який відповідає внутрішній вершині комунікаційного дерева (прототип Leaf).

За алгоритмом процес-ініціатор посилає повідомлення своїм нащадкам. Процеси типу Inner отримують повідомлення від батьківського процесу та надсилають його далі своїм нащадкам. Процеси типу Leaf отримують повідомлення, запускають задачу прийняття рішення та надсилають у відповідь підтвердження. Процеси типу Inner, отримавши підтвердження від всіх нащадків, також запускають задачу прийняття рішення, після чого передають підтвердження батьківському процесу.

На рис. 1 представлені діаграми РТС прототипів процесів алгоритму для бінарного дерева. Стани РТС зображені у вигляді еліпсів. Початкові стани: s_0, t_0, u_0. Переходи позначаються діями *rcv_left, rcv_right, snd_left, snd_right*. Дія з префіксом *rcv* відповідає отриманню повідомлення, а дія з префіксом *snd* відповідає відправці повідомлення. Переходи, для яких не вказана мітка, позначені дією τ.

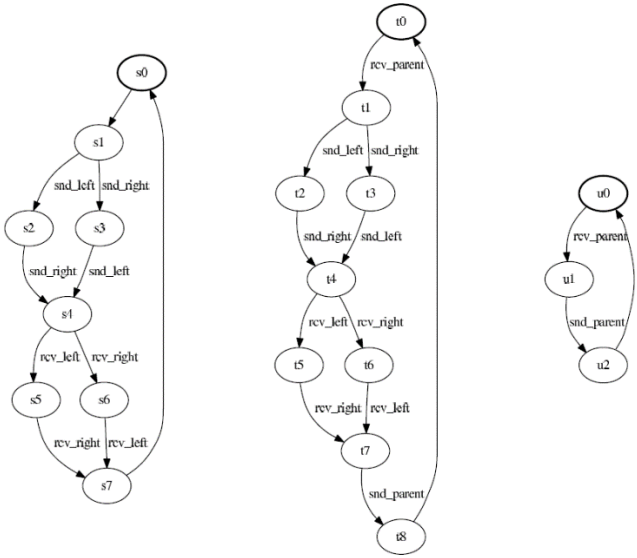

Рис. 1. Діаграми PTC процесів типу Root, Inner, Leaf

Для опису правил взаємодії на PTC визначається операція паралельної композиції. За допомогою цієї операції будується нова PTC, яка моделює асинхронне виконання та синхронний обмін повідомленнями двох вихідних PTC. Маючи PTC всіх процесів ***E,*** можна побудувати PTC всієї ***E***, виконуючи ітеративно операцію паралельної композиції.

Означення 1.9. Нехай задані PTC $M_1 = (S_1, S_1^0, A_1, R_1, \Sigma_1, L_1)$ та $M_2 = (S_2, S_2^0, A_2, R_2, \Sigma_2, L_2)$ такі, що

$$S_1 \cap S_2 = \varnothing, A_1 \cap A_2 = \varnothing, \Sigma_1 \cap \Sigma_2 = \varnothing.$$

Нехай також задана синхронізаційна пара

$$\Lambda = (\Delta, \bar{\ }), \Delta \subseteq A_1, \bar{\ } : \Delta \to A_2.$$

Тоді модель $M = (S, S^0, A, R, \Sigma, L)$ називається паралельною композицією PTC M_1 і M_2, якщо:

– вірні співвідношення

$$S = S_1 \times S_2, S^0 = S_1^0 \times S_2^0, A = A_1 \cup A_2 \setminus (\Delta \cup \bar{\Delta}), \ \Sigma = \Sigma_1 \cup \Sigma_2,$$

– вірно $L : S \to 2^{\Sigma}, \ L((s_1, s_2)) = L(s_1) \cup L(s_2),$

– відношення $R \subseteq (S_1 \times S_2) \times A \times (S_1 \times S_2)$ задається наступним чином: $((s,u),a,(t,v)) \in R$ тоді і тільки тоді, коли виконується одна з трьох умов:

а) виконується перехід першої моделі: $(s,a,t) \in R_1, u = v$;

б) виконується перехід другої моделі: $(u,a,v) \in R_2, s = t$;

в) виконується синхронний обмін повідомленнями: $(s,b,t) \in R_1, (u,b,v) \in R_2, a = \tau$.

Розглянемо операцію композиції на прикладі хвильового алгоритму. Нехай M_1 і M_2 – РТС двох процесів, отриманих з РТС прототипів Inner та Leaf відповідно шляхом перейменування за допомогою функцій ren^1 і ren^2. Визначимо синхронізатор $\Lambda = (\Delta, \bar{\ })$ наступним чином: $\Delta = \{snd\_left_1, rcv\_left_1\}$, $\overline{snd\_left_1} = rcv\_parent_2, \overline{rcv\_left_1} = snd\_parent_2$. Тоді прикладами переходів РТС $M = M_1 \|_\Lambda M_2$ є наступні переходи:

– внутрішній перехід першого процесу $(s_0,t_0) \xrightarrow{\tau} (s_1,t_0)$,

– внутрішній перехід другого процесу $(s_7,t_8) \xrightarrow{\tau} (s_7,t_0)$,

– комунікаційний перехід обох процесів $(s_1,t_0) \xrightarrow{\tau} (s_2,t_1)$.

$M = P_1 \| ... \| P_n$, де $P_1, ..., P_n$ – РТС процесів РС.

1.2. Деякі приклади використання РТС

Сучасні вимоги до програмного і технічного забезпечення сучасних інформаційних систем та критичних IT-інфраструктур спонукають до розробки нових методів їх проектування, обґрунтування та коректної реалізації. Пропонований метод проектування ілюструється прикладами синтезу сервісів критичних IT-інфраструктур типу **PaaS** (*Platform as a Service*), **IaaS** (*Infrastructure as a Service*), **SaaS** (*Software as Service*).

IaaS зазвичай надає уніфіковані апаратні і програмні ресурси, але в деяких випадках і на інфраструктурному рівні для установки ПО з оплатою pay as you go (за мірою використання). Замовлена інфраструктура може динамічно масштабуватися. На базі такого підходу побудовані Amazon EC2 Service і Amazon S3.

PaaS надає більш високий рівень сервісу, що дозволяє розробляти, тестувати і впроваджувати власні застосунки.

Вбудована масштабованість накладає обмеження на тип розробляються. Яскравим прикладом реалізації такого підходу є сервіс Google App Engine.

SaaS пропонує готове спеціалізоване ПО, що веде до спрощення використання додатків і до зменшення витрат на розробку. Одним із чудових прикладів реалізації за таким підходом є Salesforce та її онлайнова система управління відносинами з клієнтами.

Оскільки в критичних IT-інфраструктурах зосереджені потужні і досить дорогі ресурси, виникає проблема їх ефективного використання. Це стає можливим за рахунок правильного проектування та реалізації критичної IT-інфраструктури, а в подальшому використанням оптимальних методів та алгоритмів розподілу, управління і диспетчерування ресурсів і навантаження на основі комплексу відповідних математичних моделей і методів.

Приклад 1. Синтез критичного процесу для критичної IT-інфраструктури[6,7]. Роботу, пов'язану з запуском критичного процесу, можна представити у вигляді взаємодії скінченної множини підсистем. У термінах транзиційних систем застосування можна представити у вигляді синхронного добутку наступних ТС:

1. а) *ТС 1..N* – інкапсулює дії, що виконуються винятково для пошуку та виділення певних потрібних ресурсів. Залежно від складності та розподіленості підсистема виділення конкретного ресурсу може бути як одиничною системою зі строго послідовними переходами, так і мати ієрархічну структуру. Основні дії, що належать до зони відповідальності ТС 1..N: а) пошук ресурсу; б) виділення ресурсу; в) використання ресурсу; г) повернення ресурсу в пул ресурсів. ТС у формальному вигляді можна представити таким чином:

$$A = (S, S^0, A, R, \sum, L) = (\{0,1,2,3\}, 0, \{ready, \text{search\_res\_A}, res\_A\_alloc, released\}, R, \sum, L\}$$

[6] Дорогий Я.Ю. Проектування критичних IT-інфраструктур з використанням розмічених транзиційних систем. *Материалы V заочной научной конференции «Фундаментальные и прикладные исследования в современной науке»*. Харьков. 2017. Р. 48.

[7] Дорогий Я.Ю. Моделювання критичних IT-інфраструктур з використанням розмічених транзиційних систем. *Матеріали VI Міжнародної науково-практичної конференції (I Міжнародний симпозіум «Практичне застосування нелінійних динамічних систем в інфокомунікаціях», PREDT-2017)*. 9-11 листопада. Чернівці. 2017.

Структура ТС представлена на рис. 2.

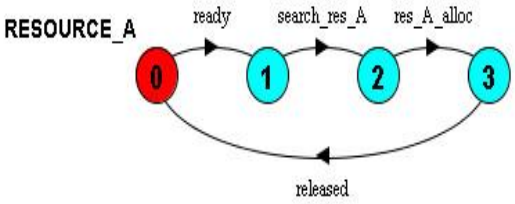

Рис. 2. Структура ТС виділення ресурсу

Множина станів така: 0 – початковий стан, 1 – стан готовності, 2 – ресурс знайдено, 3 – ресурс виділено, ресурс звільнено.

2. е) *ТС N+1* – інкапсулює дії, що виконуються винятково для запуску критичного процесу.

ТС у формальному вигляді можна представити таким чином:

$$A = (S, S^0, A, R, \sum, L) = (\{0,1,2,3,4,5\}, 0, \{ready, \text{res\_A\_alloc},$$
$$res\_\text{B}\_alloc, \text{proc\_start}, \text{proc\_killed}, released\}, R, \sum, L\}$$

Структура ТС для випадку використання двох видів ресурсу представлена на рис. 3.

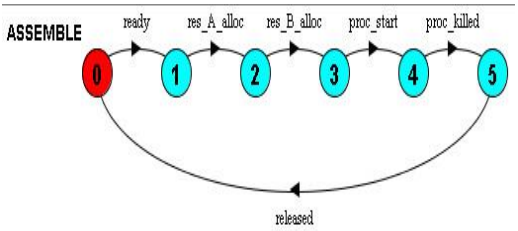

Рис. 3. Структура ТС запуску критичного процесу

Множина станів така: 0 – початковий стан, 1 – стан готовності, 2 – ресурс А виділено, 3 – ресурс В виділено, 4 – критичний процес запущено, 5 – критичний процес знищено, ресурси звільнено.

Результатом мінімізованої композиції ТС для випадку використання 2-х ресурсів буде наступна ТС (див. рис. 4):

$$A = (S, S^0, A, R, \sum, L) = (\{0..9\}, 0, A, R, \sum, L\},$$
$$де\ A = \{ready, search\_res\_A, search\_res\_B,$$
$$res\_A\_alloc, res\_B\_alloc, proc\_killed, proc\_start, released\}$$

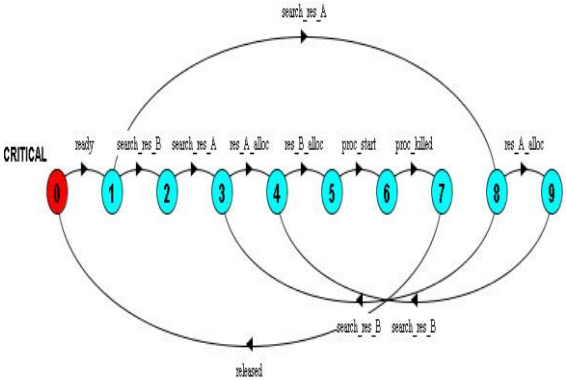

Рис. 4. Структура композиції ТС

Приклад 2. Взаємодія двох SAAS-систем (дуже спрощена модель).

Представимо нашу взаємодію систем у термінах ТС:

1. а) *ТС 1..2* – інкапсулює дії, що виконуються винятково для певних потрібних ресурсів. Основні дії, що належать до зони відповідальності ТС 1..2: а) отримання даних (ресурсу) з системи; б) запис даних (повернення ресурсу) в систему.

Структура ТС представлена на рис. 5.

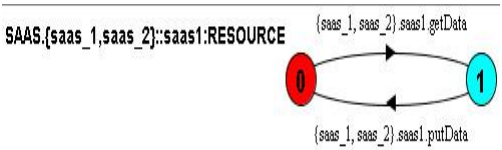

Рис. 5. Структура ТС 1..2

2. а) *ТС 3..4* – інкапсулює дії, що виконуються кожною системою SaaS при взаємодії з іншою. Основні дії, що належать до зони відповідальності ТС 3..4: а) отримання даних (ресурсу) з системи; б) запис даних (повернення ресурсу) в систему; в) обмін даними між системами; г) очікування (див. рис. 6).

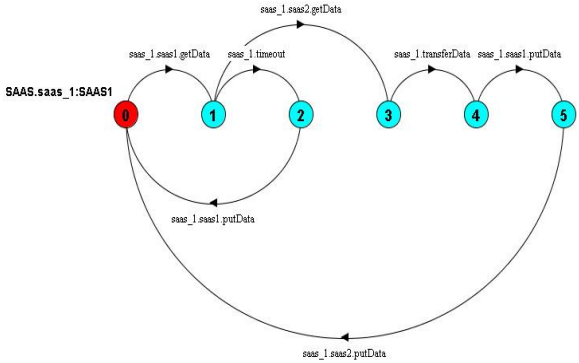

Рис. 6. ТС система SaaS

Мінімізована композиція ТС 1..4 дає можливість побачити взаємодію двох систем SaaS, яка включає взаємний обмін інформацією та її обробку (див. рис. 7).

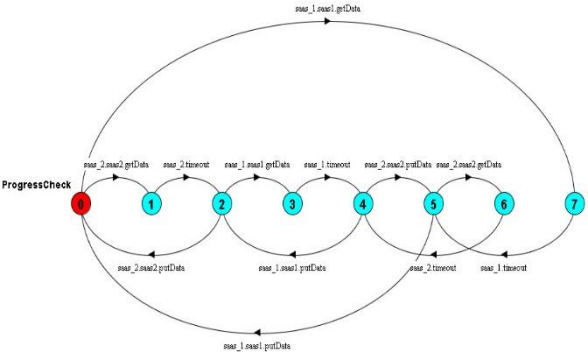

Рис. 7. Композиція ТС взаємодії систем SaaS

2. Представлення архітектури критичної IT-інфраструктури на базі розмічених транзиційних систем

У цій статті розглядається метод представлення архітектури критичної IT-інфраструктури, який отримав назву методу представлення та *обґрунтування дизайну критичної архітектури (Rationale and Evaluation of Critical Architecture Design, RECAD).*

RECAD має на меті допомогти архітекторам створювати та документувати архітектурний дизайн з акцентом на архітектурні рішення та обґрунтування проекту. RECAD охоплює три типи знань архітектури: проектні питання, проектні рішення та результати проектування. Ці об'єкти знань представлені стандартними об'єктами уніфікованого моделювання (UML). Проблема проектування – це матеріали, які впливають на прийняття дизайнерських / проектних рішень. Ця сутність інкапсулює такі поняття, як функціональні вимоги (наприклад, сценарії та діаграма співпраці), нефункціональні вимоги (наприклад, всі атрибути якості) та контексти проекту. Вона також фіксує інформацію про проектні рішення та обґрунтування проекту. Результати проектування включають в себе отримані рішення. Прикладами є класи, компоненти, інтерфейс та спосіб використання. Будь-який індивідуальний тип об'єкту архітектурного знання фіксується шляхом застосування заздалегідь визначеного тегового шаблону стереотипу. Концептуальна модель підходу представлена на рис. 8.

Фактично RECAD представляється ациклічним графом, який пов'язує елементи архітектури *CAE* з елементами обґрунтування архітектури *AR* за допомогою направленого відношення *ARtr*. *CAE* є критичним або не критичним архітектурним елементом, який приймає участь у *Рішенні* як вхідне значення (*Мотиваційна причина*) або як вихідне значення (*Результат дизайну*). AR енкапсулює результат *Обґрунтування архітектури* для *Рішення*. Оскільки *AR* має відношення типу 1:1 з *Рішенням,* для даного підходу він представляє точку прийняття рішення щодо обґрунтування архітектури. Відношення між *CAE* та *AR* представляється за допомогою направленої асоціації ARtr, яка є причинно-наслідковим зв'язком.

Означення 1.10. Модель RECAD – це РТС $RECAD = (CAE, CAE^0, AR, A, R, \Sigma, L)$, де *CAE* – множина вузлів, що представляють критичні та некритичні елементи архітектури, CAE^0 – підмножина CAE початкових станів елементів архітектури, *AR* – множина вузлів, які представляють обґрунтування архітектури, A – скінченна множина міток дій, Σ – множина змінних системи, L – функція розмітки елементів $L: CAE \to 2^{\Sigma}$, R – множина $R \subseteq (CAE \times AR) \cup (AR \times CAE)$ направлених зв'язків між вузлами, для якої виконуються такі умови:

1. Всі вузли AR повинні бути асоційовані з хоча б одною причиною та одним наслідком, тобто для $\forall r \in AR$ існує така причина $e \in CAE$, що $(e,r) \in R$ і існує такий наслідок $e' \in CAE$, що $(r,e') \in R$;

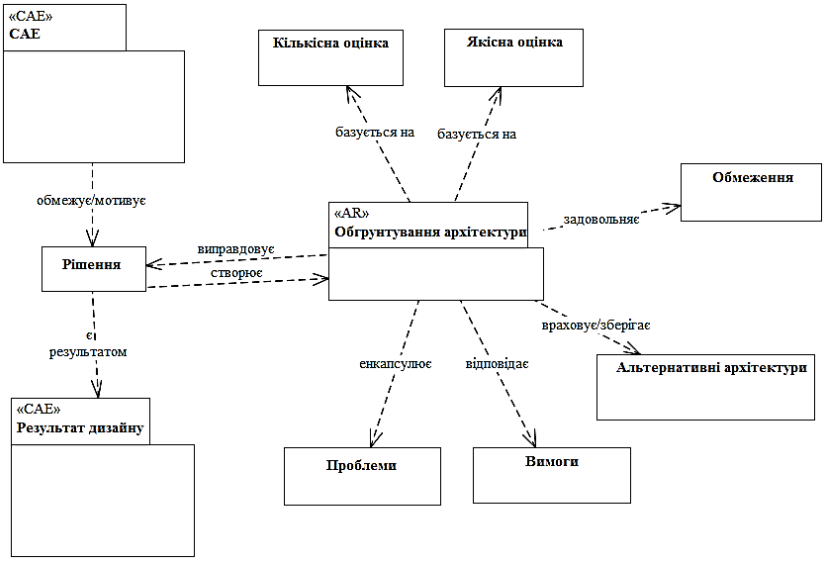

Рис. 8. Концептуальна модель RECAD

2. Не існує підмножини з множини *R*, яка утворює направлений цикл.

Основною формою модельної конструкції є шлях виду $\{\mathrm{CAE}_1, CAE_2, ...\} \to \mathrm{AR}_1 \to \{\mathrm{CAE}_a, CAE_b, ...\}$, де CAE_1, CAE_2 – входи, або причини рішення AR_1, а CAE_a, CAE_b – виходи, або наслідки цього рішення. На рис. 9 представлена UML діаграма, яка демонструє означену модельну конструкцію. Кратність відношень діаграми показує, що мотиваційні та результуючі CAE є непустими множинами, з'єднаними через єдиний елемент AR. Обмеження унікальності на діаграмі визначає, що кожний екземпляр CAE не може бути відображеним більше, ніж один раз.

Рис. 9. Причинно-наслідковий зв'язок між *CAE* та *AR*

Направлені зв'язки ARtr представляють причинно-наслідкові відношення. *CAE* призводить до *AR* через мотивацію або обмеження *Рішення,* AR своєю чергою генерує *CAE* типу *Результат дизайну*, маючи *Обґрунтування архітектури*. CAE може бути одночасно вхідним і вихідним, якщо використовується для двох *Рішень.* У якості вхідного він може представляти собою артефакт наступних типів: вимога, прецедент, клас, імплементація. Як вихідний – новий або переглянутий елемент дизайну.

В RECAD архітектурні елементи *CAE* є артефактами, які формують частини дизайну архітектури. Вони включають бізнес-потреби, які потрібно задовольнити, технічні та організаційні обмеження, що накладаються на проект архітектури, припущення, які потрібно перевірити, та об'єкти дизайну, які є результатом архітектурного проектування.

Елементи архітектури можуть також бути класифіковані за точками зору на архітектуру. Причина такої класифікації – мати можливість фокусуватись на різних аспектах проектного рішення. В цьому дослідженні, згідно з результатами аналізу ФОАП, використовуються точки зору на архітектуру підходу TOGAF. Для класифікації використано такі точки зору на архітектуру: бізнес-логіка, рівень даних, застосування, технології. А також додану нову точку зору на архітектуру, орієнтовану на критичні елементи.

На рис. 10 представлена UML-діаграма класифікації архітектурних елементів відповідно до точок зору на архітектуру. Бізнес-орієнтована точка зору на архітектуру включає архітектурні елементи, такі як функціональні та нефункціональні вимоги, бізнес-

середовище, середовище інформаційної системи, технологічне середовище. Це основні драйвери проектування архітектури.

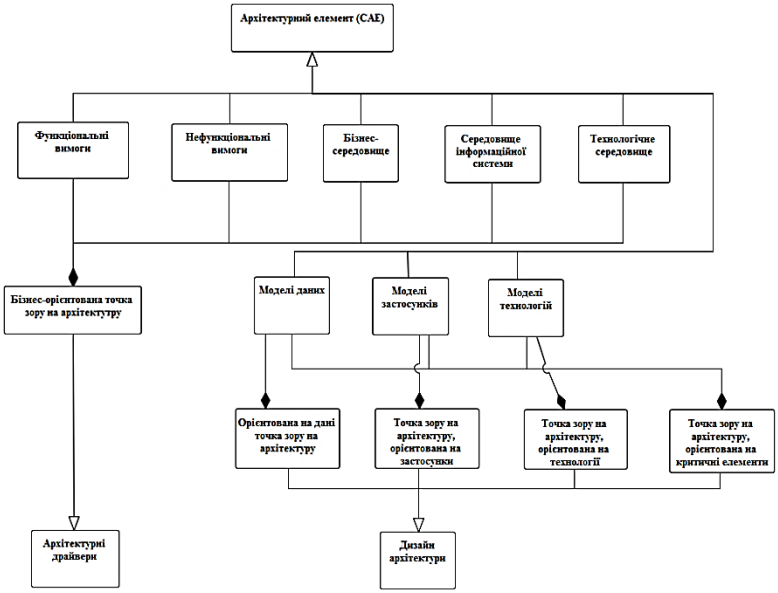

Рис. 10. UML-діаграма класифікації архітектурних елементів відповідно до точок зору на архітектуру

Архітектурний елемент як мотиваційна причина може бути представлений у вигляді:

– функціональних та нефункціональних вимог;

– припущень – документування невідомих або очікувань надає контекст для прийняття рішення;

– обмежень – можуть бути технічного, бізнес, організаційного або природного характеру;

– об'єктів дизайну – можуть впливати через свою поведінку на дизайн архітектури.

У вигляді результату дизайну архітектурний елемент може бути представлений як:

– об'єкти дизайну – можуть бути результатом рішення стосовно архітектури, які задовольняють певні мотиваційні причини;

– уточнені об’єкти дизайну – можуть бути уточнені в результаті прийняття відповідних рішень.

У цьому досліджені передбачається, що бізнес-орієнтована точка зору включає вимоги та фактори середовищ. Створено п’ять категорій бізнес-орієнтованої точки зору на архітектуру, які описують різні аспекти впливу на архітектурний дизайн. Це і є архітектурні драйвери. Системні вимоги включають в себе функціональні та нефункціональні вимоги. Така класифікація дозволяє архітектору під час аналізу відслідковувати процес обґрунтування рішень до специфічних класів кореневих причин.

Елементи «Дизайн архітектури» є результатом процесу проектування. Вони класифіковані за наступними точками зору на архітектуру:

– орієнтовані на дані – отримані та використовувані застосунками;

– орієнтовані на застосунки – логіка обробки і структура програмного забезпечення, критичні та некритичні сервіси;

– орієнтовані на технології – технології та середовища, які використовуються для впровадження та розгортання системи, критичні та некритичні компоненти.

Підхід RECAD надає можливість використовувати три типи обґрунтувань архітектури: кількісний, якісний та за допомогою альтернативної архітектури. Якісне обґрунтування представляє процес обґрунтування та аргументи у текстовій формі, фактично, за та проти для кожного проектного рішення. Кількісне обґрунтування використовує різні критерії під час оцінювання проектних рішень. Третій тип передбачає документування та зберігання відкинутих альтернативних проектних рішень та їх подальший перегляд з метою оцінки достатності наявних параметрів оцінювання наявних архітектурних проектів, а також для майбутнього використання в інших проектах.

Слід зазначити, що архітектурне рішення може еволюціонувати з часом. Причиною цьому можуть бути як зміни в бізнес-процесах підприємства, так і зміни самого бізнес-середовища. Еволюціонуючи, можна втратити вихідний архітектурний проект та опис процесу обґрунтування рішень щодо цього проекту. Тому потрібно якимось чином зберігати всю історію еволюції проекту. Для цього пропонується використовувати розширену модель RECAD.

Означення 1.11. Розширена модель RECADe – це РТС $RECADe = (CAE, CAE^0, AR, A, R, \Sigma, L)$, на якій задана бієктивна функція відображення $SSf : (CAE \to CAE_h) \cup (AR \to AR_h)$ між архітектурними елементами або обґрунтуваннями архітектури, де:

$RECAD_{cur} = (CAE_{cur}, CAE^0_{cur}, AR_{cur}, A_{cur}, R_{cur}, \Sigma_{cur}, L_{cur})$ –

актуальна модель архітектури, і виконуються наступні умови:

– $CAE_{cur} \subseteq CAE$,

– $AR_{cur} \subseteq AR$,

– $A_{cur} \subseteq A$,

– $\Sigma_{cur} \subseteq \Sigma$,

– $CAE^0_{cur} \subseteq CAE^0$,

– $L_{cur} : CAE_{cur} \to 2^{\Sigma_{cur}}$,

– $R_{cur} \subseteq R \cap ((CAE_{cur} \times AR_{cur}) \cup (AR_{cur} \times CAE_{cur}))$,

– не існує підмножини з множини R_{cur}, яка утворює направлений цикл;

$RECAD_h = (CAE_h, CAE^0_h, AR_h, R_h, \Sigma_h, L_h)$ – історична модель архітектури, і виконуються наступні умови:

– $CAE_h \subseteq CAE$,

– $AR_h \subseteq AR$,

– $A_h \subseteq A$,

– $\Sigma_h \subseteq \Sigma$,

– $CAE^0_h \subseteq CAE^0$,

– $L_h : CAE_h \to 2^{\Sigma_h}$,

– $R_h \subseteq R \cap ((CAE_h \times AR_h) \cup (AR_h \times CAE_h))$;

– не існує підмножини з множини R_h, яка утворює направлений цикл;

і для яких виконуються наступні умови:

1. $CAE_h = CAE \setminus CAE_{cur}$,
2. $AR_h = AR \setminus AR_{cur}$,
3. $A_h = AR \setminus A_{cur}$,
4. $\Sigma_h = \Sigma \setminus \Sigma_{cur}$,
5. $CAE^0_h = CAE^0 \setminus CAE^0_{cur}$,
6. $L_h = L \setminus L_{cur}$.

На рис. 11 представлена відповідна модель UML-діаграма.

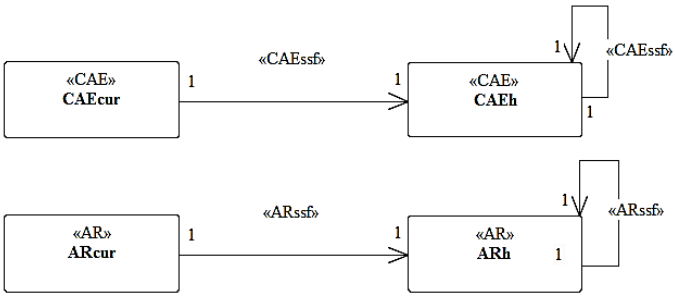

Рис. 11. UML-діаграма розширеної моделі RECADe

Імплементація підходу RECAD за допомогою UML. Для імплементації архітектурного елементу *CAE* визначено новий стереотип <<CAE>>. Цей стереотип розширює можливості UML-конструктів типу «об'єкт» та «клас» щодо підтримки властивостей відстежуваності та критичності. Це означає, що до існуючих конструктів додаються додаткові характеристики та атрибути з метою покращення процесу обґрунтування рішень. Додаткові атрибути стереотипу <<CAE>> імплементовані як теговані значення (tagged values):

– *elementID* – унікальний ідентифікатор архітектурного елементу;

– *elementVersion* – ціле число, яке ідентифікує версію архітектурного елементу і використовується для моделі CARELe;

– *elementType* – тип елементу відповідно до введеної класифікації за точками зору на архітектуру;

– *elementSubType* – підтип елементу відповідно до введеної класифікації за точками зору на архітектуру;

– *currFlag* – прапорець актуальності елементу ("Y" – остання версія, "N" – історична версія);

– *criticalFlag* – прапорець критичності елементу ("Y" – критичний елемент, "N" – некритичний елемент);

– *lastUpdateDateTime* – час останньої модифікації елементу;

– *author* – автор змін;

– *documentLocation* – місце розташування будь-яких зовнішніх документів, які описують елемент.

Під час перевірки відстежуваності параметри *elementType* та *elementSubType* дозволяють обирати фокус, щодо якого потрібно провести дослідження архітектури. Параметри *elementVersion* та *currFlag* дозволяють відстежувати процес еволюції архітектури, *criticalFlag* – критичність елементів та процесів.

З метою моделювання бізнес-орієнтованих точок зору, які містять вимоги, фактори середовищ та припущення використовується створений стереотип <<CAE>> для розширення елементів UML, таких як клас, об'єкт, артефакт та прецедент (рис. 12).

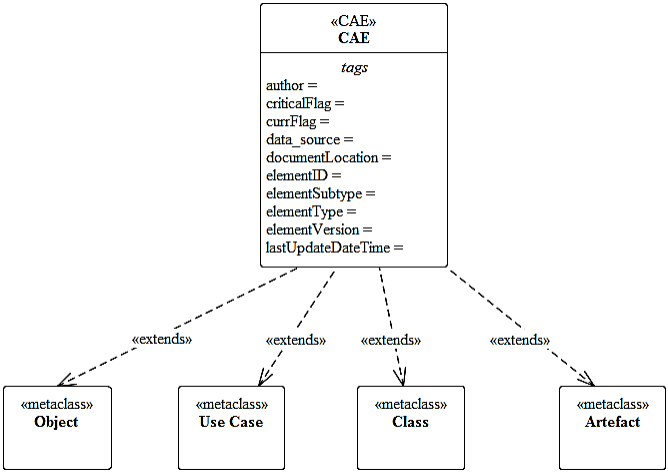

Рис. 12. Розширення елементів UML стереотипом <<CAE>>

Використання таких елементів дозволяє створювати моделі з різними точками зору на архітектуру.

Для імплементації елементу «*обґрунтування рішення*» використано стереотип <<AR>>, який розширяє можливості UML-конструкту «Пакет». Додаткові атрибути стереотипу <<AR>> імплементовані як теговані значення:

– *arID* – унікальний ідентифікатор елементу «Обґрунтування архітектури»;

– *currFlag* – прапорець актуальності елементу ("Y" – остання версія, "N" – історична версія);

– *criticalFlag* – прапорець критичності елементу (”Y” – критичний елемент, “N” – некритичний елемент);

– *lastUpdateDateTime* – час останньої модифікації елементу;

– *author* – автор особи, яка приймає рішення.

На рис. 13 представлена UML-діаграма взаємозв’язків стереотипів <<AR>> та <<AAR>>. Їх імплементація базується на конструкті «Пакет». AR може включати нуль або більше елементів AAR (обґрунтування альтернативної архітектури) залежно від кількості відкинутих альтернативних проектних рішень.

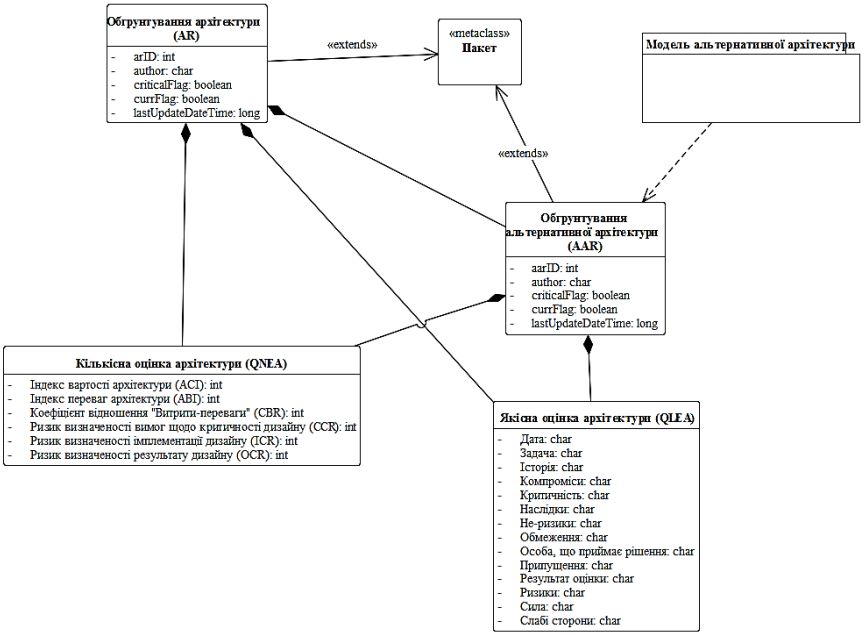

Рис. 13. UML-діаграма взаємозв’язків стереотипів <<AR>> та <<AAR>>

Стереотип <<AAR>> надає шаблон для документування альтернативних проектних рішень, які були відкинуті.

Відношення між *CAE* та *AR* представляється за допомогою направленої асоціації з новоствореним стереотипом <<ARtr>>.

Введені моделі RECAD та RECADe дозволяють представити та описати окремі конкретні проектні рішення щодо систем,

компонент та процесів, які входять до архітектури критичної ІТ-інфраструктури.

Визначимо для моделі RECAD її асинхронну паралельну композицію (Означення 1.12).

Означення 1.12. Нехай задані РТС

$RECAD_1 = (CAE_1, CAE_1^0, AR_1, A_1, R_1, \Sigma_1, L_1)$

та $RECAD_2 = (CAE_2, CAE_2^0, AR_2, A_2, R_2, \Sigma_2, L_2)$

такі, що

$CAE_1 \cap CAE_2 = \varnothing, AR_1 \cap AR_2 = \varnothing, \ A_1 \cap A_2 = \varnothing, \Sigma_1 \cap \Sigma_2 = \varnothing.$

Нехай також задана синхронізаційна пара

$\Lambda = (\Delta, \bar{}), \Delta \subseteq A_1, \bar{} : \Delta \to A_2$. Тоді модель

$RECAD = (CAE, CAE^0, AR, A, R, \Sigma, L) = \ = RECAD_1 \parallel_\Lambda RECAD_2$

називається паралельною композицією

РТС $RECAD_1$ і $RECAD_2$, якщо:

– вірні співвідношення

$CAE = CAE_1 \times CAE_2, AR = AR_1 \times AR_2, S^0 = S_1^0 \times S_2^0,$

$A = A_1 \cup A_2 \setminus (\Delta \cup \overline{\Delta}), \ \Sigma = \Sigma_1 \cup \Sigma_2,$

– вірно $L : CAE \to 2^\Sigma, \ L((cae_1, cae_2)) = L(cae_1) \cup L(cae_2),$

– відношення

$R \subseteq ((CAE_1 \times CAE_2) \times A \times (AR_1 \times AR_2)) \cup ((AR_1 \times AR_2) \times A \times (CAE_1 \times CAE_2))$

задається наступним чином: $((s,u),a,(t,v)) \in R$ тоді і тільки тоді, коли виконується одна з трьох умов:

а) виконується перехід першої моделі: $(s,a,t) \in R_1, u = v$;

б) виконується перехід другої моделі: $(u,a,v) \in R_2, s = t$;

в) виконується синхронний обмін повідомленнями: $(s,b,t) \in R_1, (u,b,v) \in R_2, a = \tau$.

Таким чином, модель архітектурного рішення критичної ІТ-інфраструктури можна представити наступним чином (Означення 1.13).

Означення 1.13. Модель архітектурного рішення критичної ІТ-інфраструктури визначається у вигляді паралельної композиції всіх окремих проектних рішень щодо систем, компонент та процесів, які входять до архітектури критичної ІТ-інфраструктури, заданих у вигляді РТС RECAD, тобто: $M = RECAD_1 \parallel_{\Lambda_1} .. \parallel_{\Lambda_{n-1}} RECAD_n$, де

$RECAD_1, ..., RECAD_n$ – відповідні РТС, $\Lambda_1, ..., \Lambda_{n-1}$ – задані на моделі пари синхронізації.

3. Верифікація моделей на базі РТС

У методах верифікації моделей (ВМ) специфікації РС зазвичай задаються за допомогою формул темпоральної логіки **Указан недопустимый источник.**. Формули темпоральної логіки описують тимчасові властивості обчислень (послідовностей переходів) моделі. Найбільше застосування в методі ВМ отримали логіки дерев обчислень (Computational Tree Logic, CTL), лінійного часу (Linear Time Logic, LTL) і узагальнення обох логік – логіка CTL⋆. Огляд методів верифікацій моделей наведено в роботі[8].

Задача верифікації моделей архітектурних рішень критичних IT-інфраструктур. Задача верифікації моделей архітектурних рішень критичних IT-інфраструктур формулюється наступним чином:

Задача 1.1. Дана модель архітектурного рішення $M = P_1 \| ... \| P_n$, де $P_1, ..., P_n$ – РТС RECAD критичної IT-інфраструктури. Задана формула (специфікація) φ темпоральної логіки відносно змінних моделі M. Необхідно перевірити виконуваність формули φ моделі M (позначається як $M \models \varphi$).

Задача верифікації параметризованих моделей архітектурних рішень критичних IT-інфраструктур.

У цій роботі розглядається узагальнення проблеми верифікації моделей. Методи верифікації застосовуються до моделей КІТІ, представлених у вигляді UML-діаграм, що складаються з кінцевої множини елементів архітектури та обґрунтувань, заданих у вигляді РТС. Як зазначалося у вступі, існують КІТІ, число процесів в яких залежить від початкової конфігурації або настройки КІТІ, а множина таких конфігурацій нескінченна. При цьому число процесів не змінюється під час роботи. Для КІТІ, в яких число процесів залежить від початкової конфігурації, можуть бути побудовані моделі з кінцевим числом станів для кожної початкової конфігурації системи. Оскільки множина початкових конфігурацій нескінченна, то і множина моделей архітектурних рішень з різним числом процесів нескінченна. Верифікація декількох, випадково обраних моделей, з цієї множини не гарантує виконуваність

[8] Dorogyy Y.A SURVEY OF PARAMETRIC MODEL VERIFICATION METHODS. *Scientific Journal «Science Rise*. 2020. Vol. 12. No. 41. pp. 42-47.

специфікації на всіх моделях множини. Для таких систем розглядається задача, яка в загальному випадку формулюється таким чином:

Задача 1.2. Дано нескінченне сімейство скінченних моделей архітектурних рішень КІТІ $\Upsilon = \{M_n\}$, параметризоване за параметром $n \in N$. Задана формула (специфікація) φ темпоральної логіки. Необхідно перевірити виконуваність формули φ на всіх моделях Υ, тобто $M_n \models \varphi$ для всіх n. Ця задача в літературі отримала назву верифікації параметризованих моделей (ВПМ).

Постановка задачі 1.2 потребує уточнення, оскільки в загальній постановці явно не вказується спосіб визначення сімейства F та специфікації φ. Тому фактично будемо розглядати наступний варіант задачі 1.2.

Задача 1.3. Дано нескінченне сімейство скінченних моделей архітектурних рішень КІТІ $\Upsilon = \{M_n\}$, параметризоване за параметром $n \in N$. Кожна модель $M_n = Q \| P_1 \| ... \| P_n$ складається з РТС фіксованої моделі RECAD Q та n екземплярів РТС альтернативних моделей P_i. Зафіксована кінцева множина $I \subseteq \Upsilon$ індексів альтернативних моделей, що спостерігаються, екземплярів прототипів P. Специфікація φ темпоральної логіки задається відносно змінних визначених моделей $P_i, i \in I$ та змінних моделі Q. Необхідно перевірити виконуваність формули φ на всіх моделях сімейства Υ, тобто $M_n \models \varphi$ для всіх .

Фактично, в постановці задачі 1.3 використовується лише одна фіксована модель та екземплярів прототипів . Можна розглядати варіант задачі, в якому присутні декілька фіксованих моделей та декілька альтернативних моделей , а модель , де . В деяких випадках ця задача зводиться до постановки задачі 1.3 за допомогою побудови РТС моделі у вигляді паралельної композиції моделей , а РТС прототипу у вигляді паралельної композиції РТС прототипів .

Запропонована постановка задач у вигляді (1.1-1.3) дозволяє застосувати для верифікації побудованих моделей широке коло формалізованих методів верифікації параметризованих моделей,

заданих у вигляді РТС. У цій роботі для верифікації побудованих моделей використано один з методів пошуку інваріантів[9].

4. Критерій оптимальності проектування та функціонування КІТІ

Під час аналізу процесу проектування нової критичної ІТ-інфраструктури потрібно враховувати:

– велику вартість та складність створюваної критичної ІТ-інфраструктури;

– наявність сучасних методологій, засобів та інструментів її створення.

Критична ІТ-інфраструктура повинна створюватись за наступними принципами:

– мінімізація чисельності необхідного людського професійного ресурсу та досягнення встановлених цілей експлуатації за мінімальних витрат цього ресурсу;

– мінімізація загальної тривалості робіт, які проводяться для критичної ІТ-інфраструктури;

– мінімізація часу зниження готовності критичної ІТ-інфраструктури;

– мінімізація часу відновлення готовності критичної ІТ-інфраструктури;

– мінімізація стороннього впливу під час проведення робіт з критичною ІТ-інфраструктурою;

– захищеність критичної ІТ-інфраструктури від можливих несанкціонованих дій;

– максимізація ефективності критичної ІТ-інфраструктури, її ресурсів та строку її служби.

Аналіз ІТ-інфраструктур та критичних ІТ-інфраструктур, цілей їх функціонування, структури та взаємозв'язків між елементами дозволяє визначити наступні основні їх особливості:

– наявність мети функціонування, що визначає її призначення та характер функціонування;

– наявність великої кількості різноманітних об'єктів управління;

– наявність ієрархічної структури управління;

[9] Konnov I. Parameterized Model Checking by Network Invariants: the Asynchronous Case. *LICS Workshop Algorithmics on Infinite State Systems*. Dubrovnik, Croatia. 2012.

– наявність мети та цілей управління для кожного підрівня ієрархічної структури;

– наявність великої кількості внутрішніх зв'язків у кожній підсистемі між її елементами;

– безперервна зміна стану функціонування елементів підсистем і системи в цілому, кількості її елементів та підсистем, критеріїв функціонування.

Оскільки в критичних ІТ-інфраструктурах зосереджені потужні і досить дорогі ресурси, виникає проблема їх ефективного використання. Це стає можливим за рахунок розподілу, управління і диспетчерування ресурсів, управління навантаженням на основі комплексу відповідних математичних моделей і методів. Формалізація проблеми ефективного використання ресурсів привела до задач нечіткого програмування[10] з широким вибором критеріїв оптимізації і врахуванням критичності, ресурсних, часових, технологічних та інших обмежень.

Для формування критерію оптимальності проектування і подальшого функціонування критичної ІТ-інфраструктури пропонується використати наступну множину параметрів[11]:

– доступність – визначає можливість мати доступ до всіх наявних ресурсів з будь-якої точки в будь-який час;

– надійність – визначає можливість безперервного користування наявним функціоналом критичної ІТ-інфраструктури;

– живучість – визначає можливість виконувати свої функції у разі втрати ресурсів, підсистем тощо;

– забезпеченість – визначає показник максимальної кількості процесів та сервісів, що обслуговуються;

– відновлюваність – визначає показник тривалості відновлення готовності до експлуатації;

– економічність – визначає показник витрат різноманітних ресурсів на забезпечення функціонування критичної ІТ-інфраструктури;

[10] Liu B. Theory and Practice of Uncertain Programming. UTLAB, 2009.

[11] Дорогий Я.Ю., Дорога-Іванюк О.О. Критерій оптимальності побудови критичної ІТ-інфраструктури. *Актуальні проблеми розвитку науки і техніки: Матеріали другої міжнародної науково-технічної конференції*, 20 грудня. Київ. 2015. Р. 10.

– безпечність – визначає показник неможливості виконання несанкціонованих дій, спрямованих на порушення роботи критичної ІТ-інфраструктури чи її частин;

– адаптивність – визначає можливість критичної ІТ-інфраструктури динамічно реагувати та адаптуватися до змін у бізнес-процесах, наявних ресурсах і таке інше;

– строк життя;

– ефективність – визначає можливість поєднання вищезгаданих параметрів в кожному окремому випадку під визначену задачу.

Під час формування вимог критерію проектування критичної ІТ-інфраструктури необхідно також враховувати особливості розв'язуваних нею задач. Слід зазначити, що визначення всієї множини параметрів не можна повністю звести до системи формалізованих процедур, бо деякі з них вимагають якісного аналізу. Для такого аналізу слід використати метод структуризації, який дозволяє поділити задачу на підзадачі, визначитись за допомогою експертів або без них з методами розв'язання цих підзадач, обмеженнями використання цих розв'язків та методами поєднання розв'язків.

Враховуючи викладені аспекти та стадії життєвого циклу критичної ІТ-інфраструктури, пропонується особливо звернути увагу на наступні питання під час створення критичної ІТ-інфраструктури:

– формування образу критичної ІТ-інфраструктури;

– визначення цілей, побудова дерева цілей та вибір шляхів їх досягнення, формування вимог та критерію оптимальності критичної ІТ-інфраструктури;

– оцінку альтернативних варіантів побудови критичної ІТ-інфраструктури;

– обґрунтування вибору архітектурних рішень.

Образ критичної ІТ-інфраструктури формується з врахуванням прогнозу її розвитку та наробіток. При цьому попередньо визначаються структура критичної ІТ-інфраструктури (тобто перелік її наявних та можливих складників, можливих конструктивних рішень і таке інше), умови експлуатації (перелік характеристик, що визначають умови використання; фактори зовнішнього середовища, що впливають на функціонування, і так далі).

Наступний крок – визначення цілей функціонування системи. На даному етапі потрібно врахувати вимоги замовника, стан технічного та наукового розвитку в даній галузі. На базі ітеративного процесу визначення цілей функціонування будується дерево цілей та шляхів їх досягнення, яке покриває всі елементи системи та етапи їх експлуатації. Виконується оцінка можливості досягнення поставлених цілей. Побудоване дерево дає можливість визначити основні вимоги до критичної ІТ-інфраструктури, а також можливі рекомендації по конструктивним рішенням окремих підсистем та елементів.

Для визначення найбільш ймовірного варіанту побудови критичної ІТ-інфраструктури потрібно:

– оцінити можливість якісного стрибка в розвитку на базі нових технічних ідей;

– оцінити можливість розвитку на базі попередніх рішень шляхом покращення окремих характеристик.

Підсумовуючи все вищесказане, пропонується визначити досяжність поставлених перед критичною ІТ-інфраструктурою цілей шляхом визначення узагальнених показників (про деякі з них згадувалось вище).

Узагальнені показники визначаються за допомогою відповідних математичних моделей, які будуються для кожної критичної ІТ-інфраструктури.

Задача побудови критичної ІТ-інфраструктури належить до задач багатокритеріального аналізу і в загальному випадку має наступний вигляд:

$$U_{opt} = opU(x) \qquad (1)$$

де:

U_{opt} – оптимальне рішення;

$x = \{x_i \mid i = 1..n\}, x \in D_{x_i}$ – вектор експлуатаційних параметрів, а D_{x_i} – множина обмежень цих параметрів;

$U = \{U_i \mid i = 1..n\}, U \in D_{U_i}$ – вектор узагальнених експлуатаційних вимог, а D_{U_i} – підмножина їх допустимих значень;

op – оператор оптимізації.

Під час оцінки ефективності роботи критичної ІТ-інфраструктури на ранніх стадіях створення потрібно враховувати витрати на реалізацію, оскільки наявність ресурсів обмежена.

В загальному випадку для кожного окремого випадку функціонування критичної ІТ-інфраструктури задача зводиться до пошуку компромісу між необхідними характеристиками функціонування. Наприклад, в одному випадку вимагається максимальна безпека функціонування, а всі інші характеристики не такі важливі, в іншому – потрібно забезпечити максимальну надійність функціонування.

ВИСНОВКИ

У цій статті представлений детальний опис методу представлення архітектури критичної ІТ-інфраструктури на базі розширених конструктів UML, наведено основні базові елементи математичного апарату розмічених транзиційних систем, використаних для опису моделей даного методу представлення архітектури у формалізованому вигляді. Запропонований метод дозволяє представити сам процес відображення процесу міркувань щодо архітектури критичної ІТ-архітектури в досить зручний спосіб та дає можливість досліднику накопичувати історію цих міркувань. Також представлена адаптація задачі верифікації параметризованих моделей, яка дозволяє застосувати для верифікації моделей архітектурних проектів, представлених у вигляді UML-діаграм і формалізованих у вигляді PTC математичний апарат темпоральних логік і, як результат, дозволяє використовувати для розв'язання цієї задачі широке коло різноманітних методів верифікації моделей без їх значного доопрацювання та модифікації.

АНОТАЦІЯ

У статті розглядаються питання, пов'язані з проблемою представлення проектних рішень щодо архітектури критичної ІТ-інфраструктури.

По-перше, наведено опис математичного апарату розмічених транзиційних систем, основні означення, який використано для визначення математичних моделей представлення архітектури критичної ІТ-інфраструктури. Наведені деякі приклади використання цього математичного апарату для представлення складних систем і процесів.

По-друге, в статті наведено детальний опис методу застосування розширених UML-діаграм для представлення проектних рішень щодо архітектури критичної IT-інфраструктури. Описані основні його елементи та спосіб переходу від UML-діаграм до формалізованих математичних моделей на базі розмічених транзиційних систем.

Запропоновано спосіб використання раніше створених методів верифікації параметризованих моделей шляхом визначення нової постановки задачі верифікації, яка дозволяє це зробити.

По-третє, визначено критерій оптимальності побудови критичної IT-інфраструктури.

Результати, викладені в статті, допоможуть дослідникам і розробникам архітектури критичних IT-інфраструктур вибрати найбільш придатний варіант стратегії прийняття рішень під час проектування власної архітектури, а також ознайомитися з критеріями її проектування та оцінки, з способами представлення проектних рішень. Варто мати на увазі, що не існує універсального підходу, який би підійшов для розробки будь-якої архітектури, тому так важливо визначити слабкі та сильні сторони кожного з них на етапі проектування системи, а також критерії проектування. Від того, наскільки вірно було визначено ці критерії, буде залежати і вибір конкретних технологій, і програмного забезпечення, а також успішність проектування.

ЛІТЕРАТУРА

1. Tel G. Introduction to Distributed Algorithms. Cambridge: Cambridge University Press, 2000. P. 610.

2. Tanenbaum A., van Steen M. Distributed Systems: Principles and Paradigms. Upper Saddle River: Create Space Independent Publishing Platform, 2016. P. 704.

3. van Steen M., Tanenbaum A.S. A brief introduction to distributed systems. *Computing*. 2016. № 98 (10). С. 967–1009.

4. Теленик С.Ф., інш. Методы исследования свойств высокопроизводительных инфраструктур. Обзор. *Фундаментальные и прикладные проблемы информатики и информационных технологий*. 2015. № 1. С. 3–13.

5. Крывый С.Л., Максимец А.Н. Верификация программ: состояние, проблемы, результаты. *Кибернетика и системный анализ*. 2013. Т. 49. С. 3–14.

6. Дорогий Я.Ю. Проектування критичних ІТ-інфраструктур з використанням розмічених транзиційних систем. *Материалы V заочной научной конференции «Фундаментальные и прикладные исследования в современной науке»*. Харьков. 2017. Р. 48.

7. Дорогий Я.Ю. Моделювання критичних ІТ-інфраструктур з використанням розмічених транзиційних систем. *Матеріали VI Міжнародної науково-практичної конференції (І Міжнародний симпозіум «Практичне застосування нелінійних динамічних систем в інфокомунікаціях», PREDT-2017)*. 9-11 листопада. Чернівці. 2017.

8. Dorogyy Y. A SURVEY OF PARAMETRIC MODEL VERIFICATION METHODS. *Scientific Journal "Science Rise"*. 2020. Vol. 12. No. 41. pp. 42–47.

9. Konnov I. Parameterized Model Checking by Network Invariants: the Asynchronous Case. *LICS Workshop Algorithmics on Infinite State Systems*. Dubrovnik, Croatia. 2012.

10. Liu B. Theory and Practice of Uncertain Programming. UTLAB, 2009.

11. Дорогий Я.Ю., Дорога-Іванюк О.О. Критерій оптимальності побудови критичної ІТ-інфраструктури. *Актуальні проблеми розвитку науки і техніки: Матеріали другої міжнародної науково-технічної конференції*, 20 грудня. Київ. 2015. Р. 10.

Information about author:
Dorogyy Ya. Yu.,
Candidate of Technical Sciences, Associate Professor,
Associate Professor in Department of Automation
and Control in Technical Systems
National Technical University of Ukraine
"Igor Sikorsky Kyiv Politechnic Institute"
37, Victory av., Kyiv, 03056, Ukraine