Лекция № 1. Определение визуального моделирования программного обеспечения.

В этой лекции рассказывается о роли чертежей в инженерных областях (строительстве, машиностроении, электротехнике и т.д.), рассматриваются ключевые свойства программного обеспечения – сложность, незримость, согласовываемость, изменчивость, и обсуждаются перспективы использования чертежей в программировании. Вводится понятие метафоры визуализации, визуальное моделирование определяется как визуализация, основанная на графах.

О пользе чертежей. Потребность в создании стандартных, одинаково используемых всеми, методов разработки ПО, способных гарантировать предсказуемый результат разработок, привела к поискам аналогий в других промышленных областях. Было замечено, что в строительстве, машиностроении, электротехнике и пр. работы разбиваются на проектирование и реализацию, проектирование проводят другие люди, нежели те, кто собственно изготовляет систему. Проектирование осуществляется с помощью чертежей – схематичных изображения создаваемой системы, – которые служат хорошим интерфейсом между проектировщиками и рабочими, строителями, электромонтерами, инженерами-сборщиками и т.д. Последним нет необходимости принимать какие-либо серьезные решения по поводу создаваемых систем – все решения уже приняты, нужно смотреть внимательно в чертежи и делать так, как там обозначено. Более того, наличие «пространства проектирования» позволяет накапливать и повторно использовать успешный опыт разных разработок, способствуя стандартизации и унификации производства.

Чертежи хорошо «работают», так как формулируют основные идеи системы в терминах ее реальных геометрических форм (без деталей, в масштабе и пр.). Поскольку у современного человека доминирующий канал восприятия информации – зрительный, то, если ему удается увидеть будущую систему хотя бы в упрощенном, схематичном виде, то ее разработка существенно упрощается. Участникам проекта легче понять друг друга, поскольку они вместе видят на чертеже объекты физического мира, вызывающие в памяти устойчивые образы других похожих объектов, которые уже существуют и в создании которых эти же люди принимали участие.

Аналогично образом хотели использовать чертежи и в программировании, однако натолкнулись на ряд особенностей по, не позволяющих использовать чертежное проектирование as is.

Характеристические признаки программного обеспечения. Программное обеспечение заметно отличается от других систем, создаваемых человеком, деятельность по его созданию имеет особенности по сравнению с иными видами деятельности человека. Этот тезис – не просто вводные слова, которые обычно говорят в начале статей, книг, лекций и пр. Осознание особенностей ПО и процесса его разработки важно, чтобы понять, что из подходов, созданных в иных областях, можно и нужно применять в программных проектах, а где нужно создавать специфические, свойственные только программной индустрии, подходы.

Фредерик Брукс выделил следующие четыре фундаментальные свойства ПО, существенно отличающие его от других от рукотворных объектов и систем – инженерных сооружений и машин, научных теорий, произведений искусства и пр.:

- сложность;
- невидимость;
- согласовываемость;
- изменчивость.

Все эти свойства очень сильно связаны друг с другом, подкрепляют и усиливают друг друга, являясь, по сути, разными гранями одного и того же – см. рис. 1.

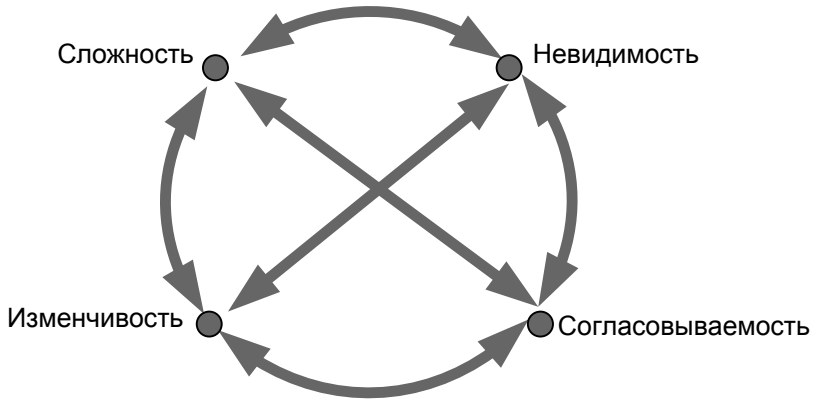

Рис. 1.1.

Сложность не позволяет создавать ПО на основе простых законов, подобно тому, как естественные науки описывают природу, сводя многообразие явлений к небольшому множеству законов, выраженных математически. Таким образом, не выделить не построить единой сквозной структуры, емкого описания ПО – всегда будут какие-то исключения, неточности, всегда сформулированное будет некоторой ассоциацией, приближением. Таким образом, возникает проблема поддержания концептуальной целостности ПО, так как не удается ее выразить просто и емко.

Незримость ПО сильно подкрепляет эту сложность. Программные системы не являются объектами физической реальности, как, например, здания, различные приборы и устройства. Они виртуальны, и конечными пользователями воспринимаются как сервисы по предоставлению определенных информационных услуг, часто управляя при этом различными физико-механическими устройствами. Из-за незримости возрастает сложность ПО, поскольку, фактически, 90% информации человек воспринимает зрительно. Очень сложно постичь незримое, коллективно конструировать незримое, определять и согласовывать его функции вместе с очень сильно различающимися по менталитету, образованию и взглядам на мир людям – программистам, инженерам, пользователям и т.д. Часто бывает, что каждый из членов этого сообщества представляет систему как-то по-своему, даже относительно очень простых и базовых ее свойств. А между тем согласовываемость ПО является его следующим важным свойством.

Согласовываемость ПО означает, что принципиальную важность в процессе его создания человеческих договоренностей, вырабатываемых в совместном общении. Ведь ПО основывается не на объективных посылках, как, например, закон всемирного тяготения (тела, прошенные вниз, падают…), а на многочисленных людских мнениях об удобстве, эффективности в той целевой области, где ПО будет работать, эргономичности и пр. Мнение о невидимом вынуждены высказывать очень многие люди, не объединенные одной какой-то областью, в рамках которой они могли бы выработать единое представление, требования, порядок (что хорошо, что плохо) и уровень субъективизма в такой ситуации чрезвычайно высок.

Наконец, известно, что ПО очень *изменчиво*. Все (и программисты, и менеджеры, и пользователи/заказчик) знают, что ПО очень легко немного изменить – вставить новый

диалог, добавить новую функцию, исправить что-либо и пр. Поэтому требования к программным системам часто меняются, и тут трудно установить барьер, отделяющий незначительные и легко реализуемые изменения от тех новшеств, привнесение которых требует существенных работ. На изменчивость требований влияет также и сложность ПО, поскольку оказывается трудным сразу точно определить, что оно должно делать, как должны выглядеть интерфейс системы и т.д. Кроме того, ПО часто изменяется «изнутри», что связано с трудностью предварительного проектирования архитектурных решений. Часто бывает, что, реализовав значительную часть системы, разработчики приходят к необходимости существенно переделывать свою работу[1]. Забегая вперед отметим, что изменчивость ПО поднимает проблему сопровождения дополнительных по отношению к программному коду артефактов разработки – всякой документации, моделей (в том числе и визуальных). Все это особенно трудно поддерживать в целостности (то есть согласовано менять), поскольку у него нет четкой концептуальной основы.

Чертить ПО… В силу сложности ПО невозможно полностью завершить его предварительное проектирование и создать подробные спецификации до начала разработки, как, например, при строительстве дома. Оказывается, что проектные спецификации склонны меняться при разработке, что совершенно не допустимо для проекта здания – проект меняется во время строительства лишь в исключительных случаях, а при создании ПО это обычное явление. Часто ситуация с проектированием ПО соответствует высказыванию Наполеона «сначала нужно ввязаться в бой, а там будет видно…».

Далее, в силу невидимости ПО, его чертежи не приносят в проект той магической ясности и определенности, как чертежи здания (пусть даже очень сложные). Чертежи ПО почти невозможно читать вне контекста проекта, без их авторов и т.д. Мы не можем свести новую, незнакомую сложность к потоку одинокого воспринимаемых всеми зрительных форм, тем самым упростив и прояснив ситуацию. Не ясно что чертить, как чертить, соответственно, если что-то начерчено, то не автору не ясно, что это….

Изменчивость ПО приводит к тому, что постоянно меняются все артефакты проекта, поддержка их согласованности оказывается не простой задачей. Чертежи ПО часто отстают от самого ПО и не хватает ресурсов на поддержание их в актуальном состоянии.

Однако все эти обстоятельства не зачеркивают идею использовать чертежи при создании ПО. Мы склонны считать ситуацию не безнадежной, а иной. Поиск ответов и решений на тему того, как проектировать ПО с помощью чертежей приводит к созданию специального раздела программной инженерии – визуальному моделированию.

Визуальное моделирование, визуализация ПО, метафоры визуализации. Задача визуализации в программировании имеет более широкий аспект, чем способы предварительного проектирования, например:

- разработка визуальных языков программирования;
- способы анимации и визуального представления различных аспектов работающего ПО (параллельных процессов, отладки, различных исполняемых характеристик – быстродействия, загрузки памяти и пр.);
- способы наглядно представлять разные артефакты разработки, например, парадигма WYSIWYG (What You See Is What You Get), используемая, например, в средствах разработки пользовательских интерфейсов;

[1] Необходимость переделок получила свое очередное официальное оправдание перед менеджерами и заказчиками в лице техники под названием рефакторинг. Существует много разных стратегий рефакторинга кода, а также программных инструментов поддержки.

- различные виды интерфейсов целевого ПО – например, метафора письменного стола, используемая в Windows;
- компьютерная визуализация различных видов электронной информации.

В силу невидимости ПО, то есть отсутствия естественной формы визуализации, центральным моментом здесь оказывается поиск различных подходящих метафор визуализации.

Когда нам нужно объяснить что-то человеку или некоторой аудитории, мы часто пользуемся аналогиями из опыта и областей знаний, доступных тем, кому мы объясняем. То есть мы используем метафоры.

В компьютерной визуализации термин «метафора» имеет дополнительный смысл. Под ним понимают способ сопоставления абстрактным и невидимым человеческому глазу абстракций ПО некоторых графических образов. Метафорично ли это сопоставление, то есть используются ли какие-либо знакомые зрительные аналогии, или конструируются принципиально новые зрительные образы – вопрос философский. Важно лишь всем договориться, что ПО видим и изображаем так то и так то, тем самым задействовав зрительный канал при проектировании и разработке ПО, передаче знаний о нем, обучении и т.д. В этом смысле неоценимую пользу оказывает развитие стандартных визуальных языков разработки ПО, сегодня, в первую очередь, UML. Люди привыкают к изображениям классов, пакетов, объектов, процессоров и пр., к определенному набору диаграмм. Они привыкают мыслить, строя те или иные диаграммы UML. А другие легко читают эти мысли в этих диаграммах.

Более детальную информацию по визуализации ПО, метафорам и методам визуализации, можно найти в работах [3], [4], [5].

На рис. 2 показан пример метафоры визуализации ПО из области визуального программирования.

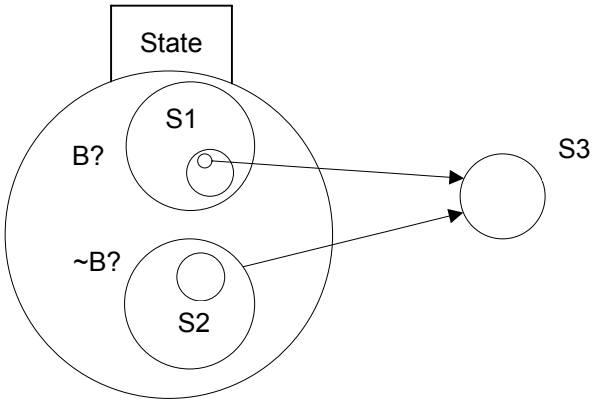

Рис. 1.2

На этом рисунке с помощью графического языка VIPR (Visual Imperative Programming) графически изображена условная конструкция if B then S1 else S2; S3.

Графовая метафора и определение визуального моделирования. Среди различных метафор визуализации ПО выделяются математические графы – вершины, изображаемые по-разному, и ребра – стрелки, связи, зависимости и т.д. На рис. 3 (а), (б), (в) приводится несколько типов диаграмм UML.

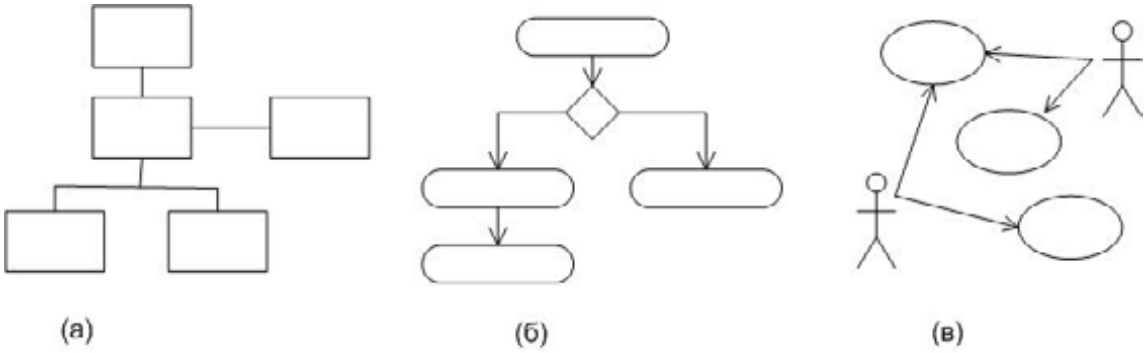

Рис. 1.3

Из этого рисунка видно, что мы видим по-разному изображенные графы. На настоящий момент, несмотря на многочисленные попытки, не создано другой общеупотребительной метафоры визуализации программ.

Определим *визуальное моделирование* как подход, позволяющий в процессе разработки и эволюции ПО выполнять его визуализацию на основе графов. Эта визуализация выполняется при проектировании, при изучении и сопровождении ПО, при его разработке и т.д.

Однако, отметим, что не все виды диаграмм, традиционно используемые в рамках визуального моделирования, являются графами. Например, диаграммы последовательностей (interaction diagrams) или временные диаграммы (timing diagrams) UML. Однако из 13 диаграмм UML 2.0 графов 11, а не графовых всего 2.

Основная литература

1. F. Brooks. No Silver Bullet. – Information Proceeding of the IFIP 10th World Computing Conference. 1986. – P. 1069-1076. (Русский перевод: Ф. Брукс. Мифический человеко-месяц или как создаются программные системы. – СПб Символ, 2000).
2. Д.В. Кознов. Языки визуального моделирования: проектирование и визуализация программного обеспечения. – Учебное пособие. Изд-во СПбГУ, 2004. – 143с.

Дополнительная литература

3. Л.Авербух. Метафоры визуализации. – Программирование, 2001, N 5. – С. 3-17.
4. В.Л.Авербух. К теории компьютерной визуализации. http://cv.imm.uran.ru/articles/cvtheory_w23print.pdf.
5. D. Gracanin, K. Matkovic, M. Eltoweissy. Software Visualizatiотn. – Innovation in Systems and Software Engineering. A NASA Journal. V. 1, No. 2 September 2005, Springer. – P. 221- 230.