Лучшее - враг хорошего.
Народная мудрость

Лекция 3.
ОБЩИЕ ПРИНЦИПЫ РАЗРАБОТКИ ПРОГРАММНЫХ СРЕДСТВ

Специфика разработки программных средств. Жизненный цикл программного средства. Понятие качества программного средства. Обеспечение надежности - основной мотив разработки программного средства. Методы борьбы со сложностью. Обеспечение точности перевода. Преодоление барьера между пользователем и разработчиком. Обеспечение контроля правильности принимаемых решений.

3.1. Специфика разработки программных средств.

Разработка программных средств имеет ряд специфических особенностей [3.1].

- Прежде всего, следует отметить некоторое противостояние: *неформальный* характер требований к ПС (постановки задачи) и понятия ошибки в нем, но *формализованный* основной объект разработки - программы ПС. Тем самым разработка ПС содержит определенные этапы формализации, а переход от неформального к формальному существенно неформален.
- Разработка ПС носит *творческий характер* (на каждом шаге приходится делать какой-либо выбор, принимать какое-либо решение), а не сводится к выполнению какой-либо последовательности регламентированных действий. Тем самым эта разработка ближе к процессу проектирования каких-либо сложных устройств, но никак не к их массовому производству. Этот творческий характер разработки ПС сохраняется до самого ее конца.
- Следует отметить также особенность продукта разработки. Он представляет собой некоторую совокупность текстов (т.е. *статических* объектов), смысл же (семантика) этих текстов выражается процессами обработки данных и действиями пользователей, запускающих эти процессы (т.е. является *динамическим*). Это предопределяет выбор разработчиком ряда специфичных приемов, методов и средств.

- Продукт разработки имеет и другую специфическую особенность: ПС при своем использовании (эксплуатации) не расходуется и не расходует используемых ресурсов.

3.2. Жизненный цикл программного средства.

Под *жизненным циклом* ПС (*software life cycle*) понимают весь период его разработки и эксплуатации (использования), начиная от момента возникновения замысла ПС и кончая прекращением всех видов его использования [3.1-3.4]. Жизненный цикл охватывает довольно сложный процесс создания и использования ПС (*software process*). Этот процесс может быть организован по-разному для разных классов ПС и в зависимости от особенностей коллектива разработчиков.

В настоящее время можно выделить 5 основных подходов к организации процесса создания и использования ПС [3.5].

- Водопадный подход. При таком подходе разработка ПС состоит из цепочки этапов. На каждом этапе создаются документы, используемые на последующем этапе. В исходном документе фиксируются требования к ПС. В конце этой цепочки создаются программы, включаемые в ПС.
- Исследовательское программирование. Этот подход предполагает быструю (насколько это возможно) реализацию рабочих версий программ ПС, выполняющих лишь в первом приближении требуемые функции. После экспериментального применения реализованных программ производится их модификация с целью сделать их более полезными для пользователей. Этот процесс повторяется до тех пор, пока ПС не будет достаточно приемлемо для пользователей. Такой подход применялся на ранних этапах развития программирования, когда технологии программирования не придавали большого значения (использовалась интуитивная технология). В настоящее время этот подход применяется для разработки таких ПС, для которых пользователи не могут точно сформулировать требования (например, для разработки систем искусственного интеллекта).
- Прототипирование. Этот подход моделирует начальную фазу исследовательского программирования вплоть до создания рабочих версий программ, предназначенных для проведения экспериментов с целью установить требования к ПС. В

дальнейшем должна последовать разработка ПС по установленным требованиям в рамках какого-либо другого подхода (например, водопадного).

- Формальные преобразования. Этот подход включает разработку формальных спецификаций ПС и превращение их в программы путем корректных преобразований. На этом подходе базируется компьютерная технология (CASE-технология) разработки ПС.
- Сборочное программирование. Этот подход предполагает, что ПС конструируется, главным образом, из компонент, которые уже существуют. Должно быть некоторое хранилище (библиотека) таких компонент, каждая из которых может многократно использоваться в разных ПС. Такие компоненты называются *повторно используемыми* (*reusable*). Процесс разработки ПС при данном подходе состоит скорее из сборки программ из компонент, чем из их программирования .

В нашем курсе лекций мы, в основном, будем рассматривать водопадный подход с некоторыми модификациями. Во-первых, потому, что в этом подходе приходиться иметь дело с большинством процессов программной инженерии, а, во-вторых, потому, что в рамках этого подхода создается большинство больших программных систем. Именно этот подход рассматривается в качестве индустриального подхода разработки программного обеспечения. Исследовательское программирование исходит из взгляда на программирование как на искусство. Оно применяется тогда, когда водопадный подход не применим из-за того, что не удается точно сформулировать требования к ПС. В нашем курсе мы этот подход рассматривать не будем. Прототипирование рассматривается как вспомогательный подход, используемый в рамках других подходов, в основном, для прояснения требований к ПС. Компьютерной технологии (включая обсуждение жизненного цикла ПС, созданного по этой технологии) будет посвящена отдельная лекция. Сборочное программирование мы в нашем курсе рассматривать не будем, хотя о повторно используемых программных модулях мы говорить будем, обсуждая свойства программных модулей.

В рамках водопадного подхода различают следующие стадии жизненного цикла ПС (см. рис. 3.1): разработку ПС, производство программных изделий (ПИ) и эксплуатацию ПС.

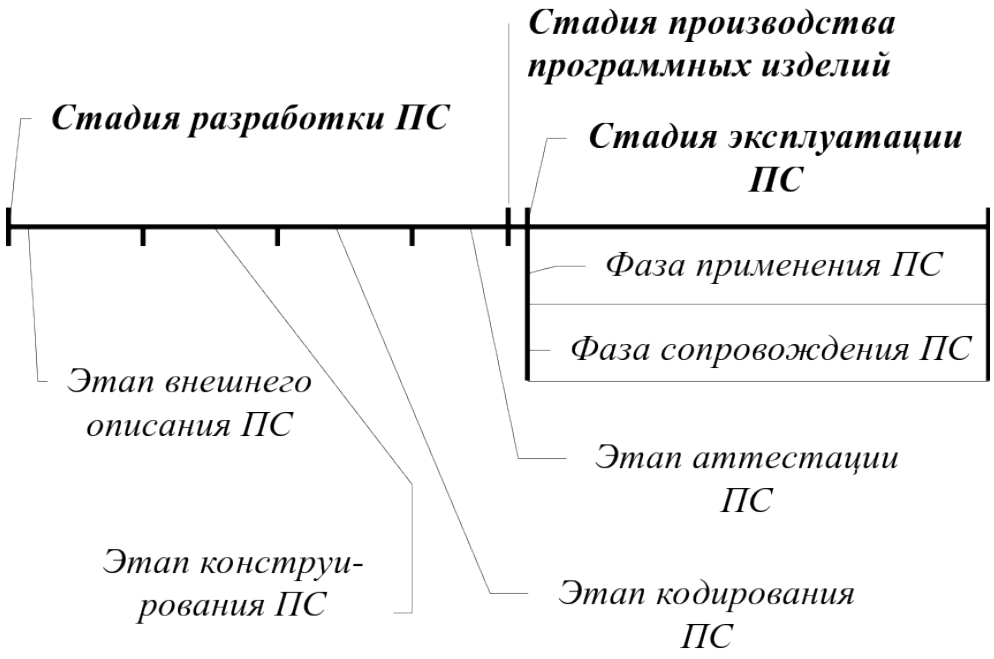

Рис. 3.1. Стадии и фазы жизненного цикла ПС.

Стадия *разработки* (*development*) ПС состоит из этапа его внешнего описания, этапа конструирования ПС, этапа кодирования (программирование в узком смысле) ПС и этапа аттестации ПС. Всем этим этапам сопутствуют процессы документирования и управления (*management*) ПС. Этапы конструирования и кодирования часто перекрываются, иногда довольно сильно. Это означает, что кодирование некоторых частей программного средства может быть начато до завершения этапа конструирования.

Этап *внешнего описания* ПС включает процессы, приводящие к созданию некоторого документа, который мы будем называть *внешним описанием* (*requirements document*) *ПС*. Этот документ является описанием поведения ПС с точки зрения внешнего по отношению к нему наблюдателя с фиксацией требований относительно его качества. Внешнее описание ПС начинается с анализа и определения требований к ПС со стороны пользователей (заказчика), а также включает процессы спецификации этих требований. *Конструирование* (*design*) ПС охватывает процессы:

разработку архитектуры ПС, разработку структур программ ПС и их детальную спецификацию.

Кодирование (*coding*) ПС включает процессы создания текстов программ на языках программирование, их отладку с тестированием ПС.

На этапе *аттестации* (*acceptance*) ПС производится оценка качества ПС. Если эта оценка оказывается приемлемой для практического использования ПС, то разработка ПС считается законченной. Это обычно оформляется в виде некоторого документа, фиксирующего решение комиссии, проводящей аттестацию ПС.

Программное изделие (ПИ) - экземпляр или копия разработанного ПС. *Изготовление* ПИ - это процесс генерации и/или воспроизведения (снятия копии) программ и программных документов ПС с целью их поставки пользователю для применения по назначению. *Производство* ПИ - это совокупность работ по обеспечению изготовления требуемого количества ПИ в установленные сроки [3.1]. Стадия производства ПИ в жизненном цикле ПС является, по существу, вырожденной (не существенной), так как представляет рутинную работу, которая может быть выполнена автоматически и без ошибок. Этим она принципиально отличается от стадии производства различной техники. В связи с этим в литературе эту стадию, как правило, не включают в жизненный цикл ПС.

Стадия *эксплуатации* ПС охватывает процессы хранения, внедрения и сопровождения ПС, а также транспортировки и применения ПИ по своему назначению. Она состоит из двух параллельно проходящих фаз: фазы применения ПС и фазы сопровождения ПС [3.4, 3.5].

Применение (*operation*) ПС - это использование ПС для решения практических задач на компьютере путем выполнения ее программ.

Сопровождение (*maintenance*) ПС - это процесс сбора информации о качестве ПС в эксплуатации, устранения обнаруженных в нем ошибок, его доработки и модификации, а также извещения пользователей о внесенных в него изменениях [3.1, 3.4, 3.5].

3.3. Понятие качества программного средства.

Каждое ПС должно выполнять определенные функции, т.е. делать то, что задумано. Хорошее ПС должно обладать еще целым рядом свойств, позволяющим успешно его использовать в течении длительного периода, т.е. обладать определенным качеством. *Качество (quality*) ПС - это совокупность его черт и характеристик, которые влияют на его способность удовлетворять заданные потребности пользователей [3.6]. Это не означает, что разные ПС должны обладать одной и той же совокупностью таких свойств в их наивысшей степени. Этому препятствует тот факт, что повышение качества ПС по одному из таких свойств часто может быть достигнуто лишь ценой изменения стоимости, сроков завершения разработки и снижения качества этого ПС по другим его свойствам. Качество ПС является удовлетворительным, когда оно обладает указанными свойствами в такой степени, чтобы гарантировать успешное его использование.

Совокупность свойств ПС, которая образует удовлетворительное для пользователя качество ПС, зависит от условий и характера эксплуатации этого ПС, т.е. от позиции, с которой должно рассматриваться качество этого ПС. Поэтому при описании качества ПС, прежде всего, должны быть фиксированы *критерии* отбора требуемых свойств ПС. В настоящее время *критериями качества ПС* (c*riteria of software quality*) принято считать [3.6-3.10]:

- функциональность,
- надежность,
- легкость применения,
- эффективность,
- сопровождаемость,
- мобильность.

Функциональность - это способность ПС выполнять набор функций, удовлетворяющих заданным или подразумеваемым потребностям пользователей. Набор указанных функций определяется во внешнем описании ПС.

Надежность подробно обсуждалась в первой лекции.

Легкость применения - это характеристики ПС, которые позволяют минимизировать усилия пользователя по подготовке исходных данных, применению ПС и оценке полученных результатов, а также вызывать положительные эмоции определенного или подразумеваемого пользователя.

Эффективность - это отношение уровня услуг, предоставляемых ПС пользователю при заданных условиях, к объему используемых ресурсов.

Сопровождаемость - это характеристики ПС, которые позволяют минимизировать усилия по внесению изменений для устранения в нем ошибок и по его модификации в соответствии с изменяющимися потребностями пользователей.

Мобильность - это способность ПС быть перенесенным из одной среды (окружения) в другую, в частности, с одного компьютера на другой.

Функциональность и надежность являются обязательными критериями качества ПС, причем обеспечение надежности будет красной нитью проходить по всем этапам и процессам разработки ПС. Остальные критерии используются в зависимости от потребностей пользователей в соответствии с требованиями к ПС. Обеспечение этих критериев будет обсуждаться в подходящих разделах курса.

3.4. Обеспечение надежности - основной мотив разработки программных средств.

Рассмотрим теперь общие принципы обеспечения надежности ПС, что, как мы уже подчеркивали, является основным мотивом разработки ПС, задающим специфическую окраску всем технологическим процессам разработки ПС. В технике известны четыре подхода обеспечению надежности [3.11]:

- предупреждение ошибок;
- самообнаружение ошибок;
- самоисправление ошибок;
- обеспечение устойчивости к ошибкам.

Целью подхода предупреждения ошибок - не допустить ошибок в готовых продуктах, в нашем случае - в ПС. Проведенное рассмотрение природы ошибок при разработке ПС позволяет для достижения этой цели сконцентрировать внимание на следующих вопросах:

- борьба со сложностью,
- обеспечение точности перевода,
- преодоление барьера между пользователем и разработчиком,
- обеспечение контроля принимаемых решений.

Этот подход связан с организацией процессов разработки ПС, т.е. с технологией программирования. И хотя, как мы уже отмечали, гарантировать отсутствие ошибок в ПС невозможно, но в рамках этого подхода можно достигнуть приемлемого уровня надежности ПС.

Остальные три подхода связаны с организацией самих продуктов технологии, в нашем случае - программ. Они учитывают возможность ошибки в программах. Самообнаружение ошибки в программе означает, что программа содержит средства обнаружения отказа в процессе ее выполнения. Самоисправление ошибки в программе означает не только обнаружение отказа в процессе ее выполнения, но и исправление последствий этого отказа, для чего в программе должны иметься соответствующие средства. Обеспечение устойчивости программы к ошибкам означает, что в программе содержатся средства, позволяющие локализовать область влияния отказа программы, либо уменьшить его неприятные последствия, а иногда предотвратить катастрофические последствия отказа. Однако, эти подходы используются весьма редко (может быть, относительно чаще используется обеспечение устойчивости к ошибкам). Связано это, во-первых, с тем, что многие простые методы, используемые в технике в рамках этих подходов, неприменимы в программировании, например, дублирование отдельных блоков и устройств (выполнение двух копий одной и той же программы всегда будет приводить к одинаковому эффекту - правильному или неправильному). А, во-вторых, добавление в программу дополнительных фрагментов приводит к ее усложнению (иногда - значительному), что в какой-то мере мешает методам предупреждения ошибок.

3.5. Методы борьбы со сложностью.

Мы уже обсуждали в лекции 2 сущность вопроса борьбы со сложностью при разработке ПС. Известны два общих метода борьбы со сложностью систем:

- обеспечения независимости компонент системы;
- использование в системах иерархических структур.

Обеспечение независимости компонент означает разбиение системы на такие части, между которыми должны остаться по возможности меньше связей. Одним из воплощений этого метода

является модульное программирование. Использование в системах иерархических структур позволяет локализовать связи между компонентами, допуская их лишь между компонентами, принадлежащими смежным уровням иерархии. Этот метод, по существу, означает разбиение большой системы на подсистемы, образующих малую систему. Здесь существенно используется способность человека к абстрагированию.

3.6. Обеспечение точности перевода.

Обеспечение точности перевода направлено на достижение однозначности интерпретации документов различными разработчиками, а также пользователями ПС. Это требует придерживаться при переводе определенной дисциплины. Майерс предлагает использовать общую дисциплину решения задач, рассматривая перевод как решение задачи [3.11]. Лучшим руководством по решению задач он считает книгу Пойа "Как решать задачу" [3.12]. В соответствии с этим весь процесс перевода можно разбить на следующие этапы:

- Поймите задачу;
- Составьте план (включая цели и методы решения);
- Выполните план (проверяя правильность каждого шага);
- Проанализируйте полученное решение.

Подробно обсуждать этот вопрос мы здесь не будем.

3.7. Преодоление барьера между пользователем и разработчиком.

Как обеспечить, чтобы ПС выполняла то, что пользователю разумно ожидать от нее? Для этого разработчикам необходимо правильно понять, во-первых, чего хочет пользователь, и, во-вторых, его уровень подготовки и окружающую его обстановку. Ясное описание соответствующей сферы деятельности пользователя или интересующей его проблемной области во многом облегчает достижение разработчиками этой цели. При разработке ПС следует привлекать пользователя для участия в процессах принятия решений, а также тщательно освоить особенности его работы (лучше всего - побывать в его "шкуре").

3.8. Контроль принимаемых решений.

Обязательным шагом в каждом процессе (этапе) разработки ПС должна быть проверка правильности принятых решений. Это позволит обнаруживать и исправлять ошибки на самой ранней стадии после ее возникновения, что, во-первых, существенно снижает стоимость ее исправления и, во-вторых, повышает вероятность правильного ее устранения.

С учетом специфики разработки ПС необходимо применять везде, где это возможно,

- смежный контроль,
- сочетание как статических, так и динамических методов контроля.

Смежный контроль означает, проверку полученного документа лицами, не участвующими в его разработке, с двух сторон: во-первых, со стороны автора исходного для контролируемого процесса документа, и, во-вторых, лицами, которые будут использовать полученный документ в качестве исходного в последующих технологических процессах. Такой контроль позволяет обеспечивать однозначность интерпретации полученного документа.

Сочетание статических и динамических методов контроля означает, что нужно не только контролировать документ как таковой, но и проверять, какой процесс обработки данных он описывает. Это отражает одну из специфических особенность ПС (статическая форма, динамическое содержание).

Упражнения к лекции 3.

3.1. Что такое жизненный цикл программного средства (ПС)?
3.2. Что такое внешнее описание ПС?
3.3. Что такое сопровождение ПС?
3.4. Что такое качество ПС?
3.5. Что такое смежный контроль?

Литература к лекции 3.

3.1. Е.А. Жоголев. Введение в технологию программирования (конспект лекций). - М.: "ДИАЛОГ-МГУ", 1994.
3.2. М. Зелковец, А. Шоу, Дж. Гэннон. Принципы разработки программного обеспечения. - М.: Мир, 1982. - С. 11.
3.3. К. Зиглер. Методы проектирования программных систем. - М.: Мир, 1985. - С. 15-23.

3.4. Дж. Фокс. Программное обеспечение и его разработка. - М.: Мир, 1985. - С. 53-67, 125-130.
3.5. Ian Sommerville. Software Engineering. - Addison-Wesley Publishing Company, 1992. - P. 5-10.
3.6. Criteria for Evaluation of Software. ISO TC97/SC7 #383.
3.7. Revised version of DP9126 - Criteria of the Evaluation of Software Quality Characteristics. ISO TC97/SC7 #610. - Part 6.
3.8. Б. Боэм, Дж. Браун, Х. Каспар и др. Характеристики качества программного обеспечения. - М.: Мир, 1981. - С. 17-24.
3.9. В.В. Липаев. Качество программного обеспечения. - М.: Финансы и статистика, 1983. - С. 18-30.
3.10. Б. Шнейдерман. Психология программирования. - М.: Радио и связь, 1984. - С. 99-103.
3.11. Г. Майерс. Надежность программного обеспечения. - М.: Мир, 1980. - С. 32-48.
3.12. Д. Пойа. Как решать задачу. - М.: Наука, 1961.
3.13. В.В. Липаев, Б.А. Позин, А.А. Штрик. Технология сборочного программирования. – М.: Радио и связь, 1992.?