УДК 681.518:681.327.8

Є.В. Буров
НУ "Львівська політехніка",
кафедра "Інформаційні системи та мережі"

СИСТЕМА ФОРМАЛЬНИХ СПЕЦИФІКАЦІЙ МЕРЕЖІ СЕРВІСІВ ТА ПРОЦЕСОРІВ ДЛЯ ПРОЕКТУВАННЯ РОЗПОДІЛЕНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

Actual trends in information technologies area present new challenges to system automation design tools and concepts. This paper propose an methodology and formal specification system to assist information design process on services and processors structure definition design stage.

Запропоновано систему формальних специфікацій для автоматизації проектування розподілених інформаційних систем на стадії проектування мережі сервісів та процесорів. Ця система може бути покладена в основу програмних засобів автоматизації проектування.

Зростання складності інформаційних систем, появa нових типів інформаційних потоків та ускладнення вимог щодо їх передавання покликло до життя нові технології контролю та забезпечення якості передавання. Усе це вимагає створення нових засобів автоматизації проектування інформаційних систем (РІС). В статті [1] була подана загальна структура системи формальних специфікацій для проектування РІС. В [2] ми зупинилися на формальних специфікаціях етапу проектування структури бізнес-процесів. Метою цієї статті є розглянути формальні специфікації наступного етапу проектування – проектування мережі сервісів та процесорів.

На етапі побудови специфікації мережі процесорів відбувається подальша деталізація проектного рішення, прийняття реалізаційних вирішень для задач логічного та параметричного проектування.

На цьому етапі продовжується розробка функціонально-параметричної структури ІС. При цьому визначають складніші агрегати функцій – системні сервіси та сеансові процеси. Відбувається перехід від суто процесного, функціонального подання системи до функціонально-об'єктного та об'єктного. В загальній системі специфікацій визначають функціональні компоненти багаторазового використання, які покладають в основу системних сервісів, а пізніше реалізують програмно або апаратно. Тут же вирішують класичні задачі побудови структури даних системи та деталізують процеси до рівня побудови мініспецифікацій.

Паралельно з подальшою розробкою функціонально-параметричної структури на цьому етапі вводиться та розробляється структура процесорів – об'єктів, які виконують процеси, розглядаються проблеми керування виконанням процесів.

Вхідним даним для проектування мережі процесорів є специфікація мережі процесів. При цьому розглядають всі рівні деталізації цієї мережі.

На етапі розробки мережі сервісів та процесорів розглядають нові типи специфікаційних об'єктів.

Сеансовий процес – це різновид процесу, результат діяльності якого досягається в ході взаємодії з багатьма іншими процесами. Сеансовий процес пам'ятає стан сеансу, має початок та закінчення, мету та алгоритм виконання. Процеси, які задіяні у сеансовому процесі, можуть знаходитися на різних процесорах. Цим сеансовий процес відрізняється від деталізації звичайного процесу. Різновидом сеансового процесу є транзакція.

Сервіс є логічним компонентом багаторазового використання. Переважно, він подається як функціональний агрегат множини процесів. Для нього стандартизують інтерфейс вхідних та вихідних функцій, інтерфейс керування. Реалізація функціональності системи за допомогою множини системних сервісів, незалежних від конкретної реалізації та параметрів бізнес-процесів підприємства, надає системі гнучкості та можливості адаптуватися до змін бізнес-процесів. Тому одним із завдань цього етапу проектування є досягти максимального використання системних сервісів.

Об'єднання процесів (логічна зона) є проміжним, допоміжним компонентом, який використовується для визначення сервісів та процесорів. В логічну зону об'єднують процеси, що мають спільні ознаки (наприклад, тип, місце розташування, спільні бізнес-правила виконання).

Процесор є виконувачем процесів та виступає носієм параметрів, які характеризують виконання процесів. Один процесор може виконувати декілька процесів. Кожен процес мусить мати свій процесор. Можлива динамічна міграція процесів між процесорами. Для керування виконанням процесів процесор має окремий процес керування, який проводить політику щодо виконання процесів.

Процеси керування розташовані на процесорах. Вони керують виконанням процесів, закріплених за процесором, змінюють параметри процесора, підтримують сеанси та транзакції, визначають логіку опрацювання процесором робочих процесів.

Потоки керування змінюють параметри налаштування процесора.

На етапі побудови мережі сервісів та процесорів вирішують задачі подальшої декомпозиції процесів та специфікації їх функціонування (зокрема, сеансових процесів), проводять групування процесів, побудову їх об'єднань, ідентифікацію системних сервісів та процесорів. Для визначених сервісів та процесорів уточнюють їх параметри та процедури функціонування. Аналогічно до попереднього етапу тут вирішуються задачі ідентифікації та дослідження параметричних залежностей, прикладні задачі проектування.

Мережа процесорів – це множина процесорів та зв'язків між ними:

$$NNs = \{M(Ns), M(LNs)\}$$

Процесор Ns – це елемент специфікації, що агрегує процеси та є носієм параметрів виконання процесів.

$$Ns = \{idNs, TyNs, LNs, CoNs, PtNs, PcNs, MdNs, PbNs, DtNs, CmNs\},$$

де idNs – ідентифікатор процесора, LNs – множина потоків керування процесора, CoNs – процес керування процесора, PcNs – множина процесів, що виконуються цим процесором, PrNs – вказівник на старший процесор, MdNs – параметрична модель виконання, PbNs – блок параметрів процесора, DtNs – деталізація процесора, CmNs – коментар.

Процесор може входити в інший процесор.

$$PtNs_i = idNs_j \perp Ns_i \subset Ns_j.$$

Кожен з процесів, закріплених за даним процесором є елементом мережі процесорів:

$$PcNs \subset M(Pc).$$

Тип процесора визначає структуру його параметрів та модель виконання. Наприклад, можливі такі базові типи процесорів:

- організація,
- підрозділ,
- комп'ютер,
- принтер,
- людина - виконавець.
- комунікаційна мережа.

На етапі розробки мережі процесорів базові типи процесорів поділяються:

- за типами процесів TyPc, що виконуються процесором,
- за приналежністю цих процесів до однієї організації чи підрозділу (єдине місце виконання).

Типи процесорів накладають обмеження і на відношення вкладеності процесорів. Деякі процесори не можуть бути вкладені в інші. Наприклад, процесор-організація не може бути вкладена в процесор-принтер. Окремі типи процесорів можуть мати підтипи. Таким чином, типи процесорів утворюють ієрархію IrTyNs:

$$IrTyNs = M(TyNs).$$

Ієрархія IrTyNs є інструментальним даним проектування.

Розглядаючи типи процесорів, треба розрізняти:

- базові типи. Це первинні типи. Деякі з них можуть виступати контейнерами для інших базових типів. Базовому типу відповідає певний шаблон параметрів та методів. Цей шаблон досить загальний;
- похідні типи. Похідні типи конкретизують певний базовий тип та утворюють свою ієрархію типів та підтипів. Наприклад, базовий тип – підрозділ. Похідні типи: відділ збуту, постачання, маркетингу та ін. Похідному типу відповідають доповнені шаблони властивостей та методів, властивих цьому підтипу. Похідні типи створюються проектувальником та передаються між проектами;
- реалізацію типу. Створюється проектувальником та існує в конкретному проекті.

Процес керування CoNs визначає логіку опрацювання процесів процесором. Він створює нові процеси, присвоює їм пріоритети, визначає інші дані, що впливають на виконання процесів, опрацьовує ситуацію початку та закінчення процесів, проводить в життя політику виконання процесів та ін. При цьому він працює з параметрами керування (власними та процесора). Процес керування процесора приймає та відсилає інформаційні потоки керування.

Процес керування

$$CoNs = \{idCoNs, MdCoNs, PlCoNs, PbCoNs, DtCoNs, CmCoNs\},$$

де idCoNs – ідентифікатор процесу керування, MdCoNs – параметрична модель, PlCoNs – політика виконання процесів, PbCoNs – блок параметрів, DtCoNs – деталізація, CmCoNs – коментар.

Треба підкреслити відмінність процесу керування CoNs від моделі виконання процесора MdNs. При необхідності створення нового процесу CoNs створить його, проаналізує поточну ситуацію згідно зі своєю логікою керування, присвоїть процесу пріоритет, може змінити деякі параметри керування. MdNs же проаналізує ситуацію в цілому, змінить робочі параметри процесора (наприклад, його завантаженість), спрогнозує час закінчення виконання процесу та наступну подію у системі. Наприклад, якщо CoNs визначає алгоритм роботи ОС та логіку зміни параметрів керування, які використовує ОС, то MNs специфікує зміну робочих параметрів комп'ютера. MdNs розглядає CoNs як один з процесів, яких потребують ресурси і мають свою модель виконання.

Процес керування специфікується аналогічно до звичайного процесу. Він реагує на визначений перелік подій, ініціалізує процеси, визначає їх параметри виконання згідно з політикою виконання.

Політика виконання процесів PlCoNs – це набір параметричних правил, які залежно від параметрів процесу, процесора та параметрів керування визначають пріоритет виконання процесу, частку ресурсів процесора й інші параметри, які впливають на параметри якості обслуговування процесу.

Модель виконання MdCoNs специфікує вплив опрацювання процесом керування робочих процесів на параметри процесора. Одним з найважливіших типів подій для процесу керування є активізація потоку до якогось з передбачених, закріплених за цим процесором процесів. У цьому випадку процес керування звертається до бібліотеки шаблонів, в якій записана модель виконання кожного типу процесу. Для повного опису такої моделі достатньо визначити для кожного робочого процесу $PcNs_i \in Ns$ процедуру φ_i.

$$\varphi PcNs_i = \{\varphi PcNs_{i1}, \varphi PcNs_{i2}\}.$$

В найпростішому випадку опрацювання процесу складається з опрацювання двох подій – початку процесу та його закінчення. Процедура φ_{i1} визначає дії процесу керування залежно від параметрів самого процесора, вхідного потоку робочого процесу на початку опрацювання процесу, а φ_{i2} – при закінченні.

Фактично ці функції – це бізнес-правила опрацювання процесу залежно від поточних параметрів самого процесу та процесора. Таким чином, процес керування розглядає роботу процесора як систему моделювання за подіями.

Якщо спрощена модель робочої операції у вигляді подій початку та закінчення не задовольняє проектувальника, може розглядатися деталізація $DtPcNs_i$, яка точніше описує перебіг виконання процесу з точки зору моделювання подій та зміни парамерів.

Випадок опрацювання робочих процесів в різних умовах завантаженості процесора може бути промодельований або моделюваннм потоку завдань, або зміною параметрів завантаженості.

Функції моделі виконання можуть бути специфіковані, наприклад, мовою ESTELLE, яка дозволяє визначити для кожного стану та набору параметрів відповідний набір дій.

Блок параметрів PbCoNs визначає параметри самого процесу керування і є множиною параметрів.

$$PbCoNs = M(PrCoNs).$$

Кожен параметр

$$PrCoNs = \{idPr, ZnPr, TyPr, DnPr, CmPr\},$$

де idPr – ідентифікатор параметра, ZnPr – його значення, TyPr – тип параметра, DnPr – домен визначення, CmPr – коментар.

Тип параметра визначається

- його розмірністю, приналежністю до ієрархії (класифікації) типів параметрів,
- сферою використання параметра.

Розрізняють:

- параметри керування (вони задаються як зовнішніми процесами керування, так і самим CoNs та впливають на поведінку);
- фіксовані параметри (визначають характеристики самого процесу керування – наприклад, споживання системних ресурсів) – задаються проектувальником;
- робочі параметри (поточні параметри CoNs, які встановлюються, як і для будь-якого іншого процесу, параметричною моделлю виконання MdNs).

Деталізація процесу керування DtCoNs будується за правилами деталізації DtPc довільного процесу Pc.

У параметричному блоці PbNs містяться параметри процесора, методи опрацювання об'єкта – процесор та параметричні залежності. Він складається з множин параметрів та методів і параметричних залежностей.

$$PbNs = \{M(PrNs), M(MeNs)\}.$$

Кожен параметр PrNs – це

$$PrNs = \{idPrNs, ZnPrNs, TyPrNs, DnPrNs, CmPrNs\},$$

де idPrNs – ідентифікатор параметра, ZnPrNs – значення параметра, TyPrNs – тип параметра, DnPrNs – домен визначення, CmPrNs – коментар.

Аналогічно до інших параметрів параметри процесора можуть бути поділені на параметри керування, фіксовані, робочі параметри.

Модель виконання MdNs відслідковує всі події зміни стану процесора і відповідно змінює його параметри. Її можна розглядати як сукупність параметричних залежностей від подій та станів процесора. Наприклад, вона може реагувати на такі події:

- опрацювання процесом керування подій початку та закінчення виконання робочого процесу. Для цього парі функцій процесу керування $\varphi PcNs_{i1}$, $\varphi PcNs_{i2}$ мусить відповідати пара функцій моделі виконання $\psi PcNs_{i1}$, $\psi PcNs_{i2}$, визначена на домені параметрів процесора Dn(PrNs) і яка відображає зміну параметричного стану процесора;

  $$\psi PcNs_{i1}: PbNs \rightarrow PbNs'$$

  $$\psi PcNs_{i2}: PbNs' \rightarrow PbNs''$$

- перехід параметрично визначеного стану процесора у визначену область (виникнення умови спрацювання параметричної залежності);
- спонтанні події, не викликані зовнішніми причинами. Відбуваються з певною ймовірністю та моделюються давачами випадкових чисел.

Для кожної визначеної події в моделі виконання специфіковано параметричну залежність, яка змінює робочі параметри процесора.

Сервіс – це агрегат процесів з фіксованим інтерфейсом, логічна компонента введення якого спрямована на виділення та оформлення в загальному наборі функцій однакових або споріднених функцій з метою їх подальшої реалізації у компонентах (логічних, програмних та апаратних) багаторазового використання. Цей підхід дає такі переваги:

- здешевлює розробку;
- робить розробку більш надійною, оскільки інкапсуляція функцій в межах компонента дає змогу краще протестувати його;
- робить систему більш продуктивною при однакових витратах за рахунок спеціалізації компонента,
- система, орієнтована на сервіси, гнучкіша до змін в ній – нема прив'язки до конкретної структури БП,
- використання стандартних компонент та сервісів дає змогу застосувати динамічну реплікацію, дублювання з меншими витратами.
- відділення інтерфейсу від реалізації надає можливість зміни реалізації сервісу без зміни інтерфейсу та оточення.

У специфікації сервісу необхідно розділити інтерфейсну та реалізаційну частини і забезпечити їх незалежні опис та змінення.

Таким чином, сервіс:

$$Se = \{idSe, InSe, ReSe, CmSe\},$$

де idSe – ідентифікатор сервісу, InSe – інтерфейсна частина, ReSe – реалізаційна частина, CmSe – коментар.

$$InSe = M(inLSe, M(ouLSe)),$$

тобто інтерфейсна частина – це множина пар елементів, кожна з яких специфікує вхідний потік та множину вихідних потоків (відповідей), пов'язаних з цим вхідним потоком.

$$ReSe = \{M(PcSe), CoSe, PbSe, MdSe\},$$

де M(PcSe) – множина процесів (операцій сервісу), CoSe – процес керування, PbSe – блок параметрів, MdSe – модель виконання.

Процес керування сервісу CoSe відображає логіку опрацювання сервісом окремих процесів (якщо вони взаємозалежні або сеансові). У цьому сервіс – це об'єкт, подібний до процесора.

Сеансовий процес – це процес, який виконується на декількох процесорах і, таким чином, не має однозначної локалізації. Сеансовий процес є різновидом бізнес-процесу. Це означає, що він має початок та закінчення, мету та описує перетворення вхідного потоку даних у вихідний. Водночас деталізація такого процесу розносить його підпроцеси по процесорах та описує взаємодію з процесом керування кожного процесора

Сеансовий процес та його деталізація специфікується аналогічно до звичайного процесу. Процеси $PcDtNs_i$ виконуються на різних процесорах. Відмінність сеансового процесу полягає у поведінці процесу керування CoNs, який за необхідності створює процес керування сеансом, який зберігає дані сеансу аж до його закінчення. Момент закінчення сеансу визначається процесом керування сеансом (при цьому можлива участь і процесу керування процесора) і може не збігатися з моментом закінчення активності операцій сеансового процесу.

Міжпроцесорні потоки LNs реалізують керування параметрами процесора. Потік керування LNs або LPc (залежно від того, хто виступає приймачем потоку – процесор або процес) – це команда зміни керуючих параметрів процесора або процесу, які впливають на їх поведінку щодо виконання головних функцій.

Рішення про зміну параметрів приймає і виступає передавачем, генератором потоку керування

- процес керування старшого (охоплюючого) процесора у випадку адаптивної системи або
- адміністратор, проектувальник, системний аналітик.

Таким чином, потік керування – це завжди потік між процесом та процесором або процесом керування.

$$LNs = \{idLNs, inLNs, ouLNs, PbLNs, DtLNs, CmLNs\},$$

де idLNs – ідентифікатор потоку, inLNs – ідентифікатор джерела потоку, ouLNs – ідентифікатор приймача потоку, PbLNs – блок параметрів потоку, DtLNs – деталізація потоку, CmLNs – коментар. Деталізація потоку DtLNs дає змогу розглядати його як об'єднання потоків.

Одним з важливих елементів мережі процесів є ресурси. Вони розглядаються як носії параметрів та параметричної моделі. Важливо зазначити відмінності між процесом, процесором та ресурсом.

Зв'язок між процесом та ресурсом – це зв'язок одного рівня або від старшого до молодшого. Процес використовує ресурс. Декілька процесів можуть використовувати один і той же ресурс.

Зв'язок між процесором та процесом – це також зв'язок від старшого до молодшого. Процесор виконує процеси, закріплені за ним.

Процес може бути зв'язаний з ресурсом в межах охоплюючого процесора. Водночас ресурс, може бути одночасно і вкладеним процесором. Ресурс можна розглядати як інкапсульований процесор. У випадку, коли процес, який використовує ресурс, не деталізується, може бути використана узагальнена модель ресурсу (процесора). Така операція і такий зв'язок може розглядатися на всіх етапах проектування.

Для більш детального аналізу та розуміння використання ресурсу процес Pc деталізується (будується DtPc). У складі деталізації виділяються процеси Pc`, які повністю виконуються на процесорі-ресурсі Ns. Моделюється виконання цих процесів на процесорі-

ресурсі, визначаються часові та інші характеристики виконання частини первинного процесу на процесорі - ресурсі. З формальної точки зору для такого дослідження необхідно, щоби

$$\exists\, Pc' \in DtPc \perp Pc' \in Ns.$$

Логічна зона – це проміжний, робочий елемент специфікації. Логічна зона – це іменоване об'єднання процесів.

$$Lz = \{idLz, LiPc, TyLz, CpLz, CmLz\},$$

де idLz – ідентифікатор логічної зони, LiLz – список процесів, які належать до зони, CmLz – коментар, CpLz – властивості, які характеризують спільність, TyLz – тип спільності процесів, об'єднаних у зону.

Як правило, у логічну зону об'єднують процеси, які мають спільні ознаки. Такими ознаками, наприклад, можуть бути:

- приналежність до організації або підрозділу;
- знаходження в одному будинку;
- сильний міжпроцесний інформаційний зв'язок;
- робота з одним провайдером.

Типи спільності процесів утворюють попередньо визначену ієрархію IrTyLz. Елементи ієрархії визначаються задачами, які використовують логічні зони як вхідні дані.

Задачі, які вирішують під час побудови мережі процесорів та її дослідження, поділяють на:

- задачі побудови мережі сервісів та процесорів;
- задачі параметричних досліджень та моделювання;
- прикладні задачі.

Задачі побудови мережі сервісів та процесорів у свою чергу поділемо на

- задачі групування процесів;
- задачі декомпозиції процесів та вибору архітектури обчислень;
- задачі ідентифікації процесорів.

Розв'язання задач групування та декомпозиції процесів тісно пов'язані з розв'язанням задачі вибору комплексу архітектурних вирішень методами та засобами, властивими для наступної специфікаційної мережі – мережі прототипів.

Архітектура (у системі проектування PIC) – це загальне для деякого процесора-контейнера та типу сервісу вирішення, яке висуває додаткові вимоги щодо внутрішньої організації цього процесора, зокрема наявності певних типів процесів або вкладених процесорів.

Вибір архітектури обчислень проводиться для системи в цілому або для окремого її фрагмента (підсистеми, наприклад, ввіддаленої підсистеми). Визначають типи та підтипи архітектур. Для однієї системи може існувати декілька архітектур різних типів, але тільки один підтип кожного типу.

В CASE-системі блок визначення архітектури може мати просту експертну систему, в якій формулюються параметричні правила, які дають змогу рекомендувати проектувальнику одну з архітектур. У простішому випадку функції експертної системи можуть бути передані проектувальнику, але це ставить додаткові вимоги щодо його кваліфікації.

Система може перевіряти правильність застосування архітектурного підходу шляхом визначення підтипів процесів та їх зв'язків з іншими процесами. Наприклад, для архітектури "клієнт-сервер" повинні бути процеси-клієнти та процес-сервер.

Ієрархія типів архітектур – це множина типів архітектур:

$$IrAr = M(Ar),$$

де Ar – тип архітектури.

$$Ar = \{idAr, DfAr, DcAr, ReAr, LiAr, [LiTyPc], CmAr\},$$

де idAr – ідентифікатор типу архітектури, DfAr параметричне визначення архітектури, DcAr – процедура прийняття рішення, ReAr – вимоги до структури та типів процесів, LiAr – список підтипів, CmAr – коментар. LiTyPc – список типів процесів, з якими працює архітектура. Ієрархія типів архітектур відноситься до інструментальних даних проектування.

Задача визначення логічних зон на множині процесів вирішується паралельно з вирішенням задачі декомпозиції процесів. Метою цієї задачі є визначити спільноти процесів, які можуть бути використані як для визначення процесорів, вибору архітектур системи, так і для вирішення задач аналізу та оптимізації. Визначення логічних зон – це творча, неформальна задача, яка вирішується проектувальником при підтримці програмної системи.

Певну допомогу у визначенні логічних зон можуть надати програми аналізу та групування процесів. Такі програми аналізують параметри процесів та будують їхнє розбиття на зони згідно з вимогами певного типу архітектури. Такі розбиття безпосередньо використовуються для вирішення задачі вибору архітектури системи. Наприклад:

- для правильного вибору мережевої архітектури треба спочатку розбити процеси на групи за ознакою розміщення. Наприклад, можна виділити процеси, розташовані у головному офісі та філіях і комунікаційні процеси між ними;
- для вибору архітектури роботи з базами даних в одну групу треба виділити процеси, які працюють з ними;
- при побудові структури мережі можна визначити групи процесів, які сильніше інформаційно пов’язані.

Для різних типів та підтипів архітектур існують *шаблони розбиттів*, орієнтовані на застосування певних технологічних вирішень. Задачу розбиття можна розглядати як оцінку параметрів системи та пошук придатного шаблону розбиття. Для того, щоби застосувати якусь архітектуру, процеси зони повинні відповідати певним вимогам. З цієї точки зору розбиття на відповідність шаблонам можна розглядати як підготовчу операцію щодо вибору архітектури. Загалом така операція не визначає однозначно архітектури. Для одного розбиття можна застосувати декілька архітектур. Шаблон розбиття відображає прийняту на практиці процедуру декомпозиції системи. Провести розбиття з урахуванням шаблону – це виділити підмножини процесів, для яких можна застосувати конкретне вирішення певного архітектурного типу. Тому операції вибору архітектури та визначення логічних зон пов’язані та виконуються почергово.

Послідовність визначення об’єднань та визначення об’єднання в межах об’єднання іншого типу встановлюється проектувальником. Можна рекомендувати таку послідовність, яка випливає з існуючих технологічних реалій:

- розбиття по організаційних ознаках та ознаках власності,
- розбиття по ознаках розміщення,
- розбиття по інтенсивності потоків та обсягах інформації,
- розбиття по типах процесів або по типах ресурсів, які вони використовують.

Розбиття по організаційних ознаках та ознаках власності виконується тоді, коли організація чи підрозділ порівняно зі своїм оточенням відрізняються спеціальними бізнес-правилами та вимогами. Наприклад, ставиться вимога підвищеної таємності та захисту інформації. Розбиття по ознаці власності проводиться тоді, коли Замовник планує враховувати цю ознаку під час проектування та роботи системи. Наприклад, він цікавиться обліком витрат на володіння окремими підсистемами, або існують підсистеми, що належать іншим власникам.

Правила поділу цілої системи на частини диктуються вимогами, сформульованими для загального типу архітектури. Наприклад, сучасні технології в галузі телекомунікацій вимагають поділити комунікаційні процеси на зони, що відповідають локальним мережам, кампусним мережам та зв'язкам глобальних мереж між ними. При цьому загальноархітектурні вимоги охоплюють всю систему, всі процеси певного типу. На рівні типу архітектури діють процедури, які визначають коректність декомпозиції системи.

Поточний поділ множини процесів на логічні зони визначає декомпозицію множини процесів M(Pc):

$$DcPc = \{Lz \perp (\bigcup LiPc = M(Pc)) \wedge (\forall i,j\ LiLz_i \bigcap_{i \neq j} LiLz_j = \varnothing)\},$$

тобто декомпозиція певної множини процесів – це множина логічних зон, процеси яких в сукупності покривають всі процеси цієї множини та не перетинаються.

Метою задачі декомпозиції процесів є досягнення рівня деталізації процесів достатнього для:

- побудови специфікації логічного проектування мовою мініспецифікацій;
- подальшого групування, побудови сервісів та об'єднань процесів. Для цього кожен процес найнижчого рівня деталізації повинен виконуватися одним виконавцем (або в одному розміщенні).

Деякі розподілені або сеансові процеси можуть не деталізуватися або деталізуватися надалі.

Задача декомпозиції проводиться на етапі, що передує визначенню процесорів.

Вхідними даними для кожної операції декомпозиції є специфікація процесу Pc, результатом – деталізація специфікації DtNp. Аналізуючи задачу декомпозиції, можна визначити декілька типів процесів декомпозиції, які проводяться над одними і тими ж даними, але з різною метою. Це такі задачі:

- задача декомпозиції процесів, що відбувається у процесі логічного проектування. Мета такої декомпозиції – визначити структуру процесів, зробити можливим їхню програмну реалізацію;
- виділення типових процесів та операцій з метою їх подальшого об'єднання для реалізації у сервісах. Такий підхід доцільно застосувати для проектування програмних компонент.
- визначення комплексу архітектурних вирішень системи та проведення декомпозиції з урахуванням обраних архітектур.

Декомпозицію проводить проектувальник при підтримці програмної CASE системи. При цьому проектувальник:

- проводить декомпозицію процесу;
- пропонує архітектурні вирішення;
- приймає рішення щодо архітектурного вирішення.
- задає параметри нових процесів;
- визначає тип нових процесів,

Система:

- перевіряє правильність запропонованої декомпозиції, вказує на невідповідності;
- зберігає нові процеси у репозиторії проектування,
- слідкує за цілісністю даних репозиторію,
- пропонує шаблони параметрів для нових процесів,
- пропонує архітектурні вирішення за результатами аналізу параметрів системи;

Задача декомпозиції включає в себе функції аналогічної задачі мережі процесів.

Одним з різновидів задачі декомпозиції процесів є задача декомпозиції зідно з вимогами певних типів архітектур. Можна стверджувати, що процес декомпозиції для такої задачі керується прийнятими щодо архітектури вирішеннями. За своєю структурою – це ітеративна задача, перебіг якої, наприклад, може бути таким:

- оцінюється система загалом і приймається рішення щодо архітектури рівня 1;
- проводиться декомпозиція та визначення логічних зон згідно з вимогами архітектури рівня 1;
- для кожної логічної зони приймається рішення щодо архітектури рівня 2;
- проводиться декомпозиція згідно з вимогами архітектури цього рівня;

Процес може продовжуватися для наступних рівнів архітектур. Кожен рівень архітектури має свої вимоги щодо декомпозиції загалом та її частин. Як правило, кожна частина відповідає вимогам однієї з архітектур нижчого рівня і придатна для її застосування.

Задача ідентифікації процесорів виходячи з попереднього визначення об'єднань процесів визначає процесори. Вхідними даними для неї є визначені попередньо логічні зони. Не всім логічним зонам буде поставлено у відповідність процесор. Рішення про визначення процесора на базі логічної зони приймає проектувальник. При цьому він може визначати процесор, керуючись такими правилами:

- для логічної зони існують окремі бізнес-правила, що впливають на роботу процесорів;
- логічною зоною об'єднані однотипні процесори або процесори, що потребують одного типу ресурсу,
- визначення процесора диктується обраною архітектурою. Часто архітектура вимагає наявності певних процесорів та закріплення за ними визначених процесів.

Складові частини задачі ідентифікації процесорів:

- прийняти рішення про визначення процесорів на базі логічних зон;
- для кожного процесора визначити алгоритм обслуговування (виконання) процесів. Залежно від ситуації можна розглядати різні ступені деталізації алгоритму роботи процесу керування: від детального опису алгоритму до подання простої моделі, яка пов'язує параметри виконання кожного процесу з параметрами інших процесів цього процесора та параметрами самого процесора;
- для кожного процесора визначити параметри керування та налаштування. Відокремити параметри внутрішнього керування, що змінюються процесом керування, та параметри керування, доступні ззовні;
- оцінити повноту списку параметрів, тобто чи всі істотні фактори враховані.

Визначення процесорів виконує проектувальник при підтримці програми проектування. При цьому проектувальник:

- приймає рішення щодо визначення процесора,
- визначає процес керування, відповідний алгоритм та параметри,

Система:

- рекомендує проектувальнику визначити процесори згідно з обраною архітектурою,
- заносить визначення процесора у репозиторій,
- формально перевіряє запропоноване проектувальником вирішення на відповідність обраній архітектурі.

Задачі моделювання та дослідження параметричних залежностей вирішуються після ідентифікації процесорів та визначення множини їх параметрів.

Структура задач групи:

- ідентифікація параметричних залежностей;
- аналіз параметричної моделі виконання, визначення ступеня її адекватності та корекція у разі необхідності;
- дослідження поведінки процесора за його моделю в різних режимах домену визначення параметрів процесу та процесорів;
- аналіз впливу параметрів керування та налаштування на параметри поведінки процесора;

За результатами аналізу можуть бути вироблені рекомендації щодо покращання алгоритму роботи процесора, визначені екстремальні параметри його роботи. Можливість вимірювання та динамічного налаштування параметрів різних типів процесорів знайшла свій вираз в концепції інтелектуальних активних мереж [3].

Під час проектування визначається попередня модель поведінки процесора, яка випливає з відомого алгоритму його роботи. Ця модель уточнюється та змінюється після завершення проектування та апаратної реалізації на основі вимірів. Задачі моделювання параметричних залежностей вирішує проектувальник та адміністратор системи при підтримці системи проектування.

Запропонована система специфікацій для проектування мережі сервісів та процесорів може бути використана для створення інтелектуальних засобів підтримки процесу проектування розподілених інформаційних систем протягом всього їх життєвого циклу.

1. Буров Є.В. Система формальних специфікацій для проектування розподілених інформаційних систем//Вісник НУ "Львівська політехніка" – 2000. – № 406. 2. Буров Є.В. Система формальних специфікацій проектування структури бізнес-процесів в САПР розподілених інформаційних систем// Вісник НУ "Львівська політехніка" – 2000. – № 406. 3. Иванов П. Активные сети //Сети. – 1999 – № 10.