Константин Берлинский

12+

Набор серебряных пуль

Справочник удачных проектных решений при разработке ПО

Константин Берлинский

Набор серебряных пуль

«ЛитРес: Самиздат»

2004

Берлинский К. К.

Набор серебряных пуль / К. К. Берлинский — «ЛитРес: Самиздат», 2004

ISBN 978-5-532-05602-2

Войны ИТ-методологов не затихают. Каждые несколько лет нам преподносится совершенно новая, быстрая, легкая, простая, эффективная методика (или новая версия «старой»). И уж она наконец-то решит главную проблему – построение качественного ПО в срок... Я думаю, что правда о методологиях заключается в том, что их не существует… Есть лишь УПР – удачные проектные решения, – которые могут сработать (или нет) в конкретной ситуации и проекте. Цель этого справочника – собрать их вместе, дать им краткое описание и подвигнуть ИТ–сообщество к дальнейшему их поиску и классификации…

ISBN 978-5-532-05602-2

Содержание

Введение

«Ну вот!» – скажете Вы, прочтя заголовок данной книги. «Ещё один новоявленный пророк – самозванец учит всех жизни, как нужно выполнять программные проекты! У нас и так есть методология, которая отлично справляется со всеми проблемами. Мы адаптировали её под свои нужды, и вроде бы проблем стало меньше…»

И Вы будете правы, … но наполовину. Я ни в сколькой мере не считаю себя новоявленной мессией, который «наконец-то расскажет, как добиться успеха». Но меня действительно интересуют методы эффективной разработки ПО (и как следствие этого знания – повышение своего профессионального мастерства разработчика ИС).

Эта книга не является обоснованием в письменном виде против какой-либо определённой методологии. Хотя ранее, на форумах тематических сайтов, я позволял себе резкие высказывания по поводу различных новомодных методик разработки, которых с религиозным пылом фанатиков отстаивали их ярые приверженцы.

Вы не найдёте здесь и доводов в пользу какой-нибудь определённой методологии, как можно было бы предположить исходя из того, что я долгое время являлся ревностным сторонником методологии RUP, активно занимался её изучением и внедрением в деятельность компании, в которой работал. Причем, в тот момент времени, во мне действительно была уверенность в том, что жесткое разделение участников разработки по ролям (и соответствующим функциональным обязанностям) сможет сделать процесс разработки управляемым и успешным, а его участников – более удовлетворёнными своей работой.

В связи с этим, своей книгой я рискую навлечь на себя праведный гнев сторонников и защитников всех ныне существующих (и которые появятся годами позже) методик разработки ПО. Однако каждый должен иметь возможность высказать своё мнение, и я воспользуюсь этим правом.

Я уверен, что правда о различных методологиях заключается в том, что их не существует…

А теперь, приведите в чувство всех упавших в обморок, и признайтесь самому себе – что представляет собой сверхновая методология разработки ПО, о которой Вы узнали из последнего маркетингового заявления неважно какой корпорации? Или та методика, которую Вы уже используете в своей повседневной работе, и в которую вложено огромное количество ресурсов (учебные материалы, курсы для ведущих специалистов с выездом в другой город/страну, и, наконец, самое ценное – время)?

Вы думаете, что методика служит организующим фактором разработки, что она четко и ясно говорит, как нужно работать, чтобы добиться успеха в ИТ-области. И, наконец, Вы надеетесь получить конкурентное преимущество, «пуская пыль в глаза» потенциальным заказчикам малопонятными для них фразами типа «мы находимся на 6-ом уровне CMM», «реинженеринг бизнес-процессов», «автоматизация хаоса путем выделения ролей в альтернативных деятельностях» и т.п.

Однако, внедрение одной и той же методики в разных организациях и проектах (а зачастую и в фазах разработки одного проекта!) даёт почему-то совершенно различные результаты. То, что в одних случаях работало хорошо и является стимулирующим фактором, в других – наоборот, тормозит разработку.

В чем же секрет, спросите Вы? Я думаю, что успех проекта зависит от двух факторов:

1) доступные ресурсы (в первую очередь, это качество разработчиков, а второе – это время);

2) способ их взаимодействия.

Если есть доступные ресурсы, и они взаимодействуют максимально эффективным способом, то я считаю, что проект имеет значительно больше шансов родить что-то действительно стоящее.

Что касается методологий – то мне кажется, что все они описывают конечный набор различных способов эффективного использования ограниченных ресурсов. Моя точка зрения состоит в том, что число этих способов (удачных проектных решений – УПР) бесконечно и не нужно ограничивать себя только подмножеством их, в рамках методологии X. Если мы хотим продвинуться в плане успешной разработки программ, то нужно собирать эти решения (аналогично паттернам проектирования) и учиться применять их в нужный момент. Эта книга является попыткой собрать известные мне методы успешной разработки в одном месте.

Приятного чтения и да прибудет с Вами великая сила!

Зачем эта книга была написана?

Эта книга была написана для того, чтобы собрать в единую коллекцию то, что я называю «золотые крупицы знания», распылённые по многочисленным источникам, таким как Интернет, литература и просто народное творчество. Фраза «одна голова хорошо, а две лучше» и принцип «разделяй и властвуй», известный еще со времен Римской империи, сделали для развития программной инженерии больше, чем Microsoft и IBM вместе взятые.

После прочтения любой книги, статьи или сообщения на форуме, меня всегда интересовало – что конкретно этот источник может дать мне полезного? Содержится ли в нём какая-нибудь новая мысль, неизвестная мне ранее? К счастью сказать, в большинстве случаев новые идеи присутствовали, но, к сожалению, были основательно «разведены» посторонней информацией, только замутняющей суть дела.

Поэтому, я старался, по мере возможности, после прочтения очередного труда, составлять его «конспект» с перечнем того, какие основные идеи (неизвестные или мало-очевидные в тот момент для меня) пытался высказать автор.

Например, вот такой результат «препарирования» у меня получился от книги [3]:

1. описание бизнес-процесса в виде текста по объему намного меньше, чем в виде графики (это действительно так – самой большой проблемой при работе с диаграммами было то, что рабочий принтер не поддерживал печать на формате A3 и A2…)

2. заказчики быстрее прочитают текст, поймут и подпишут его (согласятся с ним или выскажут свои претензии), чем выучат UML (у меня, например, много времени уходило на объяснение стрелки include/extended для связей между вариантами использования)

3. нового сотрудника можно быстрее научить писать текст в формате прецедентов Коберна, чем заставить *правильно* использовать UML и продукт, его поддерживающий (например, Rational Rose – графический дизайнер которой оставляет желать лучшего)

4. хорошая классификация целей – всегда нужно работать на одном уровне целей. Уровень повышается, если задать вопрос «Почему?» и понижается, если задать вопрос «Как?» (это большое искусство – быть на нужном уровне цели – диаграммы получаются мелкие и общие, или слишком большие и детализированные)

5. отличный, интуитивно понятный шаблон описания прецедента (основной сценарий из 10 шагов без «если» + расширения основного сценария + дополнительная информация)

6. хорошая идея об использовании средств многомерного анализа собранных требований (например, с помощью электронных таблиц) – применение сортировки, группировки по различным атрибутам (важность, срок, модуль, роль)

7. и, наконец, на мой взгляд, самое важное – «чем ниже уровень описания целей, тем менее полезными и наглядными становятся диаграммы». Из этого следует идея о «разделении труда» между различными CASE-средствами: для составления диаграмм высокого уровня (сценарий бизнес-процесса, схемы движения информации, диаграмма состояний основного документа системы, взаимодействие программных модулей) использовать инструменты, эффективно применяемые для рисования диаграмм, их связи друг с другом (Rational Rose, MS Visio), а для более детального описания применять «прецеденты по Коберну».

Должен признаться, что написание этой книги было несколько авантюрным занятием – всегда можно будет получить «укол в спину» от какого-нибудь «мастодонта» с 20-летним стажем разработки и отдавшего большую часть своей жизни продвижению методологии X, в виде фразы «сделай столько проектов, сколько я, а потом будешь предлагать решения под своим соусом…».

Мне же, с 24-мя годами общего возраста (из которых всего 3 – опыт коммерческой разработки ПО) и 5-ю реализованными проектами, «по общепринятым правилам» нужно подождать лет до 40-ка, а потом уже излагать свои мысли в письменной форме.

Но к черту все эти правила! Нужно жить сегодняшним днём, и если тебе представилась возможность «прыгнуть выше головы», то нужно делать это *здесь и сейчас*. Вероятность поднять здоровенный железный шкаф ничтожна мала (см. фильм «Пролетая над гнездом кукушки»). Но это вообще никогда не удастся, если не сделать хотя бы одну попытку.

Мне *действительно* интересно то ремесло, которой я обладаю, и каждый день я стараюсь сделать свою жизнь (и профессиональную деятельность) как можно лучше и счастливее.

Конечно, на самом деле это несерьёзно – думать, что написание книги, статьи или прочего «пиара самого себя» увеличит мою или чью-то ещё образованность (подробнее, см. статью [2]). Однако всё же это имеет смысл.

Написание книги увеличивает мою «виртуальную образованность» (т.е. то, насколько хорошо меня *воспринимают* потенциальные и текущие работодатели). Всё это (встреча с новыми людьми, познавание новой информации) увеличивает мои шансы повысить свою «реальную образованность» (т.е. та *реальная польза*, которую я приношу проектам, в которых участвую). Повышение «реальной образованности» толкает меня на публикацию нового материала. И так далее.

Этой книгой мне хотелось доказать (в первую очередь самому себе), что мир проще, чем кажется на первый взгляд и добиться в нём успеха действительно реально, независимо от тех препятствий, которые постоянно возникают на нашем пути.

Есть такая байка про миллионера, который сказал: «Я могу рассказать вам, каким образом я заработал каждый свой миллион. Кроме первого». Действительно, если воспринимать то, что называется «жизненным успехом» в качестве восхождения наверх по бесконечной лестнице, то наиболее трудно сделать именно первый шаг. Я постарался сделать для себя эту книгу «экзаменом» на право подняться на первую ступень.

Ну и, наконец, это на самом деле огромный кайф – писать книгу. Аналогично сочинению музыки, рисованию картины или вырезанию скульптуры. Красота на самом деле «спасёт мир, меня и программные проекты». Когда ты видишь, как обрывки мыслей и фраз, после терпеливой обработки (мучительного поиска синонимов и выделения деепричастных оборотов) превращаются в связный текст с логичным сюжетом, испытываешь ни с чем не сравнимое удовольствие.

Программирование тоже дает такое ощущение. Это связано с тем моментом, когда разрозненная информация, требования, приказы, инструкции, слухи и досужие домыслы, относящиеся к системе, как огромные неповоротливые булыжники вдруг складываются у тебя в голове в единую скульптуру. Каждый пазл становится на своё место и остаётся только брызнуть на них живой водой. Каменная статуя просыпается и рукотворное существо начинает делать свои первые шаги…. Это одна из самых больших радостей нашей профессии.

Что было задумано

Изначально, у меня возник следующий замысел относительно книги (планировалась только «бумажная» версия). В начале, сделать обзор современных методологий разработки ПО. Затем высказать основную идею о том, что все методологии используют подмножество из общего пространства УПР (см. рис. 1).

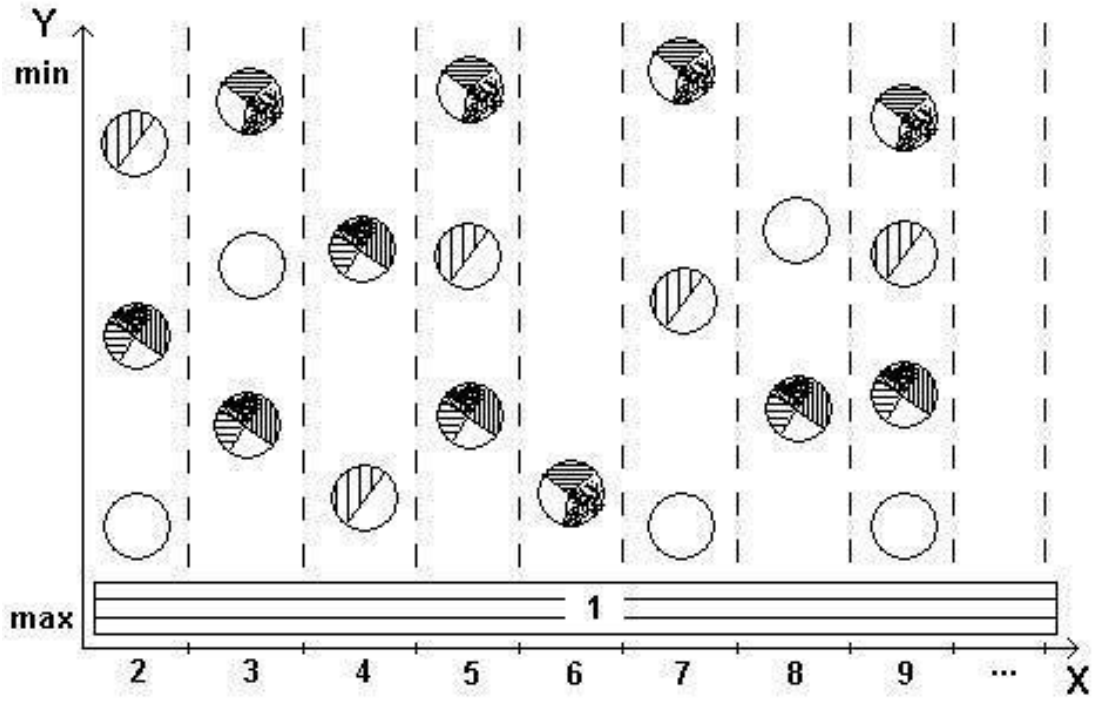

Рис. 1 – Единое пространство УПР.

X – ось этапов ЖЦ ПО;
Y – степень важности УПР для конкретного этапа;
1..9 – ЖЦ (1-фундамент проекта – «Управление»);
Круги – УПР, известные на данный момент;
Виды штриховок – различные методологии разработки ПО, включившие решение в перечень своих постулатов.

В середине – дать пятьдесят чистых листов. Каждый лист представлял бы собой форму для ввода удачного решения, найденного читателем (скорее писателем) в процессе разработки очередного проекта. Форма имела бы стандартную форму: название, код этапа ЖЦ, оценка эффективности, описание и дополнительные источники информации по решению.

Заканчивалась бы книга главами «Содержание» и «Библиография». Естественно, они бы тоже заполнялись вручную. У каждого есть свой «золотой набор», состоящий из книг, WEB-ресурсов и телефонов ближайших пиццерий.

Идея была в том, чтобы каждый написал свою собственную книгу и даже вписал свои инициалы на обложку.

Но затем я просмотрел свою домашнюю библиотеку, и обнаружил, что во многих из них идея о «дополнении» книги читателем уже реализована. Это так называемые листы «для заметок». Заметки у меня отсутствовали…

Поэтому, я решил подойти к творческому процессу более профессионально, и таким образом, книга обрела настоящий вид.

Также, в мои планы входит (в случае положительной оценки книги квалифицированными разработчиками и наличия у меня свободного времени) дальнейшее развитие теории «единого пространства УПР» в виде сайта в сети Интернет.

Сайт представлял бы собой информационный портал для обмена информацией между разработчиками ПО и пополнения БД УПР. Основные компоненты сайта – форум разработчиков, разбитый по темам (этапам ЖЦ, методологиям, продуктам и прочее), гостевая книга, профили пользователей сайта и самое главное – БД УПР, доступную для публичного доступа. Каждое новое УПР (или изменение описания уже существующего) оценивалось бы пользователями сайта, и в случае положительного результата, добавлялось бы в БД.

Однако получилось то, что получилось. Что касается продолжения книги, то я думаю, что оно будет написано в тот момент, когда мои профессиональные познания выйдут на качественно новый уровень. Может быть, на это понадобиться лет двадцать (как у Брукса), а может раньше.

Кстати, по поводу Брукса. Вы заметили различия между посвящениями 1975 и 1995 годов? В первом упоминаются непосредственный и самый главный начальник. Во втором же – «Нэнси, Божьему дару для меня». Наконец-то в списке жизненных ценностей появляется семья и всё, что с этим связано. Не в этом ли заключается «сущность и акциденция программной инженерии»?

Благодарности

Итак, кто оказал на мое развитие наибольшее влияние и кому я за это благодарен:

– Мои родители – они меня родили ;-) Хотелось бы жить не во время победившего мирового терроризма, но ничего уж тут не поделаешь.

– Моя сестра – за поддержку.

– Технический Университет Молдовы – за получение не самого хорошего высшего технического образования, но лучшего из возможных в республике. Как сказано в книге [15] – «Поставь себе цель. Получи такое образование, какое только можешь, но затем, ради Бога, делай что-нибудь!».

– Стратан Виктор – за отличное объяснение в теории и практике паттернов проектирования, ООП и т.п. Первая версия Rational Rose 98, полученная мной – его заслуга.

– Сердцев Виталий – после наблюдения его шаманства с ассемблером, C++, низкоуровневого программирования, я стал «серьёзно» увлекаться собственной профессией и до сих пор об этом не жалею. Вспоминаю слова нашего завкафедры В. Бешлиу: «Вы пришли в Политех из разных мест и с разными целями, но если вы его закончите, то станете настоящими патриотами своей профессии».

– Смолов Александр – когда баланс между радостями и печалями профессии начинает сдвигаться в сторону последних (см. книгу [6]), я сажусь за компьютер и в течение месяца борюсь с пришельцами в старом добром XCOM (см. сайт [15]), принесенных мне Александром на трёх дискетах со слониками. Грустные мысли от этого проходят, но потом еще долго снятся зеленые человечки ;-)

– Драгонер В.В. – куратор моего диплома. Была поставлена амбициозная цель по созданию аналога Rational Rose. Я написал компилятор исходных текстов C++; Сердцев – графическую оболочку для браузера объектов и диаграмм; Смолов – озвучивание на русском языке результатов анализа текстов через движок MSTextToSpeech. Проект благополучно провалился (к сдаче диплома было готово не более 50% заявленных требований), но неудачи иногда более полезны, чем успехи. Был получен опыт работы с Розой, прочтены материалы о Rational и многое другое.

– Папроцкий И.Б. и Старащук А.И. – работа в ГП «Registru» (см. статью [1]) под их руководством имела огромное значение на мой профессиональный и карьерный уровень.

– Золотухина Е.Б. – за потрясающий курс по системному анализу и разработке ИС.

– Дурлештяну Эдуард (компания BitGenerator) – вместе работать было тяжело, но очень интересно. К большому моему сожалению, тяжесть «перевесила» интерес.

– И, наконец, Топорец Игорь – мой непосредственный начальник на данный момент и настоящий профессионал своего дела. Многие поставленные цели мне удалось достичь быстрее благодаря нему. Очень надеюсь и на обратное (т.е. что я тоже помог достижению его целей).

Методологии разработки ПО

RUP

Rational Unified Process – Рациональный Унифицированный процесс.

RUP был создан в 1996г. корпорацией Rational при участии Гради Буча, Айвара Якобсона и Джима Румбаха.

Это поистине фундаментальная работа по описанию успешных методик создания ПО. Жизненные циклы ПО, потоки работ, роли, деятельности, артефакты, продукты, поддерживающие большую часть этапов ЖЦ. Не зря эту методику называют «тяжеловесной». Поддержка языка UML. RUP в качестве самостоятельной базы знаний по объему и значению может сравниться разве что с MSDN.

Основная идея RUP – максимально четко распределить работу каждого участника процесса разработки. Самая большая проблема, по моему мнению, заключается в том, что трудно (или даже невозможно) найти таких людей, которые будут делать только то, что им сказано. Как скоро им надоест заниматься одним и тем же? Не теряется ли чувство ответственности (и как следствие – удовлетворение от результата) за выполненную работу при жестком фиксировании участков работ? Служит ли наличие согласованных интерфейсов по обмену информацией гарантией качества и эффективности работы?

Продукты, поддерживающие методику – в первую очередь, это продукты самой компании Rational. Основной продукт Rose (используется практически на всех этапах разработки), SoDA (автодокументирование), RequisitePro (управление требованиями), ClearQuest (запросы на изменения), ClearCase (версионность), Administrator (управление репозиторием проекта), WorkBrench (настройка корпоративных процессов), Quantify (тестирование скорости кода), Purify (определение утечек памяти), PureCoverage (тест охвата кода), Robot (автоматизированное тестирование), SiteLoad (нагрузочное тестирование), SiteCheck (проверка «мертвых» Web-ссылок). В настоящее время доступен RUP, интегрированный в среду разработки Microsoft .NET (под названием Rational XDE).

XP

Extreme Programming – Экстремальное программирование.

Рождением (а точнее датой «официальной» регистрации) можно считать 2001г., когда в США, штате Юта 17 сторонников «легковесных» процессов разработки выработали манифест с основными постулатами. Идеологами методики можно считать Кента Бека, Уорда Каннингема и Рона Джеффриса.

Основные принципы: тесная коммуникация, постоянное тестирование, минимум документации и максимум гибкости. Впрочем, лучше привести текст манифеста (см. книгу [4]):

«Мы открываем лучшие способы разработки ПО, создавая его сами и помогая другим. Благодаря этой работе мы стали ценить:

Индивидуумов и взаимодействия выше процессов и инструментов
Работающее ПО выше всеобъемлющей документации
Сотрудничество с заказчиками выше согласований условий договора
Реагирование на изменения выше соблюдения плана

Это означает, что, хотя элементы в правой части также имеют свою ценность, но больше мы ценим элементы, расположенные слева».

Довольно краткие и понятные формулировки, делающие доступными высказанные идеи самым широким массам. XP впервые озвучила некоторые совершенно революционные принципы разработки. «Это вам не понадобится», «Ищите самое простое решение, которое может сработать», «Любые сидящие рядом два разработчика могут поменять всё что угодно в системе», «Заказчик в любой момент может изменить требования» и др.

Однако я уже высказывал свою критику по поводу XP. Большая часть претензий к XP снимается, вследствие более детального знакомства с ней. Можно считать, что последние проекты, в которых я участвовал, были «в духе XP».

Осталась следующая критика:

1. Идея «нахождения представителя заказчика в одной комнате с программистами» по моему мнению, мягко говоря, является фантастическим пожеланием. Никто не спорит, что коммуникация с заказчиком жизненно необходима. Но решение проблемы скорее находится, во-первых, в сфере разработки стандартных форм документов по взаимодействию (ТЗ, ТП). Во-вторых, в более быстрых итерациях (нужно помочь заказчику сформулировать и откорректировать своё видение системы)

2. Отрицание этапа «большого предварительного проектирования» допустимо только при разработке тривиальных систем (несложные сайты с уклоном в сторону дизайна, а не программирования, простейшие однопользовательские программы с минимумом бизнес-логики и т.п.). Цитируя одного из авторов: «Я рассматриваю здесь решения, полезные при разработке *сложных* систем. Если приложение простое, зачем тратить на него своё время?». В сложном и интересном проекте, с «богатой» предметной областью было бы преступной халатностью не сформировать *в самом начале* каркас системы, основные принципы его функционирования. Конечно, «настоящие XP-шники» стоически воспримут информацию в середине проекта о том, что обнаружился прецедент, заставляющий переделать большую часть кода (или ещё хуже выбросить его). Но я считаю это совершенно недопустимым.

3. Десятки userstory (бумажки с несколькими предложениями, характеризующими прецедент пользователя), оставшиеся после завершения проекта, не могут служить в качестве надежной документации. Получается, что XP нацелена на быструю *разработку* ПО, но не на его *сопровождение*. Приведу цитату из книги [4] по поводу неудачи проекта 3C, выполненного

посредством XP (расчет зарплаты для Chrysler – могу подтвердить, что зарплата при своей внешней простоте одна из самых трудных областей бухгалтерии, в которой «утонула» не одна команда программистов). «Когда ушло достаточно много сотрудников, незаписанные сведения о проекте и групповая память были утрачены». Я думаю, что апологеты XP переусердствовали с минимизацией документации. Что для студентов может выглядеть привлекательным, серьезных разработчиков должно насторожить.

Должен поделиться собственным ощущением от «работы в стиле XP». В отличие от RUP (или любой другой методики с *фиксированием* результатов этапов, посредством тех. задания, тех. проекта и т.п.) XP дает ощущение неуверенности и анархии *в начале* проекта. Бизнес-требования и архитектура меняются так быстро, что, просидев пару дней дома можно потом не узнать структуру базовых классов (даже если ты сам её придумал).

Сразу начинаешь понимать положение XP о 40-часовой рабочей недели без переработок. Зачем до 22_00 трудиться в поте лица над модулем, который завтра может вообще не понадобиться?

В середине архитектура стабилизируется, *конец* же проекта характеризуется полной уверенностью в себе. Никакое из изменений требований, которое может высказать заказчик, не кажется ужасным. За время постоянных модификаций архитектуру *приходится* перерабатывать таким образом, чтобы изменения выполнялись максимально просто.

Программисты выступают в роли пользователей созданного дизайна программы – если он имеет дефекты, то систему трудно менять ещё на этапе разработки и это вызывает дискомфорт. В результате, дизайн *вынужденно* улучшается, а система легко переносит любые изменения бизнес-требований.

Продуктов, поддерживающих методику просто нет. И это, наверное, самое главное достоинство. Не станешь же, в самом деле, считать «XP-продуктом» текстовый редактор или средство обеспечения версионности.

SADT

Structured Analysis and Design Technique – Методология структурного анализа и проектирования.

Известна как разработка компании SofTech, либо как только функциональный вариант в правительственной версии (IDEF0). Её начали применять с 1973г. во многих областях, таких как бизнес, производство, оборона, связь и организация проектирования.

Диаграммы в стандарте IDEF0 имеют несомненное преимущество для функционального моделирования системы. Однако представление модулей системы в виде блоков с набором входов и выходов и набором управляющих воздействий на них хорошо раскрывает только *высокоуровневые структурные особенности* системы. В этом SADT успешно конкурирует с диаграммами деятельности языка UML.

В SADT я не нашел механизмов для дальнейшей разработки системы, от сбора требований к реализации, тестированию и внедрению. Возможно, такая задача и не ставилась идеологами. Методика серьезно проигрывает остальным по охвату этапов ЖЦ ПО, сконцентрировавшись только на сборе требований и бизнес-моделировании.

SADT в полной мере реализует идею «большого предварительного проектирования в начале разработки» с целью уменьшить «просачивание» ошибок на более поздние этапы ЖЦ, где дороже их исправление (см. книгу [24]). Из-за жесткой декомпозиции процесса разработки методику можно считать «прародителем» RUP.

MSF & MOF

Microsoft Solutions Framework (Набор решений от MS) и Microsoft Operations Framework (Набор операций от MS).

По замыслу составителей методики, она объединяет лучшие принципы каскадной и спиральной моделей разработки ПО, так сказать «лучшее из двух миров» (см. сайт [23]). Естественно, для достижения успешного результата предлагаются решения и продукты непосредственно от Microsoft.

Однако нельзя не признать огромную заслугу компании Microsoft в сфере компьютерной индустрии. Стоит изучить MSF только из-за того, что «так делает лидер».

Конец ознакомительного фрагмента.