Программная инженерия

Кодирование (разработка) ПО:
управление рисками

Вопросы

- Управление рисками
  - Определения
  - Основные виды и источники рисков в программных проектах
  - Методы анализа и управления рисками
- Контроль качества
  - Стандарты программирования
  - *Метрики для исходного кода*
- Организация и инструментальные средства коллективной разработки

Определения

Характеристики

РИСКИ

Риски vs. проблемы

- Риск – неопределенное событие или условие, наступление которого отрицательно или положительно сказывается на целях проекта
- Риск это проблема, которая еще не возникла, а проблема – это риск, который материализовался

Риски: характеристики – 1/2

- Причина или источник
  - Явление, обстоятельство обусловливающее наступление риска
- Симптомы риска
  - Указание на то, что событие риска произошло или вот-вот произойдет
- Последствия риска
  - Проблема или возможность, которая может реализоваться в проекте в результате произошедшего риска

Риски: характеристики – 2/2

- Влияние риска
  - Влияние реализовавшегося риска на возможность достижения целей проекта. Воздействие обычно касается стоимости, графика и технических характеристик разрабатываемого продукта
  - Многие риски происходят частично и оказывают соразмерное отрицательное или положительное воздействие на проект

Определения

Этапы

Методы

УПРАВЛЕНИЕ РИСКАМИ: ТЕОРИЯ

Управление рисками – 1/2

- Управление рисками это процесс принятия и выполнения управленческих решений, направленных на
  - снижение вероятности возникновения неблагоприятного результата и
  - минимизацию возможных потерь, вызванных его реализацией
- В экономике: управление рисками базируется на:
  - Полезности
  - Регрессии
  - Диверсификации

Управление рисками – 2/2

- Полезности
  - Способность удовлетворять какую-нибудь человеческую потребность
- Регрессия
  - Частный случай ошибки селекции, когда группы отбираются на основе крайних показателей (сдвиг среднего значения сильно зависит от «краев» выборки)
- Диверсификация
  - Расширение ассортимента производимых товаров и/или услуг

Этапы управления рисками – 1/2

- Выявление риска:
  - Собственно выявление
  - Оценка вероятности его реализации
  - Оценка масштаба последствий и определение (максимального) возможного убытка
- Выбор методов и инструментов управления выявленным риском
- Разработка риск-стратегии с целью снижения вероятности реализации риска и минимизации возможных негативных последствий

Этапы управления рисками – 2/2

- Реализация риск-стратегии
- Оценка достигнутых результатов и корректировка риск-стратегии

Методы анализа рисков

- Виды (этапы?):
  - Качественный – определение факторов, областей и видов риска
    - Осуществляется экспертным путём на основе опыта работы по данному направлению
  - Количественный – означивание количественных характеристик рисков
    - Дает возможность численно определить возможный объём потерь по каждому виду риска

Методы количественного анализа – 1/2

- Аналогий – использование данных по другим предприятиям (случаям, проектам)
- Статистический – основывается на изучении имеющейся статистики
  - Аналогий vs. Статистический
- Экспертный – сбор мнений квалифицированных специалистов

  ? «А судьи кто?»

  ?

Методы количественного анализа – 2/2

- Моделирования (имитационного) – позволяет оценить развитие ситуации по модели
  - Может быть основой СППР
  - Как построить модель?
- Смеси методов

Методы управления рисками

- Базовыми методами управления являются:
  - ~~Игнорирование~~
  - Отказ от риска
  - Снижение
    - Профилактика или диверсификация
  - Передача
    - Аутсорсинг
  - Принятие
    - Формирование резервов или запасов
  - ~~Скрытие (улыбаемся и машем)~~

Методы управления рисками

- 5.5 Методы снижения риска.
  - Наиболее распространённым способом борьбы с риском в инвестиционных проектах являются:
  - а) распределение риска;
  - б) страхование риска;
  - в) резервирование средств;
  - г) использование метода частных рисков.
  - Распределение риска осуществляется в процессе подготовки плана проекта и контрактных документов. Как правило, ответственность за конкретный риск возлагают на ту сторону, по чьей вине или в зоне чьей

Риски в разработке ПО

Примеры

УПРАВЛЕНИЕ РИСКАМИ: БЛИЖЕ К ДЕЛУ

Риски в разработке ПО – 1/3

- Связанные с требованиями
  - Изменение требований в процессе разработки
  - Ошибка в оценках (условий, требуемых характеристик продукта, etc.)
- Продукт не прошел приемку у заказчика: сделали не то
  - Сменился бизнес
  - Сменились люди у заказчика
  - «Потеря в канале» (разработчики архитектуры не прочли ТЗ, кодеры не прочли эскизный проект)

Риски в разработке ПО – 2/3

- Срыв сроков
  - «Благодаря своей неощутимой природе и уникальности программного обеспечения, процесс разработки ПО сложно оценить и расписать» (с) Гениальный [российский] программист
- Социальные риски
  - Уход ключевых сотрудников (групп)
  - Набор новых

Риски в разработке ПО – 3/3

- Не успели / не смогли
  - Низкая продуктивность
  - «Синдром студента»
  - Недооценка времени на развертывание / внедрение / отладку
- Технические риски
  - Сложность предметной области
  - Новые технологии / архитектуры

Пример управления: изменение ТЗ

- Согласованность в работке заказчики и исполнителя (наличие горизонтальных связей)
- Правильное документирование стадий:
  - ТЗ разбивается на CR и DR
  - Архитектура
- Работающие процедуры управления изменениями
- Приоритетизация
- Разбиение проекта на фазы

Пример управления: ошибка в оценке

- «Правильные» люди в команде проектирования и разработки
- Улучшение качества требований
- Использование проверенных методологий и/или инструментов
- Обучение персонала
- Прототипирование

Пример управления: сделали не то

- Квалифицированный менеджер проекта
- Постоянный контакт с заказчиком (прототипирование, документация, демонстрации, тестовая обкатка)
- Доступность аналитиков на протяжении проекта
- Улучшение качества проектной документации (в том числе ТЗ)

Пример управления: не успели

- Квалифицированный менеджер проекта
- Вовлечение команды в процесс планирования и оценки
- Экстим: экстремальное программирование и семинары
- Резюмируя:
  - Уменьшить длину шагов
  - Увеличить «прозрачность» прогресса

Пример управления: шмена шоштава

- Документирование!
- Взаимодействие сотрудников
  - Экстрим: парное программирование
- Сплочение коллектива
- Предусмотренное обучение новых сотрудников
- Наличие в коллективе специалистов разного уровня подготовки
- NB! Выделение лидеров и ключевых разработчиков сначала увеличивает производительность, затем тормозит и блокирует развитие

«Так что же делать?» – 1/2

- На этапе планирования:
  - Найти критические пути
  - Выделить время и ресурсы на нивелирование негативных влияний:
    - «Хвост по частям»
    - Отдельные «пустые» места
  - Уменьшить длину непрерывных шагов
  - Предусмотреть контроль выполнения

«Так что же делать?» – 2/2

- На этапе выполнения:
  - Контроль
  - Обнаружение симптомов
  - Реагирование

Стандарты

- ГОСТ Р ИСО/МЭК 31010—2011 – Менеджмент риска. Методы оценки риска (ISO/IEC 31010:2009)

Стандарты программирования

КОНТРОЛЬ КАЧЕСТВА ПО

Стандарты программирования: что

- Под стандартом программирования понимают свод правил, описывающих требования к:
  - Оформлению кода программ
  - Типовым программным решениям
  - Организации кодирования и отладки
  - Организации совместной работы
  - Прочие требования

Стандарты программирования: свойства

- Не являются всеобщими. Типичные источники:
  - Распространённой печатная работе по ЯП
    - Пример: Стандарт кодирования на языке Си K&R – описан Керниганом и Ричи в их книге
  - Широко применяемые библиотеки и их API
  - Ведущие компании-разработчики ПО (MS, Sun, etc.)
  - Внутренние документы компаний или консорциумов
- Их нарушения «очевидны и видны невооруженным взглядом»:
  - Используются для анализа кода и проектов
  - Все же есть некоторые общепринятые правила, с применением которых соглашаются даже те, кто их принципиально не соблюдает сам
- Зависят от языка программирования

Стандарты программирования: зачем – 1/2

- Качество кода
  - Одинаковое решение однотипных задач → ясность кода → упрощает сопровождение
- Скорость разработки
  - Разработчику не приходится решать все задачи и принимать решения «с нуля»

Стандарты программирования: зачем – 2/2

- Уровень взаимодействия в команде
  - Многие мелкие вопросы решены → устраняет ненужные дебаты
  - Облегчает понимание и поддержку чужого кода членами команды
- Согласованность в работе
  - Разработчики сосредотачивают усилия на решении действительно важных задач
  - (Deadline же!) В цейтноте проще делать что-то обычное, чему их учили, натренированное

Примеры: стиль оформления кода – 1/4

- NB! ~~Без фанатизма!~~ Без мелочности
- Отступы
- Правила именования
  - Переменных (например: венгерская нотация)
  - Классов / структур / модулей
  - Функций / методов
- Правила оформления управляющих структур
- (?) Длины строк, длины имен, пробелы или табуляции, etc.

Примеры: стиль кодирования – 2/4

- Комментарии
- Один вход – один выход
- Сокрытие информации
- Глобальные и совместно используемые данные

Примеры: организация работы – 3/4

- Компилируйте без замечаний при максимальном уровне предупреждений
- Используйте автоматические системы сборки программ
- Используйте систему контроля версий
- Регулярно просматривайте код всей командой

Примеры: правила Detailed Design – 4/4

- Масштабируемость
- Сложность алгоритмов
- Кросс-платформенность
- Многонитевость < Многопоточность < {Распределенные алгоритмы | Распределенные приложения }

Психологические аспекты

Организация

Инструментальные средства

КОЛЛЕКТИВНАЯ РАЗРАБОТКА

Эффект Рингельмана – 1/3

- 1927 – серия психологических экспериментов → «эффект Рингельмана»
- Суть эксперимента:
  - Обычные люди поднимали тяжести. Для каждого фиксировали максимальный вес, который он «потянул»
  - После чего людей объединяли в группы, сначала – по двое, потом – четыре человека, восемь.
- Ожидания:
  - если один человек может поднять – условно – 100 кг, то двое вместе должны поднять:
    - 200 кг (простое суммирование)
    - > 200 ег (т.к. «групповая работа позволяет достичь большего, ее результат превосходит сумму отдельных результатов членов группы)

Эффект Рингельмана – 2/3

- Реальный результат:
  - Двое людей поднимали лишь 93% от суммы их индивидуальных показателей
  - Восемь лишь 49%
- Попытки исправить положение:
  - Использовали других задания (например, перетягивание каната)
  - Увеличивали размер групп – процент падал

Эффект Рингельмана – 3/3

- Психологическое объяснение:
  - Когда человек рассчитывает только сам на себя, он прилагает максимум усилий
  - В группе можно и сэкономить силы: никто ж не заметит
- Принцип «пассивности»:
  - Когда человек действует, он запоминает свои усилия и фиксирует их уровень для себя
  - В дальнейшем он/она прикладывает именно столько усилий или меньше

→Формируется пассивное отношение к делу, в которое вовлечен вместе с другими сам

Организация

- Организационные средства
  - Выделение ответственных лиц (руководителя проекта, технолога, и т.д.)
  - Создание и контроль исполнения инструкций
- Технические
  - Библиотека проектной документации
  - Программные средства поддержки

Инструментальные средства – 1/2

- Централизованные хранилища документации:
  - Файловые хранилища
  - Web-сервера
  - Хранилища документов
  - Wiki-подобные средства

Инструментальные средства – 2/2

- Программные средства поддержки
  - Системы управления версиями
  - Системы автоматической компиляции
  - Системы автоматического тестирования
  - Система автоматического документирования
  - Системы отслеживания ошибок

Назначение

Рабочий цикл

СИСТЕМЫ УПРАВЛЕНИЯ ВЕРСИЯМИ

Назначение – 1/2

- Система управления версиями (Version Control System (VCS), Revision Control System) позволяет:
  - Хранить документы
  - Сохранять историю **всех** изменений каждого документа. Все версии доступны – можно при необходимости возвращаться к более ранним версиям
  - Координировать работу нескольких авторов
  - Назначать и использовать управление уровнями доступа, включая аутентификацию пользователей и управление доступом

Назначение – 2/2

- Избегать возникновения конфликтующих версий файлов и помогает разрешать возникающие:
  - Хранит данные о том, кто и когда внес то или иное изменение
  - Показывает различия между версиями одного и того же файла
  - Хранит изменения в версии документа совместно с описанием причины внесения изменений

Свойства репозитория – 1/2

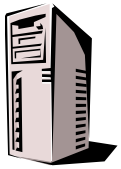
- История всех изменений всех файлов и директорий
  - Есть возможность восстановить любую предыдущую версию файла
  - Запоминает перемещенные и удаленные файлы / директории
- Контроль доступа
  - Разрешения на операции чтения / записи для пользователей и групп
  - Разрешения могут назначаться на уровне репозитория, директории, файла

Свойства репозитория – 2/2

- Журнал изменений. Записываются:
  - Автор изменения
    - Автоматически, как логин пользователя системы
  - Дата/время внесения изменения
    - Автоматически, как системное время
  - Причина внесения изменений
    - Вручную
    - Может ссылаться на документы / запросы на изменения / зарегистрированные ошибки в проекте

Типичный рабочий цикл

client



server

checkout

Скачать текущую (наиболее актуальную) версию (N)

Внесение изменений, тестирование

Запросить текущую версию (M > N)

check status

update

Скачать текущую (наиболее актуальную) версию (M > N)

Разрешение конфликтов (слияние версий проектов / файлов)

commit

Выгрузить свою версию как наиболее актуальную (M + 1)

Внесение изменений, тестирование

… и так далее

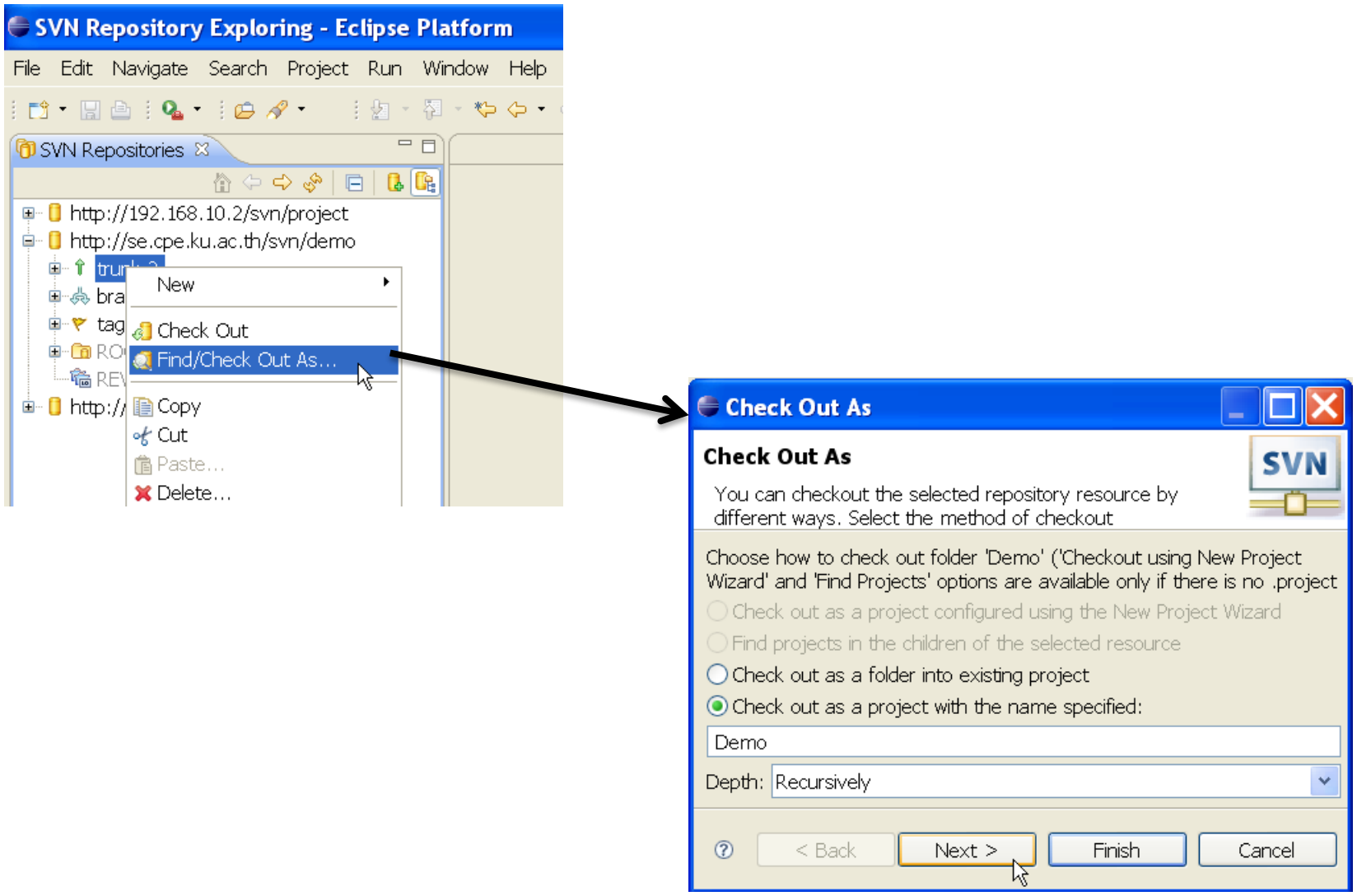
Примеры

- Распределенные – каждый разработчик использует свой локальный репозиторий, синхронизация работы – отдельная задача
  - Open-source: GNU arch, Bazaar, **Git**
  - Proprietary: BitKeeper, Code co-op, TeamWare
- Клиент-серверные – разработчики используют единый репозиторий на сервере
  - Open-source: CVSNT, OpenCVS, **CVS, Subversion**
  - Proprietary: a lot

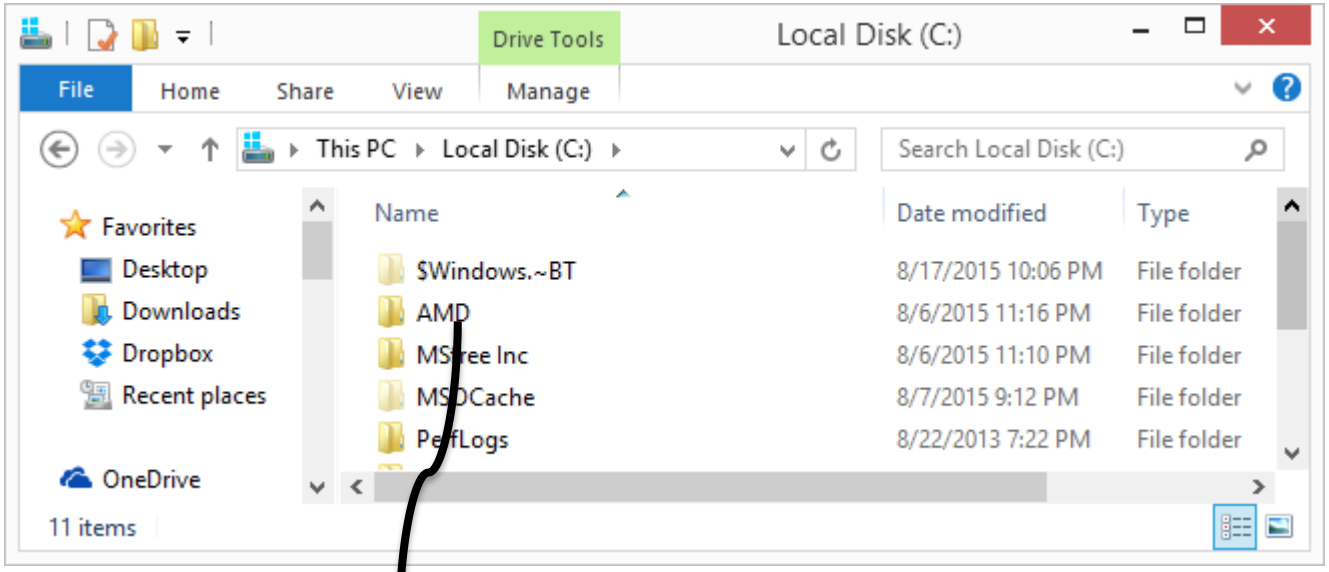
CVS (Subversion)

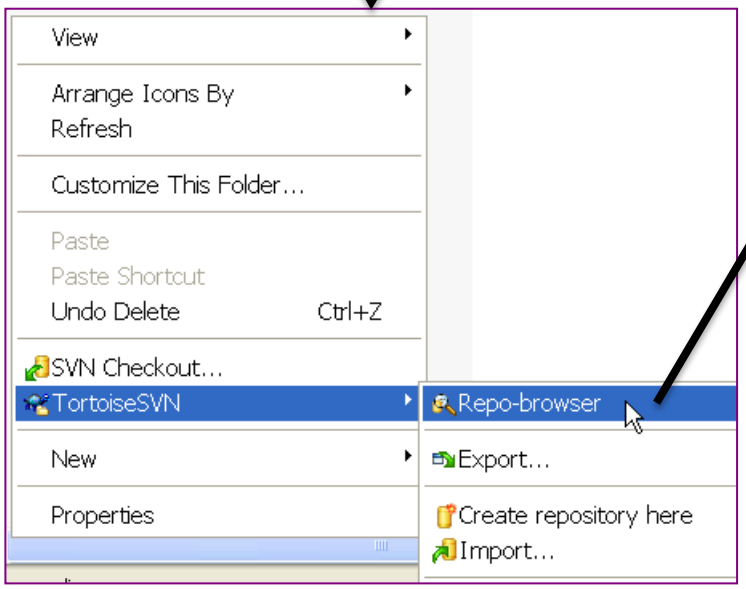
- Concurrent Versions System
- Особенности:
  - Клиент-серверная архитектура
  - Невозможность удаления файла из архива
  - Создает «скрытые» папки
  - Наиболее широко распространена
  - Большое число клиентов
  - Интегрирована во многие средства разработки ПО

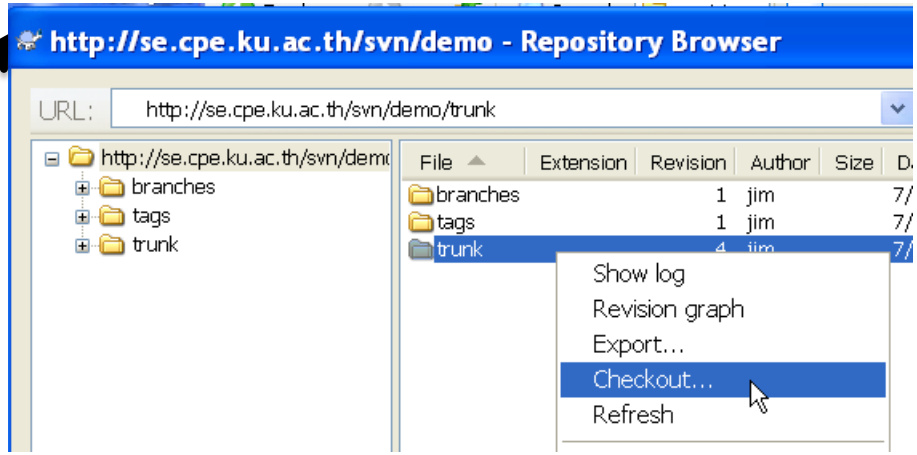
CVS: Интеграция с Eclipse



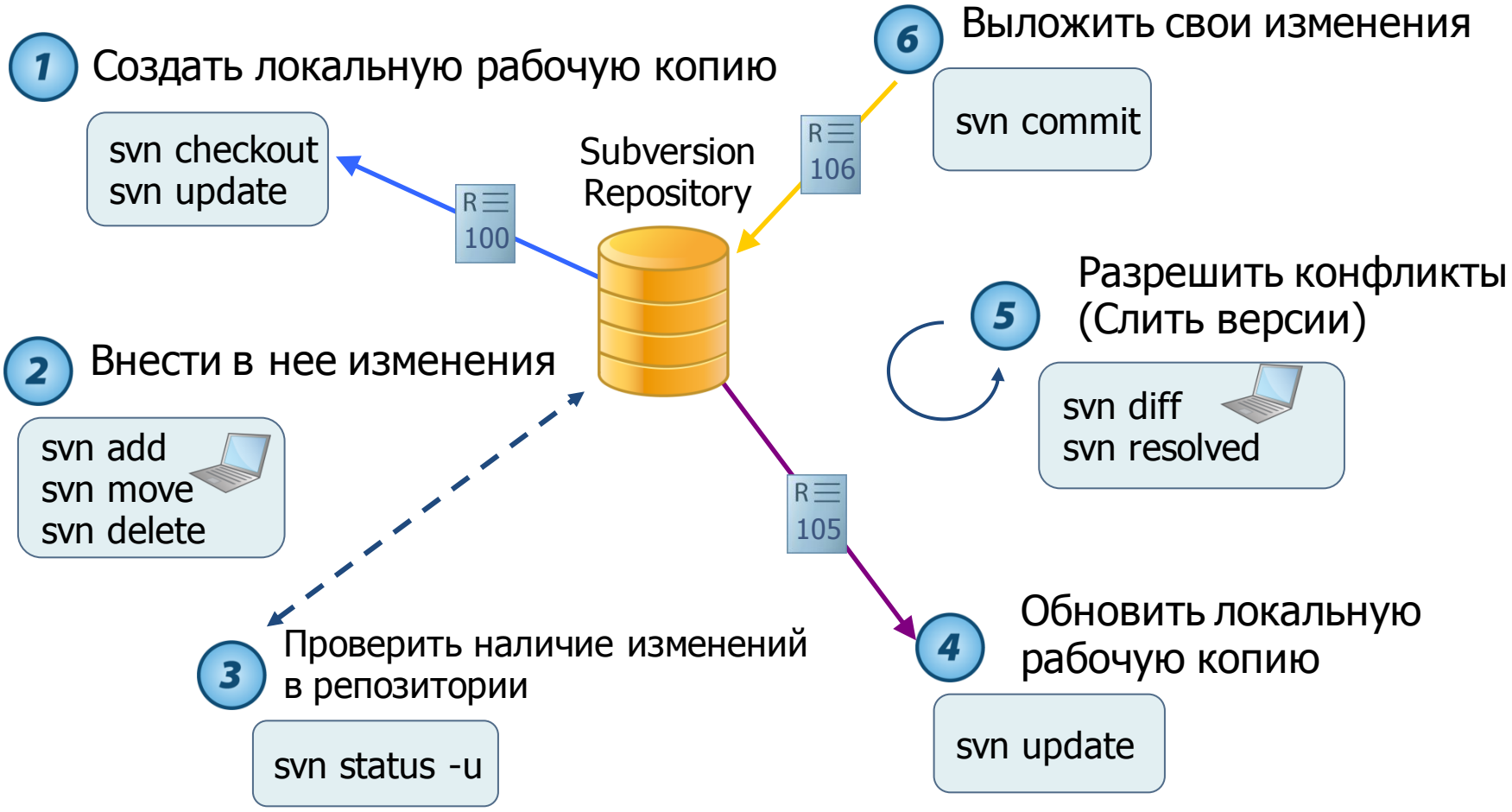
CVS: TortoiseSVN



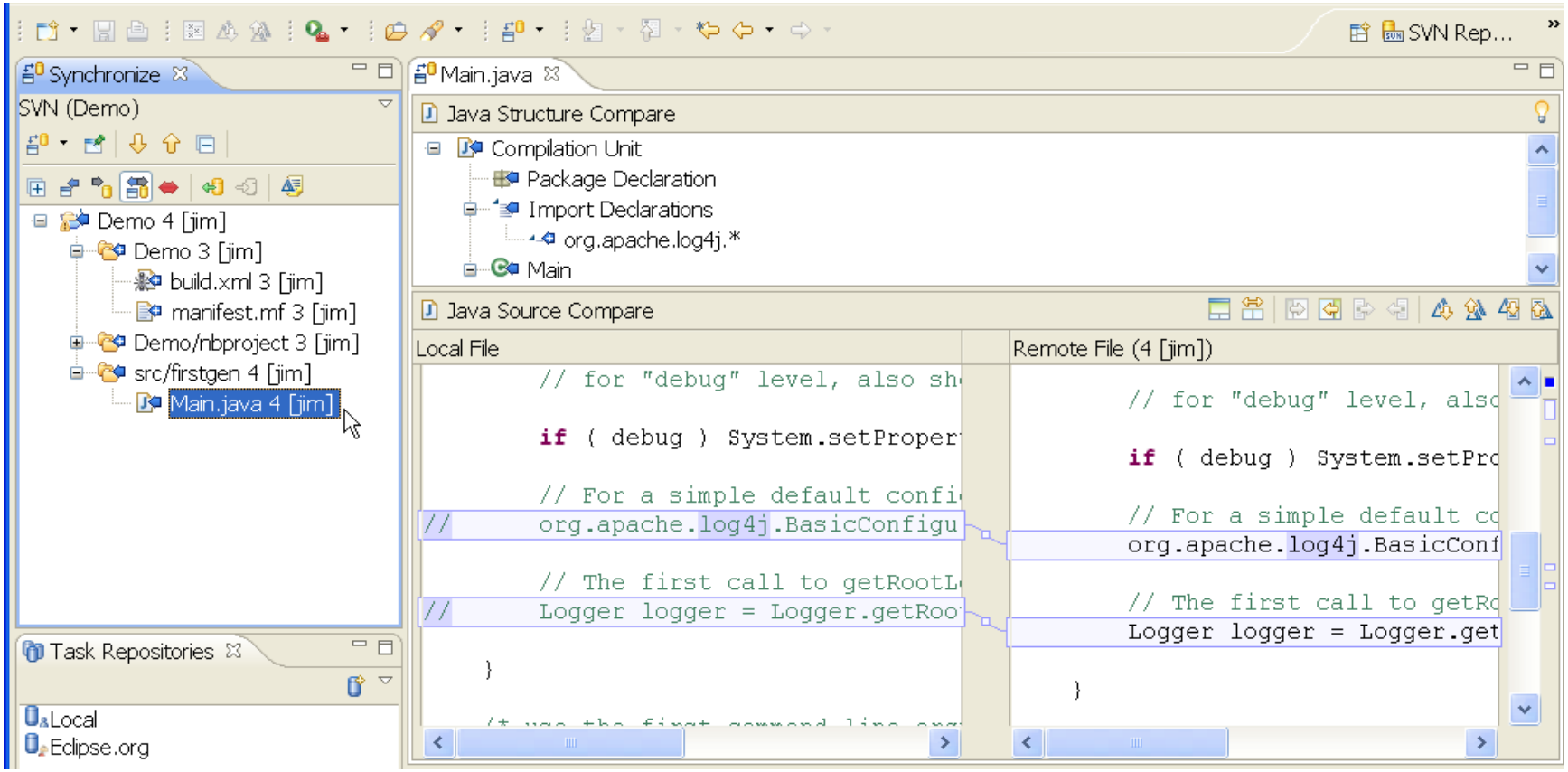




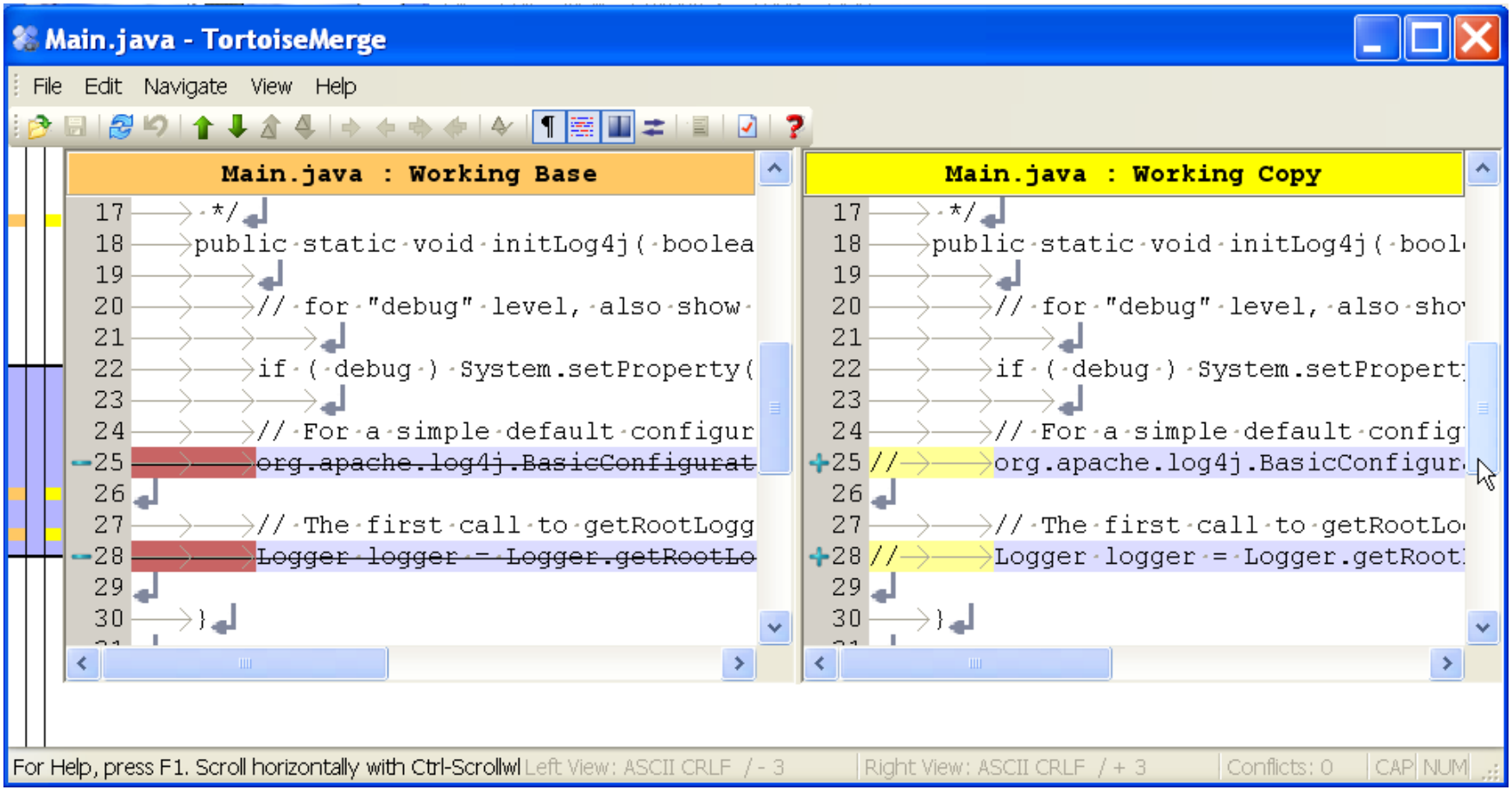
Типичный рабочий цикл



Разрешение конфликтов в Eclipse



Разрешение конфликтов в TortoiseSVN



Git

- Concurrent Versions System
- Особенности:
  - Распределенная архитектура
  - У каждого разработчика всегда есть полная копия репозитория
  - Большая часть действий доступна локально
  - Мощная система ветвлений
  - Широко распространена и поддерживается Линусом Торвальдсом
  - Большое число клиентов

Определение

Doxygen

ГЕНЕРАТОРЫ ДОКУМЕНТАЦИИ

Генератор документации – 1/2

- Генератор документации (documentation generator) — программа или пакет программ, позволяющая получать документацию, предназначенную для программистов (документация на API) и/или для конечных пользователей системы, по особым образом комментированному исходному коду и, в некоторых случаях, по исполняемым модулям (полученным на выходе компилятора)

Генератор документации – 2/2

- Примеры генераторов документации:
  - DoxyGen
  - JavaDoc

Doxygen – общее описание

- Кроссплатформенный
- Поддержка языков: C++, Си, Objective-C, Python, Java, IDL, PHP, C#, Фортран, VHDL и, частично, D
- Документация создается в форматах: HTML, LATEX, man, RTF, и др. Конвертируется в postscript, PDF
- Интегрируется со многими средствами разработки
- GNU General Public License

Doxygen: использование – 1/2

1. Скачать с doxygen.org
2. Единственный раз для [новой] директории с исходными кодами программ выполнить команду:

   doxygen –g

   Команда создает конфигурационный файл doxygen с именем Doxyfile. Файл имеет текстовый формат и может быть отредактирован

Doxygen: использование – 2/2

3. Отредактировать файл Doxyfile:
   - Убедиться, что все переменные EXTRACT* установлены в YES
   - Для языка C (НЕ C++), установить конфигурационный параметр OPTIMIZE_OUTPUT_FOR_C = YES
4. Далее по желанию [после изменения исходного кода и необходимости обновить документацию] выполнять команду:

   doxygen

   Команда обновляет документацию в поддиректории HTML

Doxygen: Что описывать?

- Файлы исходного кода
- Классы
- Методы и функции
- Переменные

Doxygen: Описание файлов

- Разместить в начале файла комментарий вида:

```
/**
 * \file        ImageData.java
 * \brief       Содержит определение класса ImageData
 *          (Обратите внимание, что класс абстрактный)
 *
 *          <Больше описаний тут>
 *
 * \author      Имя автора
 */
```

Doxygen: Описание классов

- Разместить непосредственно перед классом комментарий вида:

```
/** \brief JImageViewer класс
 *
 *   Дли-и-и-и-инное описание
 */
public class JImageViewer extends JFrame implements
    ActionListener {
…
```

Doxygen: Описание функций

- Разместить непосредственно перед функцией комментарий вида:

```
/** \brief Главная точка входа приложения
 *
 *  \param args Каждое имя файла переданное как args
 *  приведет к отображению на экране изображения
 *  из этого файла
 *  \returns ничего
 */
public static void main ( String[] args ) {
…
```

Doxygen: Описание переменных

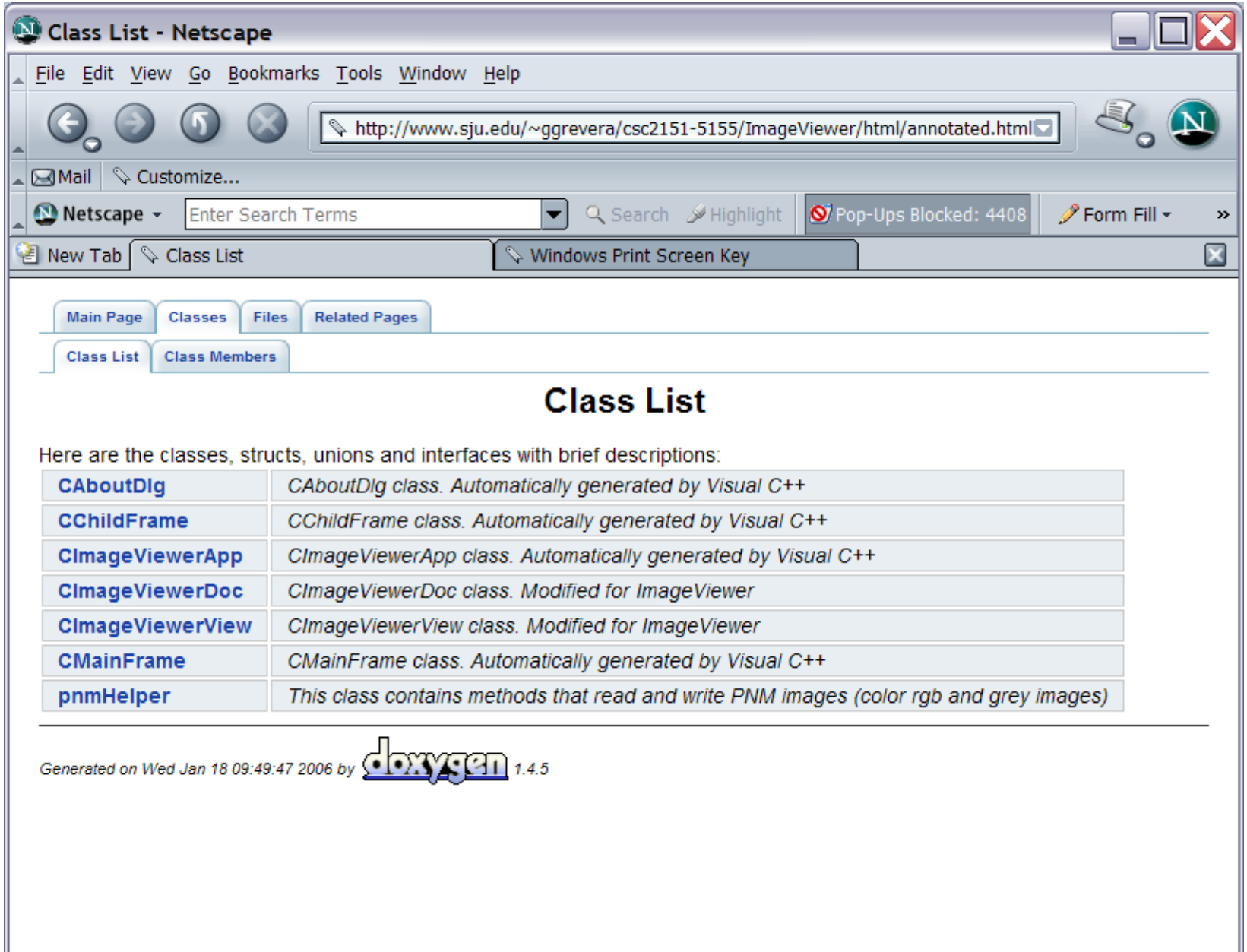
- Имеет смысл описывать глобальные, статические переменные, а также переменные-члены классов
- Для этого после переменной размещают комментарий вида:

```
int    g_nImageWidth;        ///< Ширина изображения
int    g_nImageHeight;       ///< Высота изображения
static int nMin;             ///< Минимум
static int nMax;             ///< Максимум
int[]  m_aimageData;         ///< Массив изображений
```

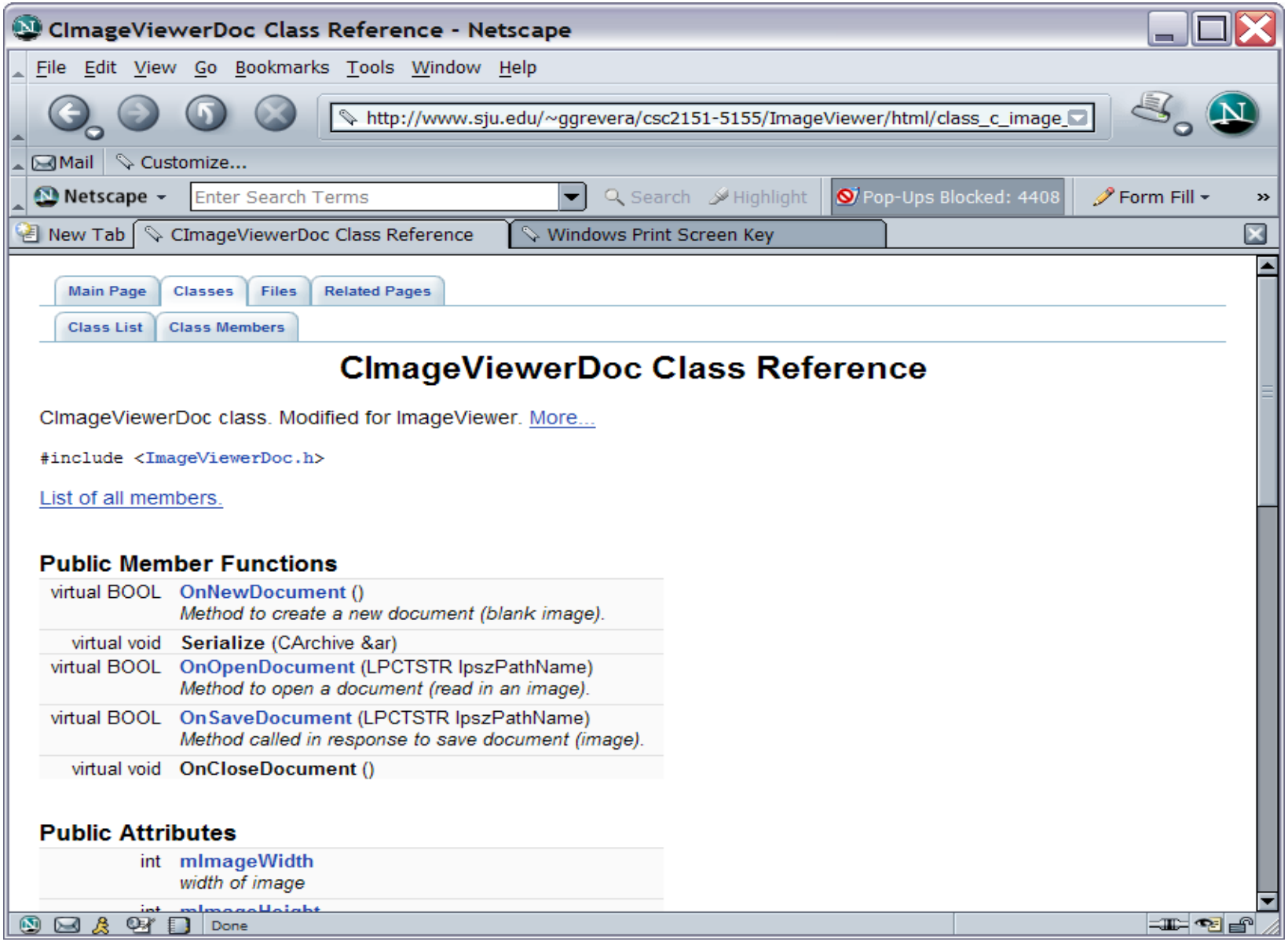
Doxygen: Самые полезные теги

- \file
- \author
- \brief
- \param
- \returns
- \todo

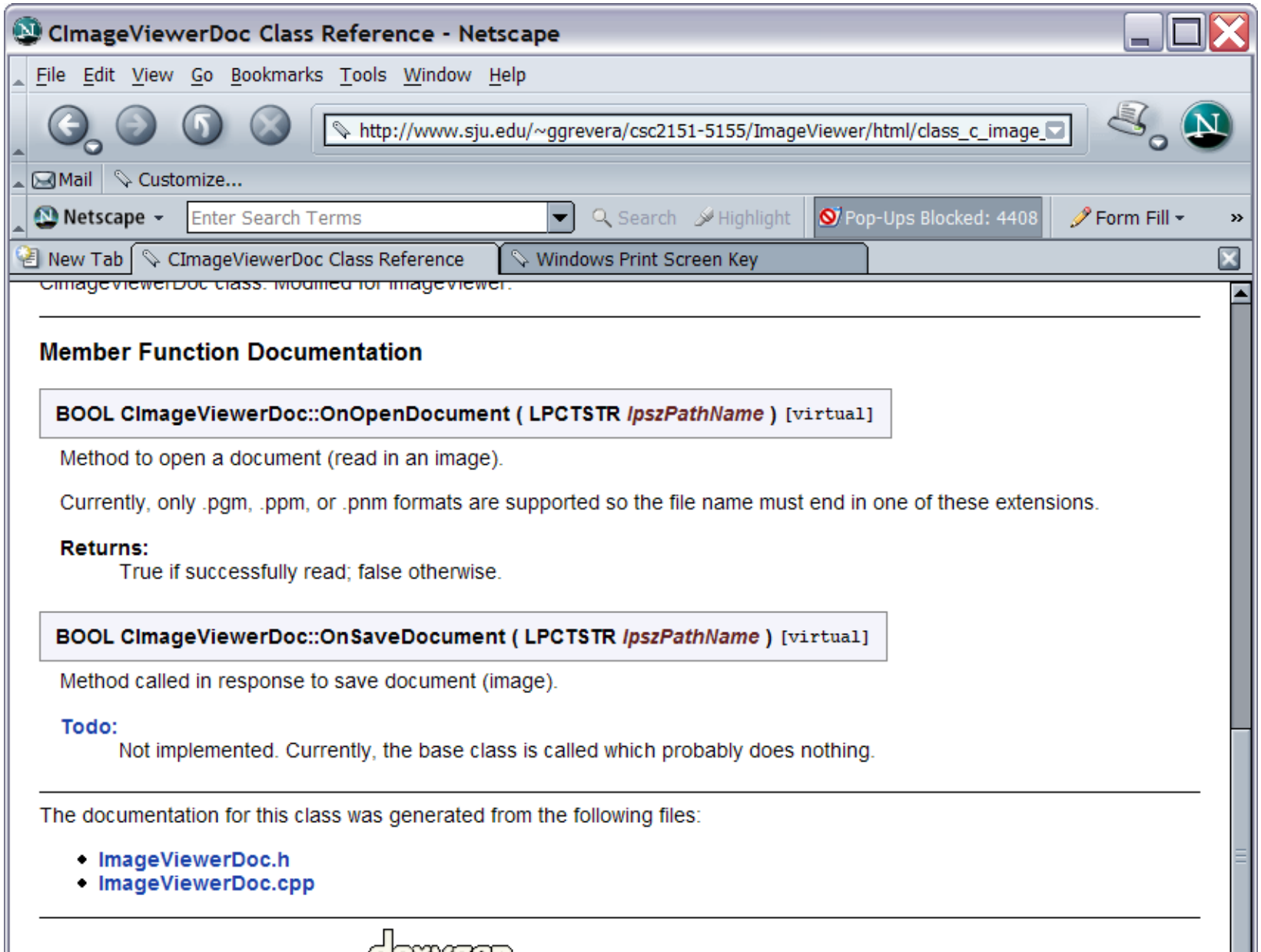
Doxygen: Пример результата – 1/5



Doxygen: Пример результата – 2/5



Doxygen: Пример результата – 3/5



Doxygen: Пример результата – 4/5

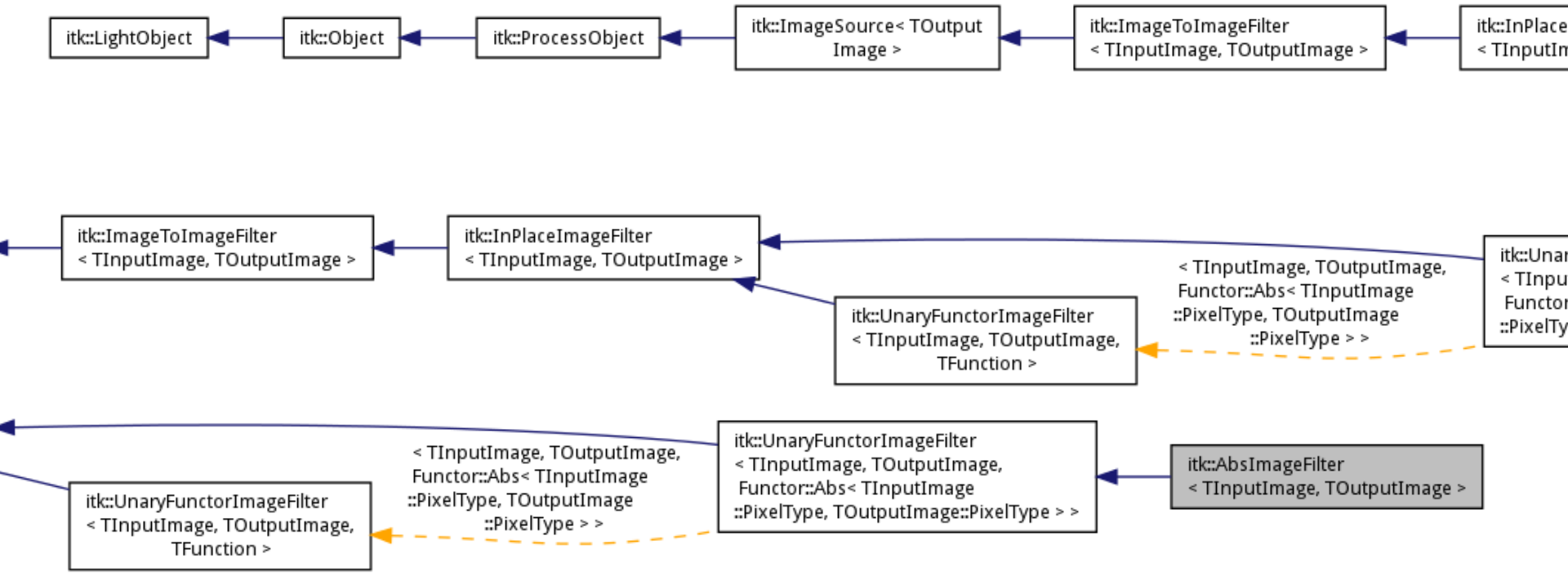


Диаграмма наследования

Doxygen: Пример результата – 5/5

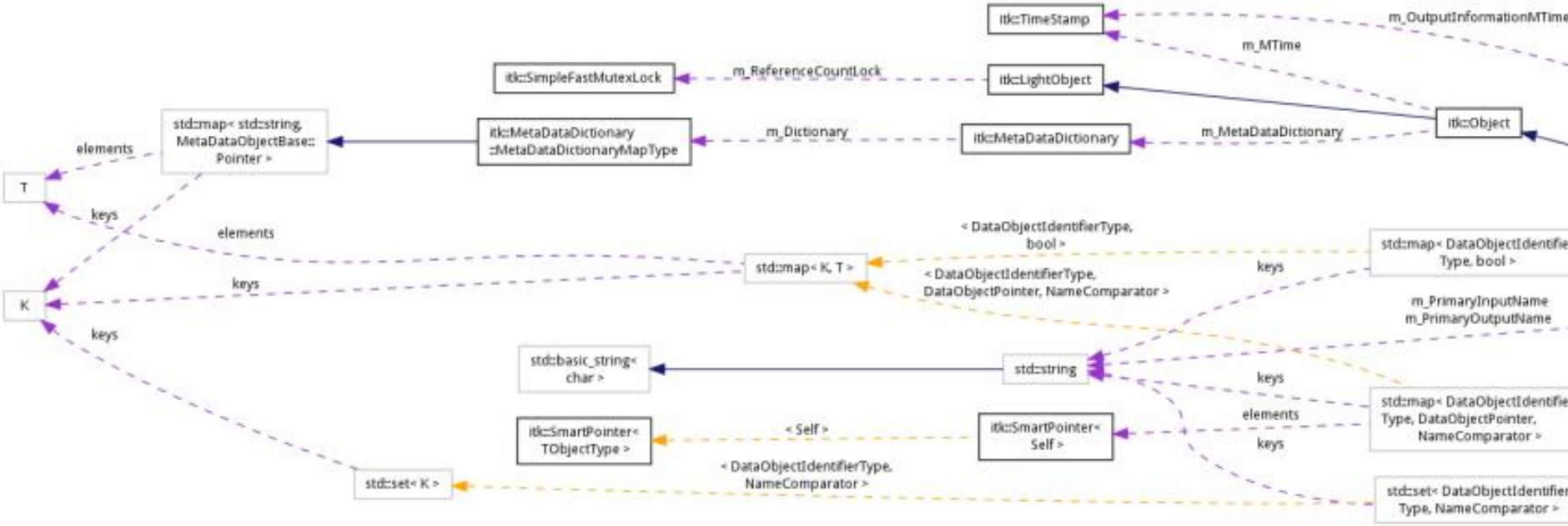


Диаграмма коммуникации (взаимодействия)
Communication diagram

Определения

Область применения

Дефекты

Жизненный цикл

СИСТЕМЫ ОТСЛЕЖИВАНИЯ ОШИБОК

Системы отслеживания ошибок

- Bug (defect) tracking system, issue tracking system
- Прикладная программа (система) помогающая разработчикам ПО (кодерам, тестировщикам и др.) учитывать и контролировать:
  - Ошибки
  - Неполадки
  - Пожелания пользователей

  а также следить за процессом отработки зарегистрированных дефектов

Области применения – 1/2

- С внешним доступом:
  - Позволяют конечным пользователям напрямую регистрировать обнаруженные проблемы
- Только внутреннее использование:
  - Компании-разработчики ПО интегрируют системы отслеживания ошибок в ПО управления проектами
  - Отчеты системы могут использоваться для оценки эффективности работы сотрудников
- Наличие системы отслеживания ошибок является признаком правильной организации (обратное еще вернее! ^_~)

Области применения – 2/2

- Local Bug Tracker (LBT) – отдельная программа, используемая службой поддержки для регистрации запросов пользователей
  - Скрывает от пользователей особенности реализации ПО
  - Позволяет использовать разные «языки» для разработчиков и пользователей

Архитектура, интеграция

- Обычно центральной частью системы отслеживания ошибок является база данных, хранящая информацию о зарегистрированных дефектах
- Существуют распределенные системы, предназначенные для совместного использования с распределенными системами управления версиями
- Системы отслеживания ошибок интегрируются с Системами управления тестированием

Жизненный цикл дефекта – 1/2

- Новый — дефект зарегистрирован
- Открыт повторно — дефект вновь найден в другой версии
- Назначен — назначен ответственный за исправление дефекта
- Разрешён — работа с дефектом завершена. Варианты разрешения (пример):
  - Исправлено – в версию номер…
  - Дублирование – такой дефект уже зарегистрирован
  - См. далее

Жизненный цикл дефекта – 2/2

- Варианты разрешения (продолжение):
  - Не является ошибкой – работает в соответствии со спецификацией
  - Отложено – имеет слишком низкий приоритет, исправление отложено до следующей версии, etc.
  - Невоспроизводимо – требуется дополнительная информация об условиях, в которых дефект проявляется
- Проверено – тестировщики подтвердили, что дефект устранен
- Закрыт – руководитель закрыл актиивность

Описание дефекта – 1/2

- Общее описание:
  - Идентификатор (номер)
  - Краткое описание
  - Кто сообщил о дефекте
  - Дата и время обнаружения
  - Серьёзность (критичность)
- Подробное описание:
  - Версия продукта, в которой обнаружен дефект
  - Конфигурация ПО / оборудования
  - Описание шагов для воспроизведения дефекта
  - Ожидаемый и фактический результаты

Описание дефекта – 2/2

- Управление:
  - Приоритет решения
  - Ответственный за устранение
  - Обсуждение возможных решений и их последствий
  - Текущее состояние (статус) дефекта
  - Версия продукта, в которой дефект исправлен

Примеры систем

- Bugzilla
- Mantis

Mantis – 1/4

- MantisBT — свободно распространяемая система отслеживания ошибок
- Гибко конфигурируется, что позволяет использовать её не только при разработке ПО, но и в качестве системы учёта заявок для службы поддержки
- Интерфейс клиента реализован как веб-интерфейс
- Название Mantis (богомол) происходит от того, что богомол питается жуками (bug)

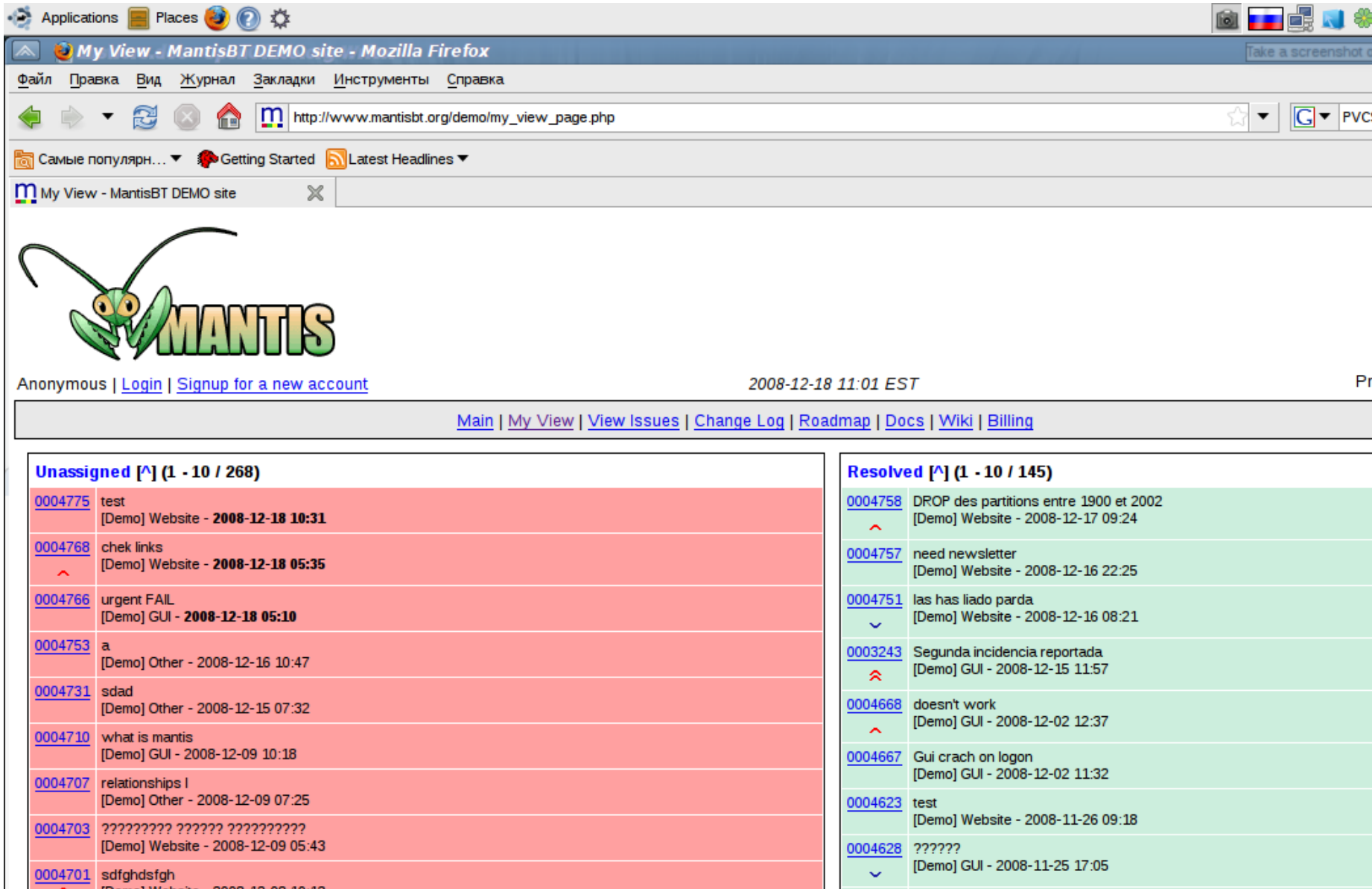
Mantis – 2/4

- Достоинства:
  - Бесплатна: GNU General Public License (GPL)
  - Цветовая индикация статуса дефекта
  - Настраиваемые пользователем поля
  - Удобные фильтры
  - Скорость работы
  - Уведомления по e-mail
  - Большое количество плагинов, расширяющих функциональность
  - Возможна интеграция с wiki-движком для создания документации (DokuWiki)

Mantis – 3/4

- Достоинства? 0_0:
  - Код на PHP свободно модифицируем
  - Понятно написанный код
- Недостатки:
  - Если доступных через интерфейс настроек не хватает, то надо менять код

Mantis – 4/4


Applications
Places
My View - MantisBT DEMO site - Mozilla Firefox
Файл Правка Вид Журнал Закладки Инструменты Справка
http://www.mantisbt.org/demo/my_view_page.php
Самые популярн...
Getting Started
Latest Headlines
My View - MantisBT DEMO site
MANTIS
Anonymous | Login | Signup for a new account
2008-12-18 11:01 EST
Main | My View | View Issues | Change Log | Roadmap | Docs | Wiki | Billing
Unassigned [^] (1 - 10 / 268)
0004775 test [Demo] Website - 2008-12-18 10:31
0004768 chek links [Demo] Website - 2008-12-18 05:35
0004766 urgent FAIL [Demo] GUI - 2008-12-18 05:10
0004753 a [Demo] Other - 2008-12-16 10:47
0004731 sdad [Demo] Other - 2008-12-15 07:32
0004710 what is mantis [Demo] GUI - 2008-12-09 10:18
0004707 relationships I [Demo] Other - 2008-12-09 07:25
0004703 ????????? ?????? ??????????? [Demo] Website - 2008-12-09 05:43
0004701 sdfghdsfgh
Resolved [^] (1 - 10 / 145)
0004758 DROP des partitions entre 1900 et 2002 [Demo] Website - 2008-12-17 09:24
0004757 need newsletter [Demo] Website - 2008-12-16 22:25
0004751 las has liado parda [Demo] Website - 2008-12-16 08:21
0003243 Segunda incidencia reportada [Demo] GUI - 2008-12-15 11:57
0004668 doesn't work [Demo] GUI - 2008-12-02 12:37
0004667 Gui crach on logon [Demo] GUI - 2008-12-02 11:32
0004623 test [Demo] Website - 2008-11-26 09:18
0004628 ?????? [Demo] GUI - 2008-11-25 17:05

Определение

СИСТЕМЫ АВТОМАТИЧЕСКОГО ТЕСТИРОВАНИЯ

Все пропало

Ужас-ужас

ПРАКТИЧЕСКИЕ ЗАДАНИЯ