КОМПОНЕНТНАЯ ПРОГРАММНАЯ АРХИТЕКТУРА МУЛЬТИВЕРСИОННЫХ СИСТЕМ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ

Поздняков Дмитрий Александрович

Аннотация

В статье предложены методы компонентного программирования для построения мультиверсионного программного обеспечения отказоустойчивых систем управления и обработки информации.

Основное содержание

На сегодняшний день разработаны различные методы проектирования отказоустойчивого программного обеспечения. Среди них одним из наиболее перспективных является метод мультиверсионного проектирования. Он состоит в том, что в систему включаются несколько программных модулей, дублирующих друг друга по своему целевому назначению.

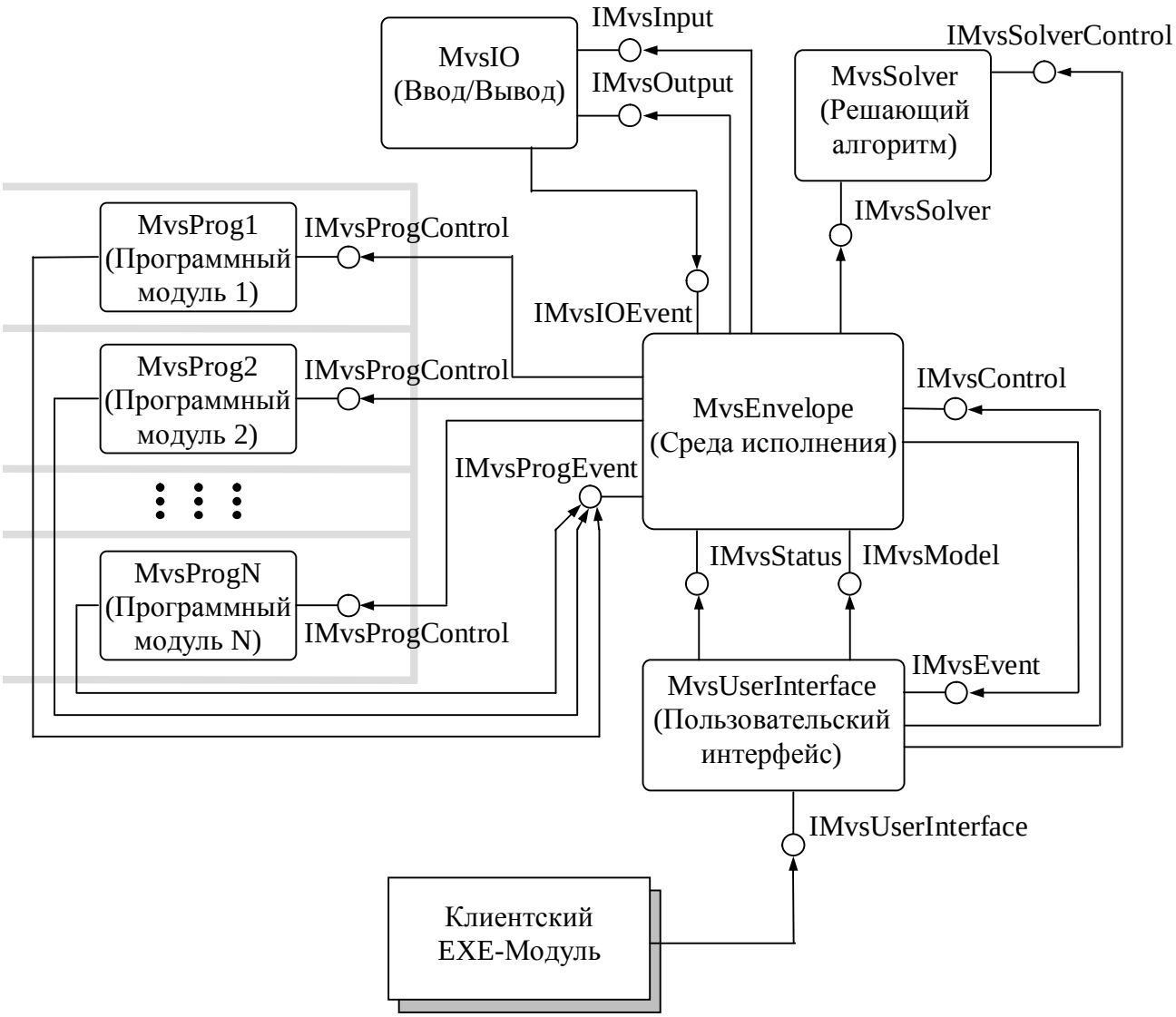
Мультиверсионное программирование предполагает, что большинство модулей, реализующих ту же самую функциональность, производят сбой в различных точках, так что сбои могут быть обнаружены и исправлены. Независимость сбоев различных модулей может быть достигнуто с помощью ввода разнообразия в процесс разработки. Но эта независимость сбоев находится на уровне исходных кодов – это является ключевой проблемой. На стадии выполнения мультиверсионных модулей независимость сбоев теряется из-за того, что остались не учтенными возможные взаимодействия модулей в рамках всей программной системы. Модули работают в едином адресном пространстве памяти, разделяют одни и те же ресурсы операционной системы, и из-за этого возникают дополнительные зависимости между модулями программного обеспечения. Вследствие этого, сбой одного модуля может привести к сбою других или даже к отказу всей системы в целом. Для решения этой проблемы необходимо сохранить введенное разнообразие при разработке модулей и перенести его на стадию выполнения. Для этого были сформулированы и обоснованы требования, которым должны соответствовать разрабатываемые мультиверсионные модули.

1. Динамическое подключение модулей, цель которого – реализовать возможность замены модулей во время работы программной системы.
2. Требование инкапсуляции следует из требования динамического подключения модулей. Модули подключаются друг к другу посредством инкапсулирующих интерфейсов.
3. Требование межмодульной защиты обеспечивает безопасное взаимодействия модулей в рамках программной системы. Необходимо организовать межмодульное взаимодействие таким образом, чтобы исключить нарушения в работе модуля, вызванные внешними причинами, а именно, несанкционированным вмешательством других модулей.
4. Требование независимости модулей от языка программирования отчасти вытекает из требования об инкапсуляции, которое описано выше. Другим основанием для подобного требований является экономическая целесообразность.

Для реализации среды и модулей мультиверсионного программного обеспечения, с учетом сформулированных требований, была предложена технология компонентной объектной модели Microsoft (COM) на основе которой был реализован комплекс программ MVS.

Основной особенностью комплекса является реализация программных модулей в виде независимых компонент, которые могут исполняться в отдельных от среды исполнения

процессах. Более того, программные компоненты могут находиться на разных компьютерах и взаимодействовать со средой исполнения посредством сети. Это, во-первых, решает проблему ограниченности вычислительной мощности одного компьютера, во-вторых, это обеспечивает защиту программных модулей друг от друга и от среды исполнения. Другие функциональные модули системы так же реализованы в виде отдельных компонент, что позволяет модернизировать и заменять отдельные части системы с минимальными усилиями.



Условные обозначения:

– компонент

– ссылка на интерфейс

– интерфейс компонента

– границы процессов

Рисунок 1. Модель мультиверсионного программного обеспечения построенного на основе технологии COM объектов

Наиболее важной задачей является защита среды исполнения от разрушительного влияния извне, поэтому взаимодействие пользователя и программных модулей со средой исполнения происходит только через специальные интерфейсы.

В случае отказа одного из программных модулей, когда работа модуля прекращается, необходим его перезапуск с восстановлением его последнего состояния. Для этого был

применен метод контрольных точек и рестарта. После каждого удачного вычислительного цикла, программный модуль формирует данные контрольной точки, которые передаются в среду исполнения вместе с результатами работы модуля. Если на следующем цикле один или несколько программных компонент терпят отказ, то их состояние восстанавливается по данным предыдущей контрольной точки, выбранной решающим алгоритмом.

Разработанная компонентная архитектура мультиверсионных программных систем позволяет исключить взаимное влияние мультиверсионных программных модулей друг на друга, что обеспечивает независимость отказов отдельных модулей. Кроме того, разработанная на базе компонентной архитектуры среда исполнения, позволяет построить распределенную мультиверсионную систему, которая обеспечивает возможность подключения к среде исполнения любого количества программных модулей, распределяя вычислительную нагрузку на множество машин.