Проектная модель для проекта FreeBSD

Содержание

Предисловие

До настоящего момента проект FreeBSD выпустил ряд описанных методик для выполнения различных частей работы. Однако, из-за растущего числа участников проекта, необходима модель проекта, обобщающая его структуру. [1] Данная статья предоставляет такую модель проекта и передаётся в проект документации FreeBSD, где она может развиваться вместе с проектом, чтобы в любой момент времени отражать способ его работы. Она основана на диссертации [Saers].

Я хотел бы поблагодарить следующих людей за то, что они нашли время объяснить мне непонятные моменты и проверить документ.

- Andrey A. Chernov ache@freebsd.org
- Bruce A. Mah bmah@freebsd.org
- Dag-Erling Smørgrav des@freebsd.org
- Giorgos Keramidas keramida@freebsd.org
- Ingvil Hovig ingvil.hovig@skatteetaten.no
- Jesper Holck jeh.inf@cbs.dk
- John Baldwin jhb@freebsd.org
- John Polstra jdp@freebsd.org
- Kirk McKusick mckusick@freebsd.org
- Mark Linimon linimon@freebsd.org
- Marleen Devos
- Niels Jørgenssen nielsj@ruc.dk
- Nik Clayton nik@freebsd.org
- Poul-Henning Kamp phk@freebsd.org
- Simon L. Nielsen simon@freebsd.org

Chapter 1. Обзор

Модель проекта — это способ снижения накладных расходов на коммуникации в проекте. Как показано в [Brooks], увеличение числа участников проекта приводит к экспоненциальному росту коммуникаций в проекте. За последние годы FreeBSD значительно увеличил как количество активных пользователей, так и коммиттеров, что соответственно привело к росту коммуникаций. Данная модель проекта поможет снизить эти накладные расходы за счёт предоставления актуального описания проекта.

Во время выборов в Core в 2002 году Марк Мюррей заявил: «Я против длинного свода правил, так как это удовлетворяет склонности к юриспруденции и противоречит техноцентричности, в которой проект так нуждается.» [FreeBSD]. Эта модель проекта не предназначена для того, чтобы оправдывать создание ограничений для разработчиков, а служит инструментом для облегчения координации. Она призвана описывать проект, давая обзор того, как выполняются различные процессы. Это введение в то, как работает проект FreeBSD.

Модель проекта FreeBSD будет описана по состоянию на 1 июля 2004 года. Она основана на работе Нильса Йоргенсена [Jørgensen], официальных документах FreeBSD, обсуждениях в списках рассылки FreeBSD и интервью с разработчиками.

После определения используемых терминов в этом документе будет описана организационная структура (включая описания ролей и линии коммуникации), рассмотрена модель методологии, а после представления инструментов, используемых для контроля процессов, будут описаны определенные процессы. В заключение будут представлены основные подпроекты проекта FreeBSD.

[FreeBSD] Разделы 1.2 и 1.3 описывают видение и архитектурные принципы проекта. Видение сформулировано как: "Создать наилучший пакет операционной системы, подобной UNIX®, с должным уважением к оригинальной идеологии программных инструментов, а также к удобству использования, производительности и стабильности." Архитектурные принципы помогают определить, находится ли проблема, которую кто-то хочет решить, в рамках проекта

[1] Это согласуется с законом Брукса, согласно которому добавление нового человека в задерживающийся проект сделает его ещё более задержанным, поскольку увеличит потребность в коммуникации. Модель проекта — это инструмент для снижения потребности в коммуникации.

Chapter 2. Определения

2.1. Активность

"Активность" — это элемент работы, выполняемый в ходе проекта [PMI]. У неё есть результат, который ведёт к достижению цели. Такой результат может быть либо входом для другой активности, либо частью поставки процесса.

2.2. Процесс

Процесс — это ряд действий, ведущих к определенному результату. Процесс может состоять из одного или нескольких подпроцессов. Примером процесса является проектирование программного обеспечения.

2.3. Роль (hat)

"Hat" (шляпа) является синонимом роли. Роль имеет определенные обязанности в процессе и ответственность за результат процесса. Роль выполняет действия. Четко определено, по каким вопросам участники проекта и люди вне проекта должны обращаться к ответственному, выполняющему эту роль.

2.4. Результат

«Результат» — это конечный продукт процесса. Это синоним понятия «поставляемый результат», который определяется как «любой измеримый, осязаемый, проверяемый результат, итог или элемент, который должен быть произведён для завершения проекта или его части. Часто используется в более узком смысле в отношении внешнего поставляемого результата, который подлежит утверждению спонсором проекта или заказчиком» согласно [PMI]. Примерами результатов являются программное обеспечение, принятое решение или написанный отчёт.

2.5. FreeBSD

Говоря "FreeBSD", мы подразумеваем UNIX-подобную операционную систему FreeBSD, основанную на BSD, тогда как говоря "Проект FreeBSD", мы подразумеваем организацию проекта.

Chapter 3. Организационная структура

Хотя никто не является владельцем FreeBSD, организация FreeBSD разделена на ядро, коммиттеров и участников и является частью сообщества FreeBSD, которое существует вокруг неё.

Структура проекта FreeBSD (в порядке убывания полномочий)

Группа	**Количество людей**
Основные участники	9
Коммиттеры	318
Участники	~3000

Количество коммиттеров было определено путем анализа журналов CVS с 1 января 2004 года по 31 декабря 2004 года, а список участников — путем просмотра перечня внесенных изменений и отчётов о проблемах.

Основной ресурс сообщества FreeBSD — это его разработчики: коммиттеры и контрибьюторы. Именно их вклад позволяет проекту развиваться. Обычные разработчики называются участниками (контрибьюторами). По состоянию на 1 января 2003 года в проекте насчитывается около 5500 контрибьюторов.

Коммиттеры — это разработчики, обладающие привилегией вносить изменения. Обычно это наиболее активные разработчики, которые готовы тратить своё время не только на интеграцию собственного кода, но и на интеграцию кода, предоставленного разработчиками без такой привилегии. Они также выбирают основную команду и имеют доступ к закрытым обсуждениям.

Проект можно разделить на четыре отдельные части, и большинство разработчиков сосредоточат своё участие на одной из частей FreeBSD. Эти четыре части — разработка ядра, разработка пользовательского пространства, порты и документация. Под базовой системой подразумеваются как ядро, так и пользовательское пространство.

Это разделение изменяет нашу таблицу следующим образом:

Структура проекта FreeBSD с участниками, имеющими права на запись, по категориям

Группа	**Категория**	**Количество людей**
Основные участники		9
Коммиттеры	Базовый	164
	Docs	45
	Порты	166
	Total	374
Участники		~3000

Количество коммиттеров по областям было определено путем анализа журналов CVS с 1 января 2004 года по 31 декабря 2004 года. Обратите внимание, что многие коммиттеры работают в нескольких областях, поэтому общее число больше реального количества коммиттеров. Общее количество активных уникальных коммиттеров на июнь 2022 года составляло 317.

Коммиттеры делятся на три группы: коммиттеры, занимающиеся только одной областью проекта (например, файловыми системами), коммиттеры, участвующие только в одном подпроекте, и коммиттеры, вносящие изменения в разные части кода, включая подпроекты. Поскольку некоторые коммиттеры работают над разными частями, общее количество в разделе коммиттеров таблицы выше, чем в предыдущей таблице.

Ядро является основным строительным блоком FreeBSD. Хотя пользовательские приложения защищены от сбоев в других пользовательских приложениях, вся система уязвима от ошибок в ядре. Это, в сочетании с огромным количеством зависимостей в ядре и тем, что нелегко увидеть все последствия изменения ядра, требует от разработчиков относительно полного понимания ядра. Множественные усилия по разработке в ядре также требуют более тесной координации, чем пользовательские приложения.

Основные утилиты, известные как пользовательское окружение (userland), предоставляют интерфейс, который определяет FreeBSD, включая пользовательский интерфейс, общие библиотеки и внешние интерфейсы для подключения клиентов. В настоящее время 162 человека участвуют в разработке и поддержке пользовательского окружения, многие из которых являются сопровождающими (maintainers) для своей части кода. Вопросы сопровождения будут рассмотрены в разделе Сопровождение.

Документация обрабатывается Проектом документации FreeBSD и включает все документы, связанные с проектом FreeBSD, включая веб-страницы. В течение 2004 года 101 человек внесли изменения в Проект документации FreeBSD.

Порты — это коллекция метаданных, необходимых для корректной сборки программных пакетов в FreeBSD. Например, порт для веб-браузера Mozilla содержит информацию о том, откуда загружать исходный код, какие патчи применять и как, а также как пакет должен быть установлен в системе. Это позволяет автоматизированным инструментам загружать, собирать и устанавливать пакеты. На момент написания доступно более 12600 портов [1], начиная от веб-серверов и игр до языков программирования и большинства типов приложений, используемых на современных компьютерах. Порты подробно рассматриваются в разделе Подпроект Ports.

[1] Статистика получена подсчётом количества записей в файле, загруженном portsdb на 1 апреля 2005 года. portsdb является частью порта sysutils/portupgrade.

Chapter 4. Методологическая модель

4.1. Модель разработки

Не существует определенной модели того, как люди пишут код в FreeBSD. Однако Нильс Йоргенссен предложил модель того, как написанный код интегрируется в проект.

Модель Йоргенссена для интеграции изменений

Этап	Следующий, если успешно	Следующий, если неудачно
программирование	рецензирование	
рецензирование	предварительная проверка перед коммитом	программирование
предварительная проверка перед коммитом	релиз для разработки	программирование
релиз для разработки	параллельная отладка	программирование
параллельная отладка	релиз для производства	программирование
релиз для производства		программирование

"Релиз для разработки" — это ветка FreeBSD-CURRENT ("-CURRENT"), а "релиз для производства" — ветка FreeBSD-STABLE ("-STABLE") [Jørgensen].

Это модель для одного изменения, которая показывает, что после написания кода разработчики ищут рецензирование сообщества и пытаются интегрировать это изменение в свои собственные системы. После интеграции изменения в версию разработки, называемую FreeBSD-CURRENT, оно тестируется многими пользователями и разработчиками сообщества FreeBSD. После достаточного тестирования оно объединяется с производственной версией, называемой FreeBSD-STABLE. Если каждая стадия не завершена успешно, разработчику необходимо вернуться, внести изменения в код и перезапустить процесс. Интеграция изменения в -CURRENT или -STABLE называется выполнением коммита.

Йоргенсен обнаружил, что большинство разработчиков FreeBSD работают индивидуально, то есть эта модель используется параллельно многими разработчиками в различных текущих процессах разработки. Разработчик также может работать над несколькими изменениями одновременно, поэтому, ожидая рецензирования или тестирования одного или нескольких своих изменений, он может писать другое изменение.

Поскольку каждый коммит представляет собой инкрементальное изменение, это модель с очень высокой степенью инкрементальности. Коммиты происходят настолько часто, что за один год [1], было сделано 85427 коммитов, что составляет в среднем 233 коммита в день.

В рамках "программирования" в модели Йоргенсена, каждый программист имеет свой собственный стиль работы и следует своим собственным моделям разработки. Этот этап вполне мог бы называться "разработкой", так как он включает сбор и анализ требований,

системное и детальное проектирование, реализацию и проверку. Однако, единственным результатом этих подэтапов являются исходный код или документация системы.

С точки зрения пошаговой модели (такой как каскадная модель), остальные этапы можно рассматривать как дальнейшую проверку и интеграцию системы. Эта интеграция системы также важна для определения того, будет ли изменение принято сообществом. До момента фиксации кода разработчик волен выбирать, насколько активно он будет обсуждать его с остальными участниками проекта. Чтобы -CURRENT мог выполнять роль буфера (позволяя откатывать идеи, которые оказались с невыявленными недостатками), минимальный срок, в течение которого изменения должны оставаться в -CURRENT перед слиянием в -STABLE, составляет 3 дня. Такое слияние называется MFC (Merge From Current).

Важно обратить внимание на слово "изменение". Большинство коммитов не содержат радикально новых функций, а представляют собой обновления для поддержки.

Единственными исключениями из этой модели являются исправления безопасности и изменения в функциях, объявленных устаревшими в ветке -CURRENT. В этих случаях изменения могут быть внесены напрямую в ветку -STABLE.

В дополнение к множеству людей, работающих над проектом, существует множество связанных проектов в рамках FreeBSD. Это могут быть проекты, разрабатывающие совершенно новые функции, подпроекты или проекты, результаты которых интегрируются в FreeBSD [2]. Эти проекты вписываются в FreeBSD так же, как и обычные разработки: они создают код, который интегрируется с проектом FreeBSD. Однако некоторые из них (например, Ports и Documentation) имеют привилегию применяться к обеим веткам или коммитить напрямую как в -CURRENT, так и в -STABLE.

Не существует стандартов по выполнению проектирования, также как и централизованного репозитория для хранения проектов. Основной дизайн взят из 4.4BSD. [3] Поскольку проектирование является частью этапа "Программирование" в модели Йоргенсена, каждый разработчик или подпроект сам решает, как это должно выполняться. Даже если проектирование должно храниться в централизованном репозитории, результаты этапов проектирования будут иметь ограниченную полезность, так как различия в методологиях сделают их плохо совместимыми, если вообще совместимыми. Для общего проектирования проекта, проект полагается на подпроекты, которые договариваются о совместимых интерфейсах между собой, а не на диктат интерфейсов.

4.2. Ветви релизов

Версии FreeBSD лучше всего иллюстрируются деревом с множеством ветвей, где каждая основная ветвь представляет основную версию. Минорные версии представлены ветвями основных ветвей.

В следующем дереве релизов стрелки, следующие друг за другом в определенном направлении, представляют ветку. Прямоугольники со сплошными линиями и ромбы обозначают официальные релизы. Прямоугольники с пунктирными линиями представляют ветку разработки на тот момент. Ветки безопасности обозначены овалами. Ромбы отличаются от прямоугольников тем, что они представляют развилку, то есть место, где ветка разделяется на две ветки, одна из которых становится подветкой. Например, на

4.0-RELEASE ветка 4.0-CURRENT разделилась на 4-STABLE и 5.0-CURRENT. На 4.5-RELEASE ветка разветвилась на ветку безопасности под названием RELENG_4_5.

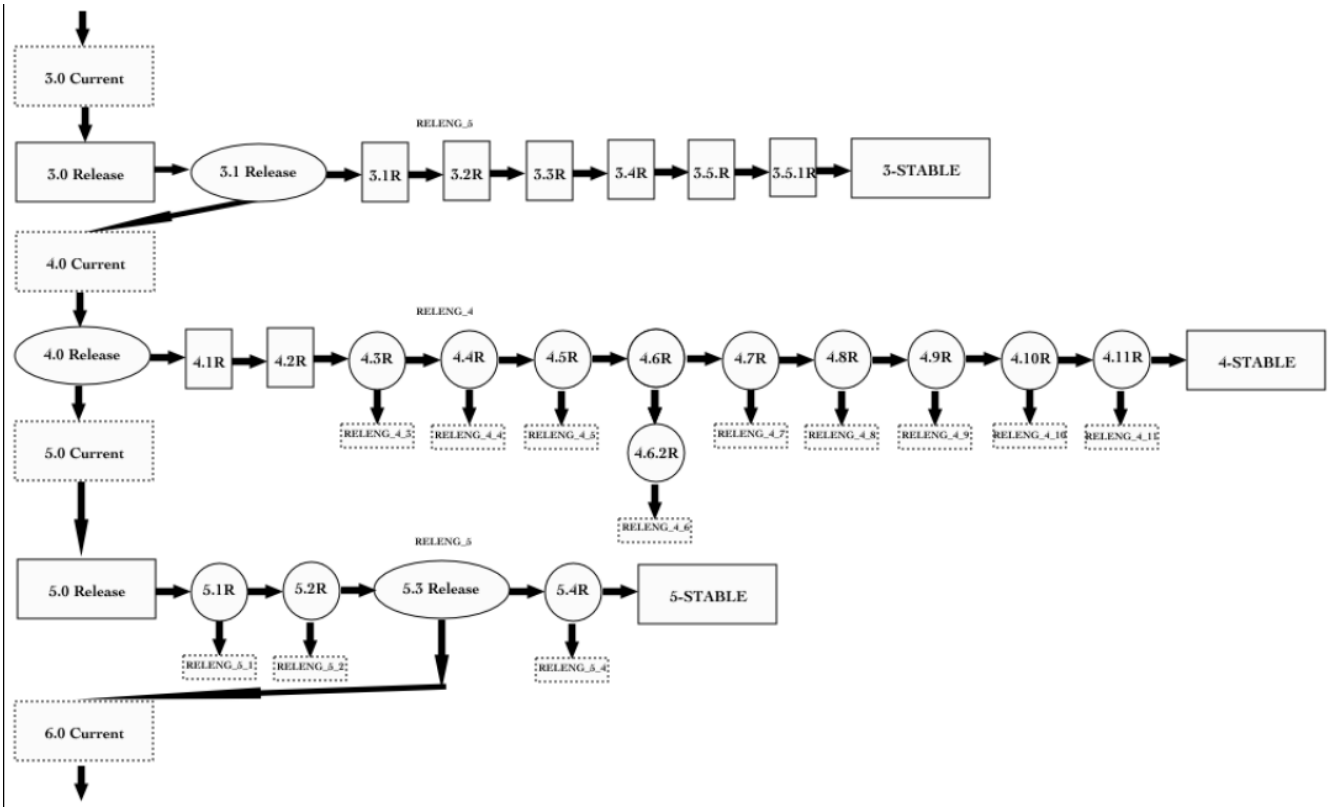

Рисунок 1. Дерево релизов FreeBSD

Основные выпуски	**Форк сделан из**	**Следующие минорные выпуски**
…		
3.0 Current (ветка разработки)		Ветки Releng 3: выпуски с 3.0 Release по 3.5 Release, ведущие к выпуску 3.5.1 Release и последующей ветке 3 Stable
4.0 Current (ветка разработки)	Релиз 3.1	Ветви Releng 4: релизы с 4.1 по 4.6 (и релиз 4.6.2), затем релизы с 4.7 по 4.11 (все, начиная с релиза 4.3, также ведущие к ветви Releng_4_n), и последующая ветвь релиза 4
5.0 Current (ветка разработки)	Релиз 4.0	Ветви Releng 5: Релизы с 5.0 до 5.4 (за исключением 5.0 и 5.3, которые также ведут к ветке Releng_5_n), и последующая ветка релиза 5
6.0 Current (ветка разработки)	Релиз 5.3	

Основные выпуски	Форк сделан из	Следующие минорные выпуски
…		

Последняя версия -CURRENT всегда обозначается как -CURRENT, а последний релиз -STABLE всегда обозначается как -STABLE. На этом рисунке -STABLE относится к 4-STABLE, а -CURRENT относится к 5.0-CURRENT после 5.0-RELEASE. [FreeBSD]

«Основной выпуск» всегда создаётся из ветки -CURRENT. Однако ветка -CURRENT не обязательно должна разветвляться в этот момент, а может сосредоточиться на стабилизации. Примером этого является то, что после 3.0-RELEASE, 3.1-RELEASE также был продолжением ветки -CURRENT, и -CURRENT не стал настоящей веткой разработки до тех пор, пока не был выпущен этот релиз и не была создана ветка 3-STABLE. Когда -CURRENT снова становится веткой разработки, за ним может следовать только основной выпуск. Ожидается, что ветка 5-STABLE будет отделена от 5.0-CURRENT примерно на момент выпуска 5.3-RELEASE. Только после отделения 5-STABLE ветка разработки получит название 6.0-CURRENT.

"Минорный релиз" создается из ветки -CURRENT после основного релиза или из ветки -STABLE.

Начиная с версии 4.3-RELEASE [4], когда выпускается минорный релиз, он становится «веткой безопасности». Это предназначено для организаций, которые не хотят следовать ветке -STABLE и потенциальным новым/изменённым функциям, которые она предлагает, но вместо этого требуют абсолютно стабильной среды, обновляемой только для внедрения исправлений безопасности. [5]

Каждое обновление в ветке безопасности называется "уровнем исправления" (patchlevel). Для каждого выполненного улучшения безопасности номер уровня исправления увеличивается, что позволяет легко отслеживать, какие улучшения безопасности были реализованы. В случаях особенно серьезных уязвимостей безопасности может быть выпущен полностью новый релиз из ветки безопасности. Примером этого является 4.6.2-RELEASE.

4.3. Сводка модели

Для подведения итогов, модель разработки FreeBSD можно представить в виде следующего дерева:

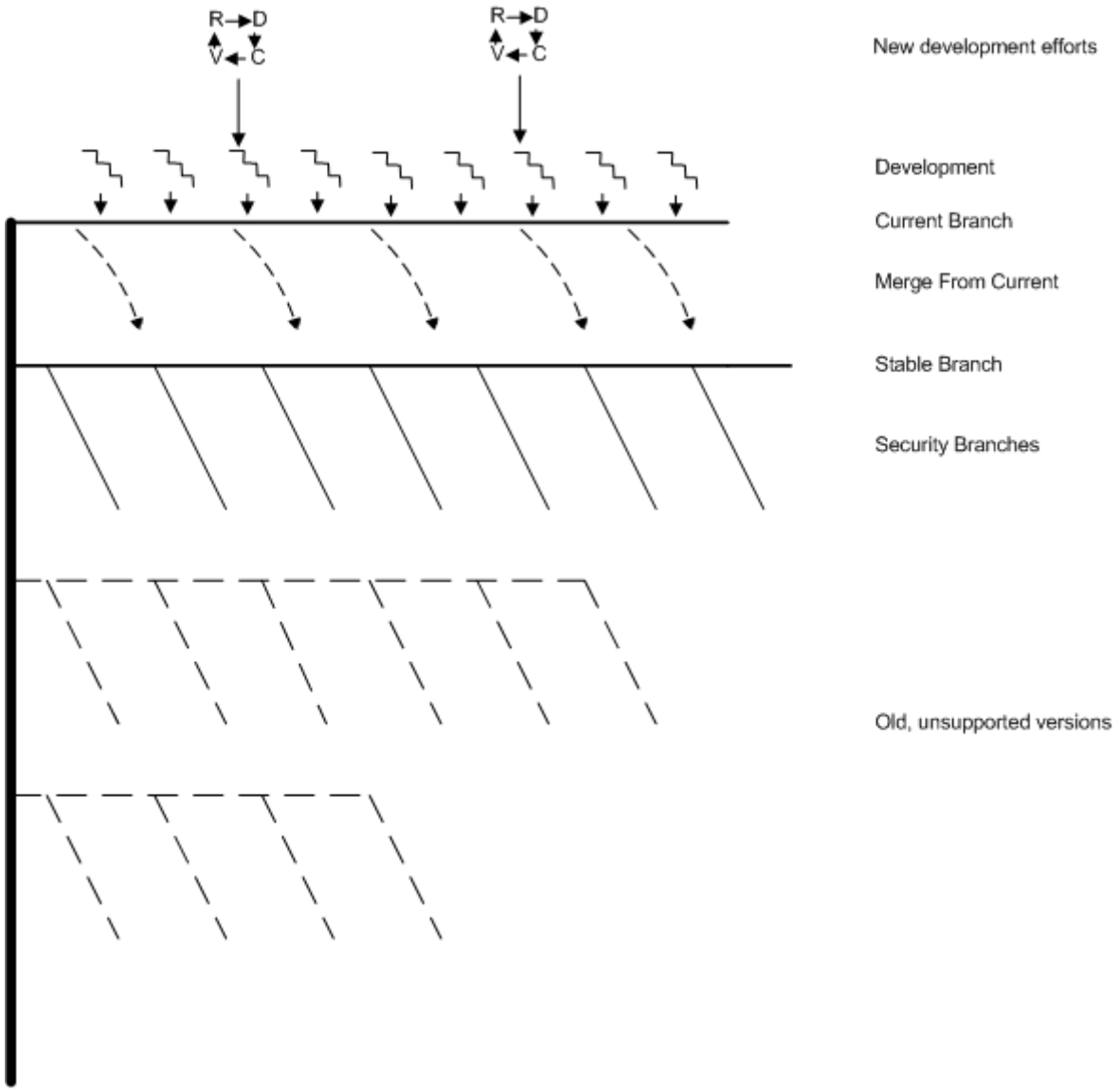

Рисунок 2. Общая модель разработки

Дерево разработки FreeBSD с текущими усилиями по разработке и непрерывной интеграцией.

Дерево символизирует версии выпусков, где основные версии порождают новые главные ветви, а второстепенные версии являются версиями главной ветви. Верхняя ветвь — это ветвь -CURRENT, в которую интегрируется вся новая разработка, а ветвь -STABLE находится непосредственно под ней. Под ветвью -STABLE находятся старые, неподдерживаемые версии.

Проект находится в тумане постоянной разработки, и разработчики выбирают модели разработки, которые считают подходящими. Результаты их работы затем интегрируются в -CURRENT, где проходят параллельную отладку, и наконец объединяются из -CURRENT в -STABLE. Исправления безопасности объединяются из -STABLE в ветки безопасности.

Многие коммиттеры имеют специальную область ответственности. Эти роли называются "hats" (шляпами). Эти роли могут быть либо проектными ролями, например, офицер по связям с общественностью, либо сопровождающим определённой части кода. Поскольку это проект, где люди добровольно уделяют своё свободное время, люди с назначенными ролями не всегда доступны. Поэтому они должны назначить заместителя, который может

выполнять эту роль в их отсутствие. Другой вариант — передать роль группе.

Многие из этих ролей не формализованы. Формализованные роли имеют устав, в котором указаны точные цели, привилегии и обязанности. Написание таких уставов — новая часть проекта, поэтому оно ещё не завершено для всех ролей. Эти описания ролей не являются формализацией, а скорее представляют собой краткое описание роли со ссылками на устав, где он доступен, и контактными адресами.

[1] Для получения этого числа был исследован период с 1 января 2004 года по 31 декабря 2004 года.

[2] Например, разработка стека Bluetooth начиналась как подпроект, пока не была признана достаточно стабильной для включения в ветку -CURRENT. Теперь это часть основной системы FreeBSD.

[3] По словам Кирка МакКузика, после 20 лет разработки операционных систем UNIX, интерфейсы в основном уже определены. Поэтому нет необходимости в большом количестве проектирования. Однако новые применения системы и новое оборудование приводят к тому, что некоторые реализации становятся более выгодными по сравнению с ранее предпочитаемыми. Одним из примеров является появление веб-браузинга, который превратил обычное TCP/IP-соединение в короткие всплески данных, а не в устойчивый поток за более длительный период времени.

[4] Первым релизом, для которого это действительно произошло, был 4.5-RELEASE, но ветки безопасности были созданы одновременно для 4.3-RELEASE и 4.4-RELEASE.

[5] Здесь вы видите терминологическое пересечение со словом «стабильный», что приводит к некоторой путанице. Ветка -STABLE по-прежнему является веткой разработки, цель которой — быть полезной для большинства пользователей. Если для системы неприемлемо получать изменения, которые не были объявлены на момент её развёртывания, такая система должна работать на ветке безопасности.

Chapter 5. Ответственные

5.1. Стандартные роли

5.1.1. Участник (контрибьютор)

Участник вносит вклад в проект FreeBSD в качестве разработчика, автора, отправляя отчёты о проблемах или другими способами способствуя прогрессу проекта. Участник не имеет особых привилегий в проекте FreeBSD. [FreeBSD]

5.1.2. Коммиттер

Человек, обладающий необходимыми привилегиями для добавления своего кода или документации в репозиторий. Коммиттер совершил коммит в течение последних 12 месяцев. [FreeBSD] Активный коммиттер — это коммиттер, который в среднем совершал один коммит в месяц в течение этого времени.

Стоит отметить, что нет технических препятствий, которые могли бы помешать кому-либо, получившему права на коммиты в основном или подпроекте, делать коммиты в частях исходного кода проекта, для которых у коммиттера нет явного разрешения на изменение. Однако, при желании внести изменения в части, с которыми коммиттер ранее не работал, следует изучить логи, чтобы понять, что происходило в этой области ранее, а также прочитать файл MAINTAINERS, чтобы узнать, есть ли у сопровождающего этой части какие-либо особые требования к внесению изменений в код.

5.1.3. Основная команда (Core Team)

Основная команда избирается коммиттерами из числа коммиттеров и выполняет функции совета директоров проекта FreeBSD. Она повышает активных участников до коммиттеров, назначает людей на четко определенные роли (hats) и является окончательным арбитром при принятии решений о направлении развития проекта. На 1 июля 2004 года в состав основной команды входило 9 членов. Выборы проводятся каждые два года.

5.1.4. Сопровождение

Сопровождение означает, что человек ответственен за то, что допускается в определённую часть кода, и имеет решающее слово в случае разногласий по поводу кода. Это включает в себя как активную работу, направленную на стимулирование участников (контрибьюторов), так и реактивную работу по рецензированию коммитов.

В исходном коде FreeBSD есть файл MAINTAINERS, содержащий краткое описание того, как каждый сопровождающий предпочитает получать вклады. Наличие этого уведомления и контактной информации позволяет разработчикам сосредоточиться на разработке, а не застревать в медленной переписке, если сопровождающий будет недоступен в течение некоторого времени.

Если сопровождающий недоступен в течение неоправданно долгого времени, и другие

люди выполняют значительный объем работы, сопровождение может быть передано без согласия сопровождающего. Это основано на позиции, что сопровождение должно быть продемонстрировано, а не заявлено.

Сопровождение определенного участка кода — это роль, которая не осуществляется коллективно.

5.2. Официальные Роли

Официальные роли в проекте FreeBSD — это более или менее формализованные и в основном административные должности. Они обладают полномочиями и ответственностью в своей области. В следующем списке показаны направления ответственности и дано описание каждой роли, включая информацию о том, кто её занимает.

5.2.1. Менеджер проекта документации

Архитектор Проекта документации FreeBSD отвечает за определение и контроль целей документации для коммиттеров в проекте Документации, за которым они присматривают.

Роль поддерживается: Командой DocEng doceng@FreeBSD.org. Устав DocEng.

5.2.2. Postmaster

Postmaster отвечает за корректную доставку почты на адреса электронной почты коммиттеров. Также он отвечает за работоспособность почтовых рассылок и должен принимать меры против возможных сбоев в работе почты, таких как троллинг-, спам- и вирус-фильтры.

Текущий руководитель: Команда почтовых серверов postmaster@FreeBSD.org.

5.2.3. Координация выпусков

Обязанности команды выпуска релизов включают

- Установка, публикация и соблюдение графика выпуска официальных релизов
- Документирование и формализация процедур выпуска релизов
- Создание и поддержка веток кода
- Согласование с командами Ports и Documentation для выпуска обновленного набора пакетов и документации вместе с новыми релизами
- Координация с командой безопасности для того, чтобы готовящиеся выпуски не были затронуты недавно обнаруженными уязвимостями.

Дополнительная информация о процессе разработки доступна в разделе Выпуск релизов.

Роль поддерживается: командой выпуск релизов (Release Engineering) re@FreeBSD.org. Устав Release Engineering.

5.2.4. Отношения с общественностью и корпоративные связи

Обязанности отдела по связям с общественностью и корпоративным отношениям включают:

- Публиковать пресс-релизы при возникновении событий, важных для проекта FreeBSD.
- Быть официальным контактным лицом для корпораций, тесно сотрудничающих с проектом FreeBSD.
- Принимать меры для продвижения FreeBSD как в сообществе Open Source, так и в корпоративном мире.
- Обрабатывать список рассылки "freebsd-advocacy".

Эта роль в настоящее время не занята.

5.2.5. Ответственный за безопасность (Security Officer)

Основная обязанность Ответственного за безопасность — координировать обмен информацией с сообществом безопасности и проектом FreeBSD. Ответственный за безопасность также принимает меры при поступлении сообщений о проблемах безопасности и способствует активному развитию в области безопасности.

Из-за опасений, что информация об уязвимостях может попасть к злоумышленникам до выпуска исправления, только Ответственный за безопасность, включающий руководителя, заместителя и двух членов Core Team, получает конфиденциальную информацию о проблемах безопасности. Однако для создания или внедрения исправления Ответственный за безопасность может обратиться к команде security-team@FreeBSD.org для помощи в выполнении работы.

5.2.6. Менеджер репозитория исходного кода

Менеджер репозитория исходного кода — единственный, кому разрешено напрямую изменять репозиторий без использования инструмента Git. В его обязанности входит оперативное решение технических проблем, возникающих в репозитории. Менеджер репозитория исходного кода имеет право отменять коммиты, если это необходимо для устранения технических проблем с Git.

Роль принадлежит: Менеджеру репозитория исходного кода clusteradm@FreeBSD.org.

5.2.7. Менеджер выборов

Менеджер выборов отвечает за процесс выборов Core Team. Он отвечает за проведение и поддержание системы выборов, а также является окончательной инстанцией в случае незначительных непредвиденных событий в процессе выборов. Крупные непредвиденные события должны обсуждаться с Core Team

Роль выполняется только во время выборов.

5.2.8. Управление веб-сайтом

Роль управления веб-сайтом отвечает за координацию развертывания обновленных веб-страниц на зеркалах по всему миру, за общую структуру основного веб-сайта и систему, на которой он работает. Управление должно согласовывать содержимое с Проектом документации FreeBSD и выступает в роли сопровождающего для дерева "www".

Роль поддерживается: веб-мастеры FreeBSD www@FreeBSD.org.

5.2.9. Менеджер портов

Менеджер портов выступает в роли связующего звена между Подпроектом портов и основным проектом, и все запросы от проекта должны направляться менеджеру портов.

Роль принадлежит: Команде управления портами portmgr@FreeBSD.org. Устав Portmgr.

5.2.10. Стандарты

Роль стандартов отвечает за обеспечение соответствия FreeBSD стандартам, которым система следует, отслеживает развитие этих стандартов и уведомляет разработчиков FreeBSD о важных изменениях. Это позволяет разработчикам действовать проактивно и сокращать время между обновлением стандартов и достижением соответствия в FreeBSD.

Текущий ответственный: Garrett Wollman wollman@FreeBSD.org.

5.2.11. Секретарь Core Team

Основная обязанность Секретаря Core Team — составление черновиков и публикация окончательных отчётов Core Team. Секретарь также ведёт повестку Core Team, гарантируя, что ни один вопрос не останется без решения.

Ответственный в настоящее время: René Ladan <rene@FreeBSD.org>.

5.2.12. Ответственный за ошибки (Bugmeister)

Ответственный за ошибки (Bugmeister) отвечает за поддержание базы данных по обслуживанию в рабочем состоянии, за корректную категоризацию записей и отсутствие недействительных записей. Они курируют исправителей ошибок (bugbusters).

Текущий ответственный: команда Bugmeister Team bugmeister@FreeBSD.org.

5.2.13. Представитель по привлечению пожертвований

Задача представителя по привлечению пожертвований — связывать разработчиков, которым что-то нужно, с людьми или организациями, готовыми сделать пожертвование.

Роль поддерживается: Отделом по привлечению пожертвований donations@FreeBSD.org. Устав Отдела по привлечению пожертвований.

5.2.14. Администратор (Admin)

(Также называется "Администратор кластера FreeBSD")

Команда администраторов состоит из людей, ответственных за администрирование компьютеров, которые проект использует для распределённой работы и синхронизации коммуникации. В основном в неё входят те, кто имеет физический доступ к серверам.

Роль поддерживается: командой администраторов admin@FreeBSD.org.

5.3. Процессозависимые роли

5.3.1. Инициатор отчёта

Лицо, изначально ответственное за подачу отчёта о проблеме.

5.3.2. Исправитель ошибок (Bugbuster)

Человек, который либо найдет подходящего специалиста для решения проблемы, либо закроет PR, если он является дубликатом или по другим причинам не представляет интереса.

5.3.3. Наставник (Mentor)

Наставник — это коммиттер, который берет на себя задачу ознакомить нового коммиттера с проектом. Это включает в себя проверку корректности настройки окружения нового коммиттера, обучение доступным инструментам, необходимым для работы, а также разъяснение ожидаемого поведения.

5.3.4. Поставщик

Лицо (лица) или организация, от которых поступает внешний код и которым отправляются исправления.

5.3.5. Рецензенты

Люди из списка рассылки, куда отправлен запрос на рецензирование.

Следующий раздел описывает установленные процессы проекта. Вопросы, не охваченные этими процессами, решаются по мере возникновения, исходя из сложившейся практики в аналогичных случаях.

Chapter 6. Процессы

6.1. Добавление новых и удаление старых коммиттеров

Основная команда (Core Team) отвечает за предоставление и отзыв прав на коммит для участников. Это может быть сделано только через голосование в списке рассылки Core Team. Подпроекты ports и documentation могут предоставлять права на коммит людям, работающим над этими проектами, но на данный момент не отзывали такие права.

Обычно кандидата в коммиттеры основной команде (Core Team) рекомендуют коммиттеры. Для участников или посторонних обращаться в Core Team с просьбой стать коммиттером считается неблагоразумным и такая просьба, как правило, отклоняется.

Если область, представляющая особый интерес для разработчика, потенциально пересекается с зоной ответственности других сопровождающих, запрашивается мнение этих сопровождающих. Однако часто именно этот сопровождающий рекомендует разработчика.

Когда участнику предоставляется статус коммиттера, ему назначается наставник. В общем случае коммиттер, который рекомендовал нового коммиттера, берет на себя обязанности наставника для нового коммиттера.

Когда участнику предоставляется право коммита (commit bit), отправляется подписанное с помощью Pretty Good Privacy письмо от Секретаря Core Team, Менеджера портов или nik@freebsd.org на адреса admins@freebsd.org, назначенного наставника, нового коммиттера и Core Team, подтверждая одобрение новой учётной записи. Затем наставник собирает строку пароля, Secure Shell открытый ключ и PGP-ключ от нового коммиттера и отправляет их Администратору. Когда новая учётная запись создана, наставник активирует право коммита и проводит нового коммиттера через остальные этапы начального процесса.

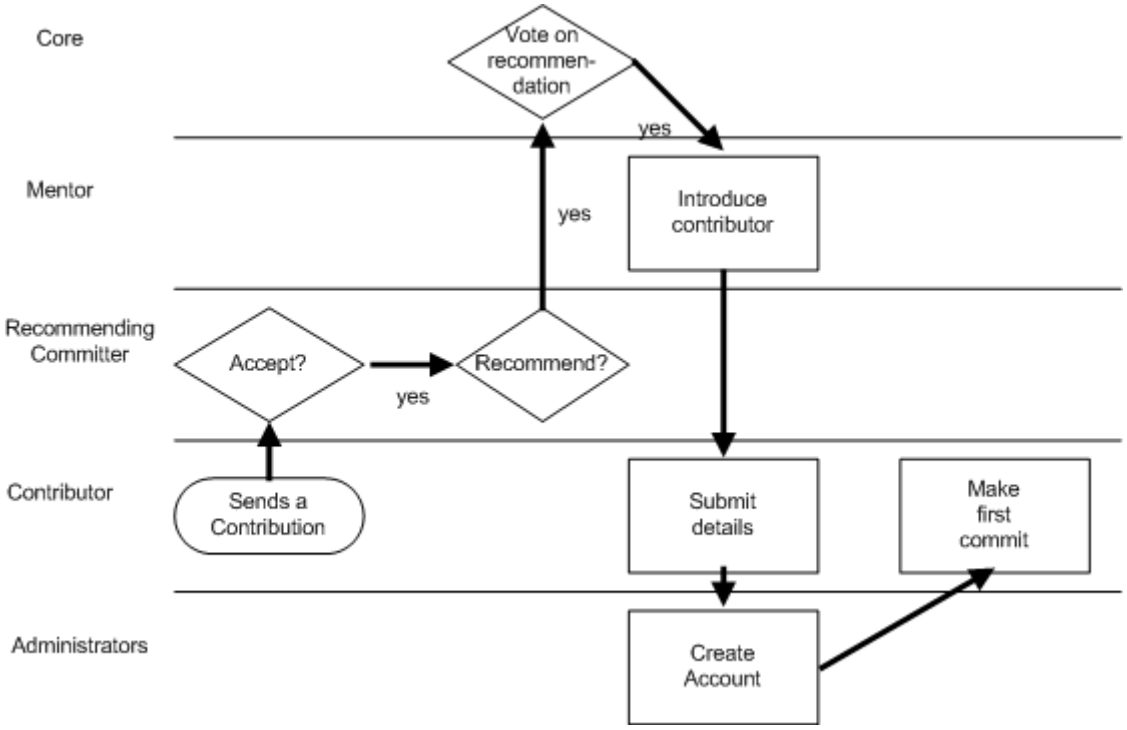

Рисунок 3. Процесс вкратце: добавление нового коммиттера

Когда участник отправляет фрагмент кода, принимающий коммиттер может предложить предоставить этому участнику права на коммит. Если он рекомендует это основной команде (Core Team), команда проводит голосование по этой рекомендации. Если голосование завершается в пользу предложения, новому коммиттеру назначается наставник, и новый коммиттер должен отправить свои данные администраторам для создания учётной записи. После этого новый коммиттер готов сделать свой первый коммит. По традиции, это делается путём добавления своего имени в список коммиттеров.

Напомним, что коммиттером считается тот, кто за последние 12 месяцев внёс изменения в код. Однако право на коммиты может быть отозвано только после 18 месяцев неактивности.

Однако не существует автоматических процедур для этого. Для действий, связанных с привилегиями коммитов, не вызванных временем, см. раздел 1.5.8.

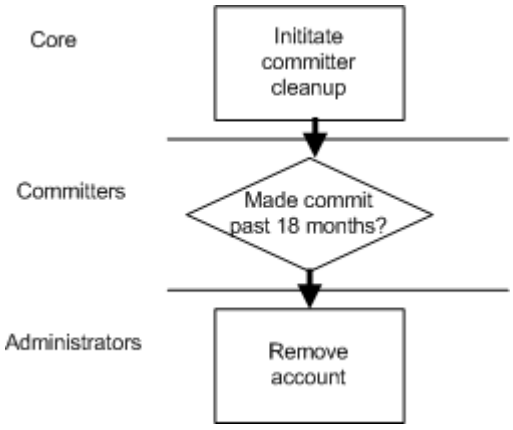

Рисунок 4. Процесс: удаление коммиттера

Когда Основная команда (Core Team) принимает решение очистить список коммиттеров, они проверяют, кто не делал коммитов за последние 18 месяцев. Коммиттеры, которые

этого не сделали, лишаются прав на коммит, и их учетные записи удаляются администраторами.

Также возможно для коммиттеров запросить отзыв их права на коммит, если по какой-то причине они больше не будут активно участвовать в проекте. В этом случае, право может быть восстановлено позже по запросу коммиттера.

Роли в этом процессе:

1. Основная команда (Core Team)
2. Участник (контрибьютор)
3. Коммиттер
4. Сопровождение
5. Наставник (Mentor)

Коммит кода

Добавление нового или изменённого кода — один из наиболее частых процессов в проекте FreeBSD и обычно происходит несколько раз в день. Фиксация кода может быть выполнена только "коммиттером". Коммиттеры фиксируют либо код, написанный ими самими, либо код, переданный им, либо код, отправленный через отчёт о проблеме.

Когда разработчик пишет нетривиальный код, он должен запросить рецензирование кода у сообщества. Это делается путём отправки письма в соответствующий список рассылки с просьбой о рецензировании. Перед отправкой кода на проверку разработчик должен убедиться, что он корректно компилируется со всем деревом исходного кода и что все соответствующие тесты выполняются. Это называется "предварительной проверкой перед коммитом". Когда получен вклад в виде кода, коммиттер должен просмотреть его и протестировать таким же образом.

Когда изменение фиксируется в части исходного кода, которая была получена от внешнего поставщика, сопровождающий должен убедиться, что патч передан обратно поставщику. Это соответствует философии открытого исходного кода и упрощает синхронизацию с внешними проектами, так как патчи не придётся применять заново при каждом новом выпуске.

После того как код был доступен для рецензирования и дальнейшие изменения не требуются, код вносится в ветку разработки -CURRENT. Если изменение применимо и для ветки -STABLE, или других веток, коммиттер устанавливает отсчёт времени для "слияния из текущей" ("MFC"). После того как пройдёт количество дней, выбранное коммиттером при установке MFC, автоматически будет отправлено письмо коммиттеру с напоминанием внести изменения в ветку -STABLE (а также, возможно, в ветки безопасности). В ветки безопасности следует сливать только критические изменения, связанные с безопасностью.

Откладывание коммита в -STABLE и другие ветки позволяет проводить "параллельную отладку", когда закоммиченный код тестируется на широком спектре конфигураций. Это приводит к тому, что изменения в -STABLE содержат меньше ошибок, что и даёт ветке её

название.

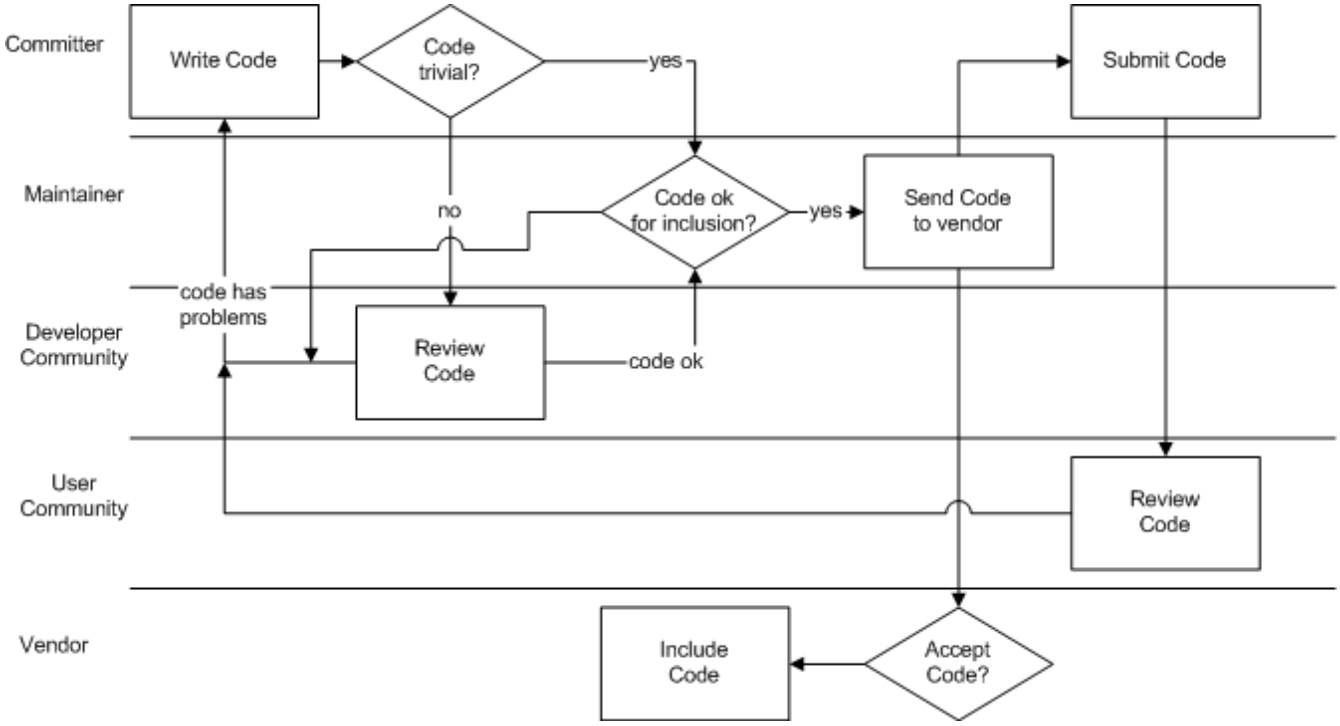

Рисунок 5. Процесс вкратце: коммиттер делает коммит кода

Когда коммиттер написал часть кода и хочет его закоммитить, он сначала должен определить, достаточно ли он тривиален, чтобы попасть в репозиторий без предварительной рецензии, или ему сначала следует сделать рецензию в сообществе разработчиков. Если код тривиален или был отрецензирован, и коммиттер не является сопровождающим, он должен проконсультироваться с сопровождающим перед тем, как продолжить. Если код предоставлен внешним поставщиком, сопровождающий должен создать патч, который отправляется обратно поставщику. Затем код коммитится и развертывается пользователями. Если они обнаружат проблемы с кодом, это будет сообщено, и коммиттер может вернуться к написанию патча. Если затронут поставщик, он может выбрать реализацию или игнорирование патча.

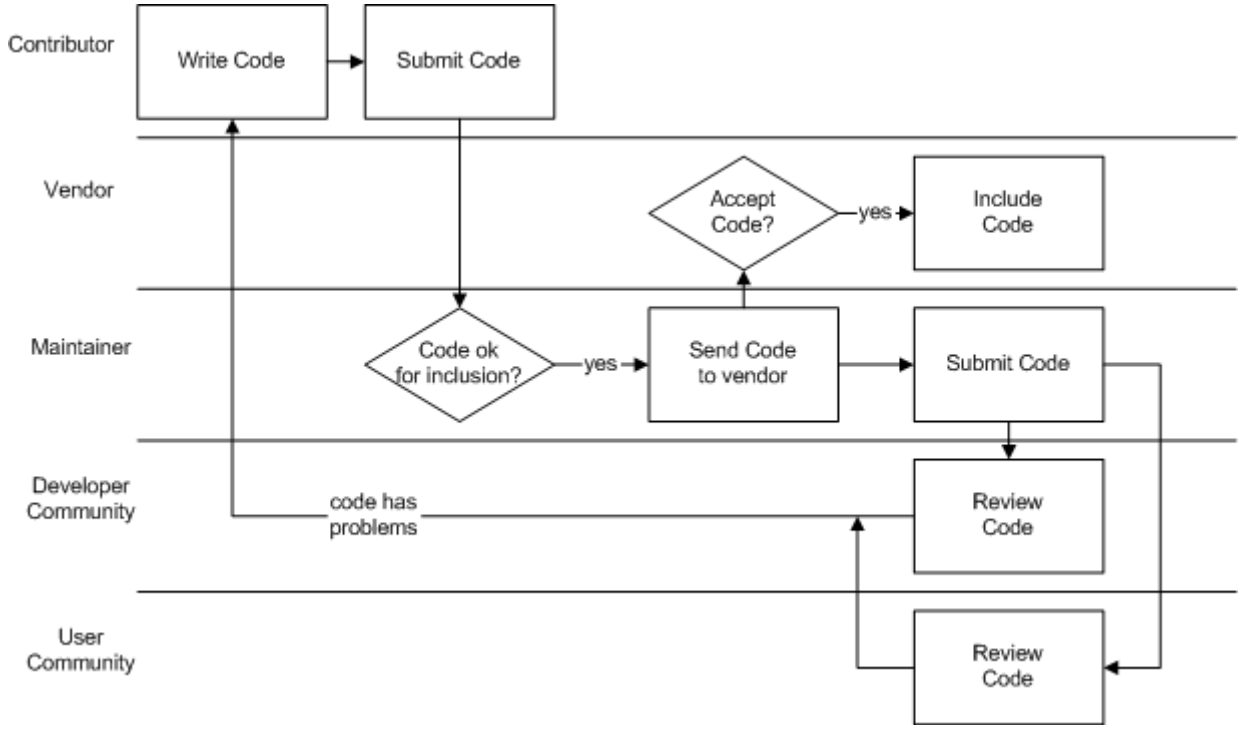

Рисунок 6. Процесс вкратце: Участник делает коммит кода

Разница, когда участник вносит код, заключается в том, что он отправляет код через интерфейс Bugzilla. Этот отчёт забирает сопровождающий, который просматривает код и делает ему коммит.

Роли, задействованные в этом процессе:

1. Коммиттер
2. Участник (контрибьютор)
3. Поставщик
4. Рецензенты

Выборы основной команды (Core Team)

Выборы Core Team проводятся не реже чем раз в два года. [1] Избираются девять участников Core Team. Новые выборы проводятся, если количество участников Core Team становится меньше семи. Новые выборы также могут быть проведены, если этого потребуют как минимум 1/3 активных коммиттеров.

Когда должны состояться выборы, Core Team объявляет об этом как минимум за 6 недель и назначает менеджера выборов для их проведения.

Только коммиттеры могут быть избраны в состав основной команды (Core Team). Кандидаты должны подать свои заявки как минимум за одну неделю до начала выборов, но могут уточнять свои заявления до начала голосования. Они представлены в списке кандидатов. При составлении своих предвыборных заявлений кандидаты должны ответить на несколько стандартных вопросов, предоставленных организатором выборов.

Во время выборов строго соблюдается правило, что коммиттер должен был сделать коммит

в течение последних 12 месяцев. Только эти коммиттеры имеют право голосовать.

При голосовании коммиттер может проголосовать один раз в поддержку до девяти номинантов. Голосование проводится в течение четырёх недель, с напоминаниями, публикуемыми в рассылке "developers", доступной всем коммиттерам.

Результаты выборов публикуются через неделю после их окончания, а новая основная команда вступает в должность через неделю после публикации результатов.

В случае ничьей при голосовании, это будет разрешено новыми, однозначно избранными членами ядра.

Голоса и заявления кандидатов архивируются, но архивы не являются общедоступными.

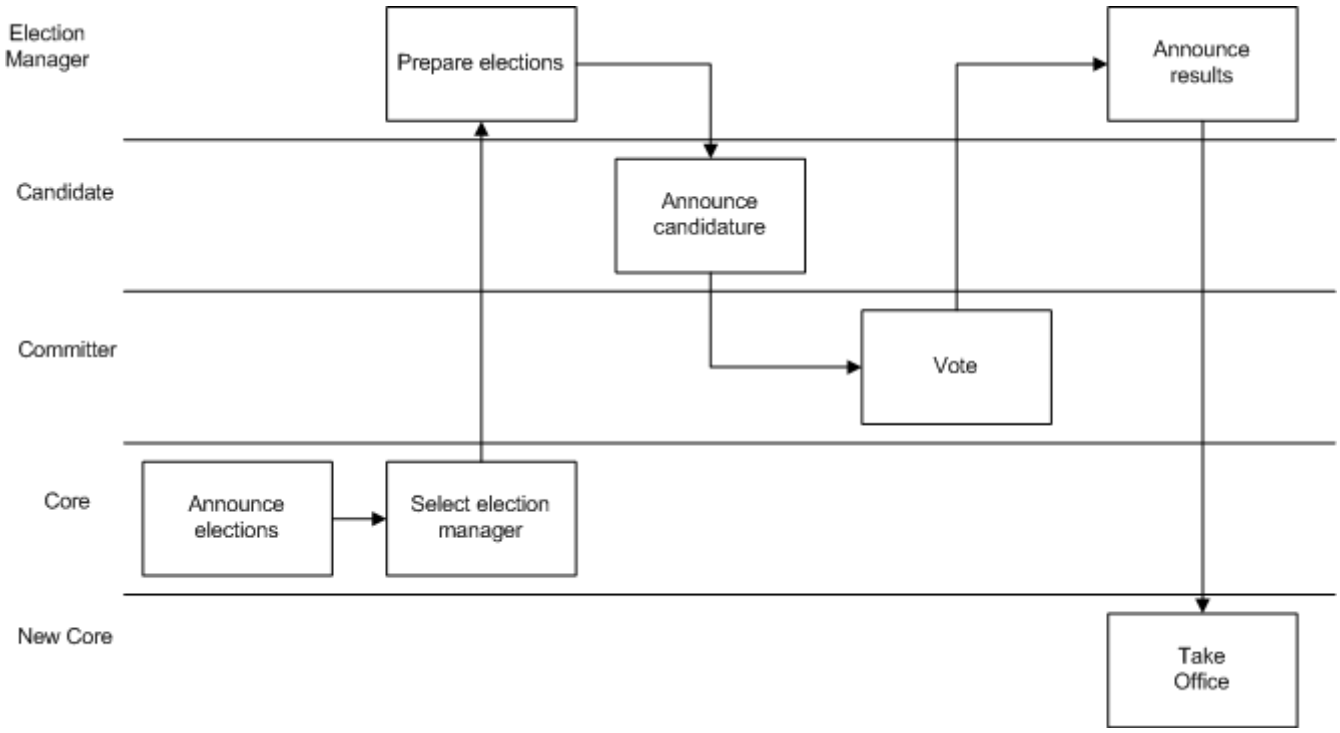

Рисунок 7. Процесс вкратце: Выборы основной команды (Core Team)

Core Team объявляет выборы и назначает руководителя выборов, который подготавливает процесс. Когда всё готово, кандидаты могут объявить о своей кандидатуре, представив заявления. Затем коммиттеры голосуют. После завершения голосования результаты выборов объявляются, и новая основная команда вступает в должность.

Ответственный за выборы Core Team:

- Основная команда (Core Team)
- Коммиттер
- Менеджер выборов

Разработка новых функций

В рамках проекта существуют подпроекты, работающие над новыми функциями. Эти проекты обычно выполняются одним человеком [Йоргенсен]. Каждый проект волен

организовывать разработку так, как считает нужным. Однако, когда проект объединяется с ветвью -CURRENT, он должен следовать руководствам проекта. Когда код хорошо протестирован в ветви -CURRENT и признан достаточно стабильным и актуальным для ветви -STABLE, он объединяется с ветвью -STABLE.

Требования проекта определяются пожеланиями разработчиков, запросами сообщества в виде прямых обращений по почте, отчётов о проблемах (Problem Reports), коммерческим финансированием разработки функциональности или вкладами научного сообщества. Пожелания, которые входят в зону ответственности разработчика, передаются этому разработчику, который расставляет приоритеты между запросом и своими собственными пожеланиями. Распространенный способ организации этого процесса — ведение списка задач (TODO-list), поддерживаемого проектом. Задачи, не входящие в чью-либо зону ответственности, собираются в списках TODO, пока кто-нибудь не возьмет на себя ответственность за их выполнение. Все запросы, их распределение и отслеживание обрабатываются с помощью инструмента Bugzilla.

Анализ требований происходит двумя способами. Поступившие запросы обсуждаются в почтовых рассылках, как в основном проекте, так и в подпроекте, к которому относится запрос или который создается этим запросом. Кроме того, отдельные разработчики подпроекта оценивают осуществимость запросов и определяют приоритеты между ними. Помимо архивов обсуждений, на этом этапе не создается никаких результатов, которые включаются в основной проект.

Поскольку запросы приоритизируются отдельными разработчиками на основе того, что они считают интересным, необходимым или за что им платят, отсутствует общая стратегия или приоритезация того, какие запросы считать требованиями, и как контролировать их корректную реализацию. Однако большинство разработчиков разделяют общее видение того, какие вопросы являются более важными, и они могут запросить рекомендации у команды инженеров по выпуску релизов.

Фаза проверки проекта состоит из двух этапов. Перед внесением кода в текущую ветку разработчики запрашивают рецензирование своего кода коллегами. Это рецензирование в основном проводится с помощью функционального тестирования, но также важна проверка кода. Когда код внесён в ветку, проводится более широкое функциональное тестирование, которое может привести к дополнительной проверке кода и отладке, если код ведёт себя не так, как ожидалось. Эта вторая форма проверки может рассматриваться как структурная верификация. Хотя сами подпроекты могут писать формальные тесты, такие как модульные тесты, они обычно не собираются основным проектом и чаще всего удаляются перед внесением кода в текущую ветку. [2]

6.2. Сопровождение

Для проекта полезно, чтобы за каждую область исходного кода отвечал хотя бы один человек, который хорошо её знает. Некоторые части кода имеют назначенных сопровождающих. Другие имеют фактических сопровождающих, а некоторые части системы не имеют сопровождающих. Сопровождающий обычно является участником подпроекта, который написал и интегрировал код, или тем, кто портировал его с платформы, для которой он был написан. [3] Задача сопровождающего — убедиться, что код

синхронизирован с проектом, из которого он получен, если это сторонний код, а также применять патчи, предоставленные сообществом, или исправлять обнаруженные проблемы.

Основной объем работы, вкладываемый в проект FreeBSD, связан с сопровождением. [Jørgensen] предоставляет схему, показывающую жизненный цикл изменений.

Модель Йоргенссена для интеграции изменений

Этап	**Следующий, если успешно**	**Следующий, если неудачно**
программирование	рецензирование	
рецензирование	предварительная проверка перед коммитом	программирование
предварительная проверка перед коммитом	релиз для разработки	программирование
релиз для разработки	параллельная отладка	программирование
параллельная отладка	релиз для производства	программирование
релиз для производства		программирование

Здесь "релиз для разработки" относится к ветке -CURRENT, а "релиз для производства" — к ветке -STABLE. "Предварительная проверка перед коммитом" — это функциональное тестирование, проводимое коллегами-разработчиками по запросу или для проверки кода с целью определения состояния подпроекта. "Параллельная отладка" — это функциональное тестирование, которое может вызвать дополнительный обзор и отладку, когда код включён в ветку -CURRENT.

На момент написания этого документа в проекте было 269 коммиттеров. Когда они вносят изменения в ветку, это создает новый выпуск. Очень часто пользователи в сообществе отслеживают определенную ветку. Мгновенное появление нового выпуска делает изменения широко доступными сразу же и позволяет быстро получать отзывы от сообщества. Это также дает сообществу ожидаемое время реакции на проблемы, которые важны для них. Это делает сообщество более вовлеченным, что, в свою очередь, позволяет получать больше и лучше отзывов, что снова стимулирует больше сопровождения и в конечном итоге должно создать лучший продукт.

Прежде чем вносить изменения в код в частях дерева, история которых неизвестна коммиттеру, коммиттер обязан прочитать журналы коммитов, чтобы понять, почему определённые функции реализованы именно так, и избежать ошибок, которые уже были обдуманы или исправлены ранее.

6.3. Сообщение о проблеме

До FreeBSD 10 в FreeBSD входил инструмент для отправки отчётов о проблемах под названием `send-pr`. Проблемы включают отчёты об ошибках, запросы функций, улучшения функций и уведомления о новых версиях внешнего программного обеспечения, включённого в проект. Хотя `send-pr` доступен, пользователям и разработчикам

рекомендуется отправлять проблемы, используя нашу форму отчёта о проблемах.

Отчёты о проблемах отправляются на электронный адрес, откуда они попадают в базу данных сопровождения отчётов о проблемах. Исправитель ошибок (Bugbuster) классифицирует проблему и направляет её соответствующей группе или сопровождающему в рамках проекта. После того, как кто-то берёт ответственность за отчёт, он анализируется. Этот анализ включает проверку проблемы и разработку решения. Часто требуется обратная связь от автора отчёта или даже от сообщества FreeBSD. Как только создаётся патч для устранения проблемы, автора отчёта могут попросить его протестировать. В итоге рабочий патч интегрируется в проект и, если необходимо, документируется. Затем он проходит стандартный цикл сопровождения, как описано в разделе Сопровождение. Отчёт о проблеме может находиться в следующих состояниях: открыт, анализируется, ожидает обратной связи, исправлен патчем, отложен и закрыт. Состояние "отложен" используется, когда дальнейшее продвижение невозможно из-за недостатка информации или когда задача требует столько работы, что в данный момент никто над ней не работает.

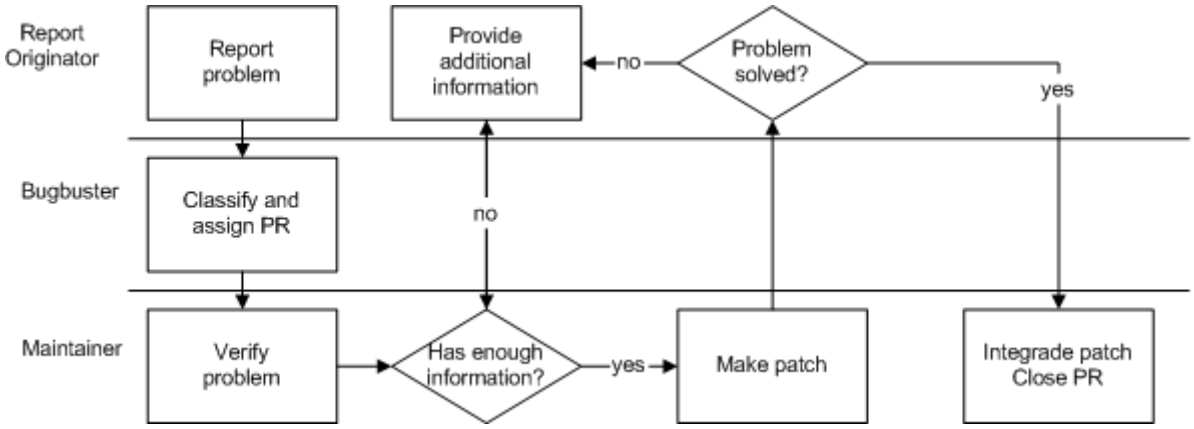

Рисунок 8. Сводка процесса: сообщение о проблеме

Проблема сообщается автором отчёта. Затем она классифицируется ответственным за обработку ошибок и передаётся соответствующему сопровождающему. Он проверяет проблему и обсуждает её с автором отчёта до тех пор, пока не будет собрано достаточно информации для создания рабочего исправления. Это исправление затем фиксируется, и отчёт о проблеме закрывается.

Роли, включенные в этот процесс:

1. Инициатор отчёта
2. Сопровождение
3. Исправитель ошибок (Bugbuster)

[FreeBSD].

Реагирование на неправильное поведение

[FreeBSD] содержит ряд правил, которым должны следовать коммиттеры. Однако случается, что эти правила нарушаются. Следующие правила существуют для того, чтобы можно было реагировать на неподобающее поведение. Они определяют, какие действия приведут к

приостановке привилегий коммиттера на тот или иной срок.

- Совершение коммитов во время заморозки кода без одобрения команды Release Engineering — 2 дня
- Коммит изменений в ветку безопасности без одобрения - 2 дня
- Войны коммитов — 5 дней для всех участвующих сторон
- Невежливое или неподобающее поведение — 5 дней

Для эффективности приостановок любой член основной команды (Core Team) может применить приостановку до обсуждения на почтовой рассылке "core". Повторные нарушители могут, при 2/3 голосов от основной команды, получить более строгие наказания, включая постоянное лишение прав на коммиты. (Однако последнее всегда рассматривается как крайняя мера из-за присущей ему склонности вызывать споры.) Все приостановки публикуются в почтовой рассылке "developers", доступной только коммиттерам.

Важно, что вас не могут приостановить за технические ошибки. Все наказания связаны с нарушением социального этикета.

Роли, участвующие в этом процессе:

- Основная команда (Core Team)
- Коммиттер

6.4. Выпуск релизов

Проект FreeBSD имеет команду инженеров по выпуску релизов с главным инженером, который отвечает за создание релизов FreeBSD для распространения среди пользователей через интернет или продажи в розничных магазинах. Поскольку FreeBSD доступна на нескольких платформах, а релизы для различных архитектур выпускаются одновременно, в команде есть ответственный за каждую архитектуру. Также в команде есть роли, отвечающие за координацию усилий по обеспечению качества, сборку набора пакетов и актуализацию документации. Под инженером по выпуску релизов подразумевается представитель команды инженеров по выпуску релизов.

Когда готовится выпуск релиза, проект FreeBSD несколько меняет свою структуру. Составляется график выпуска, включающий заморозку функциональности и кода, выпуск промежуточных релизов и финального релиза. Заморозка функциональности означает, что новые функции не могут быть добавлены в ветку без явного согласия инженеров релиза. Заморозка кода означает, что изменения в коде (например, исправления ошибок) не могут быть добавлены без явного согласия инженеров релиза. Этот процесс заморозки функциональности и кода известен как стабилизация. В процессе выпуска релиза инженер релиза имеет полномочия откатываться к более старым версиям кода и, таким образом, "отменять" изменения, если они сочтут, что эти изменения не подходят для включения в релиз.

Существует три различных вида выпусков:

1. .0 выпуски являются первым релизом основной версии. Они ветвятся от ветки -CURRENT и имеют значительно более длительный цикл разработки из-за нестабильного характера ветки -CURRENT
2. .X релизы — это релизы ветки -STABLE. Они запланированы к выходу каждые 4 месяца.
3. .X.Y — это выпуски с исправлениями уязвимостей, следующие за веткой .X. Они выходят только тогда, когда с момента последнего выпуска в этой ветке было объединено достаточное количество исправлений уязвимостей. Новые функции включаются редко, а команда безопасности участвует в этих выпусках гораздо активнее, чем в обычных.

Для выпусков ветки -STABLE процесс выпуска начинается за 45 дней до предполагаемой даты релиза. В течение первой фазы, первых 15 дней, разработчики переносят изменения из -CURRENT, которые они хотят включить в релиз, в ветку выпуска. По окончании этого периода код входит в 15-дневный период заморозки, в течение которого допускаются только исправления ошибок, обновления документации, исправления, связанные с безопасностью, и незначительные изменения драйверов устройств. Эти изменения должны быть предварительно одобрены инженером выпуска. В начале последнего 15-дневного периода создается кандидат на выпуск для широкого тестирования. В этот период вероятность внесения изменений снижается, за исключением важных исправлений ошибок и обновлений безопасности. В этот заключительный период все выпуски считаются кандидатами на выпуск. По завершении процесса выпуска создается релиз с новым номером версии, включая бинарные дистрибутивы на веб-сайтах и создание образов CD-ROM. Однако релиз не считается «действительно выпущенным» до тех пор, пока на список рассылки freebsd-announce не будет отправлено сообщение, подписанное с помощью Pretty Good Privacy, в котором явно указано, что релиз состоялся; все, что обозначено как «релиз» до этого момента, может находиться в процессе доработки и изменяться до отправки PGP-подписанного сообщения. [4].

Версии ветки -CURRENT (то есть все версии, оканчивающиеся на ".0"), очень похожи, но с вдвое большим временным промежутком. Процесс начинается за 8 недель до выпуска с объявления графика релиза. Через две недели после начала процесса выпуска вводится заморозка функциональности, и оптимизация производительности должна быть сведена к минимуму. За четыре недели до выпуска становится доступна официальная бета-версия. За две недели до выпуска код официально ветвится в новую версию. Этой версии присваивается статус релиз-кандидата, и, как и в случае с разработкой -STABLE, заморозка кода релиз-кандидата ужесточается. Однако разработка на основной ветке разработки может продолжаться. За исключением этих различий, процессы разработки релизов схожи.

*.0 выпуски выделяются в отдельную ветку и ориентированы в основном на ранних последователей. Затем ветка проходит период стабилизации, и только после того, как Команда разработки релизов решит, что требования к стабильности выполнены, ветка становится -STABLE, а -CURRENT переключается на следующую мажорную версию. Хотя в большинстве случаев это происходило с версиями *.1, это не является обязательным требованием.

Большинство выпусков происходит по достижении даты, которая считается достаточно отдалённой от предыдущего выпуска. Установлена цель выпускать основные версии каждые 18 месяцев, а промежуточные — каждые 4 месяца. Сообщество пользователей чётко дало понять, что безопасность и стабильность не могут быть принесены в жертву из-за

самостоятельно установленных сроков и целевых дат выпуска. Чтобы задержки не становились слишком длинными в вопросах безопасности и стабильности, требуется дополнительная дисциплина при внесении изменений в -STABLE.

1. Сделать график выпуска релизов
2. Заморозить функциональность
3. Заморозка кода
4. Создать ветку
5. Кандидат на выпуск
6. Стабилизировать выпуск (при необходимости вернуться к предыдущему шагу; когда выпуск считается стабильным, перейти к следующему шагу)
7. Собрать пакеты
8. Предупредить сайты-зеркала
9. Опубликовать выпуск

[FreeBSD]

[1] Первые выборы Core Team состоялись в сентябре 2000 года

[2] Однако всё больше тестов выполняется при сборке системы (make world). Эти тесты являются очень новым дополнением, и систематическая структура для них ещё не создана.

[3] sendmail и named — примеры кода, который был объединён с других платформ.

[4] Многие коммерческие поставщики используют эти образы для создания CD-ROM, которые продаются в розничных магазинах.

Chapter 7. Инструменты

Основные инструменты поддержки процесса разработки — это Bugzilla, Mailman и OpenSSH. Это инструменты, разработанные сторонними организациями, которые широко используются в мире открытого исходного кода.

7.1. Git

Git — это система для управления несколькими версиями текстовых файлов, отслеживания внесённых изменений, их авторов и причин. Проект хранится в «репозитории», а разные версии считаются разными «ветками».

7.2. Bugzilla

Bugzilla — это база данных для сопровождения, состоящая из набора инструментов для отслеживания ошибок на центральном сайте. Она поддерживает процесс отслеживания ошибок, включая отправку и обработку ошибок, а также запросы и обновление базы данных, а также редактирование отчётов об ошибках. Проект использует веб-интерфейс для отправки "Отчётов о проблемах" на центральный сервер Bugzilla проекта. У коммиттеров также есть веб- и командные клиенты.

7.3. Mailman

Mailman - это программа, которая автоматизирует управление почтовыми рассылками. Проект FreeBSD использует ее для ведения 16 общих рассылок, 60 технических рассылок, 4 ограниченных рассылок и 5 рассылок с логами коммитов Git. Она также используется для многих почтовых рассылок, созданных и используемых другими людьми и проектами в сообществе FreeBSD. Общие рассылки предназначены для широкой публики, технические рассылки в основном предназначены для разработки определенных областей интересов, а закрытые рассылки используются для внутренней коммуникации, не предназначенной для широкой публики. Большая часть всей коммуникации в проекте проходит через эти 85 рассылок [FreeBSD, Приложение C].

7.4. Pretty Good Privacy

Pretty Good Privacy, более известный как PGP, — это криптосистема, использующая архитектуру открытого ключа, чтобы позволить пользователям подписывать и/или шифровать информацию цифровой подписью для обеспечения безопасной связи между двумя сторонами. Подпись используется при отправке информации множеству получателей, позволяя им убедиться, что информация не была изменена до того, как они её получили. В проекте FreeBSD это основной способ убедиться, что информация была написана тем, кто утверждает, что её создал, и не была изменена в процессе передачи.

7.5. Secure Shell (SSH)

Secure Shell - это стандарт безопасного входа в удалённую систему и выполнения команд на ней. Он позволяет устанавливать и защищать другие соединения, называемые туннелями, между двумя взаимодействующими системами. Этот стандарт существует в двух основных версиях, и только версия два используется в проекте FreeBSD. Наиболее распространённая реализация стандарта - OpenSSH, которая входит в основную дистрибуцию проекта. Поскольку её исходный код обновляется чаще, чем выпуски FreeBSD, последняя версия также доступна в дереве портов.

Chapter 8. Подпроекты

Подпроекты создаются для уменьшения объема коммуникации, необходимой для координации группы разработчиков. Когда проблемная область достаточно изолирована, большая часть коммуникации происходит внутри группы, сосредоточенной на проблеме, что требует меньше общения с другими группами по сравнению с ситуацией, когда группа не изолирована.

8.1. Подпроект Ports

"Порт" — это набор метаданных и патчей, необходимых для загрузки, компиляции и корректной установки внешнего программного обеспечения в системе FreeBSD. Количество портов растёт с огромной скоростью, как показано на следующем рисунке.

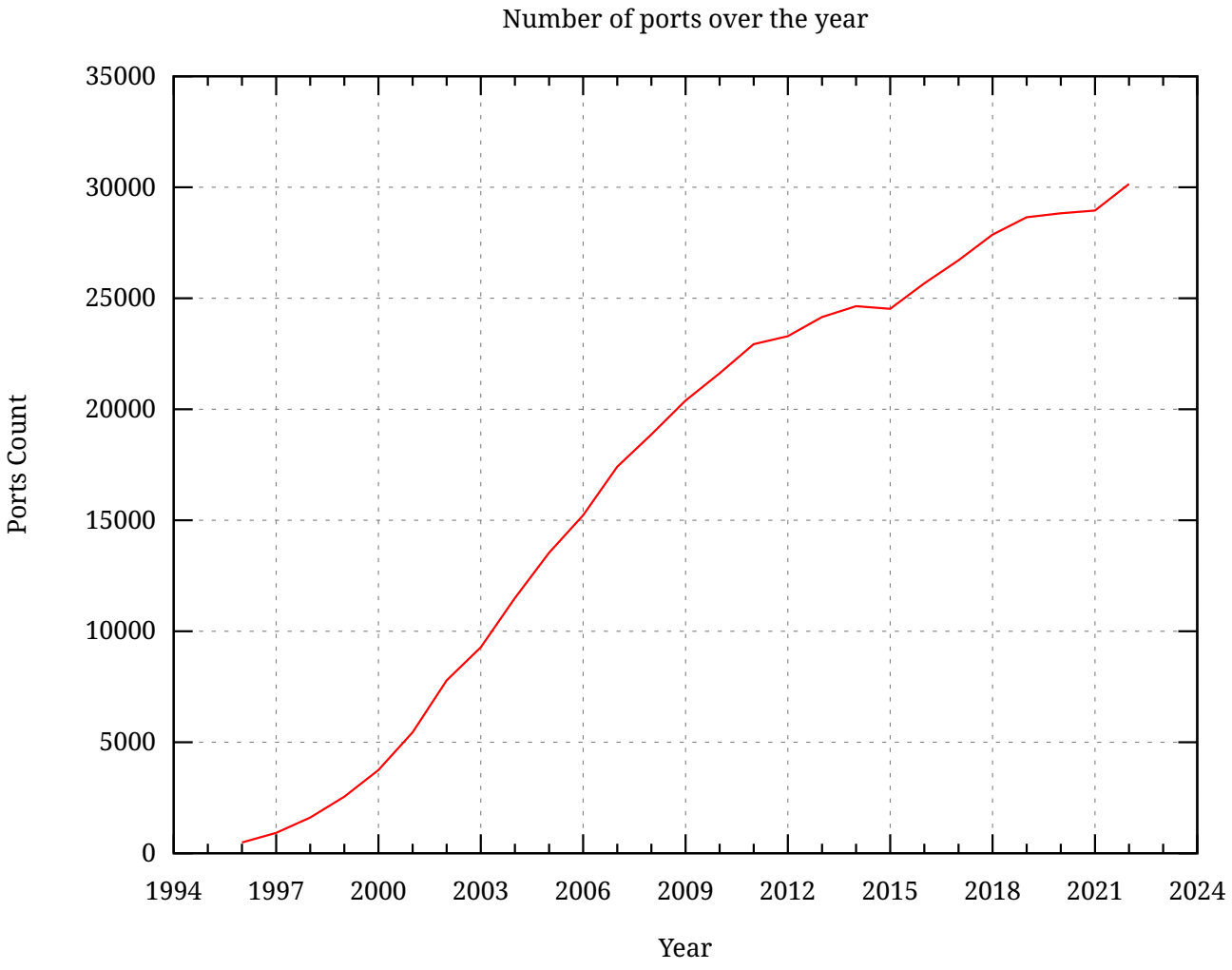

Рисунок 9. Количество портов, добавленных между 1995 и 2022 годами

image::portsstatus.svg показывает количество портов, доступных для FreeBSD в период с 1995 по 2022 год. Похоже, что кривая сначала росла экспоненциально, а затем с середины 2001 до середины 2007 года росла линейно со скоростью около 2000 портов/год, после чего скорость роста снизилась.

Поскольку внешнее программное обеспечение, описываемое портом, часто находится в

стадии активной разработки, объем работы, необходимой для поддержки портов, уже велик и продолжает расти. Это привело к тому, что часть проекта FreeBSD, связанная с портами, получила более самостоятельную структуру и все больше становится подпроектом проекта FreeBSD.

Порты имеют свою собственную основную команду с Менеджером Портов во главе, и эта команда может назначать коммиттеров без одобрения Основной команды FreeBSD (Core Team). В отличие от проекта FreeBSD, где активное сопровождение часто вознаграждается правом коммита, подпроект портов включает множество активных сопровождающих, не являющихся коммиттерами.

В отличие от основного проекта, дерево портов не разветвляется. Каждый выпуск FreeBSD следует текущей коллекции портов, что обеспечивает доступ к обновлённой информации о том, где найти программы и как их собрать. Однако это означает, что порт, зависящий от системы, может требовать изменений в зависимости от версии FreeBSD, на которой он запущен.

С неразветвлённым репозиторием портов невозможно гарантировать, что любой порт будет работать на чём-либо, кроме -CURRENT и -STABLE, в частности на старых, минорных выпусках. Для этого нет ни инфраструктуры, ни времени волонтёров.

Команды, зависящие от Ports, такие как команда выпуска релизов, для эффективности коммуникации имеют своих собственных представителей по портам.

8.2. Проект документации FreeBSD

Проект документации FreeBSD был начат в январе 1995 года. От первоначальной группы, состоявшей из руководителя проекта, четырёх руководителей команд и 16 участников, сейчас общее число коммиттеров достигло 44. Список рассылки документации насчитывает чуть менее 300 участников, что указывает на довольно большое сообщество вокруг него.

Цель проекта Документации — предоставить качественную и полезную документацию проекта FreeBSD, чтобы новые пользователи могли легче освоить систему, а также подробно описать расширенные функции для пользователей.

Основные задачи проекта Documentation — работа над текущими проектами в "Наборе документации FreeBSD" и перевод документации на другие языки.

Как и проект FreeBSD, документация разделена на те же ветви. Это сделано для того, чтобы для каждой версии всегда была обновлённая документация. В ветвях безопасности исправляются только ошибки в документации.

Как и подпроект ports, проект Documentation может назначать коммиттеров документации без одобрения основной команды FreeBSD (Core Team). [FreeBSD].

Проект документации включает в себя вводное руководство. Оно используется как для ознакомления новых участников проекта со стандартными инструментами и синтаксисом, так и в качестве справочника при работе над проектом.

Список литературы

[Brooks, 1995] Фредерик П. Брукс. Авторское право © 1975, 1995 Pearson Education Limited. 0201835959. Addison-Wesley Pub Co. Мифический человекомесяц. Эссе о программной инженерии, юбилейное издание (2-е издание).

[Saers, 2003] Никлас Саерс. Авторское право © 2003. Модель проекта для FreeBSD. Кандидатская диссертация. http://niklas.saers.com/thesis.

[Йоргенсен, 2001] Нильс Йоргенсен. Copyright © 2001. *Putting it All in the Trunk. Incremental Software Development in the FreeBSD Open Source Project*. http://www.dat.ruc.dk/~nielsj/research/papers/freebsd.pdf.

[PMI, 2000] Институт управления проектами. Copyright © 1996, 2000 Институт управления проектами. 1-880410-23-0. Институт управления проектами. Ньютаун Сквер, Пенсильвания, США. PMBOK Guide. A Guide to the Project Management Body of Knowledge (Руководство PMBOK. Руководство к своду знаний по управлению проектами), издание 2000 года.

[FreeBSD, 2000A] Copyright © 2002 The FreeBSD Project. Core Bylaws. https://www.freebsd.org/internal/bylaws/.

[FreeBSD, 2002A] Copyright © 2002 The FreeBSD Documentation Project. Руководство FreeBSD для разработчиков. Руководство FreeBSD для разработчиков.

[FreeBSD, 2002B] Copyright © 2002 Проект FreeBSD. Выборы состава основной команды (Core Team) 2002. http://election.uk.freebsd.org/candidates.html.

[FreeBSD, 2002C] Даг-Эрлинг Смёрграв и Хитен Пандья. Copyright © 2002 The FreeBSD Documentation Project. The FreeBSD Documentation Project. Рекомендации по работе с сообщениями о проблемах. Рекомендации по работе с сообщениями о проблемах.

[FreeBSD, 2002D] Даг-Эрлинг Смёрграв. Copyright © 2002 Проект документации FreeBSD. Проект документации FreeBSD. Составление сообщений о проблеме во FreeBSD. Составление сообщений о проблеме во FreeBSD.

[FreeBSD, 2001] Copyright © 2001 The FreeBSD Documentation Project. The FreeBSD Documentation Project. Справочник коммиттера. Справочник коммиттера.

[FreeBSD, 2002E] Мюррей Стокли. Copyright © 2002 The FreeBSD Documentation Project. Проект документации FreeBSD. Подготовка релизов FreeBSD. Подготовка релизов FreeBSD.

[FreeBSD, 2003A] Проект документации FreeBSD. Руководство FreeBSD. Руководство FreeBSD.

[FreeBSD, 2002F] Copyright © 2002 The FreeBSD Documentation Project. Проект документации FreeBSD. Участники проекта FreeBSD. Участники проекта FreeBSD.

[FreeBSD, 2002G] Copyright © 2002 The FreeBSD Project. The FreeBSD Project. Выборы состава основной команды (Core Team) 2002. http://election.uk.freebsd.org.

[FreeBSD, 2002H] Copyright © 2002 The FreeBSD Project. The FreeBSD Project. Политика

истечения срока действия битов коммитов. 2002/04/06 15:35:30. https://www.freebsd.org/internal/expire-bits/.

[FreeBSD, 2002I] Copyright © 2002 The FreeBSD Project. The FreeBSD Project. Процедура создания нового аккаунта. 2002/08/19 17:11:27. https://www.freebsd.org/internal/new-account/.

[FreeBSD, 2003B] Copyright © 2002 The FreeBSD Documentation Project. The FreeBSD Documentation Project. Устав команды разработчиков документации FreeBSD. 2003/03/16 12:17. https://www.freebsd.org/internal/doceng/.

[Lehey, 2002] Грег Лехи. Copyright © 2002 Грег Лехи. Грег Лехи. Two years in the trenches. The evolution of a software project (Два года в окопах. Эволюция программного проекта). http://www.lemis.com/grog/In-the-trenches.pdf.