УДК 519.876.5

ІНТЕПРЕТАТОР ДЛЯ НОВОГО ПОКОЛІННЯ СИСТЕМ АВТОМАТИЧНОГО ПРОЕКТУВАННЯ МЕРЕЖ

С.В. Котенко, канд. техн. наук

Одеський державний аграрний університет

Приведені принципи побудови інтерпретатора для систем автоматичного проектування енерго и масообмінних мереж.

ВСТУП

Розробка математичного апарату HEN-MEN-технологій (технологій аналізу тепло – і масообмінних мереж) анонсованого в відкритих наукових виданнях роботами І. Гроссмана [1,2], останніми роками призвела до наступного кроку – започатковано створення символьного процесора. Символьний процесор - програмне ядро для вирішення кардинальної проблеми – формалізації задачі пошуку найліпшого рішення при проектуванні мереж, та знаходження найліпшого рішення для систем автоматичного керування технологічними процесами. В першу чергу це стосується енерго та масо розподільчих мережі. HEN/MEN - методи апробуються також при проектуванні і синтезі систем в різних галузях: в хімічній промисловості, енергетиці, комунальному господарстві і т. ін. Головний результат використання HEN-MEN-технологій - отримання неочевидних моделей складних систем, розподілу параметрів в існуючих системах, знаходження найліпших режимів керування складними системами. Використання вже наявних напрацювань в цій сфері призводить до:

- значної економії енергоресурсу
- прискоренню термінів проектування
- безаварійності
- виключенню етапів відпрацювання режимів та технологій на дослідних та пілотних моделях**.**

Але символьний процесор потребуватиме відповідного інтерфейсу. Таким чином постає нагальна проблема створення нового, чи адаптації існуючого графічного інтерпретатора до символьного процесора. Створення графічного інтерпретатора до символьного процесора повинно перетворити мистецтво проектування складних мереж, доступне зараз тільки високоосвіченим професіоналам з багаторічним досвідом, в комп’ютерну гру для аматорів.

МЕТОДИКА ДОСЛІДЖЕНЬ

Графічні інтерпретатори – не новина. Прикладом може слугувати ISaGRAF [3]. ISaGRAF відноситься до класу програмного забезпечення SoftPLC, яке дозволяє працювати не з апаратурою, а винятково зі змінними. В ISaGRAF можна отримати програмний код з дуже високим ступенем портируємості, який можна легко перенести на іншу платформу, якщо така необхідність виникне. Досягти такої гнучкості дозволяє розділення технології ISaGRAF на дві основні складові: середовище розробки і середовище виконання. Програми створюються в середовищі розробки ISaGRAF Workbench. Це інтегрований засіб розробки, що містить редактора коду, відладчик, засіб контролю версій та інше. Саме в ньому створюються програми управління, описується вся логіка технологічного процесу, і саме Workbench компілює код для контролера. В ISaGRAF прийнято розбивати програму на складові частини. Функції, процедури, підпрограми. Для них в ISaGRAF навіть є відповідний термін - Program Organization Unit (або більш коротко - POU). Будь-який елемент коду, винесений в окрему функцію або функціональний блок, в ISaGRAF є POU.

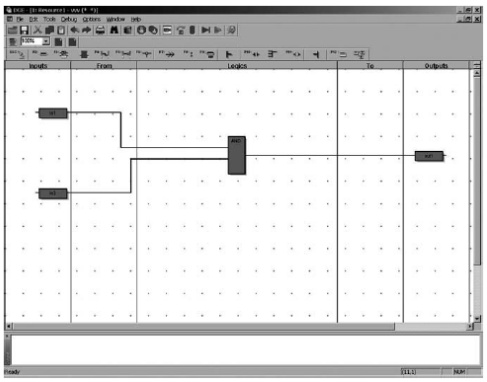

Рис.1 Скріншот ISaGRAF з намальованими блоками.

ISaGRAF взятий як приклад, хоча на ринку конкурують і багато інших програмних засобів. AnyLogic, Arena, AutoMod, Dymola, Extend, eM-Plant, EASY5, Gpss/H-Prof, iThink, Matlab, Modelica, ModelVision, MBTY, ProModel, PowerSim, Quest, Stella, Simfactory, Simplle++, Taylor, Vensim, VisSim, Witness - ось далеко не повний список продуктів, кожний з яких претендує на лідерство.

Більшість цих продуктів використовують мову програмування, в основі якої лежать функції [4], але при цьому сама мова є графічною. Якщо спробувати представити функцію в графічному вигляді, то простіше всього це буде зробити з використанням чорних скриньок. Скринька має: 1) входи, до яких підключається управляюча дія або вхідні дані, 2) саме тіло (власне, скринька), що реалізовує алгоритм обробки, який може бути скільки завгодно складним і 3) виходи, з яких знімається оброблений сигнал або нова управляюча дія.

Функціональні блоки є de-facto стандартом в системах промислової автоматизації. До речі, ще одним прикладом популярного програмного забезпечення, що використовує функціональні блоки, є відомий пакет графічного програмування LabVIEW. LabVIEW як і багато SCADA-систем

використовують програмну мову FBD, яка є частиною стандарту IEC61131 і покладена в основу концепції об'єктно-орієнтованих частин нового стандарту IEC61499.

Мова програмування FBD описує зв'язки між вхідними і вихідними змінними. Під вхідними змінними слід розуміти будь-які змінні ISaGRAF - внутрішні або ж вхідні, які "вводять" інформацію у функціональний блок. Аналогічно, вихідними будуть такі змінні, які "виводять" інформацію з блоку. Блок є функцією, яка сама по собі може бути сформована з інших блоків. Змінні і входи/виходи блоків з'єднуються за допомогою ліній зв'язку. Лінія указує напрям розповсюдження сигналу, при цьому дотримується правило, що сигнал йде зліва направо. Правий і лівий кінці ліній повинні зв'язувати дані одного типу. Окрім зв'язку типу "змінна-змінна" існує множинний зв'язок, званий дивергенцією (розбіжністю). Використовуючи її, можна поширювати дані від одного джерела на велику кількість приймаючих компонентів. Не завжди зручно мати додаткові змінні для зберігання проміжних обчислень, тому виходи одного блоку можна напряму підключати до входів іншого. Графічно блоки відображаються у вигляді прямокутників і є одиничною операцією над вхідними змінними.

Порядок роботи інтерпретатора символьного процесора майже тотожний ISaGRAF. Використання елементів – чорних скриньок, направлені лінії передачі даних між ними, вузли прийому і роздачі з довільною кількістю вхідних – вихідних гілок, кожен блок – одинична операція над даними і т. ін.

І для функціональних мов, і для стандартних інтерпретаторів, теж притаманно використання чорних скриньок – але, не дивлячись на зовнішню схожість, математична сутність їх різна. Символьні процесори базуються на лінійній алгебрі і оперують з системами лінійних рівнянь. Ефективність процесу в елементі, напрямок передачі енергії/маси залежать від змінних входу, а змінні входу – від процесів в інших елементах. Чорні скриньки символьного процесора не настільки атомарні, як чорні скриньки в функціональних інтерпретаторах. Тому використання стандартних інтерпретаторів, навіть в модифікованому вигляді, в якості графічних інтерфейсів символьного процесору неможливо.

Значно прискорити процес розробки ПЗ, уникнути багатьох потенційних помилок і в більшості випадків одержати більш швидкий код можна за рахунок вживання перевірених продуктів інших фірм, що забезпечують розширення графічних бібліотек функцій, класів і компонентів, зокрема ActiveX для задач, типових для інтерпретаторів.

За даними різних публікацій в США (наприклад, фірми IOtech [5]), в яких відображені переваги американських споживачів відносно підходу до побудови систем, можна виділити стійку трійку лідерів серед мов програмування, що лежать в основі таких рішень, — Visual Basic, C/C++ і LabVIEW. Популярність кожного з цих трьох підходів є приблизно однаковою (20–30 %), тоді як на частку всієї решти рішень по мові (стилю) програмування разом узятих, доводиться приблизно ті ж 20–30 %. Delphi помітно поступається провідній

трійці, і його використовують не більше 2–4 % споживачів. Але останні версії Delphi надали цьому ПЗ більший популярності.

Для C/C++ частіше за все використовують відповідні продукти фірм Microsoft і Borland (Inprise). Зрозуміло, що на C/C++ можна написати практично все від початку до кінця, але це буде велика робота (особливо якщо потрібне математичне забезпечення).

Розробка ж компонент, зокрема, для роботи з графічними об'єктами та систем їх підтримки, як показує досвід в Delphi зручніший ніж в Visual Basic. Це підтверджують і класичні праці, зокрема [6].

РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ

При використанні символьного процесора, для спрощення задачі, деякі елементи системи можна поєднувати в мегаелементи і розглядати їх при подальшому аналізі та розрахунках як єдину сутність. Звісно, в цьому є і технологічна потреба, бо існують стандартні прилади і стандартні технологічні системи. Але, що найсуттєвіше, це дозволяє, по-перше, значно спростити постановку задачі для непрофесіонала в математиці, по-друге, скоротити час обчислень, по-третє, при використанні значної кількості таких мегаелементів навіть знизити вимоги до конфігурації PC.

Дійсно, різноманітне технологічне обладнання має складну структуру. Наприклад, ректифікаційну колону при розрахунках можна розглядати, як систему перегінних кубів. Але, при наявності відпрацьованої математичної моделі, доцільніше аналізувати її як мегаелемент. Випарна установка може розглядатись і як складна система тепло - масообмінних апаратів, і, коли вона є субструктурою більш складної системи – як єдине ціле. Багатоступеневий компресор з міжступеневими та кінцевим охолоджувачами теж може розглядатись як один елемент. Елемент, призначення якого є трансформація електричної енергії у механічну. Число таких прикладів можна подовжити для кожної з можливих галузей використання символьного процесору.

У випадку використання мегаелементів сукупність $\boldsymbol{n}$ рівнянь, що описують систему, з $\boldsymbol{n}$ змінними буде модифіковано до математично еквівалентної сукупності $(n - \sum m_i + i)$ – рівнянь, з, відповідно, $(n - \sum m_i + i)$ змінними, де $\sum m_i$ - сума змінних (m) в i-ій кількості мегаелементів. Рівняння, що відповідають елементам в структурі мегаелементів буде замінено символьними конструкціями - алгебраїчними формулами. Складність цих конструкцій буде залежати від кількості елементів в їх структурі та зв'язків між ними. Не дивлячись на розмір формул обсяг та час обчислень за ними буде набагато економнішим, ніж обчислення системи рівнянь, яка включала в себе рівняння, що описують елементи, включені в структури мегаелементів. Самі мегаелементи, як звичайні елементи, будуть описані i-ою кількістю рівнянь в загальній системи рівнянь системи.

При цьому, звісно, треба мати в наявності бібліотеку стандартних систем (мегаелементів), кожна з котрих, представляє, з одного боку, чорну скриньку, з

іншого - математичну модель, розраховану в символьному вигляді для довільного числа елементів. Число елементів повинно задаватись користувачем, у кожному окремому випадку в діалоговому режимі з програмою.

Базова програмна концепція рекомендує побудову проекту як ієрархічного дерева програм, які є одна з іншою в певних відносинах.

Для кожної із задач повинні бути розроблені відповідні моделі знань, які відображають закономірності, нормативну базу і досвід рішення даних задач. Використання ПЗ повинно здійснюється в трьох режимах: автоматичному, питання-та-відповідь (діалоговому) та як алгоритми дій. В режимі питання-та-відповідь оператор зможе одержати відповідь на будь-які поставлені системі питання з пропонованого списку по кожній із задач. В діалоговому режимі можуть бути одержані діагнози, сформовані рекомендації, видані екстрені повідомлення.

Вимоги до інтерпретатора наступні: за допомогою одних і тих же апаратних і програмних засобів забезпечити можливість сконструювати систему, що може виконувати різні функції і мати різний, призначений для користувача інтерфейс. Управління такими системами треба здійснювати через графічний інтерфейс (Graphics User Interface - GUI) за допомогою технології Drag-and-Drop ("переніс і поклав") з використанням маніпулювання мишею через віртуальні елементи управління, розташовані на віртуальних приладових панелях (дивись Мал. 1). Віртуальні приладові панелі повинні забезпечувати роботи по проектуванню систем в трьох різних галузях – теплообмінні мережі, масообмінні мережі (зокрема дисипативні гідравлічні мережі) і електричні мережі. Перехід від одних галузевих налаштувань до інших повинен супроводжуватись зміною бібліотек стандартних мегаелементів. Окрім бібліотек треба змінювати і арго інтерфейсу, бо тезаурус фахівців в названих галузях різниться один від одного.

Інтерпретатор повинен бути оснащений запобігачами від помилок користувача, аудіовізуальними попередженнями користувача. Безумовними компонентами інтерпретатора повинен бути блок вхідної інформації для символьного процесора, блок перевірки цієї інформації на логічні протиріччя. Інтерпретатор також повинен включати стандартні засоби автоматичного проектування додатків, такі, щоб споживач міг встановлювати контрольні крапки, представляти у вигляді стендової моделі виконання програми, так, щоб бачити, як дані проходять через програму крок за кроком, щоб спростити розуміння процесів, що відбуваються.

Як платформа запропонована система програмування Delphi фірми Borland, що забезпечує гнучкий механізм створення візуальних компонент ActiveX в спеціально створеному для цих цілей середовищі програмування, званої DelphiActiveX (DAX) [6]. Використовувати технологію ActiveX треба у специфікації TMX, яка дозволяє дістати легкий доступ до бази каналів проекту автоматизації. Механізм створення ActiveX-компонент в середовищі DAX включає п'ять етапів. Підключення створеного засобами DAX-компоненту виконується за допомогою редактора представлення даних, який дозволяє розмістити на екрані проекту ActiveX-компонент з складу зареєстрованих в

системі. При установці компоненту автоматично відкривається його сторінка властивостей, в якій виконується прив'язка властивостей компоненту до каналів інтерпретатора і вибирається напрям передачі даних: прив'язка типу ВХІД - значення властивості каналу передається властивості компоненту; прив'язка типу ВИХІД - значення властивості компоненту передається властивості каналу.

Як Recommended Standard (рекомендований стандарт) протоколу обміну даними запропоновано використання RS-485. Цей протокол є напівдуплексним і знімає головне обмеження симплексних протоколів - односторонній зв'язок. Він дозволяє двом пристроям обмінюватися інформацією, причому обидва пристрої можуть бути і приймачами і передавачами, але не одночасно. Ці рекомендації прийняті з тієї точки зору, що символьні процесори можна використовувати в АСУТП (SCADA - системах) де формування моделі буде відбуватися на віддаленому сервері, а виконання – на робочих станціях.

ВИСНОВКИ

1. Сформульовані вимоги та рекомендації до створення інтерпретатора символьного процесора.
2. Вказано на неможливість використання стандартних інтерпретаторів для символьного процесора.
3. Рекомендовано для створення інтерпретатора використання утиліт та компонент Delphi.
4. Вказано на необхідну та достатню вимогу для інтерпретатора - використання стандартних бібліотек для стандартних груп елементів з наступним включенням їх у математичний аналіз в якості елемента (чорної скриньки).

ЛІТЕРАТУРА

1. Yee T. F., Grossmann I. E. and Z. Kravanja, Z. Simultaneous Optimization Models for Heat Integration – I. Area and Energy Targeting and Modeling of Multi – Stream Exchangers, Comp. and Chem. Eng., 14(10):1151 – 1164, 1990.
2. Yee T. F. and I. E. Grossmann, Simultaneous Optimization Models for Heat Integration – II. Heat Exchanger Network Synthesis., Comp. and Chem. Eng., 14(10):1165 – 1184, 1990.
3. Компания **ХОЛИТ Дейта Системс:** ISaGRAF. [Електронний ресурс] — Режим доступу: http://www.isagraf.com.ua
4. Дейкстра Э.В., Заметки по структурному программированию. [Електронний ресурс] — Режим доступу: http://khpi-iip.mipk.kharkiv.edu/library/extent/dijkstra/ewd249/index.html
5. Сайт фірми Iotech. [Електронний ресурс] — Режим доступу: www.iotech.com/

6. Конопка Рей. “Создание оригинальных компонент в среде Delphi” - К.:НИПФ - “ДиаСофт Лтд.”, 1996.- 512с.

ИНТЕПРЕТАТОР ДЛЯ НОВОГО ПОКОЛЕНИЯ СИСТЕМ АВТОМАТИЧЕСКОГО ПРОЕКТИРОВАНИЯ СЕТЕЙ

С. В. Котенко

Резюме

Приведены принципы построения интерпретатора для систем автоматического проектирования энерго и массообменных сетей.

INTEPRETATOR FOR A NEW GENERATION OF SYSTEMS OF AUTOMATIC NET-DESIGNING

Kotenko S.V.

Summary

Article is devoted by principles of construction of interpreter for the systems of the automatic designing energy and mass exchange networks.