МЕХАНИЗМ ВЗАИМОДЕЙСТВИЯ КОМПОНЕНТ В ПРОГРАММАХ С УПРАВЛЯЕМОЙ ПРОГРАММНОЙ АРХИТЕКТУРОЙ

Борисов А.В., Куриленко И.Е., Лазуткин В.А.

ГОУВПО «Московский Энергетический Институт (Технический Университет)», Москва, Россия

В данной работе рассматриваются вопросы стандартизации взаимодействия компонент программы с управляемой программной архитектурой. Управляемая программная архитектура позволяет [1]:

- сократить время разработки и повысить качество программы;
- достичь глубокой стандартизация компонент программного средства (ПС) и получить возможность применения средств автоматизации и кодогенерации;
- автоматизировать процесс построения ПС из компонент;
- реализовать автономное тестирование отдельных компонент ПС;
- изменять логику работы программы путем изменения набора использованных компонент и связей между ними в момент выполнения программы без перекомпиляции и переустановки;
- упростить модификацию приложения при изменении требований;
- получить возможность автоматической балансировки нагрузки на вычислительных узлах при работе в распределенном режиме;
- организовать автоматическое версионирование и упростить процесс выпуска редакций ПС (версий с разными возможностями).

Без стандартизированного механизма взаимодействия компонент между собой трудновыполнимыми оказываются требования независимости и слабой связности, заложенные в само определение управляемой

программной архитектуры [2]. В качестве такого механизма предлагается использовать событийно-управляемый механизм взаимодействия [3]. Он основывается на организации взаимодействия между компонентами путем обмена сообщениями через коммуникационную среду (рис. 1). Компоненты могут в произвольный момент времени подключаться к этой среде для приема либо отправки сообщений или отключаться от нее. При этом коммуникационная среда самостоятельно обеспечивает сетевую и локальную передачу сообщений. Подобная архитектура позволяет эффективно реализовать крупнозернистый параллелизм, а также дает широкие возможности для расширения программы [4]. Как указывается в работе [5], реализация бизнес-процессов, подчиненных событиям, отражает событийную природу реального мира, что дает предприятиям финансовое и стратегическое преимущество.

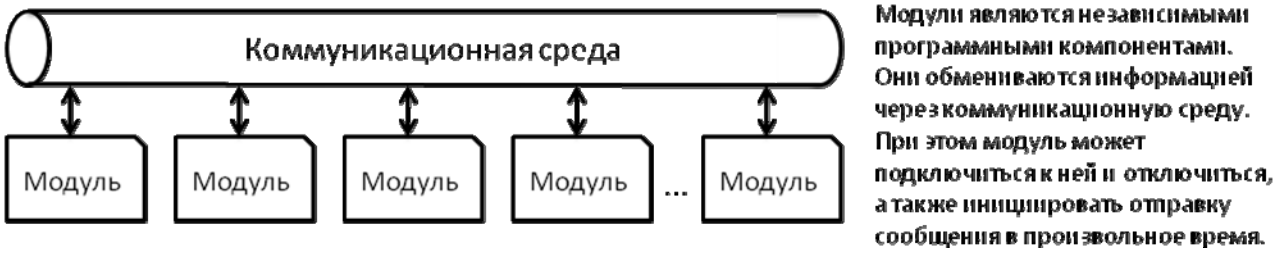

Рис. 1. Система обмена сообщениями

Рассмотрим основные требования к системе обмена сообщениями:

- обеспечить возможность синхронной и асинхронной передачи сообщений между параллельно выполняющимися компонентами в адресном и безадресном режиме как на уровне отдельно взятой электронной вычислительной машины (ЭВМ), так и на уровне вычислительной системы в целом;
- обеспечить корректное функционирование системы на локальной ЭВМ при потере связи с другими ЭВМ, а также восстановление программной связи, когда аппаратная связь будет восстановлена без перезапуска локальной системы и потери данных;

- обеспечить простоту подключения и отключения компонент во время работы системы без какого-либо останова либо перезапуска системы, перезагрузки схем взаимных соединений модулей и т.п.;
- обеспечить возможность многоуровневой фильтрации передаваемых по системе сообщений с целью снижения числа сетевых пересылок;
- обеспечить надежность на локальном уровне (при сбоях и ошибках в работе одного конкретного локально подключенного модуля не должна останавливаться ни локальная, ни сетевая рассылка сообщений для других модулей).

Обратимся к базовой архитектуре системы передачи сообщений. В соответствии с требованиями надежности и скорости система передачи сообщений разбита на два уровня – локальный уровень обмена сообщениями и сетевой уровень (рис. 2). На локальном уровне (уровне конкретной вычислительной машины в рамках локальной сети) может функционировать от одного до нескольких локальных модулей маршрутизации сообщений (ММС). Предполагается, что локальные ММС функционируют параллельно друг с другом в отдельных процессах. Каждый локальный ММС позволяет подключать от одного до нескольких компонент. Использование на локальном уровне более одного ММС позволяет развязать поток сообщений между составными элементами отдельных блоков (приложений) системы (например, сервера расчета стоимости услуг и графического приложения оператора). Такая реализация системы соответствует требованиям надежности, заявленным выше. При сбое в работе одного из локальных ММС перестанет функционировать только он и его клиенты. Если параллельно ему работает другой ММС, то благодаря вынесению в разные процессы на его работу ничего не повлияет. Если в одном из подключенных клиентских модулей происходит сбой он автоматически отключается от ММС.

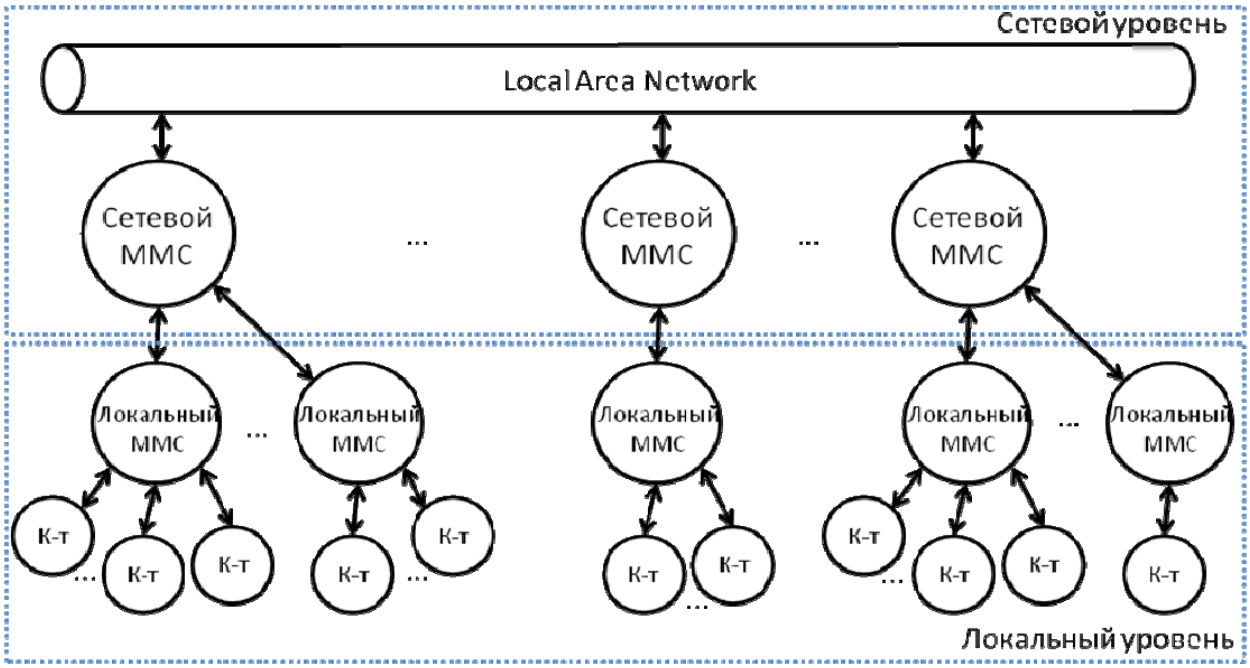

Рис. 2. Архитектура системы передачи сообщений

Благодаря наличию в приложении локального ММС, оно может состоять из слабозависимых компонент (число которых может варьироваться в зависимости от назначения приложения) взаимодействующих друг с другом посредством событий.

Локальные ММС могут взаимодействовать друг с другом и с ММС на других ЭВМ через сетевой ММС. При этом локальный ММС может быть подключен только к одному сетевому. Сетевые ММС предназначены для обеспечения передачи сообщений между вычислительными машинами (удаленное взаимодействие) и между подключенными локальными ММС (кросспроцесное взаимодействие).

Таким образом, получается три возможных уровня обмена сообщениями:

- внутрипроцессный – обеспечивается локальным ММС;
- кросспроцесный (между несколькими процессами на одной ЭВМ) – обеспечивается локальными и сетевым ММС;
- сетевой – обеспечивается сетевыми ММС.

Исходя из требований надежности, сетевые ММС физически располагаются внутри собственных процессов. Сетевой ММС способен обеспечивать рассылку сообщений даже в случае сбоя какого-либо из подключенных к нему локальных ММС (этот ММС в этом случае просто отключается).

Благодаря сетевому уровню возможна интеграция отдельных блоков системы в единое распределенное приложение (локальный уровень решает задачу интеграции параллельно работающих модулей в единые блоки системы). Также на сетевом уровне решается задача интеграции систем (реализованных в данной архитектуре) в единый распределенный программный комплекс, что организуется путем соединения между собой сетевых роутеров.

С точки зрения системы передачи сообщений подключенные клиенты могут выступать в следующих ролях: источников сообщений; приемников сообщений; и источников и приемников сообщений.

Между компонентами передаются типизированные сообщения фиксированного формата (рис. 3). Типизируются сообщения числовым кодом типа сообщения. Для сообщений с одинаковым кодом типа должен быть одинаковым тип и число аргументов в блоке аргументов.

Рис. 3. Формат передаваемого сообщения

Количество типов передаваемых сообщений произвольно. Формат блока аргументов (его размер и типы значений) определяются разработчиком компонент. За счет типизации сообщений упрощается обмен информацией

между компонентами, и становятся допустимыми дополнительные механизмы фильтрации. В частности, к возможности фильтрации по трем уровням (внутрипроцессный / кросспроцесный / сетевой), описанным выше, добавляется возможность фильтрации сообщений по типам. Для включения такой фильтрации компонент должен установить локальному ММС список принимаемых сообщений и в этом случае ему будут передаваться только сообщения с типами из этого списка (пустой список соответствует разрешению на прием сообщений всех типов). В локальном и сетевом ММС на базе списков отдельных компонент формируются совокупные списки фильтрации, которые позволяют отсекать от приема сообщения, обработка которых не представляется возможной, что позволяет снизить объем «лишних» пересылок.

Благодаря выбранному механизму взаимодействия получается программная система, компоненты которой являются слабо-связными и событийно-управляемыми. Автоматически получается развязка инициатора событий от его обработчика(ов), компонент, публикующий сообщение может не знать кто является его обработчиком и сколько их будет [6]. Описанная схема:

- позволяет задать унифицированный механизм обмена между основными компонентами приложения и добиться хорошей распределенности;
- позволяет реализовывать как среднезернистый (параллельное исполнение модулей) так и крупнозернистый (параллельное функционирование компонент приложения) параллелизм. При этом не исключается реализация мелкозернистого параллелизма (при реализации каждого из подключаемых модулей);
- дает возможность гибкой реконфигурации (или наращивания) системы (замена какого-либо компонента на более новый потенциально может быть осуществлена без остановки всей системы);

- обеспечивает простоту доработки компонент системы.

При разработке компонент в приложении, построенном по управляемой программной архитектуре, рекомендуется использовать исключительно событийный способ взаимодействия компонент между собой.

ЛИТЕРАТУРА

1. Борисов А.В., Куриленко И.Е., Чернышева Н.В. Управляемая программная архитектура // Сб. док. IX международной конференции «Информатика: проблемы, методология, технологии» в 2 т. – Т.1 – 2009. – С. 438–441.
2. Борисов А.В., Куриленко И.Е., «Применение управляемой программной архитектуры при разработке систем автоматизации парковочных комплексов», Труды междунар. науч.-техн. конф. «Информационные средства и технологии МФИ-2007». – М.: МЭИ, 2007. – Т.3. – С. 35-38.
3. Brenda M. Michelson «Event Driven Architecture Overview» Patricia Seybold Group. 2, 2006.
4. Кутепов В.П. Об интеллектуальных компьютерах и больших компьютерных системах нового поколения // Изв. РАН. Теория и системы управления 1996. № 5.
5. Черняк Л. «Архитектура, управляемая событиями» – Открытые системы, №2, 2005.
6. K. Mani Chandy «Event Driven Applications: Costs, Benefits and Design Approaches» - California Instituate of Technology, 2006.