# МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ

DOI: 10.18137/RNU.V9187.23.03.P.3 УДК 004.942

В.И. Алексеев

## ИСПОЛЬЗОВАНИЕ МЕТОДОВ ПРОГРАММНОГО МОДЕЛИРОВАНИЯ ДЛЯ РАЗРАБОТКИ СИСТЕМ БОРТОВЫХ КОМПЛЕКСОВ УПРАВЛЕНИЯ ЛЕТАТЕЛЬНЫХ АППАРАТОВ

**Аннотация.** На основе материалов из открытых информационных источников дан краткий анализ современного состояния использования методов программного моделирования для разработки систем бортовых комплексов управления летательных аппаратов, в частности бортовых вычислительных систем. Разработаны предложения по использованию программной эмуляции бортовой вычислительной системы.

*Ключевые слова:* программное моделирование, эмуляция, бортовая вычислительная система, бортовой комплекс управления, программное обеспечение, летательные аппараты.

V.I. Alekseev

## THE USE OF SOFTWARE MODELING METHODS FOR THE DEVELOPMENT OF SYSTEMS OF ON-BOARD CONTROL SYSTEMS OF AIRCRAFT

**Abstract.** Based on materials from open information sources, a brief analysis of the current state of the use of software modeling methods for the development of on-board control systems of aircraft, in particular on-board computer systems, is given. Proposals for the use of software emulation of the onboard computer system have been developed.

*Keywords:* software modeling, emulation, on-board computer system, on-board control system, software, aircraft.

### *Введение*

По своим летно-техническим и аэродинамическим свойствам современные летательные аппараты превзошли системы управления, прежде всего бортовые вычислительные комплексы. С развитием новейших устройств управления летательных аппаратов, работающих на основе новых элементов, возникает необходимость модернизации бортовых вычислительных систем, обеспечивающих управление такими устройствами.

Модернизация предполагает замену вычислительного модуля, что может быть выполнено либо путем разработки нового функционального программного обеспечения, либо путем разработки программного образа заменяемого бортового компьютера для использования на целевом бортовом компьютере с улучшенными характеристиками.

Разработка замещающего образа программного обеспечения бортового компьютера для использования на целевом бортовом компьютере с улучшенными характеристиками может быть решена с помощью системы программной эмуляции.

**Алексеев Владимир Ильич**
преподаватель, Военно-космической академии имени А.Ф. Можайского, Санкт-Петербург.
Сфера научных интересов: автоматика, телемеханика, вычислительная техника. Автор более 20 опубликованных научных работ.
Электронный адрес: vka@mil.ru

Образ программного обеспечения должен полностью повторять алгоритмы работы задач заменяемого бортового компьютера, включая временные и точностные характеристики. При разработке системы эмуляции бортового компьютера необходима отладка только комплекса программной эмуляции, что обеспечивает очевидную экономию времени и, как следствие, снижение затрат на тестирование.

*Понятие программного моделирования, эмуляции*

Имитационное (программное) моделирование – вид моделирования, при котором логико-математическая модель исследуемого объекта представляет собой алгоритм функционирования объекта, реализованный в виде программного комплекса для вычислительной системы.

Эмуляция (англ. emulation) – комплекс программных аппаратных средств или их сочетание, предназначенный для копирования функций одной вычислительной системы в другую, отличную от первой, таким образом, чтобы эмулированное поведение как можно ближе соответствовало поведению оригинальной системы. Цель – максимально точное воспроизведение поведения в отличие от разных форм компьютерного моделирования, в которых имитируется поведение некоторой абстрактной модели [10].

*Схема процесса моделирования*

Имитационное моделирование – метод исследования объектов, основанный на замене изучаемого объекта имитирующим объектом, например, функциональной моделью.

Функциональная модель – модель, предназначенная для изучения особенностей работы (функционирования) системы и ее назначения во взаимосвязи с внутренними и внешними элементами.

Процесс моделирования включает в себя следующие этапы:

- построение модели объекта-оригинала;
- изучение модели;
- экстраполяция – перенос полученных данных на область знаний об исходном объекте – интерпретация знаний;
- проверка и применение полученных знаний.

На первом этапе при осознании невозможности или нецелесообразности прямого изучения объекта создается его модель для полноценного замещения оригинала объектом-посредником, воспроизводящим его необходимые параметры.

На втором этапе производится изучение самой модели настолько детальное, насколько это требуется для решения конкретной задачи. Осуществляется наблюдение за поведением модели, проводятся над ней эксперименты, осуществляется измерение или описание ее характеристик. Целью этапа является получение требуемой информации о модели.

Третий этап (экстраполяционный) представляет собой возвращение к исходному объекту, то есть интерпретацию полученных знаний о модели, оценку их приемлемости и, соответственно, непосредственное применение их к оригиналу.

Четвертый этап предполагает проверку и применение полученных знаний.

Эти шаги реализуют своеобразный цикл моделирования, в ходе которого модель и оригинал соотносятся друг с другом (см. Рисунок 1).

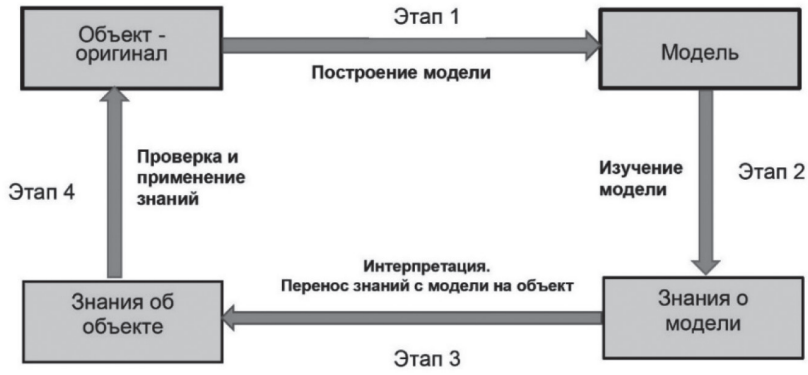


**Рисунок 1.** Структурная схема процесса моделирования

Модель является одновременно и объектом, и средством изучения, по своей познавательной функции она сопряжена с другим объектом.

Исходя из этапов моделирования легко увидеть, что структура метода моделирования содержит больше компонентов, чем наблюдение или эксперимент. В моделирование включены следующие составляющие:

- субъект, осуществляющий моделирование;
- моделируемый объект-оригинал;
- объект посредника – модель;
- контекст моделирования, к которому относятся условия времени и места, концептуальные и материально-технические средства.

*Основные подходы к программной эмуляции. Классификация моделей*

Существует по крайней мере два основных подхода к программной эмуляции [7].

Первый является *полностью программным*, то есть весь эмулируемый код разбирается и выполняется процедурами эмулятора. Данный подход самый безопасный, так как в независимости от того, какой вредоносный код будет загружен в эмулятор, он всё равно не сможет воздействовать на настоящий процессор. Также этим методом можно эмулировать произвольные архитектуры процессоров.

Однако этот подход самый трудоемкий и медленный по скорости исполнения эмулируемого кода. Трудоемкость заключается в необходимости реализовывать в процедурах каждую инструкцию процессора; медленный он потому, что исполнить инструкцию на настоящем процессоре намного быстрее, чем исполнить подпрограмму, эмулирующую эту инструкцию и состоящую из множества команд.

С учетом этих минусов, самым большой из которых – скорость исполнения, был разработан *метод частичного исполнения эмулируемого кода на настоящем процессоре.* Суть его том, что в эмулируемом коде выбирается блок безопасного кода и передается на ис-

полнение реальному процессору, после чего управление снова переходит на программу эмулятора. Безопасен код в том смысле, что он не сможет навсегда перехватить управление или обрушить программу.

Этот подход, несмотря на более высокую скорость исполнения, также имеет свои минусы. Во-первых, сложность реализации: эмулируемый код нужно выбирать или модифицировать таким образом, чтобы после передачи его реальному процессору не возникало ошибок, и после исполнения можно было бы вернуться обратно в эмулятор. Во-вторых, эмулируемый код должен быть написан под ту же архитектуру, что и эмулятор, по причине частичного исполнения на реальном процессоре.

Эмуляторы редко используют прямой доступ к оборудованию – достаточно обеспечить некоторый уровень совместимости, осуществляющих трансляцию системных вызовов эмулируемой системы в вызовы работающей системы.

Существуют следующие направления программной эмуляции [10]:

- интерпретация кода;
- динамическая рекомпиляция;
- статическая рекомпиляция.

**Интерпретация кода.** При интерпретации программа-эмулятор последовательно читает из памяти коды команд программы и выполняет их. Главный цикл работы эмулятора выглядит так:

```
while (state == EMU_RUN) {
Read(OpCode); // Чтениекода// команды.
Execute (OpCode); // Исполнение// команды.
```

Работа подобных эмуляторов проста и понятна. Их положительными чертами являются легкость встраивания отладочных функций, возможность гибкой коррекции скорости работы и легкость модификации. Основные недостатки – низкая производительность, применение мощного компьютера.

**Динамическая рекомпиляция.** Рекомпилятор сложен в написании, но при этом превосходит интерпретатор в производительности. При использовании динамической рекомпиляции программа-эмулятор преобразует фрагменты исполняемой программы в код, который может быть выполнен на другом компьютере непосредственно во время ее работы. Другими словами, рекомпилятор выступает в роли переводчика с одного машинного языка на другой.

**Рекомпиляция кода.** При использовании статической рекомпиляции программа-эмулятор обрабатывает сразу всю исходную программу и формирует такую программу, которую можно запустить на другом компьютере. Код, получающийся в результате статической рекомпиляции, функционирует очень быстро.

Наилучшие результаты достигаются при разумном совмещении всех методов.

Создание программного эмулятора состоит из следующих этапов.

1. Сбор информации об эмулируемой системе: полная информация о центральном процессоре, видеопроцессоре, портах ввода/вывода, распределении адресного пространства и др.

2. Выбор языка программирования. Из-за серьезных требований к быстродействию программы реально можно рассматривать ассемблер и язык Си.

Преимуществом ассемблера является создание максимально быстродействующей программы. При использовании регистров процессора в качестве регистров эмулируе-

мой системы многие команды могут быть напрямую заменены единственной командой центрального процессора компьютера, на котором работает эмулятор. Недостатком является то, что программы на ассемблере имеют большой объем и достаточно трудны для понимания, модификации и отладки, поиск ошибок и правка текста занимают много времени. При разработке аналогичного эмулятора, но для другого компьютера, программу придется полностью переписывать;

Достоинствами языка Си является то, что программы легко понимаются и отлаживаются. При разработке эмулятора для другого компьютера не требуется полностью переписывать программу – ее легко модифицировать, добавив эмуляцию различных дополнительных устройств. Недостаток – размер скомпилированной программы по сравнению с ассемблером увеличивается, а быстродействие уменьшается.

Наилучшего результата можно добиться при совместном использовании обоих языков: интерфейс пользователя пишется на языке Си, а критичные по времени исполнения процедуры эмуляции – на ассемблере. Наиболее разумным представляется следующий подход: сначала программа-эмулятор пишется полностью на Си, затем отлаживается, и часть, отвечающая за эмуляцию, переписывается на ассемблере [13].

*Структурный состав программного эмулятора вычислительной системы*

Обычно эмулятор вычислительной системы состоит из нескольких модулей, отвечающих за различные подсистемы эмулируемой вычислительной системы.

Основной модуль реализует интерфейс пользователя, загружает исполняемую программу, запускает непосредственно эмулятор и при необходимости вызывает встроенный отладчик.

Модуль эмуляции или симуляции центрального процессора выполняет команды центрального процессора эмулируемой системы.

Модуль эмуляции подсистемы памяти эмулирующего оперативного запоминающего устройства и постоянного запоминающего устройства обрабатывает обращения любых устройств к памяти системы.

Модуль эмуляции различных устройств ввода/вывода позволяет вводить информацию от различных систем бортовых цифровых вычислительных машин (далее – БЦВМ).

**Встроенный отладчик.** Модуль необходим при отладке программы-эмулятора.

Системные шины в целях упрощения эмуляции и увеличения производительности, как правило, не эмулируются. Вместо этого виртуальная периферия обращается непосредственно к процессору или подсистеме памяти.

Эмулятор должен быть функционально эквивалентен аппаратному прототипу беспилотных воздушных судов на уровне функциональных узлов и связей между ними и быть полностью программно совместимым с ним, обладать специализированными интерфейсами, расширяющими возможности отладки исполняемого на нем программного обеспечения и иметь возможность интеграции с реальным оборудованием в составе стенда имитационного моделирования.

В эмуляторе бортовых компьютеров моделируются архитектура процессора, оперативная память, контроллеры интерфейсов, межканального и межпроцессорного обмена, системный таймер, контроллер прерываний.

Эмулятор позволяет исполнять бинарный код, предназначенный для целевой платформы, без каких-либо изменений. Бортовая вычислительная система представляет собой систему из нескольких компьютеров, работающих синхронно в условиях жесткого

реального времени. Созданный эмулятор предоставляет возможность пошаговой отладки системы из нескольких компьютеров без нарушения их синхронной работы.

*Возможности программной эмуляции БЦВМ*

Для определения возможности программной эмуляции БЦВМ для выполнения на целевой БЦВМ необходимо провести сравнительный анализ эмулируемой и целевой БЦВМ по следующим характеристикам:

- форматы чисел;
- разрядность БЦВМ;
- объемы постоянной и оперативной памяти;
- быстродействие;
- система ввода-вывода;
- система прерываний;
- способы синхронизации и диспетчеризации.

Этапы процесса определения возможности программной эмуляции семейства БЦВМ для выполнения на целевой БЦВМ представлены на Рисунке 2.

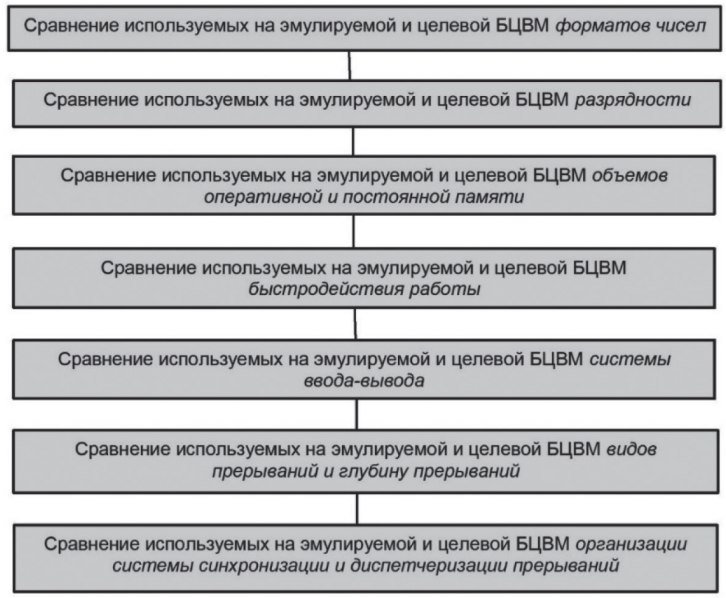


**Рисунок 2.** Этапы процесса определения возможности программной эмуляции семейства БЦВМ для выполнения на целевой БЦВМ

При несовпадении используемых на эмулируемой и целевой БЦВМ форматов чисел необходимо **провести определение возможности организации вычислений** с помощью имеющихся в целевой БЦВМ числовых форматов.

Далее предлагается **сравнить разрядности БЦВМ**. Разрядность целевой БЦВМ должна превосходить разрядности эмулируемой БЦВМ, иначе при вычислениях будет уменьшаться точность.

Затем необходимо **сравнить объемы оперативной и постоянной памяти** эмулируемой и целевой БЦВМ. Объем памяти целевой БЦВМ должен превосходить объем эмули-

руемой БЦВМ, так как на целевой БЦВМ кроме эмулируемого программного обеспечения могут выполняться другие новые задачи.

Следующий шаг – **сравнить быстродействие работы** эмулируемой и целевой БЦВМ. Целевая БЦВМ должна иметь большее быстродействие по сравнению с эмулируемой.

Далее **сравниваются системы ввода-вывода**: способы ввода-вывода, количество линий каналов ввода-вывода, частоту передачи данных. При несоответствии протоколов ввода-вывода должна быть решена задача эмуляции ввода-вывода одного вида с помощью другого. При недостатке линий каналов на целевой БЦВМ для эмуляции процесса ввода-вывода необходимо решить задачу эмуляции ввода-вывода с использованием меньшего количества каналов. Также необходимо достичь одинаковой частоты передачи данных.

Затем необходимо **сравнить виды прерываний и глубину прерываний**. На данном этапе возникает задача организации того вида прерывания, которого нет на целевой БЦВМ. Проблема появляется при нехватке глубины прерываний на целевой БЦВМ для эмулирования системы прерываний эмулируемой БЦВМ.

На последнем этапе требуется **проанализировать организацию системы синхронизации и диспетчеризации** на эмулируемой и целевой БЦВМ. Процесс выполнения задач на целевой БЦВМ должен выполняться по времени столько же, сколько на эмулируемой БЦВМ.

*Достоинства и недостатки эмуляции БЦВМ. Выводы*

Важное преимущество программных эмуляторов – возможность многократного воспроизведения рабочих ситуаций. Это особенно полезно, когда оригинальная система труднодоступна или несовместима с современным оборудованием [3].

Загрузка программы в эмулятор чаще происходит быстрее, чем в реальное железо, что очень удобно при написании и отладки программного обеспечения.

Эмуляция сохраняет также вид, поведение и ощущение от оригинальных систем, что не менее важно, чем данные сами по себе. Эмуляция позволяет использовать программное обеспечение, эксклюзивное для одной платформы, на другой платформе [6].

Несмотря на высокую изначальную стоимость создания эмулятора, со временем эмуляторы могут становиться более финансово выгодным решением.

Решение проблемы модернизации вычислительных модулей с помощью разработки системы программной эмуляции актуально и в том случае, когда вычислительные системы на борту летательного аппарата состоят из нескольких вычислительных модулей. С другой стороны, появляется возможность расширения набора функциональных задач за счет использования целевой БЦВМ с увеличенными вычислительными мощностями.

В то же время эмуляция очень ресурсоемкая задача и может требовать от компьютера намного большей производительности (скорости процессора, объема памяти), чем производительность эмулируемой системы. Чем сложнее система и выше точность эмуляции, тем большая производительность для нее требуется. Кроме того, высока изначальная стоимость создания эмулятора.

## Литература

1. QEMU Emulator User Documentation [Электронный ресурс]. Available at: URL: http://qemu.weilnetz.de/qemu-doc.html (дата обращения: 17.03.2023).

2. *Альфред Ахо, Равви Сети, Джеффри Ульман*. Компиляторы: принципы, технологии и инструменты : перевод с англ. М.: Вильямс, 2011. 768 с.

3. *Брехов О.М., Корнеенкова А.В.* Особенности программной эмуляции семейства бортовых вычислительных машин с открытой системой команд // Труды МАИ. 2006. № 25.
4. *Бровкин А. Г., Бурдыгов Б. Г., Гордийко С. В.* Бортовые системы управления космическими аппаратами. М.: МАИ-Принт, 2010. 304 с.
5. *Гатчин Ю.А., Жаринов И.О.* Основы проектирования вычислительных систем интегрированной модульной авионики. М.: Машиностроение, 2010. 224 с.
6. *Грис Д.* Конструирование компиляторов для цифровых вычислительных машин. М.: Мир, 2019. 544 с.
7. *Девятков В.В.* Мир имитационного моделирования: взгляд из России // Прикладная информатика. 2011. № 4 (34). С. 9–29.
8. *Замятина О.М.* Моделирование систем: учебное пособие. Томск: Изд-во ТПУ, 2009. 204 с.
9. *Кельтон Д., Лоу А.* Имитационное моделирование. Классика CSon : пер. с англ. СПб.: Питер, 2004. 847 с.
10. *Корнеенкова А.В.* Интерпретаторы в системах управления // Информационные средства и технологии: материалы международной конференции. М.: Станкин, 2004. С. 154–157.
11. *Куприяшкин А.Г.* Основы моделирования систем: учебное пособие. Норильск: НИИ, 2015. 135 с.
12. *Лин В.* Архитектура ЭВМ и программирование на языке ассемблера: перевод с англ. М.: Радио и связь, 2009. 316 с.: ил.
13. *Маликов Р.Ф.* Основы математического моделирования: учебное пособие. М.: Горячая линия - Телеком, 2010. 368 с.
14. *Ноженкова Л.Ф., Исаева О.С., Грузенко Е.А.* Построение программно-математической модели бортовой аппаратуры командно-измерительной системы космического аппарата // Информатизация и связь. 2014. Вып. 1. С. 87–93.
15. *Хантер Р.* Проектирование и конструирование компиляторов. М.: Мир, 2004. 212 с.

## Literature

1. QEMU Emulator User Documentation. Available at: URL: http://qemu.weilnetz.de/qemu-doc.html (accessed: 17.03.2023).
2. Alfred Aho, Rabbi of the Network, Jeffrey Ullman (2011) *Kompiljatory: principy, tehnologii i instrumenty* [Compilers: principles, technologies and tools]. Moscow: William Publishing, 768 p. (in Russian).
3. Brekhov O.M., Korneenkova A.V. (2006) [Features of software emulation of a family of onboard computers with an open command system]. *Trudy MAI*, No. 25 (in Russian).
4. Brovkin A.G., Burdygov B.G., Gordiyko S.V. (2010) [Onboard spacecraft control systems]. Moscow: MAI-PRINT Publishing, 304 p. (in Russian).
5. Gatchin Yu.A., Zharinov I.O. (2010) *Osnovy proektirovanija vychislitel'nyh sistem integrirovannoj modul'noj avioniki* [Fundamentals of designing computer systems of integrated modular avionics]. Moscow: Mashinostroenie Publishing, 224 p. (in Russian).
6. Gris D. (2019) *Konstruirovanie kompiljatorov dlja cifrovyh vychislitel'nyh mashin* [Designing compilers for digital computers]. Moscow: MIR Publishing, 544 p. (in Russian).
7. Devyatkov V.V. (2011) [The world of simulation modeling: a view from Russia]. *Prikladnaja informatika*, No. 4 (34), Pp. 9–29 (in Russian).
8. Zamyatina O.M. (2009) *Modelirovanie sistem* [Modeling of systems]. Tomsk: TPU Publishing House, 204 p. (in Russian).
9. Kelton D., Low A. (2004) *Imitacionnoe modelirovanie. Klassika CSon* [Simulation modeling. Classics of CS]. St. Petersburg: Peter Publishing, 847 p. (in Russian).

10. Korneenkova A.V. (2004) *Interpretatory v sistemah upravlenija* [Interpreters in control systems]. *Informacionnye sredstva i tehnologii: materialy mezhdunarodnoj konferencii* [Information tools and technologies: Proc. of the international conference]. Moscow: Stankin Publishing, Pp. 154–157 (in Russian).
11. Kupriyashkin A.G. (2015) *Osnovy modelirovanija sistem* [Fundamentals of systems modeling]. Norilsk: Research Institute, 135 p. (in Russian).
12. Lin V. (2009) *Arhitektura JeVM i programmirovanie na jazyke assemblera* [Computer architecture and programming in assembly language]. Moscow: Radio i Svjaz' Publishing, 316 p. (in Russian).
13. Malikov R.F. (2010) *Osnovy matematicheskogo modelirovanija* [Fundamentals of mathematical modeling]. Moscow: Hotline - Telecom Publishing, 368 p. (in Russian).
14. Nozhenkova L.F., Isaeva O.C., Gruzenko E.A. (2014) [Constructing a software-mathematical model of the onboard equipment of the command and measurement system of the spacecraft]. *Informatizacija i Svjaz'*, Iss. 1, Pp. 87–93 (in Russian).
15. Hunter R. (2004) *Proektirovanie i konstruirovanie kompiljatorov* [Design and construction of compilers]. Moscow: Mir Publishing, 212 p. (in Russian).