«Человеческие» особенности использования UML

Д. В. Кознов
dim@dk12687.spb.edu

А. Ф. Перегудов*
peregudov@dip.ru

В работе предлагается новая классификация методов использования UML (Unified Modeling Language) — широко распространенного в программной индустрии языка для создания проектных чертежей программного обеспечения. Рассматриваются следующие функции разработки программного обеспечения: проектирование принципиально новой, уникальной системы; компоновка и формализация проекта системы; изучение существующей системы; передача знаний о системе. Эти функции определяют различные психические состояния разработчиков и являются, вместе с последними, основой предлагаемой классификации.

Введение

Уникальность программного обеспечения (ПО) по сравнению с другими искусственными системами и инженерными продуктами многократно отмечалась. Например, Ф. Брукс выделял следующие особенности ПО: сложность, согласованность, изменчивость и незримость [1]. Однако у ПО существует еще одно измерение, также отличающее его от других систем, создаваемых человеком, —

* Санкт-Петербургский государственный университет кино и телевидения. 191126, ул. Правды, 13, Санкт-Петербург, Россия

богатая палитра психологических состояний, переживаемых участниками проекта по его созданию.

Одним из главных источников этого разнообразия является необходимость для членов команды совмещать в себе энтузиазм, творческий подъем, поиск нестандартных решений с такими характеристиками индустриального процесса, как надежность, контроль, строгая отчетность, дисциплина и т. д. Можно представить, как трудно одному человеку быть вдохновенным творцом новых, нестандартных решений, проявлять творческую интуицию (все это спонтанные и во многом непредсказуемые процессы!) и одновременно быть детально спланированным, предсказуемым и надежным, известным наперед, пунктуальным и точным, стабильным и равнопроизводительным.

Еще одной фундаментальной психологической особенностью процесса разработки ПО является необходимость сочетать математико-технический склад мышления с другими типами человеческого мировосприятия. Программный продукт предназначается для самых разных людей (в том числе и непрограммистов!), и с этими людьми разработчикам приходится общаться, ориентироваться на них при создании продукта, представлять и моделировать их жизнь, бизнес. Все это необходимо для того, чтобы создать систему, удовлетворяющую нуждам людей.

Сам процесс программирования требует сосредоточенности и скрупулезности, точности и дотошности, переходящей порой в подозрительность и придирчивость. Программист умеет одновременно удерживать во внимании большое число различных деталей, обладает развитыми навыками по их вычленению, виртуозно дифференцируя явления окружающего мира. Извлеченная информация облекается им в точные формулировки, максимально приближенные к логике вычислительных устройств. Его внимание направлено, как узкий и острый луч лазера, — в монитор, на который он напряженно смотрит по 8–12 часов в сутки, на листинги программ, в монотонных и внешне однообразных строках которых он различает богатую различными нюансами жизнь. В своей избирательности внимание программиста очень активно и во многом напряжено. Программиста нелегко в чем-либо убедить, он очень критичен, порой ироничен и насмешлив или неулыбчив, чрезмерно серьезен и угрюм[1]. И вот этому человеку нужно отвлечься от программиро-

[1]Мы не претендуем на то, что наш портрет программиста исчерпывающе

вания и обсудить с будущим пользователем различные свойства программной системы — как она будет выглядеть внешне (диалоги, переходы между ними, наличие в нужных местах нужных кнопок и т. д.), как она встроится в повседневную деятельность пользователя и пр.

Для успешного диалога программисту приходится сильно менять свое психическое состояние. Его внимание теряет острую направленность и напряженность, становится более расфокусированным и менее деятельным, смягчается. Критичность и неуступчивость уступают место мягкости и открытости. В поле его внимания вместо программы появляется другой человек — участник диалога — со своими задачами, вопросами, со своим собственным миром. В этом мире ничего не нужно активно вычленять, дифференцировать, выискивать и конструировать. Терпеливость и доброжелательность, искренний интерес к жизни другого человека, умение слушать и задавать нужные вопросы, плавно направлять разговор в нужное русло — такое поведение программиста дает хороший результат в этой ситуации. Тем более что собеседник часто не имеет математического или технического образования, не может хорошо отделить свои эмоции от фактов, не знает, что важно для программиста, а что второстепенно. Очевидно, что программисту очень не просто изменить свое состояние, чтобы вести такой диалог конструктивно[2]. Кроме того, программистам необходимо общаться с различными специалистами, также участвующими в разработке, но существенно отличающимися от них по складу мышления и мировосприятию. Это менеджеры, инженеры, специалисты по маркетингу и продажам, технические писатели, тестировщики и т. д. Для успешного взаимодействия с ними программисту также нужно менять свое психическое состояние.

Очень часто программисты не могут этого сделать. У них возникает раздражение на «глупых пользователей», которые «сами не знают, чего хотят», «смешивают в одну кучу все на свете». Они

полон. Но нам кажется, что наше описание содержит ряд характерных особенностей, по которым можно узнать, представить таких людей.

[2]Ситуация усложняется еще и тем, что взаимодействие с пользователем не ограничивается разовым вербальным общением — при работе программисту целесообразно не забывать пользователя, погружаясь в сложные задачи по реализации системы, активно задействовать свою память и воображение. Мы не будем здесь останавливаться на деталях и механизмах этого процесса, поскольку цель этой работы — не давать рецепты, а обозначить вопросы и лишь наметить пути их разрешения.

видят в других людях-непрограммистах неполноценных людей, не умеющих точно формулировать свои потребности и даже «просто элементарно логически мыслить». Таким образом возникает психологическая пропасть между программистами и остальными людьми, которая порождает многочисленные и хорошо освещенные в литературе проблемы: трудность работы с требованиями (их извлечение, неискажающая формализация, своевременное отслеживание их изменений [2]), различные противоречия между менеджментом и разработчиками [3], трудности коммуникаций между программистами и инженерами [4]. В [5] упоминается о конфликте между разными рабочими группами самих программистов в случае, когда эти группы фокусируются на принципиально разных задачах: одни — на поддержке общих активов семейства продуктов, другие — на разработке отдельных продуктов.

С другой стороны, нарастает психологическая дискретность процесса разработки, в частности, происходит лавинообразный рост различных событий и артефактов разработки — требования, различные спецификации, тесты, код, ошибки, отчеты начальству, запросы заказчика и т. д.[3] На события нужно своевременно реагировать, артефакты необходимо вовремя создавать, отслеживать их целостность и корректность с течением времени. Все это заставляет программистов постоянно переключать свое внимание на разные вопросы, извлекая его из одной функции и «погружая» в другую. На это требуется время (психические процессы обладают определенной инертностью), а также силы. При недостатке времени и сил, при чрезмерной частоте таких переключений у человека возникает утомление, повышается нервозность и напряженность атмосферы проекта, возрастают негативные эмоции. Человеку хочется, чтобы никто его не беспокоил, ему хочется «тихо сидеть в своем углу» и «заниматься своим делом».

Таким образом, имеется большой дефицит «человечности» при разработке ПО. Все чаще процесс создания программных продуктов оказывает разрушающее воздействие на его участников[4].

[3] В одной компании, имеющей 5-й уровень CMM (Capability Maturity Model), по свидетельству менеджеров, до 40% времени разработчиков уходит на поддержание правил формального процесса — создание соответствующих документов, подписи их у ответственных лиц и т. д.

[4] См., например, монографию Эдвара Йордана о безнадежных проектах [6]. Эта монография интересна тем, что написана с точки зрения участников таких проектов и адресована в первую очередь им же. И уже во вторую очередь речь

1. О базовых функциях и соответствующих психических состояниях

Опишем следующий набор функций процесса разработки и сопровождения ПО, базовых с точки зрения использования UML:

1) проектирование принципиально новой, уникальной системы;

2) компоновка и формализация проекта системы, «снятие вторичных противоречий» [7];

3) изучение существующей системы;

4) передача знаний о системе.

При этом мы сделаем акцент на тех психических состояниях, которые возникают у разработчиков при выполнении этих функций и которые, на наш взгляд, определяют характер использования UML в проекте. Данные функции существуют в программных проектах независимо от того, используют ли разработчики UML или нет. Последний выступает в качестве вспомогательного инструмента в проекте наряду со многими другими инструментами и может действительно помогать, резонируя с соответствующей функцией и психическим состоянием человека, накладываться на них, усиливать и добавлять новые краски. И тем самым способствовать успешности проекта. А может противоречить, дисгармонировать, «рвать» ткань проекта.

В первом случае UML используется легко, с энтузиазмом, создаются новые методики его применения, появляются интересные книги и статьи. Во втором случае UML внедряется и используется трудно, «из-под палки» или «стиснув зубы», а результаты его применения часто оказываются бесполезными для проекта: диаграммы не соответствуют программному коду, их или слишком много и в них никто, кроме авторов, не может разобраться, или их мало и они бедны и невыразительны.

При описании базовых функций основной акцент мы сделаем на психических состояниях людей, в которые они «погружаются» при выполнении этих функций. Хочется отметить, что вербально-логическое описание явлений психической жизни человека имеет существенные ограничения. Еще Юнг отмечал, что психика человека состоит не только из ментальных процессов, но включает в себя также ощущения, чувства и интуицию [8]. Как словами описать

идет о бизнесе, управлении и других традиционных для программной инженерии вопросах.

цвет, запах, вдохновение? Это можно увидеть, почувствовать, пережить. Слова же могут только намекнуть, обозначить, всколыхнуть воспоминания (эффект узнавания) — то есть прямо или косвенно вызвать сами переживания в человеке, который эти слова читает или слышит.

Именно на подобные эффекты и рассчитано предлагаемое ниже описание, которое не является системным и детальным изложением в стиле «как должно быть» или исчерпывающим описанием того, «как было». Предложены зарисовки из личного опыта авторов в которых, как мы надеемся, содержатся зерна живой человеческой жизни, могущие вызвать отклик у читателя. Мы делаем вклад в создание определенной атмосферы, а не строим фундаментальную систему знаний[5].

2. Проектирование уникальной системы

У проектировщика много разрозненной информации о будущем ПО — свой собственный опыт в данной области, различные (и часто противоречивые) пожелания и требования к конечному продукту со

[5]Здесь мы во многом следуем стилю первых монографий и статей по объектно-ориентированному анализу и проектированию конца 80-х — середины 90-х годов: работ Эдвара Йордана, Грэди Буча, Флиппа Кратчена, Петера Кода, Джима Рэмбо и многих других. Именно из этих работ родился UML. Среди современных исследователей и практиков в области UML ярким представителем такого стиля является Мартин Фаулер. Работы всех этих исследователей во многом не были каноническими исследованиями, они написаны живо и неформально, целиком и полностью основаны на собственном практическом опыте авторов. Эти исследования явились частью определенной атмосферы, сложившейся в индустрии разработки ПО в Америке в 80–90-х гг. Вне этой атмосферы данные работы очень сложно воспринимать. Мы изучали их в это же время, находясь в России, и не могли разобраться в этой кажущейся простоте, понять, как использовать эти методы на практике. Спустя значительное время нам стало понятно, что мы сделали ошибку, от которой хотим сегодня предостеречь читателей данной работы: мы пытались понять буквально, извлечь из прочитанного схемы и методики, вместо того чтобы реконструировать тот контекст, в котором родились и развились данные методы, вообразить, представить. По всей видимости, науке, подобной программной инженерии, которая призвана решать практические вопросы, не следует иметь такой разветвленный абстрактно-ментальный аппарат, как классическая наука. Наличие такого аппарата чрезвычайно затрудняет «возврат» знаний в практическую деятельность и способствует замыканию науки в себе. В связи с этим интересна монография киевского психолога О. Бахтиярова [9], рассматривающего новый вид технологий, существенно ориентированный на нементальный способ человеческого восприятия.

стороны заказчика, пользователей, специалистов по маркетингу и продажам и т.д. Но у него отсутствует единая картина того, какой будет система, у него нет ее интегрального образа или другими словами — решения задачи.

Проектировщик может вести себя, например, так. Он размышляет, задумчиво глядя в окно, отрешенно пьет кофе[6]. Он погружен в море информации, она внутри него. Однако он не тасует ее отдельные аспекты, а созерцает все вместе. При этом он может заниматься любой деятельностью, углубляющей его созерцание. Проектировщик внешне может выглядеть рассеянным и беспорядочным, но внутренне он настойчив и интенсивен. Он сосредоточен и его нелегко вывести из этого состояния каким-либо внешним образом. Если ему нужно что-то узнать и уточнить, то его вопросы могут показаться странными и нелепыми для окружающих. В какой-то момент у него появляется гипотеза. Он может ее проверить и, если ему кажется, что она незрела, то он не акцентируется на ней. Но если ему кажется, что он «напал на след», он может придать гипотезе более конкретную форму — связывает вместе отдельные части и прорабатывает детали. И после этого старается почувствовать, понять, является ли найденное решение искомым или нет. И если ее величество реальность снизойдет до его настойчивых попыток, то в определенный момент многочисленные детали начинают складываться у него в единую картину. В его воображении возникает решение, система представляется ему как единое целое. Решение именно возникает, появляется, оказывается рядом с проектировщиком. Оно либо концентрируется вокруг него как некоторая непонятная, смутная пока еще уверенность, либо возникает как само собой разумеющаяся истина и т. д. Проектировщик не создает решения, не «делает» его, не синтезирует и не производит. Он его замечает или оно само дает себя заметить. Признаком того, что это решение — «то самое», могут служить возникающие у проектировщика чувства достоверности, содержательности, адекватности, ясности и др. И — одновременно тишина и отсутствие эпатажности, бурного эмоционирования. Для продуктивного творческого поис-

[6]Отдельные детали здесь не существенны — кто и что пьет при поиске решения и т. д. Комиссар Мегрэ у Жоржа Сименона в таких ситуациях пил, например, пиво, но в целом вел себя очень похоже. Еще раз подчеркнем, что в данном случае важен не отдельный сценарий, а психическое состояние человека, которое узнается и сопереживается, а не детально описывается и точно классифицируется.

ка не нужно множества пресс-конференций, большого количества разнообразной информации, интенсивного общения. Самое главное, как правило, происходит внутри самого проектировщика — в его воображении, уме и т. д. Его созерцательность, терпеливость и настойчивость являются хорошим залогом успешности — но не гарантией![7]

В этой функции UML может использоваться, а может и не использоваться. Как отмечал известный исследователь инженерного проектирования Дж. Джонс, решения часто рождаются «на обратной стороне конверта» [7]. При этом навязывание каких-либо специальных форм работы ведет к преждевременной формализации незрелых идей, сильно отвлекает проектировщика и не дает возможности сформироваться той атмосфере, в которой может возникнуть оригинальное решение. Однако отметим, что далеко не во всех проектах по разработке ПО проектируются новые и оригинальные системы. Очень часто дело ограничивается перекомпоновкой уже созданных в данной области или в данном коллективе решений[8]. Все, что необходимо сделать в такой ситуации, — это найти нужные шаблоны и адаптировать их к незначительным особенностям данной задачи.

3. Компоновка и формализация

Итак, компоновка и формализация могут быть непосредственным началом проектирования, а могут следовать после творческого проектирования новой системы, после того как искомое решение найдено. Остановимся на последнем случае.

Сложившуюся у проектировщика целостную картину, найденное решение полезно оформить, перевести из разряда чувств, из области воображения, в зримые и осязаемые формы. Такое «оформленное» решение можно обсудить с различными людьми (а не толь-

[7]Мы предвидим, что, читая эти строки, менеджеры могут с досадой отложить статью. Их стандартной реакцией на отсутствие гарантией является раздражение, стремление во что бы то ни стало получить максимум гарантий. Однако опытные менеджеры знают, что успех — это еще и удача, и случай. Поэтому, если мы участвуем в проекте по созданию новой уникальной системы, то нужно быть готовым к неожиданностям и непредвиденным поворотам.

[8]Важно не путать творчество (то есть создание принципиально нового) с компоновкой деталей и фрагментов уже существующего. Компоновка может требовать очень высокой квалификации, и все-таки она не является творчеством.

ко с «братьями по оружию» и друзьями, понимающими проектировщика с полуслова). Оно может также служить хорошей основой для дальнейшего воплощения системы. Более того, сам процесс оформления часто вскрывает различные неразрешенности и проблемы, которых автор еще не заметил — так сказать, «вторичные противоречия» найденного решения [7]. Занимаясь компоновкой и формализацией, проектировщик меняется. Он может, например, много чертить и писать, активно работать со справочной литературой, часто и много общаться с коллегами. Проектировщик напоминает рыбака, который долго ждал, когда «клюнет», и вот теперь вытаскивает пойманную рыбу на берег. Со стороны может показаться, что он наконец занят делом. В этой функции UML очень полезен, так как позволяет наглядно выразить, в силу своей визуальной природы, сложные и непростые для понимания детали решения, сделать их очевидными и понятными для тех, кто будет воплощать это решение. Задача UML-чертежей — «схватить», продемонстрировать, обозначить, пояснить. При создании таких чертежей часто «всплывают» недоделки, «вторичные противоречия». Однако нужно дозировать применение UML, активно используя кроме него как обычные документы, так и устные объяснения. К сожалению, в области разработки ПО, в отличие от других инженерных дисциплин (строительство, машиностроение, электротехника и т. д.), не сложилось четких и однозначных рекомендаций по использованию чертежей при анализе и проектировании. Таким образом, здесь вместо формальных критериев приходится использовать интуитивные ориентиры.

4. Изучение существующей системы

Человек оказался рядом со сложной системой. Он — новичок, дилетант относительно данной системы, а ему нужно стать профессионалом, то есть досконально во всем разобраться. Цели у этого ученичества могут быть очень разные — необходимость влиться в коллектив создателей системы, освоить ее на уровне пользователя, разобраться в ней для того, чтобы создать техническую документацию, и т. д. Исходная установка ученика заключается в том, что он ничего не знает. Это — определенная внутренняя позиция, особенное состояние его психики. Это состояние можно описать как максимальную пустоту, отсутствие заранее сформулированных ожиданий, концепций, мнений, предпочтений. Если перед началом

обучения мы полны всем этим, то нам трудно воспринимать новую информацию.

Ученик тих и внимателен, спокоен перед лицом большого и неизвестного массива информации, он настойчив в изучении, но пассивен в интерпретациях. Он «вбирает» в себя знания. Он старается максимально их упорядочить, но делает это ненасильственно и лишь до тех пор, пока это легко получается. А потом он снова обращается к источнику информации. Ученик не боится незнания в любых формах, он готов носить в себе частичные знания, с лакунами и пробелами, не стремясь любой ценой их заполнить, подменяя своими домыслами существующее, но не понятное ему пока еще положение вещей. Все происходит естественно, в свое время. Однако ученик наполнен желанием постичь предмет. Процесс обучения протекает напряженно, страстно, он подобен бурной горой реке, упрямому водному потоку, пробивающемуся через многометровую толщу скалы к поверхности земли.

Постепенно ученику начинает проясняться общая картина, перед его взором «проступает» логика системы. Его знания о системе становятся все более целостными, он все увереннее ориентируется в *данной информации*. Интенсивность обучения достигается за счет активности ученика, а не его учителей. Эксперты и авторы системы, как правило, пассивны и заняты своей собственной работой. Они могут только отвечать на запросы ученика, расширяют, дополняют то, что он сам нашел, ставят следующие задачи, когда он освоил предыдущие. Ошибочно ожидать от учителей чрезмерной активности, чрезмерной заинтересованности в обучении. Ученик — самый активный человек в этом процессе. UML в данной ситуации может использоваться учеником для фиксации полученной и осмысленной информации. Такие чертежи могут быть осью работы с экспертами, поскольку, глядя на чертежи, эксперт легко видит ошибки и упущения ученика, а также его прогресс в обучении. Здесь важно, чтобы чертежей было не слишком много, чтобы они уточнялись и детализировались от встречи к встрече[9]. Целесообразно использовать структурные диаграммы (классов, компонент, размещений) как основной способ фиксации полученных знаний, описывая с их помощью структуру системы. Поведенческие диаграммы (последовательностей, кооперaций, состояний и переходов

[9]Таким образом, они используются в рамках данной функции в стиле «long term» [10].

и пр.) можно использовать для моделирования отдельных фрагментов поведения системы, таким образом выявляя недостающую информацию о ее структуре[10].

5. Передача знаний о системе

Человек является экспертом и профессионалом. Он — автор системы, владеет огромным арсеналом знаний о ней, видит большое количество незримых другим связей в системе, является неисчерпаемым источником идей по ее дальнейшему развитию. Он способен конструктивно обсуждать систему в любое время дня и ночи — данная информация всегда при нем, как меч рыцаря, и этот меч легко выходит из ножен. Для профессионала мир его системы является бесконечной вселенной. И вот перед ним появляется задача передать часть своих знаний другому человеку или некоторой аудитории (группе людей). Опишем особенности соответствующего психического состояния профессионала.

Перед тем как начать передачу знаний, профессионал тщательно соотносится с аудиторией, стараясь максимально точно уловить ее запрос, уровень подготовленности, желание и возможность работать и т. д. Для этого он может использовать свое воображение, заранее моделируя процесс обучения. При этом могут появляться чувства легкости, вдохновения, прилив идей, желание поделиться знаниями возрастает. Но вместо этого профессионал может почувствовать пустоту, тяжесть, настороженность или что-нибудь еще. В любом случае, данные переживания профессионала очень значимы и показывают ему интегральное состояние аудитории, своевременность и адекватность процесса обучения. Исходя из этого профессионал может строить свою дальнейшую работу.

Знания и опыт профессионала — это источник, из которого он черпает, сам процесс обучения — это поток, берущий начало из этого источника, а берега потока, направляющие его движение и регулирующие интенсивность, — это аудитория, ее желание и возможность работать, чтобы научиться, ее интерес и мотивации интереса к данной информации. Например, ученики усталы, едва держат-

[10]Нужно отметить, что если систему изучает программист для того, чтобы принять участие в дальнейшей разработке, то использование диаграмм UML достаточно ограничено. Главным источником информации о системе для такого человека является программный код. Если же систему изучают непрограммисты, то, как правило, они не исследуют ее кода. В этом случае роль диаграмм UML существенно повышается.

ся на ногах, их глаза слипаются. Или они бодры, жизнерадостны, но поверхностны и данной информацией интересуются лишь слегка. Или им все интересно, они полны сил, энергии и вдохновения, алертны и готовы работать. Возможны и другие варианты. В каждом из этих случаев профессионалу целесообразно действовать поразному, не навязывая своей информации и отвечая точно на запрос аудитории. Профессионал может предпринять также ряд действий по изменению состояния аудитории в необходимую для процесса обучения сторону.

Для успеха обучения профессионалу важно уметь видеть свою информацию глазами учеников. Такое переключение внимания часто бывает трудноосуществимо: ему сложно «вынырнуть» из своей информации, в которой он жил долгое время, и увидеть все, «как в первый раз», так, как видят его ученики. Если он этого не может сделать, то учебный процесс претерпевает большие сложности — профессионал рассказывает о чем-то своем, ученики недоумевают. Они существуют в разных мирах и никак не могут встретиться. Невозможно описать сложную систему «как-она-есть» — любое ее описание будет лишь взглядом на систему с определенной точки зрения[11]. Важно также, чтобы диаграммы были красивыми и гармоничными, вызывали положительные чувства у аудитории — ясность, сбалансированность, гармоничность, стройность, цельность и т. д. Когда люди видят картину, чертеж в первый раз, то они не знают еще, что там изображено, но впечатление возникает сразу — например, легкости, света, воздушности, или тяжести, перегруженности деталями, запутанности. Эти первые ментальные впечатления формируют атмосферу учебного процесса, тот контекст, в котором происходит дальнейшее восприятие материала. Поэтому UML-диаграммы, используемые в рамках передачи знаний о системе, должны создаваться с особой тщательностью, и при разработке насыщаться положительными чувствами. При этом если система, которой посвящен данный учебный процесс, не является совершенной, красивой, интересной, то трудно нарисовать соответствующие диаграммы и построить гармоничный учебный процесс. Красивые чертежи, как правило, являются свидетельством совершенства самой системы, об интересном предмете можно и нужно рассказывать интересно и захватывающе.

[11] Например, цель и точка обзора в при моделировании с помощью подхода SADT (Structure Analysis and Design Technique) [12].

Заключение

Мы рассматривали психическое состояние одного человека, а не коллектива разработчиков. Если человек попадает в некоторое состояние достаточно полно, то он оказывается его источником и ретранслятором, втягивая своих товарищей по проекту туда же просто по факту. Его товарищи начинают реагировать на это состояние, сопереживать ему, «заражаться» им. Группа оказывается моносистемой относительно такого состояния, тем более что рабочие группы программистов, как правило, подбираются из близких людей, единомышленников. Данное состояние оказывается фоном группы, фоном процесса разработки (или одной из ее фаз). Именно на этом фоне разворачивается деятельность групп[12], в частности используются те или иные инструменты.

Авторы выражают признательность своим товарищам, друзьям и коллегам, прочитавшим ранние версии статьи, сделавшим ряд ценных замечаний и вдохновивших авторов на продолжение работы: Петру Мамкину, Дмитрию Бугайченко, Александру Иванову, Игорю Соболеву.

Список литературы

[1] *Brooks F.* No Silver Bullet // Information Proceeding of the IFIP 10th World Computing Conference. — 1986.— P. 1069–1076. (Русский перевод: *Брукс Ф.* Мифический человеко-месяц или как создаются программные системы. — СПб.: Символ. — 2000.)

[2] *Sommerville I.* Software Engineering. — Addison-Wesley, 6th edition. — 2001. — 693 с. (Русский перевод: *Соммервилл И.* Инженерия программного обеспечения. — М.: Вильямс. — 2002. — 623 с.)

[3] *Humphrey W.* Managing the Software Process. — Addison-Wesley. — 1990.

[4] *Koznov D., Kartachev M., Zvereva V., Gagarsky R., Barsov A.* Roundtrip Engineering of Reactive Systems // Proceedings of the 1st International Symposium on Leveraging Applications of Formal Methods (ISoLA). — 2004. — P. 343–346.

[5] *Jaaksi A.* Developing Mobile Browsers in a Product Line // IEEE Software, July/August. — 2002. — P. 73–80.

[6] *Йордон Э.* Путь камикадзе. Как разработчику программного обеспечения выжить в безнадежном проекте. 2-е издание. — М.: Лори. — 2004. — 304 с.

[12] Пользуясь терминологией О. Бахтиярова [14], можно сказать, что из фона возникают фигуры.

[7] *Джонс Дж. К.* Инженерное и художественное конструирование. — М.: Мир. — 1976.

[8] *Юнг К. Г.* Психологические типы. — Минск: Хапвест. — 2003.

[9] *Бахтияров О.* Постинформационные технологии: введение в психонетику. — Киев: ЭКСПИР. — 1997.

[10] *Koznov D. V.* Visual Modeling and Software Project Management // Proceedings of 2nd International Workshop «New Models of Business: Managerial Aspects and Enabling Technology». — 2002. — P. 161–169.

[11] *Booch G.* Object-Oriented Analysis And Design With Application, Second edition. — The Benjamin/Cummings Publishing Company, Inc. — 1994. — 589 p. (Русский перевод: *Буч Г.* Объектно-ориентированный анализ и проектирование с примерами приложений на C++. — М.: Бином. — 2001. — 560 с.)

[12] *Marca D. A., McGowan C. L.* SADT Structured Analysis and Design Technique. — McGraw-Hill. — 1988.

[13] *Кознов Д. В., Ольхович Л. Б.* Визуальные языки проектов // «Системное программирование». — СПб.: 2004.— С. 148–168.

[14] *Бахтияров О.* Деконцентрация. — Киев: Ника-Центр. — 2002.— 126 с.