УДК 004.415.2.031.43

Бойченко О.В.,
кандидат технічних наук, доцент,
Лєнков С.В.,
доктор технічних наук, професор,
Шкуліпа П.А.,
кандидат технічних наук

СТРУКТУРНЕ ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СКЛАДНИХ ІНФОРМАЦІЙНИХ СИСТЕМ РЕАЛЬНОГО ЧАСУ

У статті розглянуто особливості системного підходу до проектування стійкого програмного забезпечення через розроблення засобів контролю і виправлення помилок у роботі інформаційної системи з використанням специфікацій, що базуються на визнанні факту можливості виникнення перекручувань у роботі обчислювальних засобів.

Ключові слова: *системний підхід, програмне забезпечення, інформаційна система, засоби контролю, ФПР-декомпозиція.*

В статье рассмотрены особенности системного подхода к проектированию стойкого программного обеспечения с использованием средств контроля и исправления ошибок в работе информационной системы на основе спецификаций, которые базируются на признании факта возможности возникновения перекручивания в работе вычислительных средств.

Ключевые слова: *системный подход, программное обеспечение, информационная система, средства контроля, ФСР-декомпозиция.*

In the article the features of approach of the systems are considered to planning of proof software with the use of controls and correction of errors in-process informative system on the basis of specifications which are based on adoptive of possibility of origin of twisting in-process of computing facilities admission.

Keywords: *approach of the systems, software, informative system, controls,* FER-decouplig.

Високі вимоги щодо рівня ефективності та експлуатаційної надійності інформаційної системи підтримки рішень (ІСПР), необхідність забезпечення якісно нового рівня показників достовірності, доступності, конфіденційності, масштабності та повноти даних, а також нагальною потребою оптимізації процесу управління інформаційними потоками відомчої ІСПР в правоохоронній діяльності при дефіциті часу визначають актуальність розроблення комплексної системи заходів удосконалення відомчої ІСПР в правоохоронній діяльності.

Одним із напрямків удосконалення інформаційної системи є структурне проектування програмного забезпечення.

Методологічною базою проектування є використання етапів, на кожному з яких розроблення ведеться по прошарках на обмеженому наборі припустимих структур [1].

На першому етапі провадиться тривимірна функціонально-подійно-режимна декомпозиція (ФПР-декомпозиція) програмного забезпечення (ПЗ) на задачі. При цьому в доповнення до класичної функціональної декомпозиції здійснюється декомпозиція за принципом реакції на події з об'єкта і по режимах роботи системи (рис. 1–3).

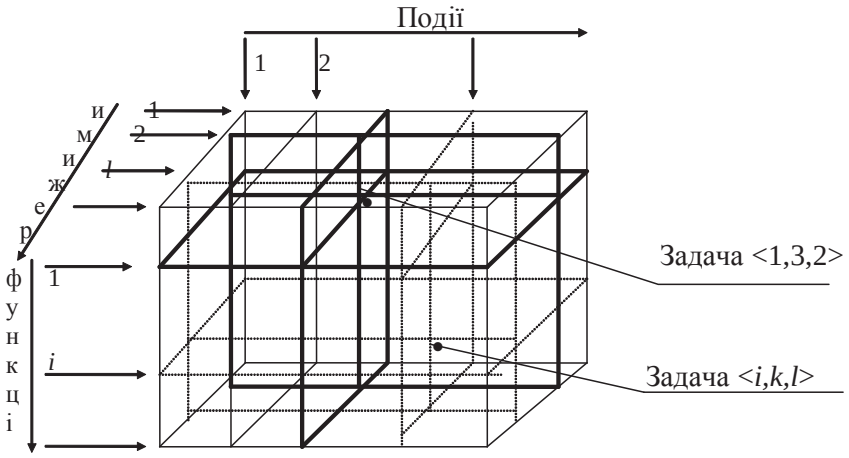

Рис. 1. Схема функціонально-подійно-режимної декомпозиції

На другому етапі в кожній задачі виділяється функціональне ядро, що не залежить від обраної операційної системи (ОС), тому що з нею не взаємодіє, і від обраного методу міжзадачного зв'язку за даними (безпосередній обмін, спільна область пам'яті, база даних). Всі функції зв'язку покладаються на частину задачі, що залишилася, інтерфейс (рис. 3).

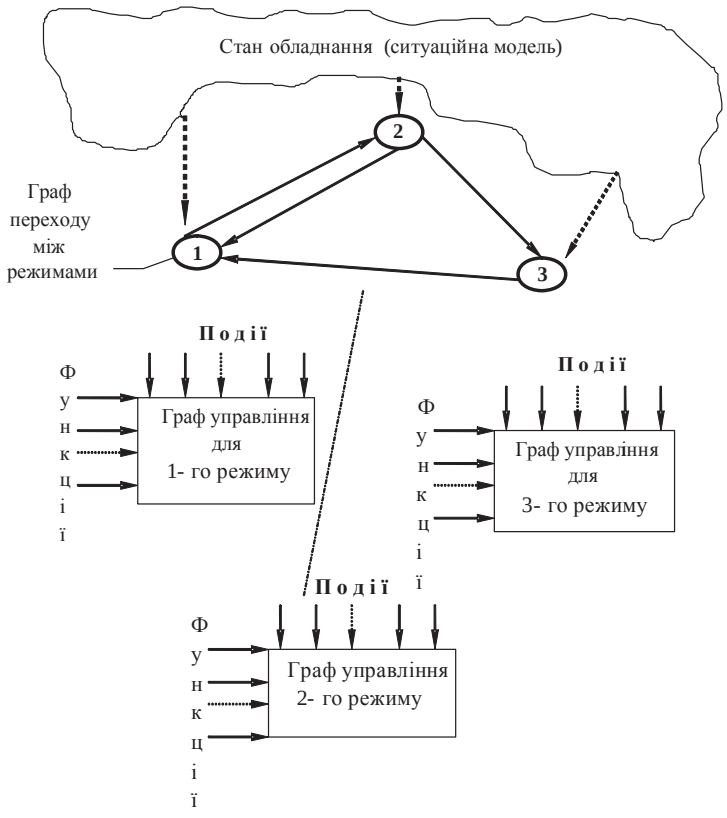

Рис. 2. Перший прошарок графу мультиструктури ПЗ АСУ

Проектування ядер може починатися незалежно від рішення загальносистемних питань. Інтерфейс є основним об'єктом для ОС, з допомогою якого здійснюється зв'язок ядер по керуванню й інформації.

Проектуванням інтерфейсів завершується складання системи. Іншими словами, починаючи з другого етапу, проектування ведеться в двох напрямках: проектування ядер відбувається по спадній гілці, а складання системи за допомогою інтерфейсів – по висхідній.

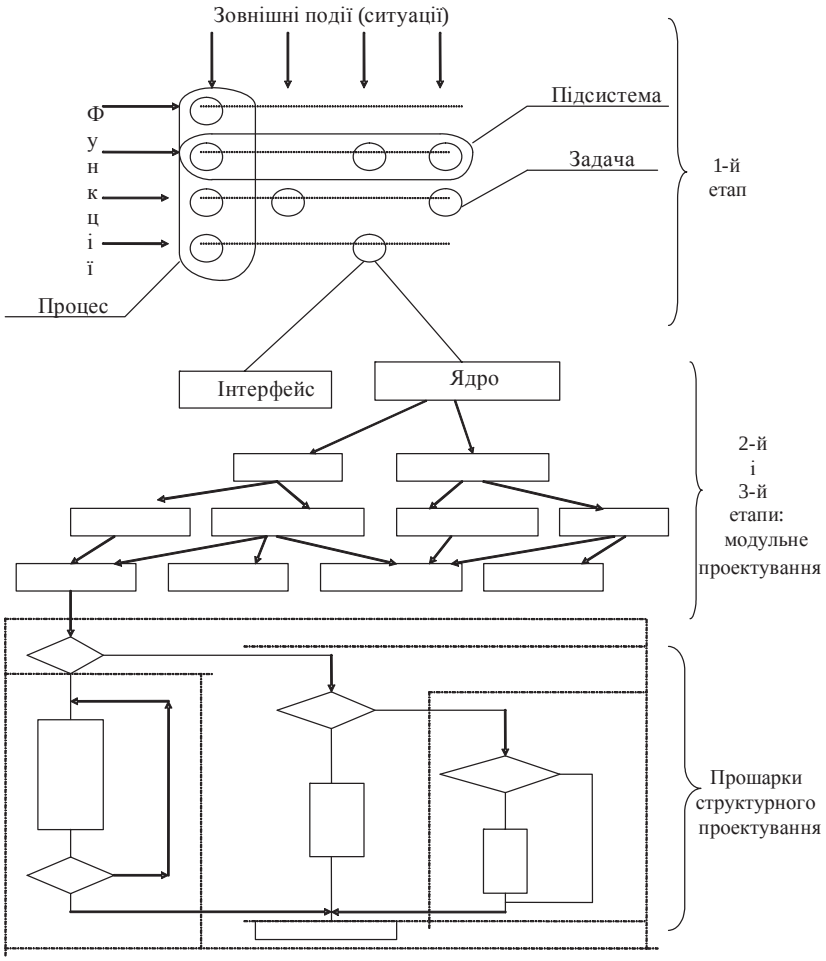

Рис. 3. Використання паролів для діагностування при збої при передачі управління

На третьому етапі відбувається пошарове модульне проектування, тобто подальша декомпозиція ядер. Цей етап реалізується одночасно з четвертим (рис. 3).

Четвертий етап – пошарова деталізація модулів із використанням узвичаєних структур структурного програмування (структурні примітиви: зчіпка, логічне розгалуження, цикл). Цей етап активно використовує мови проектування програм, типу PDL, що дозволяють "без ударно" перевести процес проектування в процес кодування (написання тексту, який безпосередньо сприймається транслятором комп'ютера) [2].

Основною відмінністю методології ФПР-декомпозиції ПЗ від інших є етап проектування системних зв'язків (системної специфікації).

На першому етапі проектування реалізується ФПР-декомпозиція ПЗ на задачі – перший прошарок опису. ФПР-декомпозиція розуміється як процес

виділення програмних одиниць, що реалізують часткову функцію системи керування, вироблену в результаті настання тієї або іншої події в одному із режимів роботи системи.

Для визначення подій використані ідеї ситуаційного керування, коли "докладний опис неозорої множини ситуацій, що укладаються в процесі функціонування реального об'єкта, по визначених правилах заміняють укрупненими "макроописами" узагальнених ситуацій", кожна з який визначає одну з можливих реакцій системи [3].

Подією для системи керування є будь-яка зміна ситуації в часі, що змінює поточний стан потоку керування в цій системі. До початку декомпозиції необхідно створити ситуаційну модель – відображення множини ситуацій на об'єкті в множину зовнішніх подій (переривань) для АСУ.

Наприклад, у системі охорони правопорядку ситуація може бути визначена залежно від положення мобільних нарядів міліції. Подією в цьому разі вважаємо вхід або вихід мобільного наряду в контрольовану зону. Іншим прикладом зовнішньої події є натискання ініціативної кнопки оператором системи (для одержання оперативної інформації тощо).

Режим розуміється як регламент роботи системи, що зберігає структуру керування. Різні режими реалізують різноманітні рівні усталеності і залежать від стану устаткування. Режими підрозділяються на нормальні й аварійні. Переходом із режиму на режим реалізуються запровадження системи в роботу і "програмний відступ" при виявленні непрацездатності елементів керуючої частини (рис. 2).

Після розбивки ПЗ на задачі відбувається опис вхідних і вихідних даних для кожній з отриманих задач, що реалізують заданий режим.

Подання кожної задачі у вигляді 2-х наборів (вхідних і вихідних даних) дозволяє побудувати орієнтований граф інформаційних зв'язків між задачами системи (інформаційний граф).

Множина вершин цього графа збігається з множиною задач, а орієнтоване ребро зв'язує вершину i із вершиною j у тому випадку, якщо перетинання вхідного набору даних задачі j із вихідним набором задачі i не порожньо. Іншими словами, якщо задача j використовує результати роботи задачі i, на інформаційному графі є ребро, що зв'язує вершину i із вершиною j.

Для відображення зв'язку задач із зовнішніми подіями в множину вершин вводиться додаткова підмножина кореневих вершин, що відповідають зовнішнім подіям. Вхідний набір фіктивної задачі, що відповідає вершині з додаткової підмножини, порожній, вихідний набір складається з одного елемента $S(i)$, що включається у усі вхідні набори задач стовпчика для даної події I.

Введення елемента $S(i)$ визначає наявність ребер із вершини, що відповідає події, у усі вершини стовпчика. На основі інформаційного графа будується граф керування для режиму.

Наявність в інформаційному графі сильно зв'язаних компонент, тобто підграфів, для будь-яких двох вершин яких існує путь із кожної вершини в кожну, свідчить про нерозумну декомпозицію на першому прошарку при проектуванні ПЗ.

Більш раціональним, із погляду ефективної організації обчислювального процесу, є об'єднання всіх задач, що утворять сильно зв'язану компоненту, в одну задачу. Така процедура відповідає розкладанню графа на максимальні сильно зв'язані підграфи.

У графі керування, отриманому з перетвореного інформаційного графа (див. такий пункт), циклів між задачами не буде. Циклічне виконання окремих задач буде виражатися петлею при вершині.

На причинно-наслідковий зв'язок між задачами, що задається графом керування (який регламентує порядок виконання задач у системі), впливають існуючі між ними інформаційні зв'язки [4].

Дійсно, якщо в графі керування між двома вершинами (i та j) існує ребро, то аналогічне ребро є й в інформаційному графі між відповідними вершинами, тобто між задачами i та j існує зв'язок за інформацією, у противному випадку задачі j нема чого очікувати завершення виконання задачі i (у цьому випадку ребро (i, j) із графа керування можна прибрати).

Вищесказане дозволяє стверджувати таке: граф керування є підграфом інформаційного графа. "Зайвими" у графі керування будуть ребра інформаційного графа, що є транзитивним замиканням (композицією) декількох ребер.

З цього випливає, що граф керування є базисним графом інформаційного графа.

Циклічне виконання задачі, що відповідає кінцевій вершині на графі керування, не впливає на ініціацію інших задач, граф у цьому випадку можна вважати ациклічним.

У випадку наявності вихідних ребер із вершини i із петлею ініціація задач, що наслідують за циклічною (тобто тих, в які входять ребра з i), повинна відбуватися щоразу після завершення задачі i.

Важливою характеристикою ПЗ є його усталеність: спроможність зберігати певний рівень працездатності, незважаючи на несприятливі впливи зовнішнього (і внутрішнього) середовища [5].

Усталеність ПЗ до впливів, що збурюють, може бути охарактеризована одним із таких рівнів:

спроможність продовжувати виконання функцій у повному обсязі (переходити на резерв у тій або іншій формі);

спроможність виконувати задачі зі зменшенням обсягу функцій і, можливо, із зниженням якості (реалізувати шлях відступу);

спроможність переходити в стан "м'якої" відмови при помилці, що не піддається усуненню.

Реалізація усталеності ПЗ забезпечується виявленням, діагностикою і корекцією помилки з наступним відновленням. В даний час є велика кількість методів і засобів контролю роботи програм, проте, широкій практичній реалізації стійкого ПЗ заважає відсутність системності підходу, недооцінка можливості прояву помилок.

У підсумку зазначимо, що застосування системного підходу до проектування стійкого ПЗ відомчої ІСПР правоохоронної діяльності дозволить створити умови для розроблення специфікацій, що базуються на визнанні факту можливості виникнення перекручувань у роботі обчислювальних засобів (ОЗ) і ПЗ; розроблення програмних засобів контролю і виправлення помилок у роботі ОЗ; розроблення структури ПЗ, що використовує зворотний зв'язок між підпорядкованим і верхнім рівнем, а також розміщення засобів контролю виконання ПЗ відповідно до рівнів ієрархії в системі.

Зазначене дозволить забезпечити розроблення ОС, що має розвинуті засоби контролю виконання програм і роботи ОЗ з максимальним використанням і

розвитком діагностичних можливостей ОС, а також резервуванням найважливіших функцій ІСПР за допомогою локальних пристроїв.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. *Йодан Э.* Структурное проектирование и конструирование программ : монография / Э. Йодан – М. : Мир, 1979. – 415 с.

2. *Дал У.* Структурное программирование : монография / У. Дал, Э. Дийкстра, К. Хоор. – М. : Мир, 1975. – 250 с.

3. *Кухтенко А.И.* Проблема многомерности в теории сложных систем / А.И. Кухтенко // Кибернетика и вычислительная техника. – Вып. 1. – К. : Наукова думка, 1969. – С. 6–35.

4. *Ларичев О.И.* Системный анализ : проблемы и перспективы / О.И. Ларичев // Автоматика и телемеханика. – 1975. – № 2 – С. 61–71.

5. *Сбитнев А.И.* Структурная организация и проектирование / А.И. Сбитнев // ПОАСУТП. – К. : УСМ, 1982. – №5. – С. 38–42.