УДК 004.312

**Є.В. Нікітенко,** канд. фіз.-мат. наук
**Р.В. Заровський,** канд. техн. наук
**С.В. Микитко**, магістрант
Чернігівський державний технологічний університет, м. Чернігів, Україна

# РОЗПОДІЛЕНА СИСТЕМА ОБРОБКИ ТА ВИВОДУ ВІДЕОДАНИХ

*Розглянуті проблеми побудови розподілених систем обробки та виводу відеоданих. Запропоновані методи вирішення виявлених проблем. Розроблена архітектура розподіленої системи обробки та виводу відеоданих.*

## Вступ

Сучасний рівень розвитку електроніки та обчислювальної техніки дозволяє будувати високопродуктивні комп'ютерні системи цифрової обробки одновимірних та двовимірних сигналів. Одним з яскравих прикладів можливостей таких систем є побудова відеостін.

Основне призначення відеостін – великомасштабне відображення інформації для колективного перегляду в диспетчерських і ситуаційних центрах, на пультах управління, а також у різних автоматизованих системах управління. Вони застосовуються в тих сферах, де необхідний оперативний контроль над інформацією і де виключно велика відповідальність за прийняті управлінські рішення: в енергетиці, на транспорті, телекомунікаціях, промисловості, у системах забезпечення безпеки, в управлінні фінансами, залах засідань, фінансових біржах, а також у рекламній сфері та сфері розваг [1]. Таким чином, у зв'язку з таким різноплановим використанням цієї системи необхідно забезпечити її універсальність для кожного роду застосування. Це означає, що сторонньому розробнику необхідно надати можливість самостійно розширювати функціональні можливості системи.

## Постановка проблеми

З технічної точки зору основна проблема побудови комп'ютерної системи, що надає зображення на відеостіну від багатьох відеоджерел, – це забезпечення можливості розширення розмірів самої відеостіни, розширення її програмної функціональності у вигляді додавання нових алгоритмів обробки відеоданих і підключення більшої кількості відеоджерел. Забезпечення таких можливостей пов'язане з нарощуванням обчислювальної потужності комп'ютерної системи на етапі її функціонування. При цьому також існує проблема управління величезним потоком відеоінформації в мережах загального призначення. Рішення цієї проблеми вимагає застосування особливих алгоритмів керування, які передбачають планування маршрутів проходження інформації [2].

Цю систему раціонально організувати як розподілену. При використанні такої топології за кожну клітинку відеостіни відповідає окремий апаратний модуль, який керується централізовано. Таке рішення дозволить досягти високої масштабованості системи, а також ряд інших переваг, таких як зниження загальної вартості системи, зменшення енергоспоживання. Також система підтримує можливість динамічного підключення додаткових програмних модулів обробки відеоінформації.

## Викладення основного матеріалу дослідження

Виходячи з аналізу необхідної функціональності, а також таких властивостей, як масштабованість і розподіленість, була запропонована архітектура системи, яка представлена на рисунку 1.

Для реалізації підсистем трансляції відеопотоку й управління відеовиводом можна скористатися технологією DaVinci. Вона являє собою комплекс засобів для конструювання багатофункціональних відеопристроїв. До складу технології входить цифровий сигнальний процесор (DSP), оптимізований для обробки відеоданих, та вбудоване програмне забезпечення, комбінації яких дозволяють створювати уніфіковані відеовідтворювальні і записуючі пристрої. Одним з принципових особливостей платформи DaVinci є розвинені можливості зв'язку таких пристроїв один з одним.

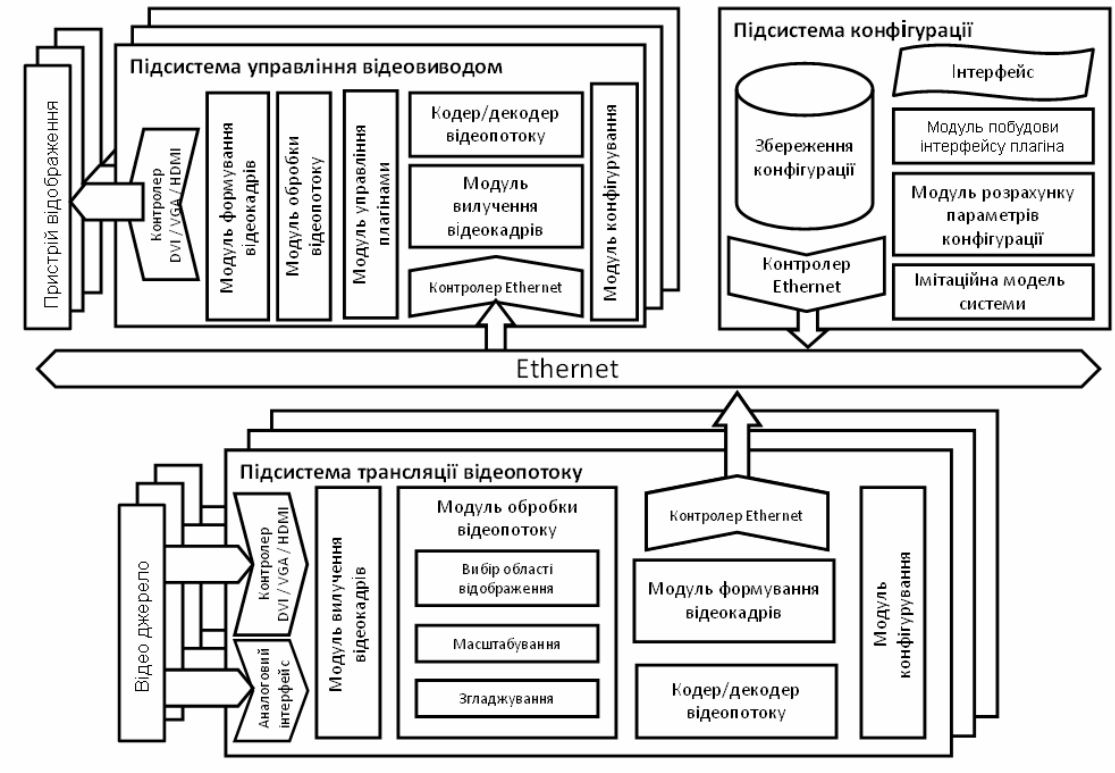


*Рис. 1. Архітектура системи*

Наявність у процесорі ядра ARM9 дає можливість застосування RTOS на базі Linux, що дозволяє використовувати велику кількість готових бібліотек і рішень, зменшити час розробки та спростити її за рахунок великої кількості готових функцій. Також використання Linux дає можливість динамічно завантажувати бібліотеки, які відіграють роль плагіна, що динамічно додається.

В якості середовища передачі інформації між рознесеними в просторі модулями можна використовувати Ethernet, який дає можливість побудови трирівневої розподіленої архітектури. На нижньому рівні архітектури знаходиться апаратне забезпечення: маршрутизатори, комутатори та ін. На середньому рівні знаходиться оверлейна мережа, що дозволяє об'єднати вузли, доступні тільки через загальну мережу. На вищому рівні знаходяться джерела відеоінформації, система управління відеостіною, модулі відображення відеоінформації на відеостіну [3].

**Структура програмного забезпечення**

Оскільки дана система забезпечує роботу з деякою кількістю відеоджерел, рознесених у просторі, а вивід на відеостіну передбачає підключення декількох пристроїв відображення, то найбільш ефективним буде застосування розподіленої архітектури системи.

У системі можна виділити 3 підсистеми:

- підсистема трансляції;
- підсистема управління відеовиводом;
- підсистема конфігурування.

Підсистема конфігурування необхідна для роботи оператора відеостіни. Її функцією є управління параметрами відображення різних відеопотоків на відеостіні. Пристрої керування відеовідображенням забезпечують прийом відеопотоку від джерела і його подальшу підготовку для виводу на екран. Підготовка відеопотоку до виведення включає декодування відео, виділення області відображення та масштабування зображення, що виводиться.

Для забезпечення гнучкості налаштування джерела відеопотоку підключаються в систему через спеціалізовані пристрої передачі (підсистема трансляції), які, крім гетерогенності джерел відеосигналу, забезпечують оптимізацію роботи системи.

Плагін – незалежно компільований програмний модуль, що динамічно підключається до основної програми, призначений для розширення її можливостей.

Запропоновано організувати плагін у вигляді двох файлів, один з яких призначений для апаратних модулів системи, на яких встановлена операційна система Linux, а другий є доповненням до програмного забезпечення (ПЗ), за допомогою якого і здійснюється централізоване конфігурування системи. Плагін підключається до апаратного модуля і буде здійснювати обробку відеозображення [4]. З технічної точки зору він є бібліотекою Linux, яка динамічно завантажується. Доповнення підключається до керуючого ПЗ і відповідає за налаштування плагіна на апаратному модулі. Це доповнення є XML-файлом, який описує інтерфейс користувача, необхідний для налаштування параметрів плагіна. Цей спосіб опису інтерфейсу користувача є універсальним і не залежить від особливостей платформи, на якій розроблено ПЗ керування відеостіною.

Ця комп'ютерна система видає зображення на відеостіну і підтримує накладання відразу декількох відеоефектів на зображення. Для взаємодії основної програми і плагіна необхідний програмний інтерфейс. Продуманість і чіткий опис цього інтерфейсу дозволяє стороннім розробникам безперешкодно розробляти плагіни для цієї комп'ютерної системи [5].

Підключення плагіна до апаратного модуля здійснюється засобами операційної системи Linux. У результаті плагін надає набір функцій, назви яких чітко визначено описом інтерфейсу, що необхідно для ідентифікації функцій основною програмою. Набір і опис функцій плагіна, що підключається до апаратного модуля, наведено в таблиці 1.

Таблиця 1

*Опис функцій плагіна апаратного модуля*

| Назва | Опис |
|---|---|
| getName | Повертає назву підключеного плагіна |
| getVersion | Повертає версію підключеного плагіна |
| getId | Повертає унікальний ідентифікаційний номер плагіна |
| getDescription | Повертає опис плагіна |
| doFilter | Здійснює обробку RGB зображення, що представлено двовимірним масивом |

Таким чином, можна представити структуру плагіна, що підключається до апаратного модуля. Структура плагіна показана на рисунку 2.

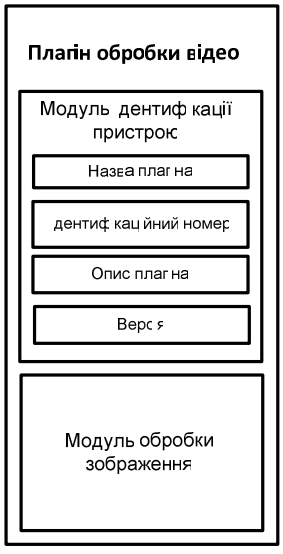


*Рис. 2. Структура плагіна, що підключається до апаратного модуля*

Оскільки на зображення, що виводиться, може накладатися відразу кілька відеоефектів, необхідно сказати, що зображення буде оброблятися кожним плагіном по черзі. Положення у черзі задається адміністратором системи. Також плагін бере на себе відповідальність за вилучення та обробку параметрів, що адресовані йому від керуючого ПЗ. З цього випливає,

що параметри, прийняті з керуючого ПЗ, передаються наскрізь у плагін, тобто як безтипові дані по відношенню до основної програми апаратного модуля. Необхідно сказати, що інформація про самі плагіни, що доступні на цьому апаратному модулі, буде завантажена під час завантаження модуля, а також за приходом службової команди з керуючого ПЗ про додавання нового плагіна. Основна програма модуля містить список всіх доступних на цей момент плагінів і коротку інформацію про них, а саме ідентифікаційний номер, назву плагіна, версію та ім'я файлу, що являє собою сам плагін. Таким чином, плагіни будуть завантажені в пам'ять модуля тільки тоді, коли вони беруть участь у ланцюжку обробки відеозображення. На рисунку 3 показана структура списку доступних плагінів.

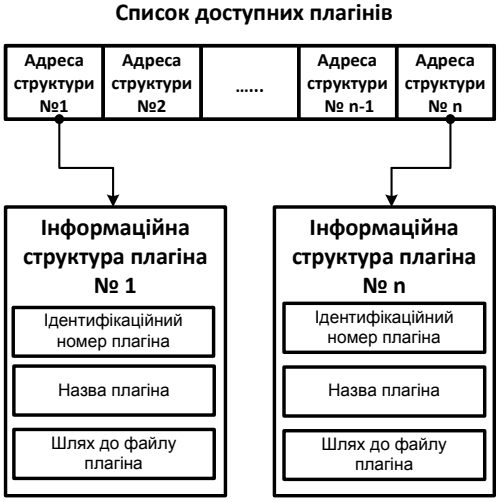


*Рис. 3. Структура списку доступних плагінів*

Відеозображення обробляється певним списком плагінів і в певній послідовності, що задано адміністратором, будується ланцюжок обробки. Всі плагіни в цьому ланцюжку повинні бути завантажені в пам'ять модуля, тому що вони містять функцію обробки відео, яка буде викликана для накладення ефектів на зображення. Схема ланцюжка обробки зображення показана на рисунку 4.

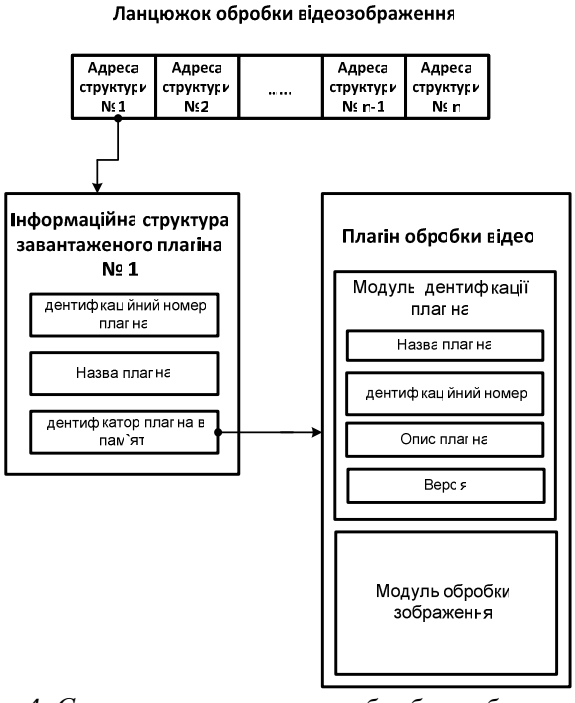


*Рис. 4. Структура ланцюжка обробки зображення*

Плагіни в цій розподіленій комп'ютерній системі поширюються централізовано з керуючого ПЗ на програмне забезпечення апаратних модулів. Таким чином, установка плагінів на апаратний модуль відбувається віддалено.

Для налаштування плагіна необхідний інтерфейс користувача. Опис інтерфейсу здійснюється за допомогою технології XML. Це значить, що інтерфейс описується XML-файлом, який, у свою чергу, аналізується керуючим ПЗ і на основі якого власне і відбувається побудова графічного інтерфейсу. Структура XML-файлу показана на рисунку 5.

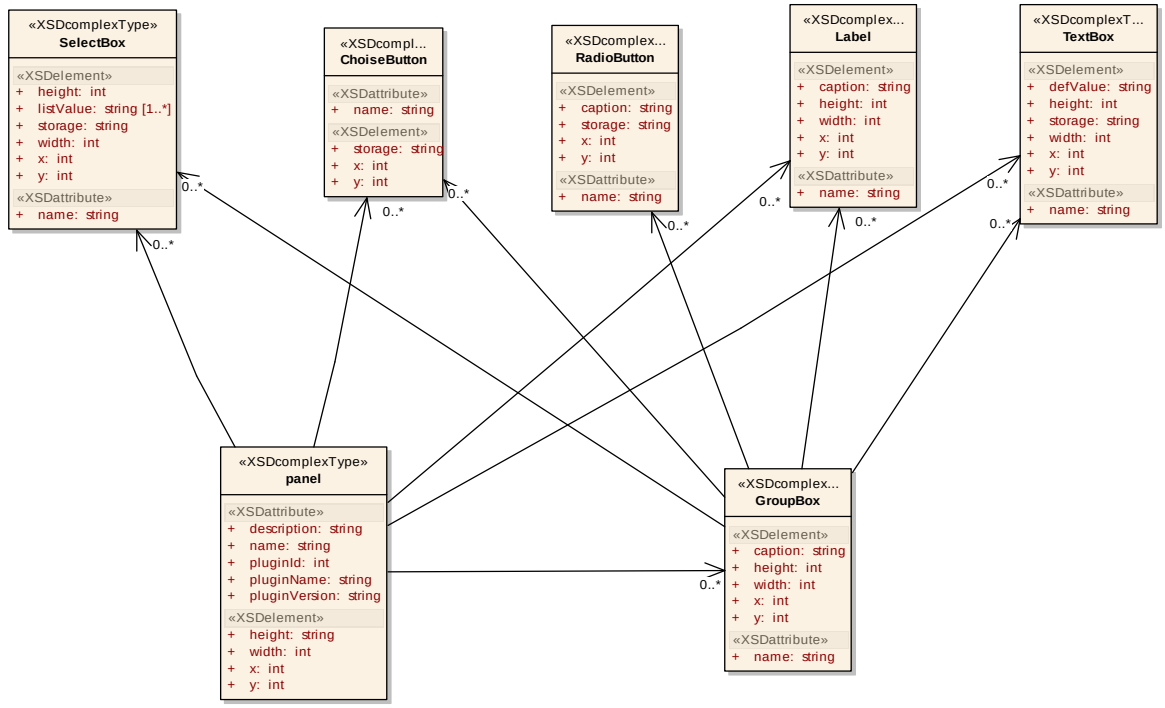


*Рисунок 5. Структура XML файлу, що описує інтерфейс*

Для підключення плагіна до керуючого програмного забезпечення необхідний XML-файл з описом інтерфейсу. Після чого система проаналізує цей файл і побудує інтерфейс користувача для налаштування плагіна. Після введення всіх налаштувань користувачем система сформує пари параметр-значення відповідно до поля storage в описі компонента в XML-файлі.

**Висновки**

Архітектура системи забезпечує об'єднання відеопотоків різного формату, в результаті чого користувач зможе спостерігати комплексне зображення. Розроблена архітектура підтримує можливість розширення функціональності системи за рахунок динамічного додавання алгоритмів обробки відеопотоку в реальному режимі часу.

Використання розподіленої архітектури системи дозволяє підвищити верхній поріг продуктивності обробки відеопотоку в реальному режимі часу за рахунок паралельного виконання алгоритмів відеообробки. А введена в систему підтримка плагінів дозволяє істотно розширити функціональність системи і надає можливість стороннім розробникам доповнювати систему власними модулями.

**Список використаних джерел**

1. Xuesong Cao, Zhaoping Wang, Ruimin Hu, Jun Chen, «An overlay-based service architecture for distributed video surveillance system», April, 2008, p. 211-214.

2. C. Y. Chan, and Jack Y. B. Lee, «A Decentralized Scheduler for Distributed Video Streaming in a Server-less Video Streaming System», 2009, p. 57-62.

3. Гук И. Краткий обзор цифровых сигнальных процессоров DaVinci / И. Гук // Компоненты и технологии. – 2007. – № 3. – С. 150-160.

4. Фисенко В. Т. Компьютерная обработка и распознавание изображений: учеб. пособие / В. Т. Фисенко, Т. Ю. Фисенко. – СПб.: СПбГУ ИТМО, 2008. – 192 с.

5. Видеостены и качество передачи изображения на составных полиэкранах [Электронный ресурс]. – Режим доступа: http://kljuch.ru/videosteny-s-kachestvo-peredachi-izobrazheniya-na-sostavnyh-poliekranah/. – Название с экрана. – Дата доступа 11.05.2011.

УДК 004.94:519.876

**Р.Ю. Лопаткин**, канд. физ.-мат. наук
**В.А. Иващенко**, аспирант
Институт прикладной физики НАН Украины, г. Сумы, Украина

# ЯЗЫК ПРОГРАММИРОВАНИЯ GROOVY КАК СРЕДСТВО ОПИСАНИЯ МУЛЬТИАГЕНТНЫХ МОДЕЛЕЙ

*В данной статье авторами предлагается использование методологии метапрограммирования для упрощения описания мультиагентных моделей, предложена архитектура среды для мультиагентного моделирования в рамках данного подхода, а также решаются связанные с этим подходом технические вопросы.*

## Введение

Перед разработчиками среды имитационного моделирования всегда стоит задача предоставления пользователю некоторого интерфейса для описания моделей. Обычно исследователю предоставляется возможность описывать модель либо с помощью специального графического интерфейса, либо используя языки программирования. Понятно, что первый подход может обеспечить значительное упрощение процесса описания моделей, а второй – его гибкость. Для совмещения преимуществ обоих подходов, с нашей точки зрения, напрашивается использование графических интерфейсов для задания параметров модели, а для описания модели – использовать язык программирования.

Описания моделей с помощью языка программирования возможно несколькими способами:

1. Интегрировать скриптовый язык в систему моделирования, написанную на компилируемом языке программирования. Примером такой связки могут быть язык программирования C++ и скриптовый язык Luna Lua или Angel Script. Преимуществом первого подхода может быть эффективное обеспечение безопасности – код скриптового языка часто (но не всегда) выполняется в специальной защищенной среде, за пределы которой скрипт не имеет доступа. Но такой подход может иметь недостатки относительно обеспечения взаимодействия среды моделирования с моделью, поскольку для полноценного взаимодействия необходимо, чтобы среда моделирования имела доступ к любой переменной, любой функции модели, а модель, в свою очередь, – наоборот, должна иметь доступ к переменным среды моделирования. Еще очевидный недостаток такого подхода – отсутствие кроссплатформенности разработанной программы, что является критическим фактором при разработке распределенных систем.

2. Описывать модели на том же языке, на котором разработана среда моделирования. Такой подход может обеспечить высокое быстродействие в случае, если идет речь о компилируемом языке программирования, но возникают вопросы с безопасным выполнением кода модели, а также с гибкостью интеграции модели и среды моделирования. Для решения вопросов безопасности в таком случае придётся создавать или использовать готовую виртуальную машину, что не всегда удобно и может збавить от преимущества в скорости выполнения. Также как и в первом случае не обеспечивается кроссплатформенность разработанного решения. Подобный подход может быть приемлем, если среда моделирования предназначена для запуска на локальной машине, но неприемлем при распределенном моделировании. Примером такой системы может