Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

СОЛОДОВНІКОВ АНДРІЙ СЕРГІЙОВИЧ

УДК 004.2:004.4’2

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ СИНТЕЗУ ПРОГРАМНОЇ АРХІТЕКТУРИ НА ОСНОВІ ГРАФОВОЇ МОДЕЛІ

05.13.06 – інформаційні технології

Автореферат дисертації на здобуття наукового ступеня кандидата технічних наук

Харків – 2017

Дисертацією є рукопис.

Роботу виконано в Харківському національному університеті радіоелектроніки**,** Міністерство освіти і науки України.

Науковий керівник: кандидат технічних наук, старший науковий співробітник,
Чайніков Сергій Іванович,
Харківський національний університет радіоелектроніки,
професор кафедри системотехніки.

Офіційні опоненти: доктор технічних наук, професор
Шматков Сергій Ігорович,
Харківський національний університет ім. В. Н. Каразіна,
завідувач кафедри теоретичної та прикладної системотехніки;

доктор технічних наук, професор
Шостак Ігор Володимирович,
Національний аерокосмічний університет ім. М. Є. Жуковського «Харківській авіаційний інститут»,
професор кафедри інженерії програмного забезпечення

Захист відбудеться «12» квітня 2017 р. о 15:30 годині на засіданні спеціалізованої вченої ради Д 64.052.08 Харківського національного університету радіоелектроніки за адресою: 61166, м. Харків, пр. Науки, 14.

З дисертацією можна ознайомитись у бібліотеці Харківського національного університету радіоелектроніки за адресою: 61166, м. Харків, пр. Науки, 14.

Автореферат розісланий «10» березня 2017 р.

Учений секретар
спеціалізованої вченої ради І. П. Плісс

ЗАГАЛЬНА ХАРАКТЕРИСТИКА РОБОТИ

Актуальність теми. Існуючі методи, технології та інструментальні засоби створення програмного забезпечення інформаційних систем такі як Agile, Service Oriented Architecture (SOA), Structured Systems Analysis And Design Method (SSADM), Meris, Test-Driven Development (TDD), Behavior-Driven Development (BDD), Dynamic Software Product Line (DSPL) дозволяють виконувати розробку як при визначених вимогах на основі водоспадної стратегії, так і при вимогах кінцевого користувача, що змінюються еволюційно, на основі стратегії гнучкої розробки.

У той же час в сучасних ринкових умовах з високою конкуренцією потрібна кастомізація програмного забезпечення, яка полягає в його адаптації до потреб кінцевого користувача, що змінюються у часі, з урахуванням специфіки конкретного підприємства і конкретного робочого місця. Необхідність у кастомізації може виникати як в процесі розробки, так і супроводу програмного забезпечення. Методи і технології, які засновані на водоспадній стратегії, недоцільно застосовувати для вирішення задачі кастомізації, оскільки вони використовують тільки апріорно відомі вимоги кінцевого користувача. Технології гнучкої розробки засновані на використанні евристик при побудові програмної архітектури і тому їх застосування в задачі кастомізації призводить до надмірних матеріальних витрат.

Виходячи з цього, невідповідність між можливостями існуючих технологій проектування і практичною необхідністю адаптації програмного забезпечення до функціональних вимог кінцевих користувачів, що змінюються в часі, призводить до виникнення проблеми розробки ефективних формальних підходів до кастомізації програмного забезпечення інформаційної системи. Розв’язання зазначеної проблеми вимагає розробки нових формальних графових моделей програмної архітектури інформаційної системи та вдосконалення інформаційної технології синтезу програмної архітектури на основі цієї моделі.

Значний внесок у розвиток теорії проектування і розробки програмного забезпечення на основі графових моделей програмної архітектури внесли В. В. Воєводін, О. Л. Перевозчикова, А. А. Шалито, Е. Л. Ющенко, а в створення і дослідження методів структурного синтезу, адаптації та кастомізації програмного забезпечення – І. Д. Зайцев, Е. М. Лавріщева, Петров Е. Г., Karl Lieberherr, Pekka Kalevi Abrahamsson, Harald F.O. von Korflesch, Matthias Bertram та інші фахівці.

Однак слід зазначити, що при розробці формальних графових моделей програмної архітектури з використанням існуючих методів еволюційні зміни вимог кінцевого користувача зазвичай не розглядаються, що призводить до труднощів при вирішенні зазначеної задачі кастомізації.

У зв’язку з цим, розробка методів структурного синтезу програмної архітектури інформаційної системи з урахуванням вимог кінцевого користувача, що змінюються під час її розробки та впровадження, є актуальною науково-практичною задачею.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційна робота виконана в рамках плану науково-дослідних робіт кафедри системотехніки

Харківського національного університету радіоелектроніки за темою № 293 «Розробка методології математичних моделей соціально-економічних систем при реалізації концепції їх стійкого розвитку» за розділом № 293-1 «Розробка математичних моделей реалізації концепції стійкого розвитку соціально-економічних систем» (№ ДР 0115U002429).

Мета та задачі дослідження. Метою дослідження є розробка моделі, методів та інформаційної технології синтезу програмної архітектури інформаційної системи для підвищення ефективності конфігурування програмного забезпечення в умовах змінних вимог кінцевого користувача для формалізованих задач.

Для досягнення мети необхідно було розв’язати такі задачі:

– аналіз методів та інструментальних засобів розробки і конфігурування програмної архітектури інформаційної системи;

– розробка ярусно-паралельної графової моделі програмної архітектури інформаційної системи;

– розробка методу об'єднання вершин графової ярусно-паралельної моделі програмної архітектури інформаційної системи на основі оцінки показників складності і зв'язності програмних модулів;

– розробка графового методу оцінки функціональної складності програмного забезпечення;

– розробка автоматного методу перевірки виконання обмежень до програмного забезпечення, що формується;

– розробка інформаційної технології структурного синтезу програмної архітектури інформаційної системи з можливістю конфігурування програмного забезпечення в умовах вимог кінцевого користувача, що змінюються;

– програмна реалізація інформаційної технології структурного синтезу програмної архітектури інформаційної системи з можливістю конфігурування програмного забезпечення в умовах вимог, що змінюються;

– впровадження результатів дослідження для вирішення практичних задач конфігурування програмного забезпечення.

Об'єктом дослідження є процеси синтезу програмної архітектури інформаційної системи в умовах вимог кінцевого користувача, що змінюються.

Предметом дослідження є графові моделі і методи структурного синтезу програмної архітектури інформаційної системи в умовах вимог кінцевого користувача, що змінюються еволюційно, на основі графових моделей.

*Методи дослідження***.** У процесі дисертаційного дослідження використані теорія графів при побудові графової моделі програмної архітектури інформаційної системи та розробці методів спрощення структурної і функціональної складності графової моделі, а також теорія автоматів і некласичні логіки при розробці автоматного методу перевірки виконання обмежень до програмного забезпечення, що формується.

Наукова новизна отриманих результатів полягає у розробці моделі та методів структурного синтезу програмної архітектури інформаційної системи на основі графових моделей, для чого:

– *вперше* запропоновано метод об'єднання вершин ярусно-паралельної

графової моделі програмної архітектури інформаційної системи, що відображають програмні модулі, який відрізняється від існуючих послідовним розрахунком показників зчеплення і зв'язності модулів та дозволяє зменшити структурну складність моделі і тим самим знизити часові витрати на адаптацію програмної архітектури;

– *вперше* запропоновано графовий метод оцінки функціональної складності програмного забезпечення, який відрізняється від існуючих використанням графової ярусно-паралельної моделі програмної архітектури інформаційної системи та включає до себе етапи визначення функціональних характеристик програмних модулів і розрахунку критеріїв функціональної складності на основі виділення підграфів функціональних задач, що дозволяє оцінити відповідність програмного забезпечення, яке формується, функціональним вимогам кінцевого користувача при конфігуруванні програмної архітектури;

– *удосконалено* автоматний метод перевірки виконання обмежень до програмного забезпечення, що формується, які виражені у формі нефункціональних вимог. Метод відрізняється від існуючих етапами синтезу автоматної моделі взаємодії модулів, перевірки виконуваності функцій програмного забезпечення, відмовостійкості в умовах апаратних обмежень і порівняння з вимогами до сценаріїв взаємодії модулів. Це дозволяє підвищити ефективність конфігурування програмної архітектури з урахуванням вимог до розгортання програмного забезпечення;

– *удосконалено* графову ярусно-паралельну модель програмної архітектури інформаційної системи, яка відрізняється від існуючих врахуванням взаємозв'язків по даним між модулями, що представлені в моделі у вигляді вершин графа. Модель дозволяє на основі оцінки структурної складності підвищити ефективність адаптації програмної архітектури інформаційної системи до вимог кінцевого користувача, що змінюються еволюційно.

Практичне значення отриманих результатів. На основі запропонованих моделі та методів удосконалено інформаційну технологію структурного синтезу програмної архітектури інформаційної системи, яка, на відміну від існуючих технологій, містить етапи синтезу графової ярусно-паралельної моделі програмної архітектури, конфігурації архітектури з урахуванням поточних функціональних вимог і перевірки виконання нефункціональних вимог на основі відповідного автоматного методу, що дозволяє знизити функціональну і структурну складність програмного забезпечення і, тим самим, знизити витрати на реалізацію функціональних завдань інформаційної системи з урахуванням вимог кінцевого користувача, що змінюються у часі. Інформаційна технологія реалізована у вигляді програмного продукту, робота якого засновується на запропонованій графовій ярусно-паралельній моделі програмної архітектури інформаційної системи, методі об'єднання вершин ярусно-паралельної графової моделі програмної архітектури інформаційної системи, методі оцінки функціональної складності програмного забезпечення, методі перевірки виконання обмежень до програмного забезпечення, що формується, та дозволяє формувати програмне забезпечення на основі змінних вимог кінцевого користувача.

Впроваджено інформаційну технологію у вигляді програмного забезпечення у

наукову та проектну діяльність Інституту фізики високих енергій і ядерної фізики Національного наукового центру «Харківський фізико-технічний інститут» (ІФВЕЯФ ННЦ ХФТІ, акт впровадження від 18.04.2016 р.) та методи спрощення структурної та функціональної складності ярусно-паралельної графової моделі програмної архітектури у проектну діяльність Харківського науково-дослідного інституту технології машинобудування (акт впровадження від 9.12.2014 р.).

Особистий внесок здобувача. Основні наукові положення і результати, які викладені в дисертації, отримані автором особисто. У роботах, опублікованих у співавторстві, здобувачеві належать: [1] – визначення основних принципів організації обчислювальних процесів і їх взаємодії, які засновані на відображенні станів обчислень на графовій моделі програмної архітектури інформаційної системи; [2] – формалізований опис графової моделі програмної архітектури інформаційної системи; [4] – алгоритм відновлення результатів роботи обчислювальних процесів на основі контрольних точок відновлення, який застосовується в інформаційній технології структурного синтезу програмної архітектури; [5] – модель кінцевих автоматів, що забезпечує взаємодію програмних модулів та використовується в автоматному методі перевірки виконання обмежень до програмного забезпечення, що формується; [6] – інформаційна технологія, що забезпечує структурний синтез програмної архітектури інформаційної системи на основі заданої моделі предметної області, а також відображені допрацьовані методи об'єднання вершин ярусно-паралельної графової моделі і автоматний метод перевірки виконання обмежень до програмного забезпечення, що формується, які виражені у формі функціональних вимог.

Апробація результатів дисертації. Основні результати дослідження, наукові висновки і рекомендації доповідалися і обговорювалися на 15-му ювілейному молодіжному форумі «Радіоелектроніка і молодіж у XXI віці" (м. Харків, 18-20 квітня 2011 р.) [8], Міжнародній науково-технічній конференції «ИСТ-2012» (А.Р. Крим, смт. Морське, 22-29 вересня 2012 р.) [9], 15-ій Міжнародній конференції «Системний аналіз і інформаційні технології (SAIT'2013)» (м. Київ, 27-31 травня 2013 р.) [10], Міжнародній конференції «Інформаційні технології в управлінні (ИВС 2014)» (м. Санкт-Петербург, 21-28 травня 2014 р.) [11], Міжнародній науково-технічній конференції «ИСТ-2014» (м. Харків, 21-27 вересня 2014 р.) [12], Міжнародній науково-технічній конференції «ИСТ-2015» (м. Харків, 21-27 вересня 2015 р.) [13].

Публікації. За результатами дисертаційного дослідження опубліковано 13 наукових робіт: 7 статей (з них 2 одноосібно), у тому числі 5 статей у наукових фахових виданнях України з технічних наук; 2 статті у закордонних виданнях (серед них 3 статті опубліковано у виданнях, що включено до міжнародних наукометричних баз даних Research Bible, Open Academic Journals Index, Index Copernicus, Eurasian Scientific Journal Index, IndianScience, The Journals Impact Factor, Directory Indexing of International Research Journals); 6 публікацій у збірниках матеріалів і тез доповідей науково-технічних конференцій.

Структура дисертації. Дисертаційна робота складається зі вступу, 4 розділів, висновків, списку використаних джерел і додатків. Повний обсяг дисертації складає

191 сторінку (з них 139 сторінок – основного тексту), та містить: 38 рисунків і 14 таблиць по тексту, список використаних джерел з 159 найменувань (з них – 97 кирилицею, 62 – латиницею) на 18 сторінках. і 4 додатки на 34 сторінках.

ОСНОВНИЙ ЗМІСТ РОБОТИ

У вступі обгрунтовано актуальність теми дисертаційної роботи, визначено мету і задачі дослідження, сформульовано новизну і практичне значення отриманих результатів. Наведено відомості про впровадження результатів дисертаційної роботи, апробацію, особистий внесок здобувача та публікації.

У першому розділі виконано аналіз існуючих методів структурного синтезу і автоматизованого конфігурування програмного забезпечення інформаційної системи. На підставі проведеного аналізу встановлено, що розв’язання задачі кастомізації програмного забезпечення залишається актуальним, і вимагає розвитку існуючих інформаційних технологій шляхом використання формальних графових ярусно-паралельних моделей програмної архітектури інформаційної системи, які забезпечують кастомізацію програмного забезпечення в умовах вимог, що змінюються еволюційно.

У разі вимог кінцевого користувача, що змінюються, технології TDD, BDD вимагають додаткових часових та трудових витрат, оскільки програмні підсистеми або компоненти, що не відповідають цим вимогам, повинні бути заново переписані розробником. А такі технології як SOA і управління лініями програмних продуктів DSPL не можуть задовольнити високі вимоги до безпеки даних. Використання технологій гнучкої розробки програмного забезпечення, таких як Agile, також призводить до надмірних матеріальних витрат. У той же час не враховується застосування методів спрощення програмної архітектури інформаційної системи на основі аналізу графової моделі його архітектури з метою зниження витрат часу на проектування та кастомізацію.

Наявність цих недоліків, необхідність зниження витрат часу на розробку програмного забезпечення та його адаптацію до потреб кінцевого користувача, що змінюються еволюційно, обґрунтовує актуальність теми, мету та задачі дисертаційного дослідження.

У другому розділі вдосконалено ярусно-пареллельну графову модель програмної архітектури інформаційної системи, розроблено метод об'єднання вершин графової моделі на основі оцінки показників зчеплення і зв'язності, а також розроблено графовий метод оцінки функціональної складності програмного забезпечення інформаційної системи.

Графова модель являє собою сукупність вершин, яким зіставляються елементарні програмні модулі, що виконуються послідовно або незалежно один від одного. Орієнтовані дуги між ними визначають тип зв'язків за даними і описуються парою множин характеристик програмних модулів. Графова модель формується з надлишковою для заданої предметної області функціональністю і забезпечує гнучку конфігурацію системи в умовах вимог, що змінюються еволюційно.

Для побудови моделі потрібні дані специфікацій вимог до кінцевого

програмного продукту, інформація, що одержується на етапах передпроектних досліджень, існуючі архітектурні рішення, патерни. При цьому мається на увазі наявність існуючих програмних модулів і моделі предметної області. На підставі цієї інформації будується графова ярусно-паралельна модель програмної архітектури:

$$M = \left\langle V_{поч}, X_{поч}, D_{in}, D_{out}, V^{H}, X^{H}, V^{M}, X^{M}, VC, PM \right\rangle, \quad (1)$$

де $V_{поч}$ – множина вершин *v*, яким зіставляються програмні модулі (або ж, в загальному розумінні, обчислювальні процеси);

$X_{поч}$ – множина орієнтованих дуг $x_{ij} = (v_i, v_j)$ графу $G_{поч}$, що визначають залежність по даним між програмними модулями;

D_{in} – вектор вхідних даних $D_{in} = \left\langle d_1^{in}, d_2^{in}, ...d_k^{in} \right\rangle$, що виділені для кожної *i*-ї вершини моделі (1) та є незмінними в процесі роботи програмного модуля;

D_{out} – вектор вихідних даних $D_{out} = \left\langle d_1^{out}, d_2^{out}, ...d_k^{out} \right\rangle$, які були модифіковані;

V^{H} – підмножина нових програмних модулів $V^{H} = \{ v_i^{н} \}$;

X^{H} – підмножина нових залежностей по даним $x_{ij}^{н} = (v_i^{н}, v_j^{н})$;

V^{M} – підмножина модифікованих модулів $V^{M} = \{ v_i^{м} \}$, причому підмножини V^{H}, V^{M} використовуються у разі вимог кінцевого користувача, що змінюються, та визначають модифікацію архітектури;

X^{M} – підмножина модифікованих залежностей по даним $x_{ij}^{м} = (v_i^{м}, v_j^{м})$;

VC – множина характеристик вершин $VC = \{ m_1^{VC}, m_2^{VC}, ..., m_n^{VC} \}$, де $m_i^{VC} = \langle isSubG, Comp \rangle$ – підмножина, що складається з двох елементів: *isSubG* – логічна змінна, яка визначає наявність вкладеного підграфу для супервершини, та підмножина *Comp* – перераховує номери вершин, що належать компоненті сильної зв'язності;

PM – множина функцій $PM = \{ p_1^{PM}, p_2^{PM}, ..., p_q^{PM} \}$, що зіставляються з вершинами.

Для моделі (1) визначається орієнтований граф:

$$G = F(G_{поч}), \quad (2)$$

де $F(G_{поч})$ – визначає множину операцій над початковою графовою моделлю, які дозволяють привести її до ярусно-паралельної форми з супервершинами, а $G_{поч}$ є початковим графом виду:

$$G_{поч} = (V_{поч}, X_{поч}). \quad (3)$$

Формування множини вершин $V_{поч}$ і дуг $X_{поч}$ графа (3) відбувається на основі виділення в предметної області відповідних функціональних підсистем об'єкта дослідження і встановлення потоків даних між ними.

Для отримання нової версії програмної архітектури інформаційної системи з графу (3) виключається підмножина вершин $V^{З} \subseteq V_{поч}$, яка замінюються підмножиною V^{M}, тобто прибираються програмні модулі застарілої версії. Така ж операція здійснюється з дугами $X^{З} \subseteq X_{поч}$. Це дозволяє визначити постійну частину графу G: $G^{П} = \left(V^{П}, X^{П}\right), G^{П} \subseteq G$, де $V^{П} = V_{поч} \setminus V^{З}$, а $X^{П} = X_{поч} \setminus X^{З}$.

Множина функцій *PM* графової моделі (1) визначається як підмножина елементів $p_i^{PM} = \langle UID, PName, PMName, PUPath, PDes, SPath \rangle$, які необхідні для опису програмних компонент або модулів, що виконують ці функції, а саме: *UID* – унікальний ідентифікатор обчислювального процесу; *PName* – ім'я процесу, що використовується компонувальником програмної архітектури при визначенні логічного зв'язку між вершиною графової моделі і програмним модулем; *PMName* – реальне ім'я програмного модуля (або бібліотеки); *PUPath* – фізична адреса файлу програмного модуля; *SPath* – фізична адреса файлу специфікації, яка визначає основні характеристики модуля; *PDes* – додаткові відомості про елемент. *PName* є синтаксичним унікальним ім'ям програмного модуля, а *PMName* – реальним фізичним (семантичним) ім'ям.

Графова модель, що отримана, має такі основні властивості. Для неї відсутні висячі вершини, для яких виконується умова:

$$\forall v \in V, \{ V' \subseteq V : |V'| > 1 : \exists v_i \in V' : deg^-(v_i) = 0 \}. \quad (4)$$

Для моделі відсутні ізольовані вершини, для яких виконується умова:

$$\forall v \in V, \{ \exists v_i \in V \,|\, deg^+(v_i) = deg^-(v_i) = 0 \}. \quad (5)$$

Для неї відсутні парні дуги:

$$\forall i, j \in \mathrm{N}, \{ v_i, v_j \in V \,|\, \exists! \, x_{ij} \in X \}, \quad (6)$$

і відсутні вершини з петлями:

$$\forall i \in \mathrm{N}, \{ v_i \in V \,|\, x_{ii} \notin X \}. \quad (7)$$

Крім цього, для ациклічної графової ярусно-паралельної моделі всі вершини, для яких заданий номер з урахуванням топологічного сортування і виконується умова

$$\begin{aligned} &\forall i, j \in \mathrm{N}, i < j, \{ \exists v_i, v_j \in V : \mu[v_i, v_j] \,|\, deg^-(v_i) = deg^+(v_i) = \\ &= deg^-(v_{i+1}) = deg^+(v_{i+1}) = \ldots = deg^-(v_j) = deg^+(v_j) = 1 \}, \end{aligned} \quad (8)$$

де $\mu[v_i, v_j]$ – маршрут від вершини v_i до вершини v_j, об'єднуються в одну супервершину. Призначенням запропонованої графової моделі є опис програмної архітектури, що формується, який дозволяє оцінити цю архітектуру до побудови програмного забезпечення і забезпечити динамічне конфігурування робочих місць користувачів, що є перевагою запропонованої моделі.

Запропоновано метод об'єднання вершин графової моделі програмної архітектури інформаційної системи на основі оцінки складності і зв'язності програмних модулів, який призначений для приведення графової моделі до виду (1) для забезпечення здійсненності властивостей (4) – (8) і для зменшення витрат часу на розробку системи шляхом об'єднання вершин в супервершини на базі модифікованого алгоритму Косарайю. Даний алгоритм додатково обробляє графову модель на основі комплексного критерію оцінки ефективності структури програмного забезпечення:

$$K_C = \frac{K_{заг}}{K'_{заг}} \cdot \frac{K'_{склад}}{K_{склад}}, \quad (9)$$

де $K'_{склад}$ та $K'_{заг}$ – значення середнього коефіцієнта складності і узагальненого

критерію, які отримані після оптимізації графової структури, а $K_{склад}$ і $K_{заг}$ – значення критеріїв до оптимізації. Середній коефіцієнт складності для всієї структури програмного засобу:

$$K_{склад} = \frac{1}{n} \cdot \sum_{i=1}^{n} C(i), \tag{10}$$

де $C(i)$ – це коефіцієнт системної складності, який розраховується за метрикою Д. Карда та Р. Гласса, а n – загальна кількість модулів. Узагальнений критерій дозволяє оцінити програмну архітектуру, грунтуючись на порівнянні топології графа з топологією дерева, так як така топологія є оптимальною за критерієм мінімізації кількості зв'язків між програмними модулями, і задається виразом:

$$K_{заг} = \frac{n \cdot (n-1) - 2 \cdot e_r}{n^2 \cdot (n-1) \cdot (n-2)} \cdot \sum_{i=1}^{n} K_{CP}(i) \cdot \sum_{i=1}^{n} K_{CH}(i), \tag{11}$$

де $K_{CH}(i) = \frac{1}{10} \cdot Ch(i)$ та $K_{CP}(i) = \frac{1}{8} \cdot (9 - Cp(i))$ – нормовані коефіцієнти зв'язності і зчеплення модуля m_i відповідно. Величина *Ch(i)* визначається відповідно до значущості зв'язності а *Cp(i)* вибирається щодо значущості зчеплення модулів.

Для критерію (9) задана характеристика h_{ij}, що показує необхідність об'єднання вершин v_i і v_j одна з іншою до однієї супервершини, якщо $h_{ij} = 1$. Та $h_{ij} = 0$, якщо інакше:

$$h_{ij} = \begin{cases} 1, K_C < 1 \\ 0, K_C \geq 1. \end{cases} \tag{12}$$

Метод об'єднання вершин графової моделі програмної архітектури інформаційної системи містить такі етапи.

Етап 1. Усунення топологічних некоректностей графової моделі: не повинно бути висячих (4) та ізольованих (5) вершин.

Етап 2. Отримання для кожного *i*-го програмного модуля кількості модулів, що викликаються (параметр $sfan_{OUT}$), кількості елементів і структур даних, що оновлюються *i*-м модулем (параметр fan_{OUT}) і значення коефіцієнта системної складності $C(i)$ за метрикою Д. Карда та Р. Гласса.

Етап 3. Розрахунок середнього коефіцієнта складності $K_{склад}$ (10).

Етап 4. Розрахунок значення критеріїв $K_{заг}$ (11) та $K_{склад}$ (10), і отримання значення комплексного критерію оцінки ефективності програмної архітектури (9) для заданої версії вимог до конфігурації системи.

Етап 5. Пошук компоненти сильної зв'язності для заданого графу на основі алгоритму Косарайю і об'єднання вершин знайденої компоненти до супервершини.

Етап 6. Пошук пари вершин за умовою (6) після того, як на графі не залишилося жодної компоненти сильної зв'язності (для будь-яких двох вершин v_i та v_j неможливо знайти двох одночасно існуючих орієнтованих шляхів $\mu[v_i, v_j]$ і $\mu[v_j, v_i]$).

Етап 7. Обчислення значення комплексного критерію оцінки ефективності програмної архітектури (9) для знайденої пари вершин.

Етап 8. Об'єднання пари вершин в одну супервершіну, якщо значення комплексного критерію строго менше одиниці, і повторення дій з етапу 3 до тих пір, поки значення комплексного критерію (9) не задовольнятиме конфігураційним вимогам. Завершення методу.

Для запропонованого методу проведено оцінку часової складності, яка співпадає з алгоритмом Косарайю, і відповідає величині $O(n)$. Перевагою запропонованого методу є те, що метод дозволяє зменшити часові витрати на конфігурування шляхом спрощення структури графової моделі, спираючись на її топологічні особливості і конфігураційні вимоги до системи.

Запропоновано графовий метод оцінки функціональної складності програмного забезпечення інформаційної системи на основі розрахунку FP-метрики кожного програмного модуля. Для цього кількості зовнішніх вводів ставиться у відповідність величина напівстепені заходу відповідної вершини графової ярусно-паралельної моделі програмної архітектури. Кількості зовнішніх виводів співвідноситься величина напівстепені виходу вершини графа. Кількість внутрішніх логічних файлів або логічних груп даних, які є частиною бази даних або окремим файлом, розраховується на основі множини D_{in}.

Для розрахунку кількості зовнішніх інтерфейсних файлів використовується множина D_{out}. При отриманні значень двох останніх інформаційних характеристик також додатково використовується елемент *PDes* підмножини p_i^{PM} елементів опису програмних компонент, що містить додаткові відомості про програмний модуль. Множина вхідних та вихідних даних D_{in} та D_{out} використовуються для визначення типів даних, з якими працюють програмні модулі, що зіставляються вершинам графової моделі.

У рамках запропонованого методу у випадку використання графової ярусно-паралельної моделі програмної архітектури інформаційної системи для кожної вершини цієї моделі необхідно зберігати розраховану FP-метрику, значення якої використовується при формуванні підграфа або графа нової функціональної конфігурації. Загальна оцінка функціональної складності програмного забезпечення ґрунтується на сумі FP-метрик всіх компонентів, які входять до нього:

$$FS_{curr} = \sum_{i=1}^{n} FP_i , \tag{12}$$

де n – загальна кількість вершин графа або підграфа, а величина FP_i розраховується за критерієм оцінки функціональної складності за формулою, яка запропонована А. Албрехтом: $FP = TotalPoints \cdot \left(0,65 + 0,01 \cdot \sum_{i=1}^{14} F_i \right)$, де *TotalPoints* – загальна оцінка, що отримується при розрахунку FP-метрики на основі рангу і оцінки складності зовнішніх вводів і виводів, запитів, внутрішніх логічних і зовнішніх інтерфейсних файлів для поточного програмного модуля, а F_i – коефіцієнти регулювання складності, що приймають значення в залежності від характеристики системних параметрів програмного модуля.

Величини *TotalPoints* та F_i обчислюються відповідно до стандартного

алгоритму розрахунку FP-метрик і дозволяють оцінити функціональну складність програмних модулів обраного підграфа або графа в цілому.

При врахуванні нефункціональних вимог до програмних модулів на кожному ярусі орієнтованого графа вирішується задача підбору програмних модулів з мінімальним значенням функціональної складності, тобто:

$$FSL_k = \sum_{i=1}^{m} FP_i, \tag{13}$$

$$FSL_k \to min, \tag{14}$$

де m – загальна кількість вершин на n-му поточному ярусі ярусно-паралельної форми графової моделі.

Виходячи з цього, загальна функціональна складність для всієї графової моделі програмної архітектури інформаційної системи розраховується таким чином:

$$FS_{curr} = \sum_{k=1}^{p} FSL_k, \tag{15}$$

$$FS_{curr} \to min, \tag{16}$$

де p – загальна кількість ярусів заданої ярусно-паралельної графової моделі.

Запропонований метод оцінки функціональної складності програмного забезпечення інформаційної системи дозволяє сформувати програмну архітектуру з мінімальним значенням функціональної складності на базі графової моделі (1) і містить такі етапи.

Етап 1. Формування підграфа задачі користувача для графу (2).

Етап 2. Формування підмножин вершин $V^M = \{ v_i^M \}$ і $V^H = \{ v_i^н \}$, які будуть заміщати або додаватися до існуючої моделі програмної архітектури інформаційної системи (1) для відповідного ярусу цієї моделі.

Етап 3. Розрахунок FP-метрики (12) для множини вершин V, яке сформоване з урахуванням підмножин V^M та V^H поточного ярусу.

Етап 4. Розрахунок загальної функціональної складності (13) для поточного ярусу графової моделі (1).

Етап 5. Перевірка відповідності значення показника (13) функціональним вимогам користувача (14). При негативній відповіді повтор дій з етапу 2.

Етап 6. При задовільному значенні показника (13) перехід на новий ярус і повтор з етапу 2.

Етап 7. На завершальній стадії обробки всіх ярусів графової моделі розрахунок показника функціональної складності всього підграфа задачі користувача (15).

Етап 8. При незадовільному (16) значенні показника (15) повернення на перший ярус графової моделі і повтор формування підмножин нових вершин з етапу 2. У разі неможливості здійснити подальшу мінімізацію функціональної складності програмного забезпечення – перехід на етап 9.

Етап 9. Компіляція сформованої програмної архітектури інформаційної системи на основі оновленої графової моделі (1), завершення методу.

На базі запропонованого методу для поточної версії вимог кінцевого користувача виконується конфігурування функціональності робочого місця з

урахуванням оцінки функціональної складності окремих програмних модулів і програмного забезпечення в цілому. Для визначення часової складності запропонованого методу береться до уваги кількість вершин графової моделі *n*, так як кількість функціональних показників FP розраховується для кожної вершини. А також враховується кількість ярусів для ярусно-паралельної форми графової моделі. Оскільки, у крайньому випадку, число таких ярусів також дорівнює *n*, то оцінка часової складності запропонованого методу відповідає величині $O(n^2)$.

Перевагою запропонованого методу є те, що метод дозволяє отримати програмну архітектуру інформаційної системи з найменшою допустимою функціональної складністю, дозволяє оцінити відповідність програмного забезпечення, що формується, функціональним вимогам користувача і забезпечити підтримку робочих версій програмного забезпечення.

У третьому розділі запропоновано автоматний метод перевірки виконання обмежень до програмного забезпечення, що формується, які виражені у формі нефункціональних вимог. Метод використовується для проектування виконувача обчислювальних процесів за допомогою моделювання сценаріїв взаємодії програмних модулів і порівняння результатів моделювання з нефункціональними вимогами до програмного забезпечення. Вхідними даними методу є графова ярусно-паралельна модель програмної архітектури інформаційної системи, для якої визначені вершини, що відповідають за формування діалогу користувача і за формування відповідних їм підграфів модулів, що реалізують пов'язані задачі системи.

Метод містить такі етапи.

Етап 1. Виділення базових станів компонентів програмного забезпечення і умов переходів на основі вхідних та вихідних даних вершин графової моделі.

Етап 2. Формування структури керуючого кінцевого автомата і його вкладених підавтоматів, що реалізують поведінку системи, з урахуванням даних специфікацій вершин графової моделі.

Етап 3. Визначення принципів взаємодії кінцевих автоматів з програмними модулями на основі архітектурних шаблонів об'єктно-орієнтованого програмування для організації розподіленого або паралельного виконання програми.

Етап 4. Формування початкової інформації, яка необхідна для організації роботи кінцевих автоматів, такої як матриці суміжності графової ярусно-паралельної моделі програмної архітектури і транспонованої матриці суміжності для керування прямим і зворотним ходом обчислювальних процесів.

Етап 5. Перевірка здійсненності функцій програмного забезпечення, відмовостійкості в умовах апаратних обмежень і порівняння з вимогами кінцевого користувача сценаріїв взаємодії модулів, що моделюються.

Етап 6. Розробка програмної архітектури інформаційної системи з урахуванням взаємозв'язків графової моделі і керуючих кінцевих автоматів.

Етап 7. Генерація програмного забезпечення на основі перевіреної програмної архітектури, завершення методу.

Для кінцевих автоматів визначено та позначено стани (табл. 1) та події, за

якими кінцевий автомат переходить в ці стани (табл. 2). Для реалізації задач прямого (автомат AForwardSM) і зворотного (автомат AbackwardSM) управління ходом обчислювальних процесів запропоновано використовувати основний керуючий кінцевий автомат, що забезпечує взаємодію двох вкладених підавтоматів. Запропоновані автоматні моделі мають такі обов'язкові властивості, що відповідають вимогам до поведінки програмного засобу.

Таблиця 1 – Переходи кінцевого автомату

Позначення	Найменування
e1	отримано дозвіл
e5	помилка виконання
e6	номер вершини менше максимального для поточного ярусу
e7	номер поточного ярусу менше кількості ярусів
e16	задано умови і параметри обчислювального процесу
e43	дані зчитано
e81/ e82	архів існує / відсутній
e101/ e102	дані архіву не відновлювалися / відновлюються
e103	дані архіву відновлено
e121/e122/e123	обчислювальний процес не запущений / виконується / виконано

Таблиця 2 – Стани кінцевого автомату

Позначення	Найменування
o3.zf22	ініціалізація обчислювального процесу для автомата AForwardSM
o3.zf24	перехід до наступного обчислювального процесу
o3.zf28/ o3.zf31	запуск обчислювального процесу / очікування його завершення
o3.zf35/ o3.zf36	архівація даних / отримання даних
o2.zf45	відновлення даних для автомата ABackwardSM

Для автомата AForwardSM це:

– обчислювальний процес $P(v_i)$ для заданої i-ї вершини буде запущений тільки за запитом користувача ($\neg(\neg e1 \boldsymbol{U} o3.zf\,28)$);

– процес $P(v_i)$ обов'язково будь-коли буде запущений, якщо не буде помилок обчислень ($\neg(e5 \boldsymbol{U} o3.zf\,28)$);

– процес $P(v_i)$ буде запущено тільки тоді, коли вершина знаходиться в стані «не запущено» або «виконано» ($\neg((\neg e121 \wedge \neg e123) \boldsymbol{U} o3.zf\,28)$);

– вершина v_i коли-небудь обов'язково буде знаходитися в стані «не запущено» або «виконано» ($\boldsymbol{F} o3.zf\,31$);

– для кожної необхідної вершини графової моделі буде виконано процес архівації даних ($e43 \boldsymbol{F} o3.zf\,35$);

– для послідовності обчислювальних процесів на основі існуючої ярусно-паралельної графової моделі можна виділити такий обчислювальний процес, який стає активним лише тоді, якщо всі попередні процеси закінчать свою роботу ($F_{общ}^{AV} = \neg(F_1^{AV} \wedge F_2^{AV} \wedge ... \wedge F_n^{AV}) = \neg F_1^{AV} \vee \neg F_2^{AV} \vee ... \vee \neg F_n^{AV}$, де $F^{AV} = \neg(\neg e122 \boldsymbol{U} o3.zf\,31) \wedge$ $\wedge \neg(e121 \boldsymbol{U} o3.zf\,22) \boldsymbol{U} (\neg(\neg e16 \boldsymbol{U} o3.zf\,28))$;

– для всієї множини обчислювальних процесів програмної системи можна виділити таку підмножину обчислювальних процесів, які розміщені на одному ярусі ярусно-паралельної графової моделі і не мають інформаційних залежностей один від одного, що дозволяє запускати їх одночасно $(F_{общ}^{PV} = \neg(F_1^{PV} \wedge F_2^{PV} \wedge ... \wedge F_n^{PV}) = \neg F_1^{PV} \vee \neg F_2^{PV} \vee ... \vee \neg F_n^{PV}$, де $F^{PV} = \neg((\neg e6 \vee \neg e7)\boldsymbol{U} o3.zf\,28\boldsymbol{U}$ $\boldsymbol{U}(\neg(\neg e122\boldsymbol{U}(o3.zf\,24 \wedge o3.zf\,36))))$.

Для автомата ABackwardSM це:

– відновлення даних буде виконано тільки на вимогу користувача $(\neg(\neg e1\boldsymbol{U} o2.zf\,45))$;

– відновлення даних буде виконано тільки при наявності архіву даних для кожної вершини підграфу обчислювального процесу $(\neg(\neg e82\boldsymbol{U} o2.zf\,45) \vee \neg(\neg e81\boldsymbol{U} o2.f\,45))$;

– для кожної вершини виділеного підграфу відновляться дані тільки в стані, якщо інший користувач не задіяв цю вершину для себе (статус «не запущено») і якщо відновлення даних не відбувалося зовсім (статус «не відновлено»), або дані вже були відновлені раніше (статус «відновлено») $(\neg((\neg e121 \vee \neg e101 \wedge \neg e103)\boldsymbol{U}$ $\boldsymbol{U} o2.zf\,45) \vee \neg((e122 \wedge e102)\boldsymbol{U} o2.f\,45))$.

Наведені властивості автоматних моделей формалізовані на базі темпоральної логіки LTL і доповнюються новими у разі появи наступних версій нефункціональних вимог. Оцінка часової складності автоматного методу відповідає величині $O(n^3 \cdot \log k)$, що розрахована на основі часової складності алгоритму пошуку шляхів фіксованої довжини на графі, де n – кількість станів кінцевого автомата, а змінна k – кількість переходів в ці стани. Сформульовані властивості є основою для визначення правил виконання обчислювальних процесів, тому автоматна модель, що розроблена відповідно до запропонованого методу, дозволяє сформувати обмеження до програмної архітектури з урахуванням змінних вимог, підвищити надійність функціонування програмного забезпечення, спираючись на дані графової моделі програмної архітектури інформаційної системи, дозволяє оцінити можливість розгортання програмного забезпечення інформаційної системи для існуючої конфігурації апаратних засобів, що відрізняє метод від технології BDD.

У четвертому розділі запропоновано інформаційну технологію структурного синтезу програмної архітектури інформаційної системи в умовах вимог, що змінюються, а також наведено результати впровадження моделі, методів та інформаційної технології при вирішенні практичних задач за допомогою розробленого програмного інструментарію «SW Design».

Запропонована технологія дозволяє відобразити вимоги поточної версії на графовій моделі і забезпечити переконфігурування модулів програмного забезпечення, а також знизити його функціональну і структурну складність.

Технологія містить такі етапи.

Етап 1. Формування початкової графової моделі програмної архітектури інформаційної системи.

Етап 2. Об'єднання вершин графової моделі і отримання графової ярусно-

паралельної моделі.

Етап 3. Формування мета-файлу специфікацій модулів.

Етап 4. Формування мета-файлу специфікації програмного проекту.

Етап 5. Формування програмної архітектури відповідно до графової моделі.

Етап 6. Перевірка виконання обмежень до програмного забезпечення, що формується.

Етап 7. Оцінка функціональної складності програмного забезпечення.

Етап 8. Компіляція програмного забезпечення, завершення методу.

Перший етап інформаційної технології (рис. 1) забезпечується інформацією з доступних документів, що описують вимоги замовника: первинні специфікації, діаграми статичного і динамічного моделювання предметної області. На основі функціональних і архітектурних вимог до програмного забезпечення для сутностей моделі предметної області визначаються функції об'єктів, які формують графову модель, яка на другому етапі перевіряється на топологічну коректність. На третьому етапі на основі вимог замовника виконується метод об'єднання вершин графової моделі і формується раціональна структура графа. На четвертому етапі на базі отриманої структури відбувається опис програмних модулів за допомогою мета-файлів специфікацій. На п'ятому етапі з використанням архітектурних шаблонів формується необхідний для генерації програмного забезпечення та динамічного конфігурування мета-файл програмного проекту, який описує принципи взаємодії модулів. Даний файл на шостому етапі є вхідним для процесу перевірки виконання обмежень до програмного забезпечення на основі верифікації послідовностей взаємодії модулів. На сьомому етапі, після успішного зменшення функціональної складності програмного забезпечення виконується його генерація та відчуження графової моделі для кінцевого користувача сформованої системи. У разі нової версії вимог процес повторюється з третього етапу.

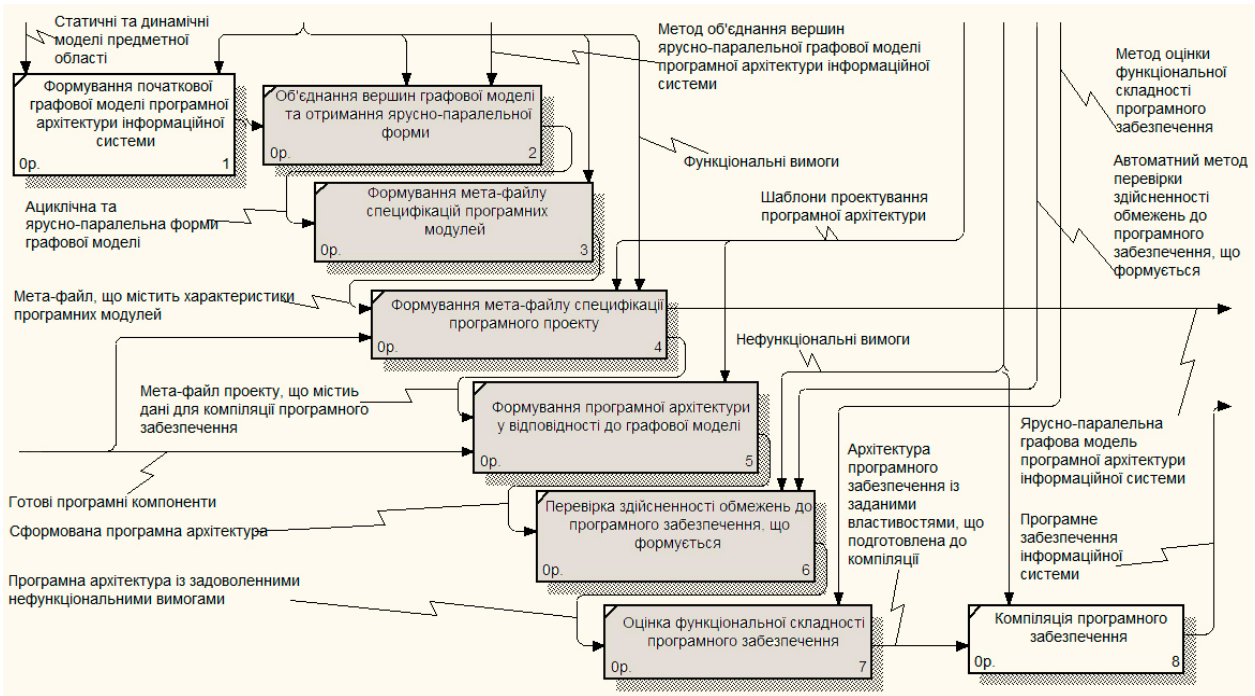

Рисунок 1 – Етапи інформаційної технології структурного синтезу програмної архітектури інформаційної системи

Інформаційна технологія дозволяє здійснити структурний синтез програмної архітектури інформаційної системи і динамічно конфігурувати архітектуру і функціональність робочих місць в умовах вимог кінцевого користувача, що змінюються, а також знизити витрати на реалізацію функціональних завдань інформаційної системи шляхом зниження функціональної і структурної складності.

Впровадження отриманих результатів виконано за допомогою розробленого інструментарію «SW Design» з архітектурою стандарту IEEE P1484.1 / D11. Було виконано генерацію програмного забезпечення «Автоматизована система підготовки виробництва складних електронних пристроїв» за наданою інформацією про предметну область. Результати отримано на основі початкових даних задачі проектування виробничого процесу складних виробів електронної техніки для державного підприємства «Харківський науково-дослідний інститут технології машинобудування». Сформовану графову модель, що складається з 14 вершин, було взято за основу аналізу процесів динамічного налаштування та відновлення даних обчислювальних процесів.

Для кожної вершини графової моделі задано фіксований час виконання 10 мс процесорного часу. Ситуація зміни вимог моделювалася шляхом додавання вершини: випадково вибирався підграф модулів системи, для якого таким же чином вибиралася вершина, після якої була додана нова. Після додавання вершини повторно проводилася часова оцінка стандартних і запропонованих методів та відновлення результатів виконання обчислювальних процесів. При цьому враховувався відсоток помилок в інформації, що вводиться користувачами системи. Відсоток ручної настройки системи, яка необхідна для динамічного конфігурування, розрахований на основі LOC-оцінок програмного коду у порівнянні зі стандартним методом конфігурування системи на основі даних звіту по впровадженню ERP-систем у виробництві за 2015 рік, згідно з яким більшість підприємств (в середньому 44 % підприємств усіх галузей промисловості і 38 % підприємств сфери машинобудування) вимагають від 26 % до 50 % змін вихідного коду відповідно до настройок і вимог замовників системи (рис. 2).

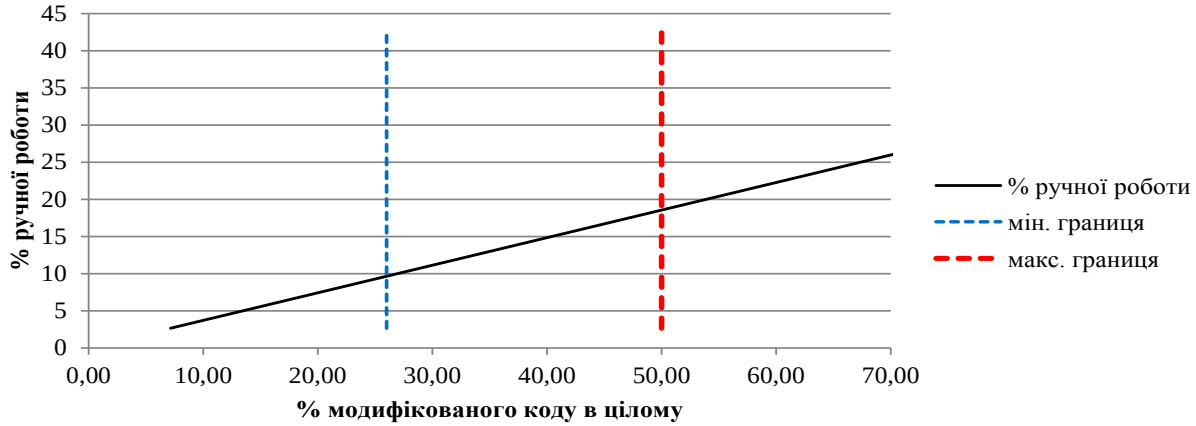

Рисунок 2 – Частка ручної настройки системи щодо відсотка зміни оригінального програмного коду

Запропонована технологія дозволила зменшити відсоток ручної роботи персоналу до 10 % для модифікації оригінального коду традиційним засобом у

розмірі 26 % і до 18 % для відповідної межі в 50 %. Тобто запропонована інформаційна технологія знижує часові витрати на переконфігурування програмного забезпечення на 36 %.

Результати порівняння (табл. 1) описуються такими показниками: $L_{заг}$ – відсоток зміни програмного коду; $L_{руч}$ – частка ручної настройки системи; $T_{сер}^{тр}$ – середній час виконання обчислювальних процесів (традиційний підхід); $T_{сер}$ – середній час виконання (із застосуванням запропонованої інформаційної технології); P_k – k-ий відсоток помилок у початкових даних.

Таблиця 1 – Порівняльні дані результатів виконання обчислювальних процесів з використанням стандартної і запропонованої технологій

№ вершини	$L_{заг}$, **%**	$L_{руч}$, **%**	$T_{сер}^{тр}$, **мс**	$T_{сер}$, **мс**				
				P_{50}	P_{40}	P_{30}	P_{20}	P_{10}
1	7,1	2,7	1546	2752	2511	2413	2705	2766
2	14,3	5,3	2937	8543	8360	8543	8543	8543
3	21,4	7,9	4509	10252	9783	9838	9649	9498
4	28,6	10,6	6023	11918	11195	11221	10683	10434
5	35,7	13,3	7863	13773	12687	12481	11774	11320
6	42,9	15,9	9667	15446	14134	13670	12739	12327
7	50,0	18,6	11087	17285	15567	14916	13764	13281
8	57,1	21,2	12046	18960	17026	16172	14832	14157
9	64,3	23,9	14794	20703	18500	17384	15971	15137
10	71,4	26,5	15989	22430	19973	18642	16995	16418
11	78,6	29,2	17358	24352	21710	19858	18172	17420
12	85,7	31,8	19557	26112	23164	21039	19247	18366
13	92,7	34,5	20399	27946	24599	22298	20249	19301
14	100,0	37,1	21685	29796	25893	23605	21252	20218
15	107,1	39,8	22924	31511	27351	24801	22320	21166
16	114,3	42,4	24892	33390	28743	26044	23393	22047

Аналіз даних (рис. 3) показує, що запропонована інформаційна технологія має переваги при значному ускладненні існуючої графової моделі, більше 12-ти додаткових функцій, що представлені вершинами графової моделі.

При цьому у початкових даних виявляється невеликий відсоток помилок, менше 20 %.

Використання запропонованої інформаційної технології дозволило поліпшити швидкодію програмної системи при зміні вимог до кінцевого програмного продукту у разі необхідності відновлення результатів роботи обчислювальних процесів в порівнянні зі стандартними методами резервування і відновлення даних в 1,12 рази для 10 % помилок, і в 1,1 рази для 20 % помилок на графовій моделі, що складається з початкової множини функціональних завдань, яка доповнена 16-ма вершинами.

Визначення принципів управління, відновлення даних і синхронізації

обчислювальних процесів в паралельних обчислювальних середовищах, а також методів формування гнучкої програмної архітектури з її адаптацією для сервіс-орієнтованої та хмарної архітектур є предметом подальших досліджень.

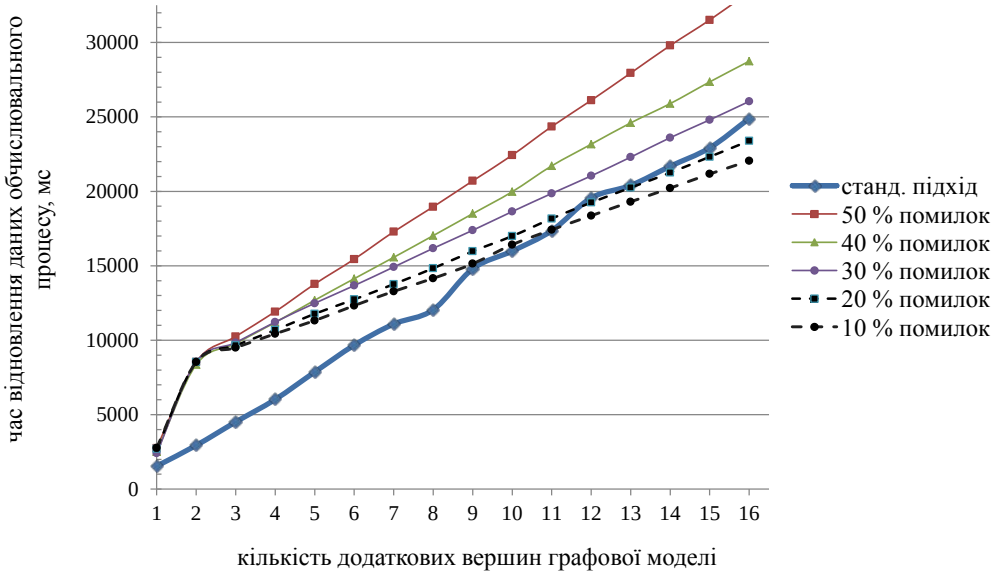

Рисунок 3 – Графік часу виконання обчислювальних процесів програмного забезпечення

У додатках наведено акти про впровадження результатів дисертаційної роботи у виробництво, текст базових програмних класів роботи з графовою моделлю та код програми верифікатора мовою PROMELA.

ВИСНОВКИ

Основні наукові та практичні результати роботи полягають у тому, що вирішено актуальну науково-практичну задачу розробки моделі, методів та інформаційної технології структурного синтезу програмної архітектури інформаційної системи в умовах вимог кінцевого користувача, що змінюються. При цьому отримані такі результати:

1. Аналіз методів структурного синтезу і кастомізації програмного забезпечення інформаційної системи показав, що сучасні методи не задовольняють вимогам до ефективної адаптації програмного забезпечення під змінні в часі вимоги кінцевого користувача, що підтверджує актуальність вирішення задачі розробки ефективних формальних підходів до кастомізації програмного забезпечення інформаційної системи шляхом використання ярусно-паралельних графових моделей програмної архітектури.

2. Удосконалено ярусно-паралельну графову модель програмної архітектрури шляхом врахування взаємозв'язків по вхідним і вихідним даним між програмними модулями інформаційної системи. Програмні модулі в моделі представляються вершинами графа. Спрямовані дуги графа відображають зв'язок за даними між програмними модулями. Доповнення моделі зв'язками за даними забезпечує можливість динамічного конфігурування програмної архітектури при зміні

функціональних вимог до системи.

3. Вперше запропоновано метод об'єднання вершин графової ярусно-паралельної моделі програмної архітектури на основі оцінки взаємозв'язків між цими вершинами по вхідним та вихідним даним. Метод включає етапи оцінювання показників зчеплення і зв'язності вершин моделі, а також об'єднання вершин на основі отриманих значень показників. Метод дозволяє зменшити часові витрати на конфігурування програмного забезпечення при зміні вимог і, тим самим, знизити витрати на конфігурування.

4. Вперше запропоновано автоматний метод перевірки виконання обмежень до програмного забезпечення, що формується, у вигляді нефункціональних вимог. Метод включає етапи формування автоматної моделі взаємодії модулів і порівняння отриманих в результаті моделювання послідовностей взаємодії модулів з нефункціональними вимогами до програмного забезпечення. Метод дозволяє сформувати обмеження на структуру програмного забезпечення, що враховують нефункціональні вимоги, і, тим самим, підвищити надійність його функціонування.

5. Запропоновано метод оцінки функціональної складності програмного забезпечення шляхом використання графової ярусно-паралельної моделі програмної архітектури інформаційної системи. Метод включає в себе етапи визначення функціональних характеристик програмних модулів і розрахунку критеріїв функціональної складності, що дозволяє оцінити відповідність програмного забезпечення функціональним вимогам користувача.

6. Запропоновано інформаційну технологію структурного синтезу програмної архітектури інформаційної системи. Технологія включає етапи синтезу графовой моделі програмної архітектури, конфігурування архітектури на базі об'єднання вершин графа і перевірки виконання функціональних вимог на основі відповідного автоматного методу. Запропонована технологія дозволяє значно знизити вартість змін програмного забезпечення.

7. Проведено впровадження моделі, методів та інформаційної технології при вирішенні практичних задач за допомогою розробленого інструментарію «SW Designer». Результати роботи були впроваджені в проектній діяльності ПАТ «Інститут автоматизованих систем», а також для вирішення завдань проектування виробничого процесу складних виробів електронної техніки на базі Харківського науково-дослідного інституту технології машинобудування, що підтверджено відповідними актами впровадження. Запропонована інформаційна технологія дозволяє знизити часові витрати на конфігурування програмного забезпечення на 36 %.

СПИСОК ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

1. Чайников С. И. Принципы организации вычислений на базе граф-модели предметной области / С. И. Чайников, А. С. Солодовников // Бионика интеллекта: науч.-техн. журнал. – 2012. – № 2 (79). – С. 72-75.

2. Чайников С. И. Формализованное описание граф-модели предметной области / С. И. Чайников, А. С. Солодовников // Бионика интеллекта: науч.-техн.

журнал. – 2013. – № 1 (80). – С. 77-81.

3. Солодовников А. С. К вопросу оценивания эффективности и сложности структуры программного средства / А. С. Солодовников // Проблеми інформаційних технологій. – 2014. – № 2 (16). – С. 125-129. (Входить до міжнародних наукометричних баз: Open Academic Journals Index, Research Bible).

4. Чайников С. И. К вопросу организации контрольных точек восстановления данных вычислительных процессов / С. И. Чайников, А. С. Солодовников // Бионика интеллекта: науч.-техн. журнал. – 2014. – № 2 (83). – С. 128-131.

5. Чайников С. И. Организация вычислительных процессов для заданной предметной области с использованием модели конечных автоматов / С. И. Чайников, А. С. Солодовников // Системи обробки інформації. – 2016. – № 2 (139). – С. 132-137. (Входить до міжнародних наукометричних баз: Index Copernicus, CiteFactor).

6. Chainikov, S., Solodovnikov, A. Information Technology of Software Architecture Structural Synthesis of Information System. EUREKA: Physical Science and Engineering. – 2016. – 4 (5). – P. 25-32. (Входить до міжнародних наукометричних баз: Journalindex, Citefactor, Open Academic Journals Index, Eurasian Scientific Journal Index, IndianScience).

7. Solodovnikov, A. Developing Method for Assessing Functional Complexity of Software Information System. EUREKA: Physical Science and Engineering. – 2016. – 5 (6). – P. 3-9. (Входить до міжнародних наукометричних баз: Journalindex, Citefactor, Open Academic Journals Index, Eurasian Scientific Journal Index, IndianScience).

8. Чайников С. И. Об одном подходе к методике разработки сложных диалоговых систем / С. И. Чайников, А. С. Солодовников // Материалы 15-го юбилейного Международного молодежного форума "Радиоэлектроника и молодежь в XXI веке" 18-20 апреля 2011 г. – Харьков: ХНУРЭ – 2011. – Том 9. – С. 160-161.

9. Чайников С. И. Использование граф-модели предметной области при разработке программных средств / С. И. Чайников, А. С. Солодовников // Информационные системы и технологии: материалы международной научн.-техн. конф., Морское-Харьков, 22-29 сентября 2012 г. – Харьков: НТМТ – 2012. – С. 66.

10. Чайников С. И. Организация обмена данными между вычислительными процессами в диалоговой системе / С. И. Чайников, А. С. Солодовников // Системный анализ и информационные технологии (SAIT’2013): материалы 15-ой Международной конференции, 27-31 мая 2013 г. – Киев.: «НТУ КПИ». – 2013. – С. 493-494.

11. Солодовников А. С. Об одном подходе к управлению вычислительными процессами в проблемно-ориентированных информационных системах / А. С. Солодовников // Информационные технологии в управлении (ИТУ-2014): материалы конференции. – Санкт-Петербург.: ОАО «Концерн «ЦНИИ «Электроприбор». – 2014. – С. 245-247.

12. Чайников С. И. Оптимизация топологии модели предметной области и оценка эффективности структуры программного средства / С. И. Чайников, А. С. Солодовников // Информационные системы и технологи (ИСТ 2014):

материалы 3-й Международной научн.-техн. конференции, Харьков, 15-21 сентября 2014 г. – Харьков: ХНУРЭ, 2014. – С. 101-103.

13. Солодовников А. С. Синтез проблемно-ориентированного программного комплекса на основе модели предметной области (по материалам диссертационного исследования) / А. С. Солодовников, С. И . Чайников // Информационные системы и технологии (ИСТ 2015): материалы Международной научн.-техн. конференции, 21 - 27 сентября 2015 г. – Харьков: НТМТ, 2015. – С. 108-109.

АНОТАЦІЯ

Солодовніков А. С. Інформаційна технологія синтезу програмної архітектури на основі графової моделі. – Рукопис. Дисертація на здобуття наукового ступеня кандидата технічних наук за спеціальністю 05.13.06 – інформаційні технології. – Харківський національний університет радіоелектроніки, Міністерство освіти та науки України, Харків, 2016.

Дисертаційна робота присвячена вирішенню науково-практичного завдання – розробці моделі, методів та інформаційної технології структурного синтезу програмної архітектури інформаційної системи в умовах вимог, що змінюються. У роботі удосконалено графову ярусно-паралельну модель програмної архітектури інформаційної системи, яка використовується для її адаптації до вимог кінцевого користувача, що змінюються еволюційно.

На основі графової ярусно-паралельної моделі запропоновано методи зниження структурної і функціональної складності програмного забезпечення інформаційної системи, до яких відносяться метод об'єднання вершин ярусно-паралельної графової моделі програмної архітектури та метод оцінки функціональної складності програмного забезпечення інформаційної системи. Також запропоновано автоматний метод перевірки виконання обмежень до програмного забезпечення, що формується.

На основі розроблених методів запропонована інформаційна технологія та програмне забезпечення структурного синтезу програмної архітектури інформаційної системи.

Ключові слова: інформаційна технологія, графова модель, структурний синтез, структурна складність, функціональна складність, кінцевий автомат, змінні вимоги, темпоральная логіка.

АННОТАЦИЯ

Солодовников А. С. Информационная технология синтеза программной архитектуры информационной системы. – Рукопись. Диссертация на соискание ученой степени кандидата технических наук по специальности 05.13.06 – информационные технологии. – Харьковский национальный университет радиоэлектроники, Министерство образования и науки Украины, Харьков, 2016.

Диссертация посвящена решению научно-практической задачи – разработке модели, методов и информационной технологии структурного синтеза программного обеспечения информационной системы в условиях изменяющихся

требований. В работе усовершенствована графовая ярусно-параллельная модель программной архитектуры информационной системы, которая используется для адаптации программной архитектуры к эволюционно изменяющимся требованиям конечного пользователя.

На основе графовой ярусно-параллельной модели предложен метод объединения вершин графовой ярусно-параллельной модели программной архитектуры. Метод отображает программные модули на основе последовательного расчета показателей сцепления и связности модулей. Метод позволяет уменьшить структурную сложность модели и тем самым снизить временные затраты на адаптацию программной архитектуры к изменяющимся во времени требованиям конечного пользователя. Также предложен графовый метод оценки функциональной сложности программного обеспечения информационной системы. Метод включает в себя этапы определения функциональных характеристик программных модулей и расчета критериев функциональной сложности для выделенных подграфов функциональных задач, что позволяет повысить эффективность конфигурирования функциональности рабочих мест пользователей информационной системы и оценить соответствие формируемого программного обеспечения функциональным требованиям пользователя.

Также усовершенствован автоматный метод проверки выполнения ограничений к формируемому программному обеспечению, выраженных в форме нефункциональных требований, который включает в себя этапы синтеза автоматной модели взаимодействия программных модулей, проверки выполняемости функций программного обеспечения, отказоустойчивости в условиях аппаратных ограничений и сравнения с требованиями к моделируемым сценариям взаимодействия программных модулей. Данный метод позволяет оценить возможность развертывания программного обеспечения информационной системы для существующей конфигурации аппаратных средств.

На основе разработанных методов предложена информационная технология и программное обеспечение структурного синтеза программной архитектуры информационной системы. Информационная технология включает в себя этапы синтеза графовой ярусно-параллельной модели архитектуры программного обеспечения, конфигурирования архитектуры с учетом текущих функциональных требований и проверки выполнения нефункциональных требований на основе соответствующего автоматного метода, что позволяет снизить функциональную и структурную сложность программного обеспечения и тем самым снизить затраты на реализацию функциональных задач информационной системы с учетом требований конечных пользователей. Экспериментальные исследования подтвердили корректность предложенной модели, методов и информационной технологии. Полученные теоретические результаты реализованы при разработке программного обеспечения «SW Design».

Ключевые слова: информационная технология, графовая модель, структурный синтез, структурная сложность, функциональная сложность, конечный автомат изменяющиеся требования, темпоральная логика.

ABSTRACT

Solodovnikov, A. Information technology of the synthesis of information system software architecture based on graph models. – Manuscript. The thesis for the degree of candidate of technical sciences, specialty 05.13.06 – information technology. - Kharkiv National University of Radio Electronics, The Ministry of Education and Science of Ukraine, Kharkiv, 2016.

The thesis is devoted to solving scientific and practical task – developing model, methods and informational technology of structural synthesis of information system software architecture in terms of changing end-user requirements. The work improved multilevel graph model of information system software architecture that is used to adapt the software architecture to the changeable requirements of the end-user.

Based on the multilevel graph model of information system software architecture, author proposed methods for reducing the structural and functional complexity of information system software architecture, including the method of union nodes of graph model and method for assessing the functional complexity of software architecture.

Also author proposed method of automatic verification of constraints to software that is formed.

Based on the developed methods, author proposed information technology and instrumental software for structural synthesis of information system software architecture.

Keywords: information technology, graph model, structural synthesis, structural complexity and functional complexity, finite state machine, changeable requirements, temporal logic.