УДК 681.51 : 004.942

ЗАБЕЗПЕЧЕННЯ ШВИДКОЇ ЗМІНИ МОДЕЛІ ПОВЕДІНКИ У ВБУДОВАНИХ СИСТЕМАХ РЕАЛЬНОГО ЧАСУ

А. В. Ярмілко, М. Ю. Багінський
Черкаський національний університет імені Богдана Хмельницького
бул. Шевченка, 81, м. Черкаси, 18000, Україна. E–mail: a-ja@ukr.net

Розглянуто проблему забезпечення швидкої зміни моделей у вбудованих системах реального часу. Запропоновано можливі варіанти та критерії оцінки ефективності її вирішення. Визначено перспективні засоби програмної реалізації задачі зміни моделей. Розроблено методику кількісної оцінки запропонованих методів і засобів. На основі синтетичних моделей експериментально досліджено властивості інтерпретаторів скриптових мов, механізму DLL і середовища Matlab як зовнішнього інструменту забезпечення зміни моделей. Подано рекомендації до забезпечення оптимального використання розроблених методів і засобів у вбудованих системах управління реального часу.

Ключові слова: модель, критерії ефективності, вбудовані системи, системи реального часу, модельно-орієнтоване управління.

ОБЕСПЕЧЕНИЕ БЫСТРОЙ СМЕНЫ МОДЕЛЕЙ ПОВЕДЕНИЯ ВО ВСТРОЕННЫХ СИСТЕМАХ РЕАЛЬНОГО ВРЕМЕНИ

А. В. Ярмилко, М. Ю. Багинский
Черкасский национальный университет имени Богдана Хмельницкого
бул. Шевченко, 81, г. Черкассы, 18000, Украина. E–mail: a-ja@ukr.net

Рассмотрена проблема обеспечения быстрой смены моделей во встроенных системах реального времени. Предложены возможные варианты и критерии оценки эффективности её решения. Определены перспективные средства программной реализации задачи смены моделей. Разработана методика количественной оценки предложенных методов и средств. На основе синтетических моделей экспериментально исследовано свойства интерпретаторов скриптовых языков, механизма DLL и среды Matlab в качестве внешних инструментов обеспечения смены моделей. Представлены рекомендации по обеспечению оптимального использования разработанных методов и средств во встроенных системах управления реального времени.

Ключевые слова: модель, критерии эффективности, встроенные системы, системы реального времени, модельно-ориентированное управление.

АКТУАЛЬНІСТЬ РОБОТИ. Інтелектуалізація сучасних систем управління супроводжується широким застосуванням моделей різноманітного призначення та складності, які забезпечують функціонування різних рівнів прийняття рішень (стратегічний, тактичний, керуючий), розпізнавання образів, прогнозування, відновлення даних тощо. Існуюча модельно-орієнтована парадигма пропонує широке використання комп'ютерних моделей об’єктів та процесів у контурах управління у масштабі реального часу [1]. За наявності великої кількості актуальних для конкретної системи моделей та/або стратегій поведінки виникає необхідність впровадження механізмів заміни моделі при виконанні конкретних функціональних завдань. Наявність такого механізму з відповідними прикладній задачі динамічними характеристиками може бути засобом надання необхідної гнучкості виконавчим або керуючим органам.

Такі вимоги, зокрема, висуває концепція формування економічно ефективної стратегії керування інтелектуальним технологічним модулем на основі вибору оптимальної стратегії функціонування [2]. Даний підхід поєднує принципи адаптивності та оптимальності управління та використовує математичний апарат теорії ігор для вибору поточної стратегії. Проте цільова платформа – вбудовані системи реального часу – накладає жорсткі обмеження на часові та матеріальні ресурси. Тому одержання достовірних ефективних рішень за умов ризику, невизначеності, нечіткості, неточності, стохастичності та трудомісткості процесів у ряді випадків стикається з суттєвими труднощами [3].

Виходячи із вище зазначеного, метою роботи є вироблення методів, критеріїв і метричних характеристик, які б дозволили адекватно оцінити та вибрати оптимальний спосіб зміни моделі при практичній реалізації систем управління, а також апробація методів зміни моделей в автоматизованих системах реального часу.

МАТЕРІАЛ І РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ. Задачі, які розв’язувались, – це розробка та дослідження механізмів швидкої зміни моделей у вбудованих адаптивних системах реального часу; можливі варіанти взаємодії математичних середовищ і прикладних систем, способи представлення моделей, механізми заміни моделей управління систем керування в реальному масштабі часу; визначалися критерії оцінки моделей та їх метричні характеристики. Експериментальна частина мала на меті апробацію запропонованої системи кількісних та якісних критеріїв оцінки та їхнє застосування стосовно розглянутих методів швидкої зміни моделей. У даному дослідженні обмеження та оптимальність топології системи не розглядалися.

Аналіз існуючих технологій дозволив виявити кілька перспективних для впровадження у вбудовані системи реального часу методів швидкої зміни моделей (моделей управління в тому числі). В цілому, запропоновані методи можна розділити на дві категорії. Перша – заміна параметрів моделі. Друга – заміна власне моделі. Прикладом моделі

першого випадку може бути поліном Колмогорова-Габора. При необхідності змінити поведінку моделі, необхідно змінити коефіцієнти при членах полінома. Другий підхід передбачає не тільки модифікацію параметрів моделі, а й зміну її виду. Наприклад, спочатку для прогнозування деякого параметра використовується проста модель лінійної регресії, яку потім буде замінено на поліном Колмогорова–Габора.

Якщо говорити про способи практичної реалізації, то в першому випадку модель може бути написана на мові програмування високого рівня, скомпільована під архітектуру виконавчого пристрою та зашита в ПЗП виконавця, а інформація про параметри моделі може надходити із-зовні по каналах зв'язку від керуючого пристрою до виконавчого. Другий випадок не є настільки тривіальним. Різні моделі можуть мати кардинально різні способи реалізації, а їх узагальнення і пошук способів уніфікації, для представлення різних видів моделі через однаковий інтерфейс, не завжди можливий.

Вирішення задачі організації середовища функціонування моделі може здійснюватися двома способами.

Перший – реалізація власних інструментів для створення та відпрацювання моделей в масштабах реального часу.

Інший варіант – використання сторонніх апробованих рішень, які забезпечують отримання моделей заданої якості. У такому випадку для інтеграції отриманих моделей у виконавчі пристрої необхідно лише створити інструменти обміну між середовищами генерації та відпрацювання моделей.

Деякі математичні середовища надають досить великі можливості для інтеграції.

Так, математичний пакет Matlab надає різні варіанти взаємодії середовища із зовнішніми програмами. Тому можливе використання Matlab як засобу, який буде виконувати модель. Дане середовище досить потужне і надає зручний рівень абстракції для вирішення різних типів задач. Тому при використанні Matlab у ролі середовища для створення та виконання моделі можна отримати додаткові переваги в забезпеченості засобами роботи із складними даними, гарантоване функціонування інструментів, які надаються системою. Варто також зауважити, що система Matlab у цілому є досить громіздкою, при цьому для роботи використовується віртуальна машина Java. Тому можливість використання Matlab у вбудованих та системах реального часу є досить сумнівною.

Іншою пропонованою альтернативою є використання механізму, який часто застосовується при необхідності розширення функціональності деякого програмного засобу – використання скриптових мов. Такий підхід часто використовується в CAD-системах, пакетах 3D графіки, комп'ютерних іграх.

Одною із скриптових мов, яка заслуговує уваги, є Python. Python є стандартним компонентом *nix систем і використовується як засіб для розробки невеликих утиліт. Відповідно, для даної мови розроблено досить багато бібліотек і модулів для виконання різних задач. Так, завдяки проекту Sage – відкритій математичній бібліотеці для Python [4], – інколи в наукових колах Python використовується як вільний замінник Matlab. Враховуючи це, Python може бути адекватною альтернативою Matlab в плані обробки та аналізу даних, створення моделей. Разом з тим, інтерпретатор Python може бути використаний як виконавчий елемент розроблених моделей. Варто відмітити можливість використання мови для вбудованих систем, оскільки доступним є вихідний код, який може бути зібраний під необхідну платформу.

Інша скриптова мова, яку варто розглядати в контексті пошуку засобів для представлення та виконання моделей – Lua. Як і Python, дана мова є відкритою і має доступну програмну реаізацію. Для роботи Lua використовує компілятор в байткод і віртуальну машину для виконання згенерованого байткоду. Варто відмітити ще те, що для даної мови розроблено компілятор LuaJIT. Даний компілятор забезпечує умови для виконання програмного коду з великою швидкістю та може бути використаний для критичних задач [5]. Реалізація мови компактна і, як зазначено раніше, відкрита. Тому є можливість використання мови у вбудованих системах. На відміну від Python, для Lua не створено такої великої кількості бібліотек, тому реалізацію частини математичного апарату потрібно реалізовувати в C/C++. Але це дозволяє використовувати для складних обрахунків оптимізований код, скомпільований під конкретну платформу.

Ще одним вартим уваги варіантом є заміна моделей як скомпільованого програмного коду. Даний підхід повинен забезпечити найбільшу швидкість, проте вимагає додаткового дослідження в плані використання для вбудованих систем.

Таким чином, за наявності кількох можливих варіантів вирішення задачі інтеграції зовнішніх моделей в системи управління реального часу, постає питання оптимальності їх вибору для конкретної ситуації. Інструменти розробки та виконання моделей, виходячи із загального трактування поняття ефективності [6], запропоновано оцінювати за наступними критеріями:

– швидкість виконання моделі;
– швидкість зміни моделі;
– вимоги до розміру ОЗП виконавчого пристрою;
– можливість інтеграції у вбудовані системи;
– тип, математичний апарат і складність моделей;
– складність створення моделі;
– складність та збитковість моделі;
– об'єм моделі та форма представлення.

На основі комплексної оцінки за даними критеріями можуть бути обрані найбільш ефективні рішення у реалізації систем адаптивного управління за модельно-орієнтованим підходом. Для обрахування значень оцінки необхідно було визначити вид характеристики за конкретним критерієм, спосіб їх отримання та узагальнення для встановлення параметрів запропонованих методів зміни моделей у системах реального часу – використання скриптових мов, динамічних бібліотек, внутрішньої мови

і середовища Matlab – з урахуванням умов, які висуваються цільовою платформою.

Однією з головних метричних характеристик ефективності зазначених методів є швидкість виконання різних етапів роботи з моделлю. Так, умовно, роботу із моделлю, яка задається програмним кодом, можна поділити на чотири етапи. Перший – ініціалізація середовища. У випадку використання для представлення моделі скриптових мов Lua, Python, внутрішньої мови Matlab робота на даному етапі буде полягати в ініціалізації інтерпретаторів або середовища Matlab. Зауважимо, що при використанні DLL для представлення моделей цей крок не виконується.

Наступний крок – завантаження представлення моделі, яка буде виконуватись на наступному етапі функціонування вбудованої системи. При цьому інтерпретатори завантажують файл із раніше підготовленим скриптом. Якщо для представлення моделі використовується код, скомпільований в DLL, то робота полягатиме у завантаженні власне динамічної бібліотеки та пошуку там адреси необхідної функції.

Варто відмітити, що передача моделі при використанні механізму DLL може реалізовуватися двома способами. Перший спосіб полягає в тому, що система отримує програмний код, виконує його компіляцію та лінковку, і вже потім завантажує як динамічну бібліотеку. При іншому підході системі може надсилатися вже зібрана динамічна бібліотека, яка зразу завантажується у її адресний простір. Переваги є як в першого, так і другого способів, і вибір необхідного варіанту може відбуватись шляхом аналізу слабких місць в апаратному забезпеченні системи. При цьому слід звертати увагу на швидкість центрального процесора та швидкість комунікативної мережі або виходити з аналізу архітектури системи, способу поділу її та виконуваних задач на різних рівнях. При використанні першого методу завантажувач моделі отримує невеликий файл, що зручно з точки зору передачі моделі, а потім компілює його в динамічну бібліотеку. При використанні математичних бібліотек для підвищення рівня абстракції коду моделі та при використанні великої кількості функціоналу в код моделі буде включено багато header-файлів. Це призведе до тривалого часу компіляції навіть невеликої моделі. При використанні другого підходу загальне навантаження на адаптивну систему буде менше. Отриману динамічну бібліотеку можна швидко і легко завантажити. Проте, зазвичай, скомпільовані вихідні коди мають в рази більший розмір і, таким чином, створюють навантаження на мережу при передачі моделі.

Наступний етап – власне виконання моделі. Система передає деякі параметри, викликає на виконання завантажену раніше модель та отримує результат виконання.

Останній етап, який не є визначальним – вивантаження середовища. Час, витрачений на виконання даної операції, суттєвий тільки в тому випадку, якщо в системі управління необхідне постійне перезавантаження модуля заміни моделей.

Для дослідження зазначених властивостей методів зміни моделей необхідно мати тестову модель, яка ідентично проявлятиме себе при різних способах її представлення. Враховуючи те, що в адаптивних системах переважно використовуються моделі у вигляді систем рівнянь, то під час обрахування моделі мають виконуватися математичні операції.

Для визначення швидкісних параметрів вище згаданих методів запропоновано використання бенчмарків, суть яких полягає у виконанні деяких операцій, які супроводжуються збиранням статистичної інформації: час виконання, кількість кадрів за секунду (при тестуванні графічних складових), час читання-запису та ін. У загальному, всі тести можна поділити на два типи.

Перший тип тестів імітує роботу реальних додатків. Так, загальновідомі бенчмарки серії 3DMark імітують ігрові додатки, що дозволяє отримати комплексну оцінку системи і визначити її придатність до виконання конкретного типу задач. Другий тип тестів – це, так звані, синтетичні тести. Наприклад, бенчмарк PiNumber використовується для оцінки швидкості роботи процесора шляхом вимірювання часу, за який даний пристрій може визначити число p з деякою точністю.

Для проведення експериментів було створено набір уніфікованих за властивостями тестових моделей.

Розроблена модель реалізується комбінацією «легких» (віднімання, додавання, зміна знаку) та «важких» (множення–ділення, показникові та тригонометричні функції) математичних операцій. Загальне математичне представлення тестової моделі має наступний вигляд:

$$\sum_{i=1}^{1000} i + \frac{\sin(i)^i}{i}. \qquad (1)$$

Дослідження швидкості роботи засобів зміни моделей проводилося з використанням оригінальних модулів відпрацювання моделі на основі використання скриптової мови Lua у двох її реалізаціях (Lua та LuaJIT), які забезпечують різні часові характеристики відпрацювання моделей, скриптової мови Python та на основі використання механізму динамічних бібліотек: LuaSpeedTest, PythonSpeedTest, CppSpeedTest.

При проведені експериментів використана наступна програмно-апаратна платформа: процесор – Intel® Core™ i3 CPU M 350 @ 2.27 Гц Ч 4; ОЗП – 3,8 ГБ DDR3; HDD – 500 ГБ; Linux Ubuntu 12.04 LTS; версія ядра – 3.2.

Було проведено вимірювання швидкості виконання різних етапів роботи із моделлю. Для вимірювання використовувалися вбудовані в тестуючу програму засоби. У табл. 1 наведено виміри тривалості виконання етапів запропонованих у роботі засобів зміни моделі.

Крім метрик швидкості, для оцінки ефективності вище було запропоновано ще ряд критеріїв, значимих при виборі засобів для конкретного випадку.

Таблиця 1 – Тривалість виконання етапів роботи при відпрацюванні синтетичної моделі

Засіб	Ініціалізація середовища	Відкривання моделі	Виконання моделі	Звільнення середовища
Lua	108	48	631	21
LuaJIT	65	32	206	31
DLL	0	33	205	7
Python	17726	98	976	3819

Проте, отримані за цими критеріями характеристики є недостатньо формалізованими і тому дають лише якісну оцінку при обґрунтуванні придатності засобів зміни моделей для певних випадків. Тому, для врахування при обчисленні загального показника ефективності, описові оцінки були замінені коефіцієнтами на основі порівняльного аналізу. Для цього були експертно встановлені показники даних критеріїв у вигляді вагових коефіцієнтів.

Також для розрахунку була встановлена оцінка важливості критеріїв у формі вагових коефіцієнтів (табл. 2).

Такий спосіб формалізації достатній для використання методу аналізу ієрархій при виборі способів заміни моделей. Варто зауважити, що важливість критеріїв дуже залежить від випадку використання і має оцінюватися відповідно до вимог, встановлених задачами розроблюваної системи.

Таблиця 2 – Вагові коефіцієнти критеріїв оцінки ефективності

Критерії	Коефіцієнт
Швидкість ініціалізації середовища	0,04
Швидкість завантаження моделі	0,29
Швидкість виконання моделі	0,3
Швидкість звільнення середовища	0,02
Розмір ОЗП виконавчого пристрою	0.01
Можливість інтеграції у вбудовані системи	0,2
Типи та складність моделей	0,02
Складність створення моделі	0,01
Складність та збитковість моделі	0,01
Об’єм моделі	0,1

Результати обрахунку комплексних оцінок засобів зміни моделей для досліджених методів наведені у табл. 3.

Таблиця 3 – Комплексних оцінок засобів зміни моделей

Засіб зміни моделей	Комплексна оцінка
Python:	0,277
DLL (C++)	0,345
LuaJIT:	0,358

Як видно, при заданих вагах критеріїв найбільшу комплексну оцінку має спосіб зміни моделей на основі використання інтерпретатора LuaJIT. Зазначимо, що спосіб представлення моделей на внутрішній мові Matlab на даному етапі дослідження не розглядався, оскільки він, як було обґрунтовано вище, малопридатний для використання у вбудованих системах реального часу.

ВИСНОВКИ. У представленому дослідженні запропоновано механізми забезпечення швидкої зміни моделей у вбудованих адаптивних системах реального часу. Такими методами є зміна моделі на основі скриптових мов Python та Lua, механізму DLL та використання середовища Matlab. Використання останнього з вказаних методів для вбудованих систем реального часу є проблематичним внаслідок великих вимог до ресурсів. Найбільш перспективним вбачається застосування скриптових мов. Проте цей метод може забезпечити найкращі результати при інтеграції більшої частини математичного апарату у модуль зміни моделі та описові моделі в скриптах на найвищому рівні абстракції, можливому в даній системі за конкретної її реалізації.

Вироблено критерії і методи оцінки ефективності запропонованих методів зміни моделей. Встановлено, що конкретна топологія програмно-апаратного рішення, яке б задовольняло зазначеним критеріям, залежна від задачі, яка вирішується. При її виборові необхідно керуватися формалізованими критеріями оцінки ефективності можливих варіантів на основі метричних характеристик кожного з методів.

Виходячи із цільової платформи – вбудовані системи реального часу, – визначено два основні критерії ефективності засобів зміни моделей: розмір ОЗП, потрібний для функціонування програмної системи, та швидкість роботи метода.

Дослідження метричних характеристик запропонованих методів, здійснене за допомогою оригінальних синтетичних тестових моделей, продемонструвало суттєву варіацію їхньої продуктивності. На основі запропонованих критеріїв та за допомогою алгоритму аналізу ієрархій було встановлено, що найефективнішою реалізацією механізму швидкої зміни моделей виявився алгоритм, заснований на використанні скриптової мови Lua. При цьому метод проявив чутливість до використання конкретного інтерпретатора даної мови. Зокрема, найвищу продуктивність забезпечив інтерпретатор LuaJIT.

Варто відмітити, що результуюча оцінка методу дуже залежить від вагових коефіцієнтів критеріїв, які визначаються із конкретної задачі, прикладної області, та умов застосування. У даному випадку при аналізові отриманих результатів головними вважалися критерії швидкості завантаження моделі, її виконання, розміру моделі, які висуваються при розробці засобів зміни моделей для автономних роботів [7].

При створенні вбудованих систем реального часу запропоновано використовувати підхід, який базується на наступних принципах:

1. При проектуванні кібернетичної системи потрібно врахувати множину математичного апарату яка буде або може використовуватися майбутніми моделями.

2. Виконати можливі апаратні оптимізації (використання SIMD наборів команд для операцій із наборами даних – векторами, матрицями), оптимізації логіки і випадків використання. Так, значного ефекту можна досягнути окремою реалізацією операцій для 2–4 вимірних векторів, матриць. Загальні реалізації дозволяють написати код, який буде працювати у будь-яких випадках. Але реалізації часткових випадків у більшості дозволяють спростити код і збільшити швидкість роботи для конкретного рішення.

3. Виконати інтеграцію реалізованого математичного апарату в середовище зміни моделі.

4. При створенні моделей максимально використовувати розроблені раніше математичні абстракції, які надаються хостовою програмою.

Забезпечені таким чином оптимізації та високий рівень абстракції скриптових моделей дозволять підвищити швидкодію та гнучкість системи керування.

Подальші дослідження передбачають визначення ефективності запропонованих методів швидкої зміни моделей на різних апаратно-програмних платформах і при використанні моделей різних типів.

ЛІТЕРАТУРА

1. Модельно-ориентированное управление как стратегия функционирования интелектуальных производственных систем / В.В. Литвинов, В.В. Казимир // Математичні машини і системи. – 2004. – № 4. – С. 143–156.

2. Ярмілко А.В. Формування стратегії керування технологічним модулем за даними поточного моніторингу та експрес-діагностики // Математичні машини і системи. – 2013. – № 1. – С. 102–110.

3. Кучеренко В.Є. Автоматизоване управління складними об’єктами за умов невизначеності на основі гібридних імітаційних моделей: автореф. дис... канд. техн. наук: 05.13.06 / Харк. нац. ун-т радіоелектрон. – Х., 2007. – 19 с.

4. Sage: Open Source Mathematics Software [Електронний ресурс]. – Режим доступу: http://www.sagemath.org/

5. LuaJIT: Performance Comparison. [Електронний ресурс]. – Режим доступу: http://luajit.org/performance.html

6. Juran J.M., Gryna F. Quality Control Handbook. M.: Mcgraw-Hill, 1998. – 1774 p.

7. Звенігородський О.С. Інтелектуальна система планування тактики руху автономного робота в квазістаціонарному середовищі: автореф. дис. на здобуття наук. ступеня канд. техн. наук: спец. 05.13.23 / Звенігородський Олександр Сергійович. – Д., 2002. – 27 с.

ENSURING THE RAPID CHANGING OF BEHAVIOR MODELS IN THE IMBEDDED REAL-TIME CONTROL SYSTEMS

A. Yarmilko, M. Bahinskyi

Bohdan Khmelnytsky National University at Cherkasy

blv. Shevchenko, 81, Cherkasy, 18000, Ukraine. E–mail: a-ja@ukr.net

The authors discuss a problem of the rapid model changing in embedded real-time systems. The most appropriate options, feasible variants and estimation criteria of the efficiency of this problem solutions were proposed in the article. Also the advanced tools and facilities of real-time models changing program implementation were defined. The authors have presented the developed technique and tools for quantitative evaluation of the proposed methods. The article also focuses on testing the methods of models changing in embedded real-time systems. The experimental research results of the model changing methods properties based on using Matlab engine, performed by the authors, are discussed in the article, as well as scripting languages and shared library concept developed with use of synthetic models. In the end of the paper there are the usage recommendations for the methods and tools developed for the real-time embedded control systems specified.

Key words: model, criterions of efficiency, embedded systems, real-time systems, model-driven engineering.

REFERENCES

1. Litvinov, V.V., Kazymyr, V.V. (2004), “Model-oriented control as strategy of functioning of the intelligent manufacturing systems”, *Mathematical machines and systems*, no. 4, pp. 143–156., IMMSP NASU, Kyiv, Ukraine.

2. Yarmilko, A.V. (2013), “The formation of the control strategy of the technological unit based on the current monitoring and rapid diagnosis”, *Mathematical machines and systems*, no. 1, pp. 102–110. IMMSP NASU, Kyiv, Ukraine.

3. Kucherenko, V.Ye. (2007), “Automated complex objects process control in terms of uncertainly based on hybrid imitating models”, Thesis abstract Cand.Sc. (engineering), spec. 05.13.06 Automated control systems and progressive information technologies, Kharkiv National University of Radio-Electronics, Kharkiv, Ukraine.

4. “Sage: Open Source Mathematics Software”, available at: http://www.sagemath.org/

5. “LuaJIT: Performance Comparison”, available at: http://luajit.org/performance.html

6. Juran, J.M., Gryna, F. (1998), Quality Control Handbook. M.: Mcgraw-Hill. – 1774 p.

7. Zvenigorodsky, A.S. (2002), “Intelligent planing system of movements autonomous robot in half stationary space”, Thesis abstract Cand.Sc. (engineering), spec. 05.13.23 Facilities and systems of artificial intelligence, Donetsk State Institute for Artificial Intelligence, Donetzk, Ukraine.

Стаття надійшла 15.02.2013.