Программная инженерия

Конфигурационное управление

Лекция 8

Лекция 8 Управление конфигурациями

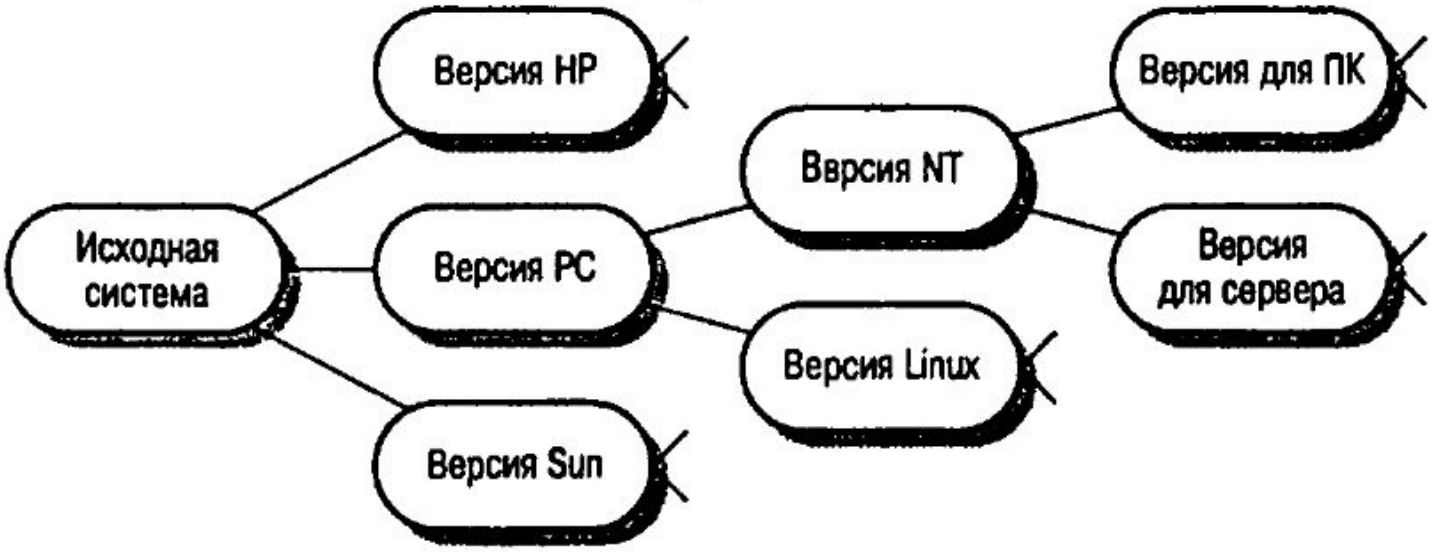
Цели

1. Понимать значение управления конфигурацией ПО
2. Знать о четырех основных процессах конфигурационного управления: планирование управления; управление изменениями; управление версиями и управление сборкой системы
3. Иметь представление о CASE средствах для поддержки процессов конфигурационного управления

Читать по Лекции 8	
Книга	**Страницы**
Д.В.*Кознов ,и др. Введение в программную инженерию*	***Лекция 6: Стр. 34-39***
Иан Соммервилл Инженерия программного обеспечения	***Глава 29: Стр. 585 - 602***

Управление конфигурацией – это процесс разработки и применения стандартов и правил по управлению эволюцией программных продуктов. Эволюционирующие системы нуждаются в управлении по той простой причине, что в процессе их эволюции создается несколько версий одних и тех же программ. В эти версии обязательно вносятся некоторые изменения, исправляются ошибки предыдущих версий; кроме того, версии могут адаптироваться к новым аппаратным средствам и операционным системам. При этом в разработке и эксплуатации могут одновременно находиться сразу несколько версий. Поэтому нужно четко отслеживать все вносимые в систему изменения.

Существует много причин, объясняющих наличие разных конфигураций одной и той же системы. Различные версии создаются для разных компьютеров или операционных систем, включающих специальные функции, нужные заказчикам, и т.д. (рис. 29.1). Менеджеры по управлению конфигурацией обязаны следить за различиями между разными версиями, чтобы обеспечить возможность выпуска следующих вариантов системы и своевременную поставку нужных версий соответствующим заказчикам.



Процесс управления конфигурацией и связанная с ним документация должны подчиняться определенным стандартам. В качестве примера можно привести стандарт IEEE 828-1983, определяющий составление планов управления конфигурацией.

Для сертификации качества своих программных продуктов организация должна придерживаться официальных стандартов управления конфигурацией, которые приведены в стандартах ISO 9000 [178] и в модели оценки уровня развития SEI

При традиционной разработке ПО в соответствии с каскадной моделью (см. главу 3) разрабатываемая система попадает в группу по управлению конфигурацией уже после полного завершения разработки и тестирования ПО. Именно такой подход лежит в основе стандартов управления конфигурацией, которые, в свою очередь, обусловливают необходимость использования для разработки систем моделей, подобных каскадной [38]. Поэтому упомянутые стандарты не в полной мере подходят при использовании таких методов разработки ПО, как эволюционное прототипирование и пошаговая разработка. В этой ситуации некоторые организации изменили подход к управлению конфигурацией, сделав возможным параллельную разработку и тестирование системы. Такой подход основан на регулярной (иногда ежедневной) сборке системы из ее компонентов.

1. Устанавливается время, к которому должна быть завершена поставка компонентов системы (например, к 14.00). Программисты, работающие над новыми версиями компонентов, должны предоставить их к указанному времени. Работу над компонентами не обязательно завершать, достаточно представить основные рабочие функции для проведения тестирования.
2. Создается новая версия системы с новыми компонентами, которые компилируются и связываются в единую систему.
3. После этого система попадает к группе тестирования. В то же время разработчики продолжают работу над компонентами, добавляя новые функции и исправляя ошибки, обнаруженные в ходе предыдущего тестирования.
4. Дефекты, замеченные при тестировании, регистрируются, соответствующий документ пересылается разработчикам. В следующей версии компонента эти дефекты будут учтены и исправлены.

Основным преимуществом ежедневной сборки системы является возможность выявления ошибок во взаимодействиях между компонентами, которые в противном случае могут накапливаться. Более того, ежедневная сборка системы поощряет тщательную проверку компонентов.

Для ежедневных сборок системы требуется достаточно строгое управление процессом изменений, позволяющее отслеживать проблемы, которые выявляются и исправляются в ходе тестирования. Кроме того, в результате возникает множество версий компонентов системы, для управления которыми необходимы средства управления конфигурацией.

Планирование управления конфигурацией

1. Определение контролируемых объектов, подпадающих под управление конфигурацией, а также формальная схема определения этих объектов.
2. Перечень лиц, ответственных за управление конфигурацией и за поставку контролируемых объектов в команду по управлению конфигурацией.
3. Политика ведения управления конфигурацией, т.е. процедуры управления изменениями и версиями.
4. Описание форм записей о самом процессе управления конфигурацией.
5. Описание средств поддержки процесса управления конфигурацией и способов их использования.
6. Определение базы данных конфигураций, применяемой для хранения всей информации о конфигурациях системы.

Для планирования процесса управления конфигурацией необходимо точно определить, какие проектные элементы (или классы элементов) будут объектами управления. Такие элементы называются *конфигурационными элементами*. Как правило, они представляют собой официальные документы. Конфигурационными элементами обычно являются планы проектов, спецификации, схемы системной архитектуры, программы и наборы тестовых данных. Кроме того, управлению подлежат все документы, необходимые для будущего сопровождения системы.

Листья дерева иерархии документов являются официальными документами проекта. На рис. 29.2 показано, что для каждого объекта требуется три формальных документа. Это описание объектов (документ ОБЪЕКТЫ), код компонента (документ КОД) и набор тестов для этого кода (документ ТЕСТЫ).

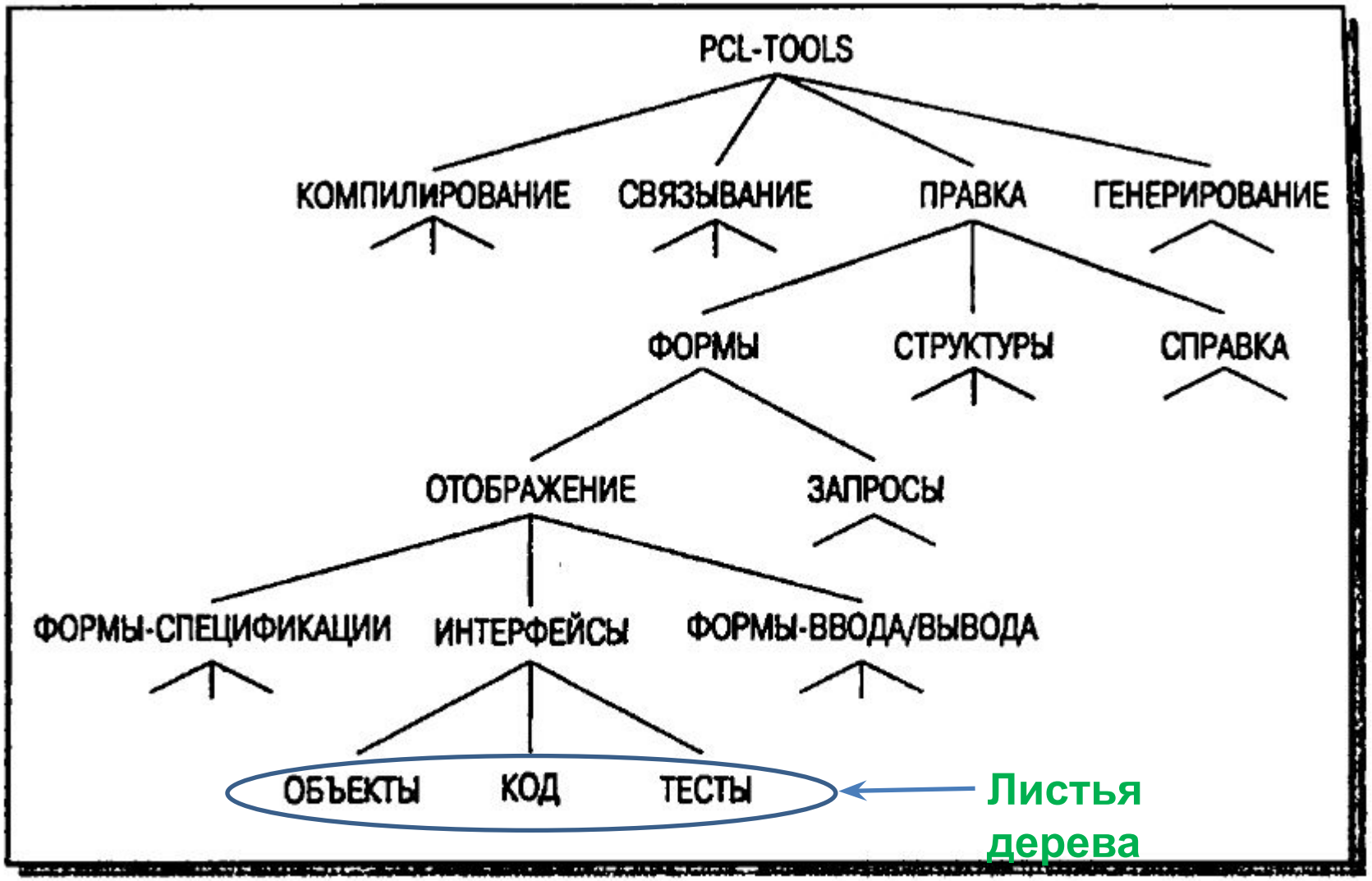


Рис. 29.2. Иерархия конфигурации

База данных конфигураций

Информация, заключенная в базе данных конфигураций, должна помочь ответить на ряд вопросов, среди которых основными и часто запрашиваемыми будут следующие.

- Каким заказчикам поставлена определенная версия системы?
- Какие аппаратные средства и какая операционная система необходимы для работы данной версии системы?
- Сколько было выпущено версий данной системы и когда?
- На какие версии системы повлияют изменения, вносимые в определенный компонент?
- Сколько запросов на изменения было реализовано в данной версии?
- Какое количество ошибок было зарегистрировано в данной версии системы?

В идеале база данных конфигураций должна быть объединена с системой управления версиями, которая создается для хранения и управления формальными проектными документами. Такой подход, к тому же поддерживаемый некоторыми интегрированными CASE-средствами, предоставляет возможность связать изменения, вносимые в систему, и с документами, и с теми компонентами, которые подверглись изменениям. В этом случае упрощается поиск измененных компонентов, поскольку установлены связи между документами (например, между документами по системной архитектуре и кодом программ) и этими связями можно управлять.

Управление

Изменения в больших программных системах неизбежны. Как отмечалось в предыдущих главах, в течение жизненного цикла системы изменяются пользовательские и системные требования, а также приоритеты и запросы организаций. Процесс управления изменениями и соответствующие CASE-средства предназначены для того, чтобы зарегистрировать изменения и внести их в систему наиболее эффективным способом.

Процесс управления изменениями (листинг 29.1) начинается после того, как программное обеспечение или соответствующая документация передается команде по управлению конфигурацией. Он может начаться во время тестирования системы или даже после ее поставки заказчику. Процедуры управления изменениями создаются для обеспечения корректного анализа необходимости изменений и их стоимости, а также для контроля за вносимыми изменения.

Все изменения, кроме тех, которые относятся к исправлению мелких недоработок, должны быть переданы в группу контроля за изменениями, где принимается решение о принятии изменения либо отказе. Эта группа оценивает воздействие изменения не с технической, а скорее с организационной или стратегической точек зрения. Во внимание принимаются такие соображения, как экономическая выгодность изменения и организационные факторы, которые оправдывают необходимость изменения.

Группа контроля за изменениями состоит из лиц, на которых возлагается ответственность за решения о внесении изменений. Такие группы со структурой, включающей старшего менеджера компании-заказчика и сотрудников фирмы-разработчика, обязательны при выполнении военных проектов. Для небольших или среднего размера проектов в эту группу может входить только менеджер проекта и один-два инженера, которые не занимались разработкой данного ПО. В отдельных случаях допускается участие аналитика по изменениям, который дает реко-

Листинг 29.1. Процесс управления изменениями

```
Запрос на изменение, заполнение формы запроса
Анализ запроса
if изменение допустимо then
    Оценка способа внесения изменения
    Оценка стоимости изменения
    Запись запроса в базу данных
    Передача запроса группе контроля за изменениями
    if запрос принят then
        repeat
            внесение изменений в ПО
            регистрация изменений
 передача измененного ПО группе управления качеством
        until качество ПО соответствует нормам
        создание ноной версии системы
    else
        запрос на изменение отвергнут
else
    запрос на изменение отвергнут
```

После принятия решения о внесении изменений программная система для внесения изменений передается разработчикам или команде по сопровождению системы. По окончании этой процедуры система обязательно должна пройти проверку на правильность внесения изменений. После этого именно команда по управлению конфигурацией, а не разработчики, займется выпуском новой версии.

Изменение каждого компонента системы должно регистрироваться. Таким образом создается история компонента. Самый лучший способ для этого – создавать стандартизированные комментарии в начале кода компонента (листинг 29.2), где содержатся ссылки на запросы изменений данного компонента. Для составления отчетов об изменениях компонента и обработки их историй используются специальные средства.

Управление версиями и выпусками

Версией системы называют экземпляр системы, имеющий определенные отличия от других экземпляров этой же системы. Новые версии могут отличаться функциональными возможностями, эффективностью или исправлениями ошибок. Некоторые версии имеют одинаковую функциоиальность, однако разработаны под различные конфигурации аппаратного или программного обеспечения. Если отличия между версиями иезначительны, они называются *вариантами* одной версии.

Выходная версия (release) системы – это та версия, которая поставляется заказчику. В каждой выходной версии либо обязательно присутствуют новые функциональные возможности, либо она разработана под новую платформу. Количество версий обычно намного превышает количество выходных версий, поскольку версии создаются в основном для внутреннего пользования и не поставляются заказчику.

В настоящее время для поддержки управления версиями разработано много разнообразных CASE-средств (см. раздел 29.5). С помощью этих средств осуществляется управление хранением каждой версии и контроль за допуском к компонентам системы. Компоненты могут извлекаться из системы для внесения в них изменений. После введения в систему измененных компонентов получается новая версия, для которой с помощью системы управления версиями создается новое имя.

Идентификация версий:Нумерация; На основе атрибутов;

1. *Нумерация версий.* Каждый компонент имеет уникальный и явный номер версии. Эта схема идентификации используется наиболее широко.
2. *Идентификация, основанная на значениях атрибутов.* Каждый компонент идентифицируется именем, которое, однако, не является уникальным для разных версий, и набором значений атрибутов, разных для каждой версии компонента [110]. Здесь версия компонента идентифицируется комбинацией имени и набора значений атрибутов.
3. *Идентификация на основе изменений.* Каждая версия системы именуется так же, как в способе идентификации, основанном на значениях атрибутов, плюс ссылки на запросы на изменения, которые реализованы в данной версии системы [244]. Таким образом, версия системы идентифицируется именем и теми изменениями, которые реализованы в системных компонентах.

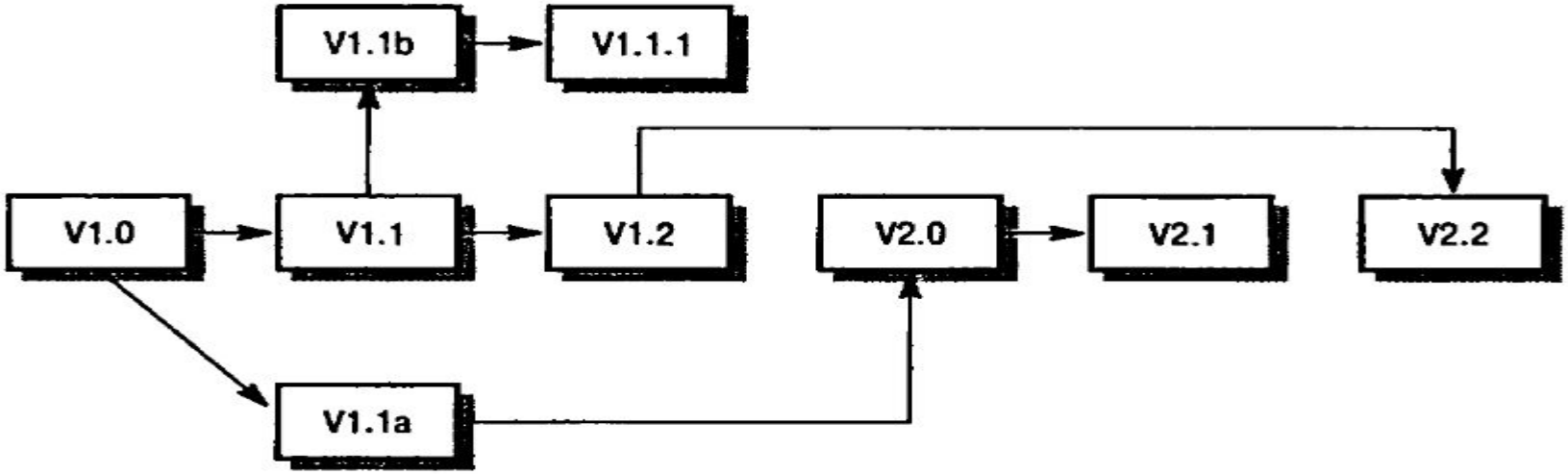


Рис. 29.3. Структура системных версий

Управление выходными версиями

Выходная версия системы включает в себя не только системный код, но также ряд компонентов.

1. *Конфигурационные файлы*, определяющие способ конфигурирования системы для каждой инсталляции.
2. *Файлы данных*, необходимые для работы системы.
3. *Программа установки*, которая помогает инсталлировать систему.
4. *Документация* в электронном и печатном виде, описывающая систему.
5. *Упаковка и рекламные материалы*, разработанные специально для этой версии системы.

Таблица 29.1. Факторы, влияющие на стратегию выпуска версий системы

Фактор	Описание
Техническое качество системы	Необходимость выпуска новой версии обусловлена зарегистрированными ошибками в существующей версии системы. Небольшие дефекты можно устранить с помощью заплат (patches), которые часто распространяются через Internet
Пятый закон Лемана (см. главу 27)	Этот закон постулирует постоянство приращения функциональных возможностей в каждой выходной версии по сравнению с предыдущей. Однако существуют и исключения, например за версией с достаточно большими изменениями следует версия с исправлением ошибок
Конкуренция	Необходимость новой версии объясняется наличием на рынке конкурирующих продуктов
Требования рынка	Отдел маркетинга компании может приурочить выход новой версии к определенной дате
Предложения заказчика об изменениях в системе	Для разработанных под заказ систем заказчик может предложить внести в систему ряд изменений, тогда новая версия выйдет сразу после реализации этих изменений

Сборка

Сборкой системы называют процесс компиляции и связывания программных компонентов в единую исполняемую программу. Перед сборкой системы полезно ответить на следующие вопросы.

1. Все ли компоненты, составляющие систему, включены в инструкцию по сборке?
2. Каковы версии компонента, перечисленные в инструкции по сборке?
3. Доступны ли все необходимые файлы данных?
4. Если на файлы данных используются ссылки внутри компонентов, то каковы имена этих файлов в выходной версии?
5. Доступны ли нужные версии компилятора и других необходимых средств? Действующие версии программных средств могут быть несовместимы с более старыми версиями, которые применялись при разработке системы.

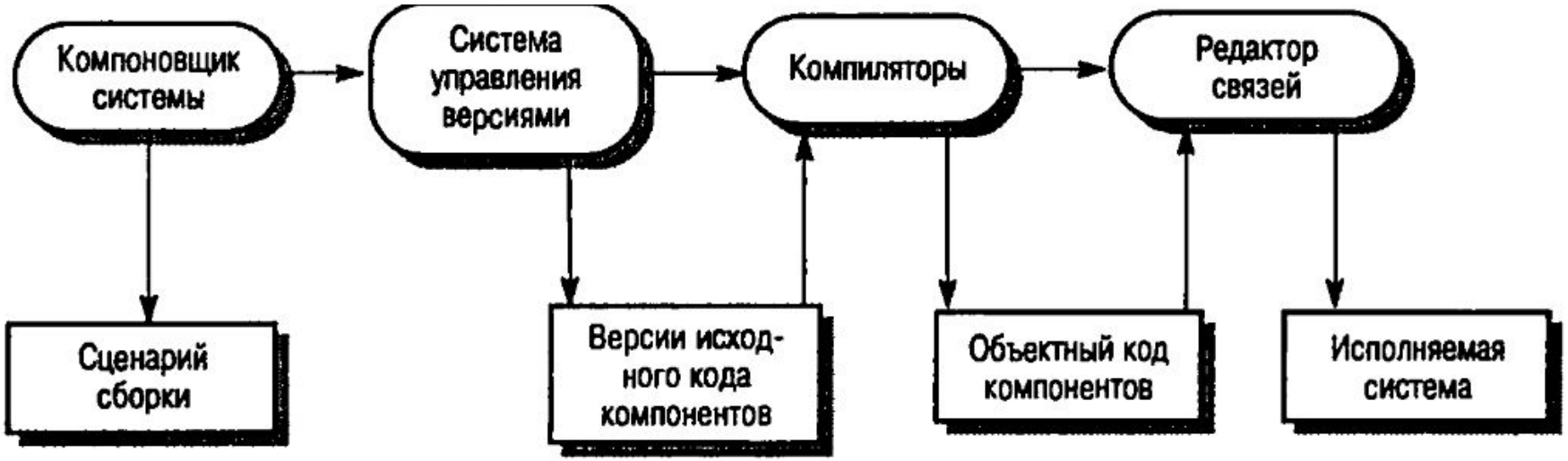


Рис. 29.4. Сборка системы