СБОРНИК **НАУЧНЫХ ТРУДОВ НГТУ. – 2012. – № 1(67) –** 121–132

УДК 62-50:519.216

МЕТОДОЛОГИЯ ПРОЕКТИРОВАНИЯ СОВРЕМЕННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ПРИМЕНИТЕЛЬНО К СТАНЦИИ УПРАВЛЕНИЯ ЛИФТОМ*

Е.В. ШОБА, А.В. МАРКОВ

На примере современной станции управления лифтом предложен алгоритм создания и проектирования программного обеспечения (ПО) современной станции управления лифтами (ССУЛ). Рассмотрены общие методологии программирования и средства проектирования. Спроектированы диаграмма последовательности и диаграмма активности. Осуществлена проверка корректности построения диаграмм при помощи аппарата сетей Петри.

Ключевые слова: UML, сеть Петри, пространство состояний, варианты использования, логическое представление, представление компонентов, представление размещения, ООП, ООАП, объект, класс, наследование, инкапсуляция, полиморфизм, ССУЛ Союз.

ВВЕДЕНИЕ

Современная станция управления лифтом – это сложный программно-аппаратный комплекс, система взаимосвязанных элементов, которая должна проектироваться и реализовываться с использованием самых последних достижений техники и информационных технологий.

Основной объем работ в создании и проектировании ССУЛ можно разделить на два больших направления.

1. Проектирование и разработка аппаратной части комплекса.
2. Проектирование и разработка программной части комплекса.

Первое направление «Проектирование и разработка аппаратной части комплекса» останется за рамками данной статьи, оно интересно, но достаточно тривиально. В общем объеме работ первое направление требует около 10–15 % затраченных временных ресурсов.

Сегодняшний уровень развития микропроцессорной техники и сопутствующей элементной базы очень высок. Все функции по контролю, управлению и измерениям можно доверить одному или нескольким процессорам, тем самым заменив множество логических микросхем, которые были необходимы ранее.

На первый план выходит не аппаратная часть, а именно программный функционал. Имея большой опыт работы в данном направлении, можно заметить, что трудоемкость разработки программных приложений на начальных этапах программирования оценивается значительно ниже того, что реально придется затратить. Это является причиной дополнительных расходов и затягивания окончательных сроков готовности проекта.

В процессе разработки программной части, как правило, изменяются функциональные требования заказчика либо появляются новые исходные данные (информация), которые были неизвестны ранее. Это также отдаляет момент окончания работы программистов.

Наличие данных моментов, усложняющих разработку и оттягивающих сроки сдачи проекта, требует уделить больше внимания именно этапу «Проектирование и разработка программной части комплекса». В дальнейшем будем называть Программную часть ССУЛ просто ПО.

1. СУЩЕСТВУЮЩИЕ МЕТОДОЛОГИИ ПРОГРАММИРОВАНИЯ

Методология процедурно-ориентированного программирования

Базовым понятием этой методологии является понятие алгоритма, под которым в общем случае понимается точно определенная последовательность действий, направленная на достижение заданной цели, или решение поставленной задачи. Примерами алгоритмов служат хорошо известные правила нахождения длины окружности, площади квадрата и т.п.

Понятие процедуры определяет общее понятие алгоритма применительно к решению задач на компьютерах. Процедура представляет собой законченную последовательность действий, направленных на решение отдельной задачи.

Главная особенность процедурного программирования заключается в том, что программа всегда имеет начало во времени или начальную процедуру – начальный блок и окончание – конечный блок.

В рамках данной методологии получило развитие нисходящее проектирование программ или программирование сверху вниз (водопад).

Наибольшая популярность структурного программирования приходится на конец 70-х начало 80-х годов прошлого века.

Методология объектно-ориентированного программирования

Со временем ситуация стала существенно изменяться. Сложность программ непрерывно росла. Набор функциональных требований также увеличивался.

Стало очевидным, что традиционные методы процедурного программирования не способны справиться ни с растущей сложностью программ и их разработкой, ни с необходимостью повышения их надежности. Во второй половине 1980-х годов возникла потребность в новой методологии программирования, которая была бы способна решить весь этот комплекс проблем. Такой методологией стало объектно-ориентированное программирование (ООП). Фундаментальными понятиями ООП являются понятия класса и объекта.

Объект – компьютерное представление сущностей реального мира (предметов реального мира или понятий, придуманных человеком). Каждый объект в системе имеет две характеристики: состояние и поведение.

Класс – это описание группы объектов с общими свойствами (атрибутами), поведением (операциями), отношениями с другими объектами и семантикой. Класс – это шаблон для создания объекта. Каждый объект является экземпляром конкретного класса.

Основными принципами ООП являются наследование, инкапсуляция и полиморфизм.

Наследование – это принцип, в соответствии с которым знание о более общей категории разрешается применять для более узкой категории.

Инкапсуляция – это принцип скрытия отдельных деталей внутреннего устройства классов от внешних по отношению к нему объектов или пользователей.

Полиморфизм – это свойство некоторых объектов принимать различные внешние формы в зависимости от обстоятельств.

Широкое распространение методологии ООП оказало влияние на процесс разработки программ. В частности, процедурно-ориентированная декомпозиция программ уступила место **объектно-ориентированной декомпозиции**, при которой отдельными структурными единицами программы стали являться не процедуры и функции, а классы и объекты с соответствующими свойствами и методами.

При использовании ООП каждая программа представляет собой бесконечный цикл ожидания некоторых заранее определенных событий. Инициаторами событий могут быть другие объекты или пользователи. При наступлении отдельного события, например, пришел вызов от другой станции управления, находящейся в группе, программа выходит из состояния ожидания и реагирует на это событие вполне адекватным образом.

Наиболее существенным обстоятельством в развитии методологии ООП явилось понимание того факта, что процесс написания программного кода может быть отделен от процесса проектирования структуры программы.

Все эти обстоятельства привели к появлению новой методологии, получившей название «Методология объектно-ориентированного анализа и проектирования» (ООАП).

Методология объектно-ориентированного анализа и проектирования

Выделение исходных или базовых компонентов предметной области, необходимых для решения той или иной задачи, представляет в общем случае нетривиальную проблему. Сложность данной проблемы проявляется в неформальном характере процедур или правил, которые можно применять для этой цели. Более того, такая работа должна выполняться совместно со специалистами или экспертами, хорошо знающими предметную область.

Появление методологии ООАП потребовало, с одной стороны, разработки различных средств концептуализации предметной области, а с другой – соответствующих специалистов, которые владели бы этой методологией. Как раз в это время появляется относительно новый тип специалиста, который получил название аналитика или архитектора проекта.

Разделение процесса разработки сложных программных приложений на отдельные этапы способствовало становлению концепции жизненного цикла (ЖЦ) ПО. Под ЖЦ программы понимают совокупность взаимосвязанных и следующих во времени этапов, начиная от разработки требований к ней и заканчивая полным отказом от ее использования.

Методология ООАП тесно связана с концепцией автоматизированной разработки программного обеспечения (Computer Aided Software Engineering, CASE). Данную концепцию реализуют современные средства проектирования (CASE-средства) в совокупности с унифицированным языком моделирования (Unified Modeling Language, UML), который ориентирован на решение задач первых двух этапов ЖЦ программ.

Средства проектирования

История развития языка UML [2] берет начало с октября 1994 года, когда Гради Буч и Джеймс Румбах из Rational Software Corporation начали работу по унификации методов Booch и OMT. Хотя сами по себе эти методы были достаточно популярны, совместная работа была направлена на изучение всех известных объектно-ориентированных методов с целью объединения их достоинств.

Усилия Г. Буча, Дж. Румбаха привели к появлению первых документов, содержащих описание собственно языка UML версии 0.9 (июнь 1996 г.) и версии 0.91 (октябрь 1996 г.).

Компания Rational Software Corporation стояла у истоков разработки языка UML. Эта компания разработала

и выпустила в продажу одно из первых инструментальных CASE-средств Rational Rose 98, в котором был реализован язык UML.

В настоящее время существует версия UML 2.0, которая и взята за основу проектирования, но существуют и более поздние версии.

2. ПОСТРОЕНИЕ ПРОЕКТИРУЕМОЙ МОДЕЛИ

Для построения правильной и полной модели поведения системы ССУЛ необходимо спроектировать 4 представления, предлагаемые Rational Rose:

1 проектирование «Вариантов Использования»:
2 проектирование «Логического Представления»;
3 проектирование «Представления Компонентов»;
4 проектирование «Представление Размещения».

Предложен следующий алгоритм создания и проектирования ПО ССУЛ:

1 анализ предметной области и формулировки требований к ПО;
2 проектирование структуры программы:
- проектирование Вариантов Использования:
  - определение актеров;
  - определение прецедентов;
  - определение потоков событий для прецедентов;
  - определение ассоциации;
  - составление диаграммы последовательности для каждого П. и для каждого потока П.;
  - составление кооперативной диаграммы для каждого П. и для каждого потока П;
- проектирование Логического Представления;
- проектирование Представления Компонентов;
- проектирование Представление Размещения;

3 реализация программы в кодах (собственно программирования);
4 внедрение программы.

На рис. 1 представлен пример диаграммы использования.

Выявлены действующие актеры системы:

Пассажир, Пожарная команда, Механик, Диспетчер, Другая ССУЛ.

Выявлены основные прецеденты системы:

Движение на этаж, Открывание дверей, Управление с крыши кабины, Управление из МП, Передачи информации о ССУЛ, Прием команд для ССУЛ, Передача информации о вызовах, Прием информации о вызовах;

Выявлены взаимодействия между ними.

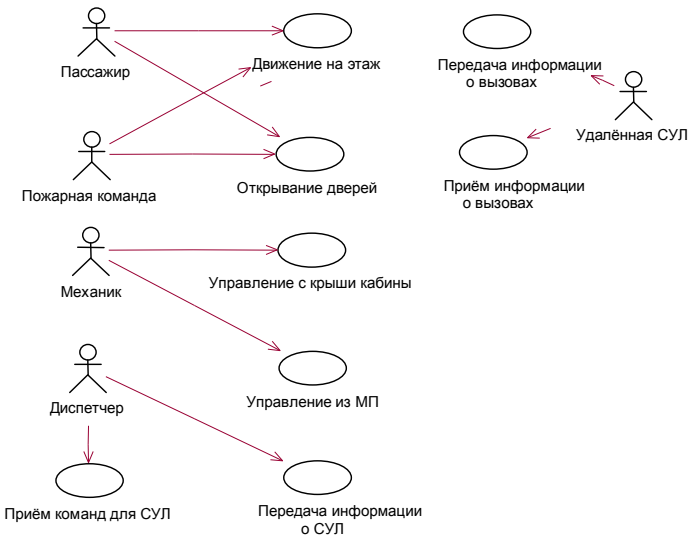

Рис. 1. Вариант диаграммы использования

На рис. 2 представлен пример диаграммы последовательности для подпотока «Не принятия вызова для обработки». Данный рисунок показывает взаимодействующие объекты системы и наборы сообщений, которыми

объекты обмениваются между собой.

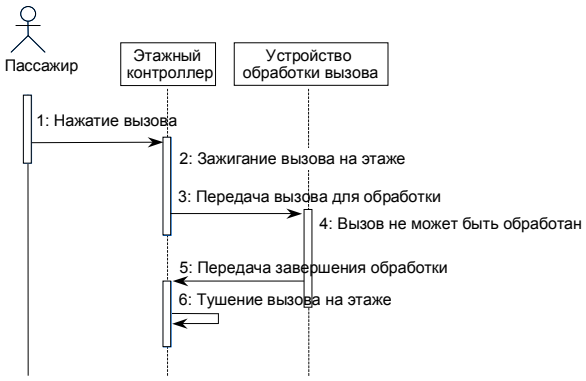

Рис 2. Вариант диаграммы последовательности (не принятия вызова для обработки)

На рис. 3 представлен пример диаграммы активности для прецедента «Эвакуатор». Данный тип диаграмм рекомендуется реализовывать в логическом представлении, а не в представлении вариантов использования; здесь он приведен для примера.

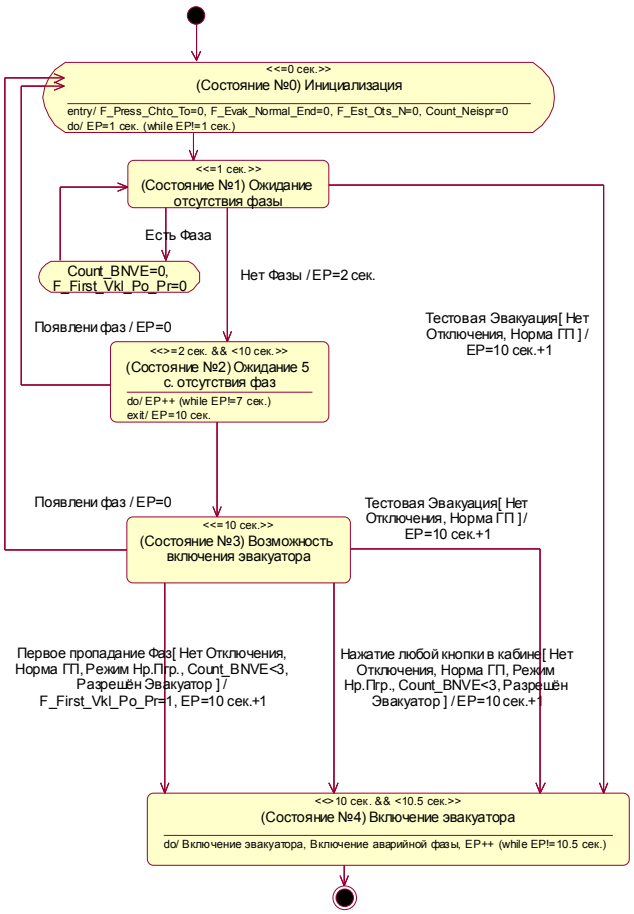

Рис. 3. Пример варианта диаграммы активности

3. ПРОВЕРКА СПРОЕКТИРОВАННОЙ МОДЕЛИ

Отличным средством верификации спроектированных UML диаграмм является аппарат сетей Петри [4]. Сеть Петри представляется как двудольный ориентированный граф, состоящий из вершин двух типов, не имеющих возможности соединяться непосредственно. Между вершинами перемещается метка (маркер, фишка). Полученные диаграммы транслировались в сети Петри [5–7].

Диаграмма последовательности, изображенная на рис. 2, при преобразовании в сеть Петри будет выглядеть так, как показано на рис. 4.

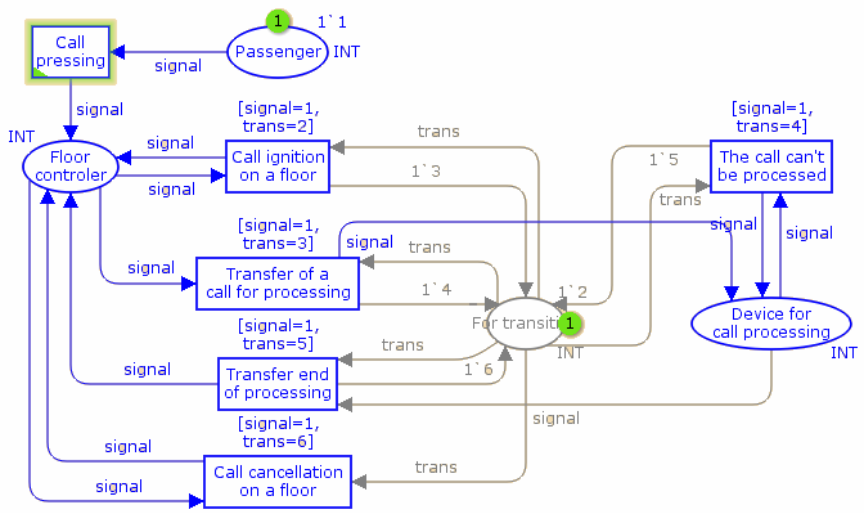

Рис. 4. Сеть Петри, транслированная из диаграммы последовательности

Моделирование сети производилось в программной среде CPN Tools, которая обладает таким весьма полезным функционалом, как проверка корректности построения сети посредством генерации пространства состояний. Верификация сгенерированного пространства состояний (табл. 1) показала, что построенная сеть, а следовательно, и диаграмма последовательности смоделированы верно.

На рис. 5 показана сеть Петри, смоделированная на основе диаграммы активности.

Таблица 1

Частичный отчет о сгенерированном пространстве состояний (диаграммы последовательности)

Результаты отчета	Выводы
State Space Nodes: 7 Arcs: 6 Secs: 0 Status: Full	Пространство состояний модели вычислено полностью за 0 секунд и содержит 7 узлов, 6 дуг.
Home Markings [7] Dead Markings [7] None	В сети 7 «домашних» маркировок и 7 «мертвых».
Dead Transition Instances None Live Transition Instances None	В сети нет «мертвых» переходов.
Fairness Properties No infinite occurrence sequences	В сети отсутствуют бесконечные последовательности срабатывания

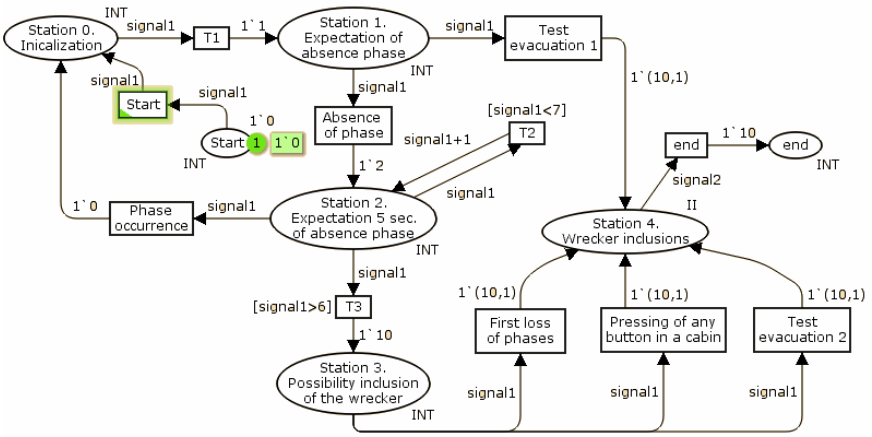

Рис. 5. Сеть Петри, полученная из диаграммы активности

В табл. 2 приводится отчет о полученном пространстве состояний, из которого видно, что спроектированная сеть и, в свою очередь, диаграмма активности построены корректно.

Таблица 2

Частичный отчет о сгенерированном пространстве состояний (диаграммы активности)

Результаты отчета	Выводы
State Space Nodes: 12 Arcs: 20 Secs: 0 Status: Full	Пространство состояний модели вычислено полностью за 0 секунд и содержит 12 узлов, 20 дуг
Home Markings [6] Dead Markings [6] None	В сети 6 «домашних» маркировок и 6 «мертвых»
Dead Transition Instances None Live Transition Instances None	В сети нет «мертвых» переходов
Transition Instances with No Fairness None	Все переходы являются живыми

ЗАКЛЮЧЕНИЕ

В ходе проведенной работы показана методология проектирования современных систем ПО. В качестве примера взята система управления лифтом. Данная модель, спроектированная и в дальнейшем разработанная, получила название «СОЮЗ». Система запущена в производство и в настоящий момент серийно производится предприятием ООО «ППК ЭССАН-Лифтэк», г. Новосибирск. Эксплуатируется на территории всей России.

Одной из главных причин надежной и безотказной работы ССУЛ «СОЮЗ» следует считать правильно разработанную и реализованную методологию проектирования программного обеспечения станции.

[1] *Трофимов С.А.* CASE-технологии: практическая работа в Rational Rose. – Изд. 2-е. – М.: Бином-Пресс, 2002. – 288 с.

[2] *Терри Кватрани.* Rational Rose 2000 и UML. – М.: ДМК, 2001. – 176 с.

[3] *Питерсон Дж.* Теория сетей Петри и моделирование систем / пер. с англ. – М.: Мир, 1984. – 264 с.

[4] *Коротиков С. В.* Применение сетей Петри в разработке программного обеспечения центров контроля и управления: дис. … канд. техн. наук. – Новосибирск: Изд-во НГТУ, 2007.

[5] *Романников Д.О.* Использование UML-диаграмм и сетей Петри в разработке программного обеспечения систем распределенной обработки информации, критических ко времени исполнения: магистерская

диссертация. – Новосибирск: НГТУ, 2011.

[6] *Марков А.В.* Поиск манипулятором кратчайшего пути в лабиринте // Сборник научных трудов НГТУ. – 2011. – № 4(66). – С. 75–90.

[7] *Романников Д.О., Марков А.В., Зимаев И.В.* Обзор работ, посвященных разработке ПО с использованием UML и сетей Петри // Сборник научных трудов НГТУ. – 2011. – № 1(63). – С. 91–104.

Шоба Евгений Владимирович – соискатель кафедры автоматики Новосибирского государственного технического университета. Основное научное направление – синтез многоканальных систем управления. Имеет 14 публикаций. E-mail: shobaev@essan.ru.

Марков Александр Владимирович – аспирант кафедры автоматики Новосибирского государственного технического университета. Тема диссертационного исследования – автоматизация написания программного обеспечения для технических систем. Имеет 6 публикаций. E-mail: muviton3@mail.ru.

E.V. Shoba, A.V. Markov
Methodology of the modern software designing with reference to station of lift management

On an example of modern station of management the lift, offers algorithm of creation and designing of software Modern Station of Lifr Management (MSLM). The general are considered programming and design tool methodology. The diagram of sequence and the activity diagram are designed. Check of a correctness of construction of diagrams by means of the device of Petri nets is carried out.

Key words: software engineering, UML-diagrams, Petri nets, CPN Tools, parallel processes, the manipulator. UML, Petri nets, space of conditions, use case diagram, logic representation, representation of components, placing representation, object-oriented programming (OOP), object-oriented analysis and designing (OOAD), Object, Class, Inheritance, encapsulation, polymorphism, Union of MSLM.