Софийски университет „Св. Климент Охридски”
Факултет по Математика и Информатика
Катедра „Софтуерни технологии”
Специалност „Информатика”
Специализация „Софтуерни технологии”

ДИПЛОМНА РАБОТА

Тема:

Софтуерен процес при адаптиране на проект с отворен код (АПОК).
Битторент клиент (extended BitTorrent Client).

Дипломант: Галин Красимиров Гроздев, ФN№ M21-640

Научен ръководител: Доц. Силвия Илиева, катедра „Софтуерни технологии”, ФМИ

СЪДЪРЖАНИЕ

1. Въведение в предметната област на темата

Темата на дипломната работа предполага описание на предложение за нова гъвкава методология. За да може да се дефинира тази гъвкава методология ще е нужно да изберем подход за описването ѝ. В дипломната работа сме избрали това да стане чрез метаезик, дефиниран при описването на съществуващите гъвкави методологии. Ще представим по-популярните гъвкави методологии, за да систематизираме приликите с нашата новопредложената гъвкава методология по:

- практики
- роли и отговорности
- особености и сходства между софтуерните процеси.

Ще опишем и именуваме новопредложената гъвкава методология, като АПОК (Адаптиране на проект с отворен код), и ще покажем че тя наистина е гъвкава. Ще бъде демонстрирано използването на гъвкавата методология при разработването на един проект използващ проект с отворен код (мрежови проект – XBTC битторент клиент). Ще опишем и самия демонстрационен проект – предметната му област, идеята и функционалността му от гледна точка на АПОК. И ще обобщим мястото на АПОК в света на гъвкавите методологии, кога е удачно да се прилага и перспективите за развитието ѝ.

1.1. Въведение в гъвкавите методологии

Гъвкавите методи за софтуерна разработка се опитват да отговорят на потребностите на „нетърпеливото" бизнес общество изискващо по-лек софтуерен процес и същевременно по-бързо реагиращ на настъпващите промени начин на разработка. Това особено важи за случаи като бързо развиваща се и бързо променяща се интернет индустрия. Представянето на една нова гъвкава методология върви ръка за ръка с много дебати относно въпросната млада гъвкава методология, публикации и нужда от описваща я литература. Ще започнем с излагането на идеята на гъвкавите методологии.

1.1.1. Идея на гъвкавите методологии

Гъвкавите методологии предлагат начин за управляването на софтуерни проекти, в които имаме значителни промени и еволюция на същия този проект през целия жизнен цикъл на проекта. Съществуват десетки гъвкави методологии, ние ще обърнем внимание само на най-често използваните. Повечето от гъвкавите методологии се опитват да намалят риска от провал чрез въвеждането на „времеви кутийки" по времето на които единица човешки ресурс разработва единица функционалност. Времевите кутиики биват обединявани в така нарчените итерации, разработвани обикновено от една седмица до един месец. Всяка една итерация представлява мини проект сама по себе си, включваща в себе си всички стъпки за създаването на нова версия на проекта с добавената нова функционалност. А стъпките обикновено са планиране, анализ на изискванията, дизайн, кодиране, тестване, интегриране и документиране. С приключването на всяка итерация разполагаме с нова версия на проекта. Често след приключването на всяка итерация се налага да се променят приоритетите на различните итерации.

Гъвкавите методологии поставят наличието на работещ на дадено ниво софтуер като главно мерило за развитието на проекта. Друга характерна особеност на гъвкавите методологии е, че при тях комуникацията (колкото се може по-близка) е заместител на обширната и всеописателна документация.

1.1.2. Сравнение на гъвкавите с останалите методологии

Гъвкавите методологии имат много общи характеристики с методологията Бързо разработване на софтуер (Rapid Application Development) [RAD] от 80те години.

Те често са характеризирани като противоположност на „дисциплинираните" и управляваните чрез стриктно планиране методологии. Това обаче не означава, че те са без планиране и недисциплинирани. Съществуващите гъвкави методологии обхващат цялата гама динамичност на промените по даден проект, или от непрекъснато предвиждане на потребностите на клиента до адаптация към текущите изисквания към проекта.

Barry Boehm и ***Richard Turner*** във книгата си „Balancing Agility and Discipline" [16] описват разликите между адаптивните (adaptive) и предвиждащите (predictive) методологии (таблица 1.1.2). Като слагат знак за еквивалентност между гъвкавост и адаптивност и съответно обявяват че план управляваните методологии са „предвиждащите" методологии. Обявената от тях скала може да се използва за да се определи доколко една методология е гъвкава.

Таблица 1.1.2: *Гъвкава/план-управлявана методология*

Гъвкава методология	План управлявана методология
Ниска критичност на задачите	Висока критичност на задачите
Преобладават опитните програмисти	Много първопрохождащи програмисти
Изискванията се променят доста често	Изискванията почти не се променят
Малък брой разработчици	Голям брой разработчици
Култура на разработка позволяваща свобода в много отношения на участниците.	Политика на компанията за налагане на ред и дисциплина.

Въпреки че новопредложената методология използва „план на проекта", съставен от използваните в план-управляваните методологии елеменити като задачи, подзадачи, разпределение на ресурсите, ограничения и.т.н., (създава се неофициален план за разработването на проекта, които непрекъснато търпи промени) тя носи характерните особености на една гъвкава методология.

1.2. Битторент технология. Peer-to-peer проекти

1.2.1. Peer-to-peer мрежи. Класификация

Peer-to-peer ("пийр-към-пийр" – p2p) компютърните мрежи разчитат главно на bandwidth и изчислителната мощ на участниците в тази мрежа, а не на сървъри. Към Peer-to-peer мрежите се свързват PC потребители използващи peer-to-peer клиент, за да споделят мултимедия, и обменят данни в реално време. Всеки един peer-to-peer клиентите може да представлява едновременно и клиент и сървър за споделяне на данни. Всеки клиент същевременно може да направи множество връзки с други клиенти с интерес към определени данни, или въпросния клиент е създал пийр.

Peer-to-peer мрежите могат да бъдат класифицирани като:

- Централизирана като Napster
- Децентрализирана като KaZaA
- Структурирана като CAN, BitTorrent protocol
- Неструктурирана като Gnutella

1.2.2. Битторент технология

Битторент представлява peer-to-peer протокол за комуникация и обмен на файлове обединени в пакети наречени torrents. Това е начин за дистрибуция на голям обем данни. За да изясним какво представлява битторент протокола от потребителска гледна точка, ще опишем начина му на употреба.

Всеки един „възел" (свързан към мрежата PC) в битторент мрежата представлява един клиент имплементиращ битторент протокола. Всеки един такъв клиент е способен да подготвя, изисква и трансферира по мрежата всякакъв вид файлове обединени в колекция ("torrent data") използвайки протокола. За всеки един битторент клиент, всеки един PC на който е стартирана инстанция на битторент клиент, с който въпросния клиент има някакви взаимоотношения, представлява пийр. За да се споделят група от файлове някой пийр трябва първо да създаде

обединяващия ги по тематика торент. Това е малък файл съдържащ metadata относно файловете които ще бъдат споделяни, и информация относно tracker-а , компютърът който координира дистрибуцията на файлове. Самият тракер съдържа най-често в база данни списък на всички клиенти имащи някакво отношение към всеки един торент. Пийровете които искат да свалят дадени данни първо свалят торент файла, и се свързват към указания в него тракер който информира нашия и останалите пийрове посредством „анонси" (тематични съобщения между пийровете и тракера) за разпределението на торент данните из мрежата. Нашия клиент избира от кой пийрове да сваля парчета от данните – leeching и едновременно с това да предоставя свалените парчета данни на останалите заинтересовани пийрове – seeding. Самия торент файл представлява колекция от checksum на всяко едно от парчетата данни на които са разделени торент данните. Торентите могат да бъдат разпространявани в:

- централизирана peer-to-peer мрежа, като биват регистрирани на някой тракер и най-често разпространявани от някой website.
- децентрализирана peer-to-peer мрежа или *trackerless system,* където всеки един пийр играе ролята и на тракер. Този начин на разпространение е имплементиран в много клиенти използвайки метода на distributed hash table (DHT).

Доказателство за това че тази предметна област се развива и в момента е факта че през Април 2007 клиентът Azureus предлага нов уникален начин за реализиране на DHT несъвместим с този на останалите клиенти. Битторент технологията (Technologies built on BitTorrent) се развиват непрекъснато. Последното подобрение на технологията, за повишаване скоростта на споделяне на данни между пийровете, също датира от Април 2007 - Similarity Enhanced Transfer (SET).

1.3. Цел на дипломната работа

Настоящата дипломната работа засяга една сравнително нова и бързоразвиваща се технология – битторент. Но тази технология е представена дотолкова доколкото е нужно за да се запознаем с предметната област на разработвания проект. Не сме се задълбочавали в битторент технологията поради факта че темата на дипломната работа не е представянето на дадена технология а **дефинирането, класифицирането** и **демонстрация на използването** на една новопредложена гъвкава методология. Хронологично погледнато две цели стоят пред нас:

1) Да се предложи подходящ софтуерен процес придържащ се към принципите на гъвкавите методологии за проекти с отворен код.
2) Да се избере проект с отворен код и да се приложи предложения софтуерен процес, като се направят изводи и препоръки за бъдещи проекти.

За постигането на тези цели пред нас стоят следните задачи:

1. *Сравнителен анализ със съществуващите гъвкави методологии* – откриването на особеностите и приликите им, които ще ни помогнат за дефинирането на една нова гъвкава методология.
2. *Предложение на нова гъвкава методология за проекти с отворен код.* – **дефинирането** на методологията Адаптация на проекти с отворен код (АПОК).
3. Избор на проект (изследване) с отворен код и приложение на софтуерния процес – **демонстрация на използването** на АПОК.
4. Да се опише добавената функционалност, необходимостта от нея и пътя до достигането и:
   - internall peer-to-peer firewall.
   - error handling and logging.
   - enhancing the frontend-backend communication protocol.
   - other minor features and fixes.
5. Анализ на получените резултати и препоръки. – **класифицирането** на АПОК в света на гъвкавите методологии.

1.4. Полза от реализацията на дипломната работа

Дипломната работа се разглежда от няколко гледни точки:

- От гледната точка на софтуерния инженер, който би проследил анализа на гъвкавите методологии от гледна точка на дипломанта и цитираните от него източници. Същия би разгледал новопредложената гъвкава методология с интерес от теоретична гледна точка.
- От практическа гледна точка е предоставен един нов модел на работа за разработчиците използващи проекти с отворен код при създаването на софтуерния си продукт. Доброто класифициране на новопредложената методология би я направило алтернативен избор на съществуващите гъвкави методологии при разработването на подобни проекти.
- От гледна точка на участника в проекти с отворен код – Виждаме как да доближим един разработван на добра воля проект с отворен код (ПОК) към бизнеса.

Много от гъвкавите методологии са представени без реални демонстрационни проекти, postmortems или значителна по обем библиотека от “case studies” (Начин на изучаване на специфична предметна област, базиран на детайлно описание и анализ на специфични проблеми. В нашия случай най-вероятно този „проблем” представлявал проект, част от проект, ...). Описанието на демонстрационния проект освен, че ни показва „гъвкавост” при използването на новопредложената гъвкава методология в реални условия на разнородни ограничения и динамика при клиентските изисквания, описва и един типичен „шаблон” за разработването на такъв тип проекти използвайки тази гъвкава методология.

Анализът на получените резултати представлява изводи, готови за използване при разработването на бъдещи проекти, следвайки предложената гъвкава методология АПОК. Препоръките за начин на употреба на тази гъвкава методология е може би най-отворената за нови идеи област, която неминуемо би довела да развитието на АПОК. Използването на тази методология излиза извън рамките на ПОК и е възможно да бъде използвана при работа с компании разработващи компоненти или дори в компании разработващи различни аспекти на един продукт.

И не на последно място самата предметна област на използвания ПОК е актуална и вероятно сама по себе си би ни заинтригувала да се запознаем с целия труд.

1.5. Структура на дипломната работа

Дипломната работа е организирана в следните секции:

1) В първа глава „*Въведение в предметната област на темата”* ще дадем постъпково кратко описание на пътя за достигането на целта на дипломната работа. Кратко описание на структурата на дипломната работа:

- *Въведение в гъвкавите методологии* – кратко запознаване на читателя с предметната област на гъвкавите методологии. Идеята на гъвкавите методологии , с какво се различават от останалите методологии и различни факти които ще ни помогнат да докажем че новопредложената гъвкава методология е гъвкава.
- *Битторент технология. Peer-to-peer проекти* – Кратък преглед на peer-to-peer мрежите с цел класификация на битторент мрежата. Запознаване на читателя с основите на битторент технологията. Термини и идея.
- *Цел на дипломната работа, Полза от реализацията на дипломната работа, Структура на дипломната работа,* Списък на използваните фигури, таблици, съкращения и термини – описание на идеята и структурата на дипломната работа.

2) Във втора глава “*Сравнителен анализ на съществуващите гъвкави методологии”* се представя детайлен анализ на гъвкавите методологии. Анализът се извършва от различни гледни точки и по различни показатели:

- *Предмет на сравнителния анализ,* подхода който ще се използва за направата на сравнителния анализ – с използването на метаезик, *Концепция на гъвкавите методологии* – кратък преглед на събирателните черти на гъвкавите методологии от практическа гледна точка (за разлика от въведението в гъвкавите методологии). *Основни Дефиниции в гъвкавите методологии* – дефиниране на основните понятия в областта на гъвкавите методологии като метаезик.

- *Кратко описание и сравнителен анализ на по-популярните гъвкавите методологии по ключови особености и жизнен цикъл* – разгледани са XP, Scrum, DSDM, FDD, Crystal и ASD. От разгледаните методологии е взаимствано при разработването на модела на новопредложената гъвкава методология. Също така е и додефиниран използвания метаезик за описание на гъвкавата ни методология.
- *Общи белези на гъвкавите методологии и заключение*– заключително сравнение на всички гъвкави методологии с план управляваните методологии и OSS от гледна точка на различните участници в разработваните проекти. *Преход към „Предложението за нова гъвкава методология"* – Обобщение на втора глава и описване на нуждата от дефинирането на нова гъвкава методология.

3) В трета глава „*Предложение на нова гъвкава методология за проекти с отворен код*" се описва теоретично, по дефинирания в предходната глава метаезик, какво представлява нововъведената гъвкава методология. Също така и именуваме методологията спрямо предназначението и – *Адаптиране на проект с отворен код* (*АПОК*).

- *Софтуерния процес* е разделен на три части подобно на DSDM - *Проекти с отворен код* (Начина на работа в ПОК обществото и традициите които са се обособили с времето при разработка на ПОК), *Адаптация (*Разглежда се бизнес ориентирания под-процес*)* и *Логистика* (Връзката между бизнес света и ПОК обществото). Всеки един под-процес е разгледан по дефинираната схема за разглеждане на софтуерния процес, ролите и отговорностите , практиките и особеностите.
- Описание по метаезика*: Практики в Адаптацията и ПОК, Роли и отговорности* – Практики при ПОК са наследени от традициите на разработка на ПОК, а практиките на Адаптацията са разделени на основни и допълнителни – тези, чието използване се налага при наличие на проблеми или мащабни промени.
- *Схема на софтуерния процес* – скицирана блок схема на софтуерния процес. Използвана за направата на „табло за презентиране на дипломната работа".

4) В четвърта глава „*Избор на проект с отворен код(изследване) и приложение на софтуерния процес"* се демонстрира използването на АПОК. Постъпково демонстрацията се извършва на следните етапи:

- *Иницииране на търсенето и избор на подходящия ПОК* – разглеждане на различните ПОК, които биха били подходящи – Azureus, CTorrent, XBT. Изследване на предметна област на ПОК, дефиниране на „мисията на продукта", избор на архитектура, проучване на намерените резултати сред които е и успешно избрания за целта XBT client.
- *Софтуерен процес* – Програмни инструменти и политика на използване – Описване на използвания шаблон от програмни инструменти и подходящи роли и практики от АПОК за такъв тип проекти. Описание на прототипирането, планирането , моделирането , прехода от сивата към бялата фаза във софтуерния процес. *Комуникация между бизнес средата и ПОК обществото.*
- *Ограничения и адаптация към* допълнителните фактори влияещи на проекта - *Критичен път, Промяна на отговорностите* – Описва се подхода използван за синхронизиране на различните участници и екипи с цел минимизиране на критичния път на проекта. Демонстриране на „гъвкавост" при използването на АПОК в реални условия на неясни изисквания и изменящи се ограничения. *Бъдещо развитие и перспективи на XBT ПОК.*
- *Сурова схема на процеса на разработка на проекта разбит на три итерации* – Блок схема обобщаваща начина на разработка на този проект използвайки АПОК.

5) В пета глава *„Описание на добавената функционалност"* описваме добавената към ПОК функционалност, необходимостта от нея и пътя до достигането и – програмния аспект на дипломната работа.

- *Описание на добавената функционалност* - Подобряване на frontend-backend протокола за комуникация, Обработка и съхранение на грешки, Добавена минорна функционалност и оправени проблеми в проекта, internall peer-to-peer firewall
- *Спецификация. Програмно Приложение* - Локация на пача и скрипта за тестването му, описание на изисквания за модификацията „internal peer-to-peer firewall", описание на едно от минорните изисквания – разширената „close command", описание на комуникационния протокол на XBTC – backend to frontend , примери за използването на xbt_cli и настройки на XBTC.

6) В шеста глава „*Анализ на получените резултати и препоръки"* класифицираме АПОК методологията ,и се опитваме да покажем мястото и в света на гъвкавите методологии, начина и на приложение.

- *Мястото на АПОК в света на гъвкавите методологии* – Кога е удачно да се използва АПОК – в какви проекти по отношение на обем, критичност и предметна област. *Класификация на наличните ПОК* - по популярност , перспективност и други критерии.
- *Подходящи среди и алтернативни начини за прилагането на АПОК* - Сравнение с гъвкавата методологията Google, Благоприятни постановки за прилагането на АПОК, Развиване на ПОК обществото чрез прилагането на АПОК. използването на АПОК, като вместо ПОК „компонент" търсим комерсиален такъв дистрибутиран от друга компания. Или използваме АПОК за менажиране на разработката на компоненти в една компания, или за менажирането на модулното разработване на проекти с голям обхват.

7) *Заключение*
8) *Литература*

1.5. Списък на използваните фигури таблици, съкращения и термини

Таблица 1.5. Списък на термините и съкращенията:

Термини и Съкращения	Пълно наименование или превод
ПОК	Проект с отворен код
Scrum	Вид гъвкава методология
XP	Extreme programming
DSDM	Dynamic Systems Development Method
FDD	Feature-Driven Development
ASD	Adaptive Software Development
AM	Agile Modeling
OSS	Open source software
COTS	Commercial-off-the-shelf
PP	Pragmatic programming
RUP	Rational Unified Process
API	Application programming interface
CBSE	Component-Based Software Engineering
UML	Unified Modelling Language
Revision	Версия на продукта
Review	Преглед на единица функционалност с цел отсранявЗане на проблеми
Refactoring	Преразработване на единица функционалност с цел подобряване на работата и.
Crystal ...	Фамилия гъвкави методологии. (Orange, Clear …)
peer-to-peer	Компютър към компютър
Torrent file	Служебен файл съдържащ информация за дистрибутираните данни
DHT	Distributed Hash Table
tracker	Система служеща да следи и информира различните пийрове
peer	Мрежови клиент свързан към друг такъв

Съписък на използваните таблици:

Съписък на използваните фигури:

2. Сравнителен анализ на съществуващите гъвкави методологии

2.1. Предмет на сравнителния анализ

Целта на тази глава е да демонстрира разликите и приликите между отделните гъвкави методологии, за целта читателя трябва да е запознат с тях тъй като детайлното описване на всеки един метод би изместило темата. Затова:

1) първо ще дефинираме и охарактеризираме „гъвкавостта" в един софтуерен процес. Какво е необходимо за една методология за да бъде смятана за гъвкава.
2) второ ще ги анализираме и ще подчертаем техните прилики и разлики отделяйки повече внимание на процесите, практиките и обхвата на тези методологии, които са по близки до нашата.
3) В следващата глава на базата на тези анализи ще проведем изследването за необходимостта, спецификацията и ползите от нашия нов метод.

2.2. Концепция на гъвкавите методологии

2.2.1. Основни Дефиниции в гъвкавите методологии

Основните „стойности" на гъвкавостта са както следва:

1) *Индивидуализма и интерактивността са по-ценни от строго дефинирания софтуерния процес и програмните инструменти*: Гъвкавите методологии наблягат на комуникацията и отношенията между софтуерните разработчици. Използването на дадени програмни инструменти в една гъвкава методология би могло да бъде представено като едно „case study" за използването и. Такъв е случая и с нашата гъвкава методология където именно програмните инструменти ни вкарват в рамката на работния процес.
2) *Наличието на работещ софтуер е по важен от всеобхватната документация*: „Еженощно" работещ софтуер е необходим. Така наречения nightly build. В нашия случай документацията само отразява текущото състояние на проекта, без да описва подробно функционалността му, тъй като се счита че активните участници в проекта (Stakeholders) разполагат с цялата информация която е налична. Документацията е твърде активна за да се описва детайлно, тъй като проекта се развива в много аспекти едновременно. *Оказва се че активната комуникация между участниците в проекта замества документацията.* Разработчиците изкарват чести releases и се стремят да пазят кода лесен за разбиране доколкото е възможно.
3) *Набляга се на въвличането на клиента в жизнения цикъл на проекта*: Вместо да се подписват точни договори с ясни и строги клаузи. Клиента става по близък до разработчиците, това подпомага за поддържането на една приятелска атмосфера между всички участници в проекта. В нашия случай ние играем и ролята на клиента. Както клиента така и разработчиците трябва да бъдат добре информирани и компетентни достатъчно за да обмислят бъдещи промени по време на цикъла на разработка. Тоест трябва участниците в проекта да са готови дори и за архитектурна промяна. И практиките в гъвкавите методологии, програмните инструменти които използват са ориентирани така че да са подготвени за този момент.
4) *Реагирането на промени стои над необходимостта от следването на плана*: В предната глава разгледахме разликите между гъвкавите методологии и план-управляваните. Въпреки че в някои методологиите въвеждат „времевите кутийки" (интервал от време за имплементирането на дадена задача), способността да се реагира бързо на ново клиентски изискване или промени в плана поради вътрешни за екипа ни отличават гъвкавите методологии от план-управляваните.

Cockburn [15] (автор на „Crystal Clear”) дефинира ядрото на гъвкавите софтуерни методи за разработка и като набор от „леки и достатъчни” правила за разработката на проект, и като комуникационно и човеко-ориентирани правила. Лекотата означава да запазим маневреност, а достатъчността на правилата, както той се изразява означава да останеш в играта.

2.2.2. Кратко описание на по-популярните гъвкавите методологии

Според книгата на Song & Osterweil 1991”Comparing Design Methodologies through Process Modeling” [17] задачата за сравняване на дадени гъвкави методологии е доста трудна и по-скоро базирана на интуицията и субективния опит на автора на този анализ.

През 1992 гореспоменатите Song и Osterweil определят и два алтернативни метода за сравнение: неформален и „квази-формален”. Квази-формалния метод се опитва да премине през субективните ограничения на неформалния метод за сравнение.

Самия „Квази-формален метод за сравнение” може да бъде извършен по пет различни начина:

1) Описване на идеализиран метод и сравняване на останалите методологии с него поединично.
2) Определяне на някои важни особености основни за няколко методологии и сравняване на изискванията за всеки метод срещу тях.
3) Формулиране на „първичната хипотеза” за изискванията на гъвкавите методологии и пораждане на някакъв модел за изграждане на гъвкава методология от емпиричните данни на няколко методологии.
4) Дефиниране на метаезик като комуникационно средство и рамки от указания описани чрез този език, на базата на които описваш въпросните методологии.
5) По метода на общите проблеми, да опитаме да свържем тези методологии в някакви релации.

Song & Osterweil определят втория и четвъртия начин за извършване на сравнителния анализ като най-научни. Всъщност ние ще използваме четвъртия метод (този със дефинирането на метаезик като комуникационно средство). Тоест ще използваме топология от сорта: процес, роли, отговорности, практики, опит и възприемане, обхват на употребата и настоящи изследвания върху дадената гъвкава методология.

При „Сравнителния анализ” често оценяваме една методология спрямо друга. По-долу сме описали най-популярните гъвкави методологии, наблягайки на онези практики или роли които вероятно ще бъдат заимствани или сравнявани с дефинирани такива в нашето предложение за гъвкава методология. Тук са дефинирани и много термини използвани и по-късно.

Различните гъвкави методологии споделят до голяма степен една и съща философия. Те биват и дефинирани от сходни характеристики и практики. Но от гледна точка на имплементацията, всяка си има своя собствена рецепта от практики, терминологии и тактики.

Нека да се подсетим какво унифицира гъвкавите методологии като Scrum, Extreme Programming (XP), Crystal, Dynamic Systems Development Method (DSDM), Feature-Driven Development (FDD) …

2.2.2.1. Scrum

Scrum [10] е олекотена система за управление със широки възможности за менажиране и контролиране на итеративни проекти от всички типове. Ken Schwaber, Mike Beedle, Jeff Sutherland както и много други са допринесли значително за еволюирането на Scrum през последно десетилетие.

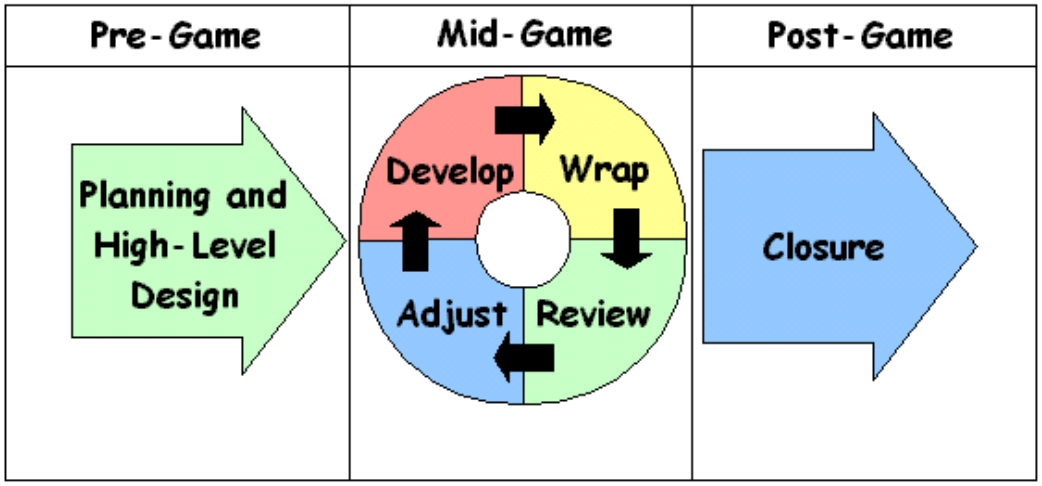
В Scrum “Собственика на продукта” работи заедно и близко с екипа за да се идентифицира и задава приоритети на функционалността във формата на така наречения „Product Backlog”. Въпросната „история на продукта” се състои от ново въвеждана функционалност, оправени грешки, нефункционални изисквания и .т.н. – каквото е нужно за съставянето на работеща софтуерна система. Със задаването на приоритети , зададени от „собственика на продукта”, многофункционални екипи оценяват задачите и се задължават да доставят потенциално ценни нововъведения в софтуера по време на така наречените „Спринтове” (Sprints) с продължителност 30 дена (в повечето случаи).

Веднъж „историята на продукта” за текущия „Спринт” да бъде установена, не може да бъде добавяна нова функционалност освен от екипа. След като „Спринта” е извършен, „историята на продукта” е анализирана и се задават наново приоритетите на задачите, ако е необходимо. И следващата по важност функционалност е отлагана за следващия „Спринт”.

Самият софтуерен процес при Scrum се дели на три части:

- Подготвителна фаза(Pre-Game Phase) – Тук се включва планирането, построяването на „product backlog list” на базата на честите му обновявания вследствие от задаването на приоритети и преоценка на усилията, които трябва да положи екипа за изграждането на проекта. В тази фаза се моделира и архитектурата на проекта.
- Фаза на разработка (Mid-Game) – Тук се взимат новите изисквания извадени от „product backlog list”-а, които дават старта на следващия спринт, където тези изисквания се анализират, прави се дизайн, системата „еволюира”, тества се и се прибавя нова функционалност към продукта. Или се преминава към следващата фаза.
    - Develop – разработка на продукта – имплементиране на нова функционалност, тестване и документиране.
    - Wrap – интеграция и оценка на еденицата въведена функционалност.
    - Review – преглед на извършената работа в спринта.
    - Adjust – следене, преоценка и отразяване на евентуални промени в изискванията на клиента в плана на проекта.
- Заключителна фаза (Post-Game Phase) – Тук се извършва интеграцията, системното тестване, изготвянето на документация и евентуалното изготвяне на крайния, завършен продукт, когато стойностите на променливите време, качество, конкурентоспособност и цена ни задоволяват.

На фиг. 2.2.2.1.1 е представена схема на софтуерния процес на SCRUM:



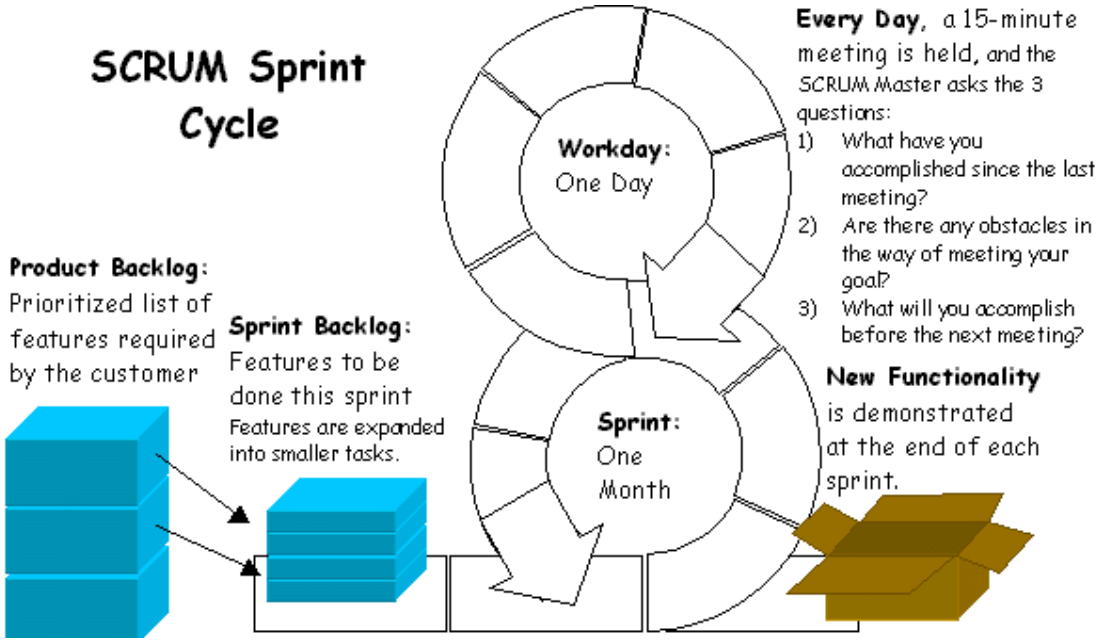
Фигура 2.2.2.1.1: Схема на софтуерния процес на SCRUM

Има шест лесно идентифицируеми роли в Scrum, които имат различни задачи и цели в софтуерния процес: Scrum Master, Product Owner, Scrum team, Customer, User и Management. Тези роли са дефинирани и представени от Schwaber и Beedly (2002).

По идея Scrum не се нуждае от специални практики или програмни инструменти, които да използва, но за да намали непредсказуемостта и хаоса Schwaber (1995) дефинира такива:

- “product backlog” – Всичко което ни е необходимо да знаем за крайния продукт базирано на текущите ни знания
- Effort estimation – Product Owner-а и Scrum Team-а са отговорни за оценяването на въпросния „backlog”
- Спринт (Sprint)
- Среща за планирането на спринта (Sprint Planning meeting)
- Sprint Backlog – това което ще се имплементира в следващия Спринт.
- Ежедневна среща (Daily Scrum meeting) – 15 минутни срещи за да се следи ежедневно прогреса до проекта.
- Среща за преглед на спринта (Sprint Review meeting) – демонстрация на извършената работа в един спринт.

На фиг. 2.2.2.1.2 е представена схема на Sprint цикъла:



Фигура 2.2.2.1.2: Sprint цикъл

Scrum се е доказал като удобна гъвкава методология и при използването и от множество вързани екипи в огромни организации (над 800 човека). Но е ефективна методология и за малки екипи. Използва се и от екипи по малки от 10 човека.

2.2.2.2. Екстремно програмиране (Extreme Programming(XP))

Гъвкавата методология XP, описана за пръв път от Кент Бек, се е превърнала в една от най-популярните гъвкави методологии, доставяща ни висококачествен софтуер бързо и последователно. Тя се гради на базата на голямата въвлеченост на клиента в проекта, бързия отговор от клиента, непрекъснатото тестване и планиране, и близката работа в екип. „Работещо копие" на софтуера се доставя на много кратки интервали (от 1 до 3 седмици).

Оригиналната XP рецепта е базирана на четири основни ценности : простота, комуникация, отговор от клиента, „кураж"(да поемаш отговорност), и 12 практики които биват следвани в една или друга степен:

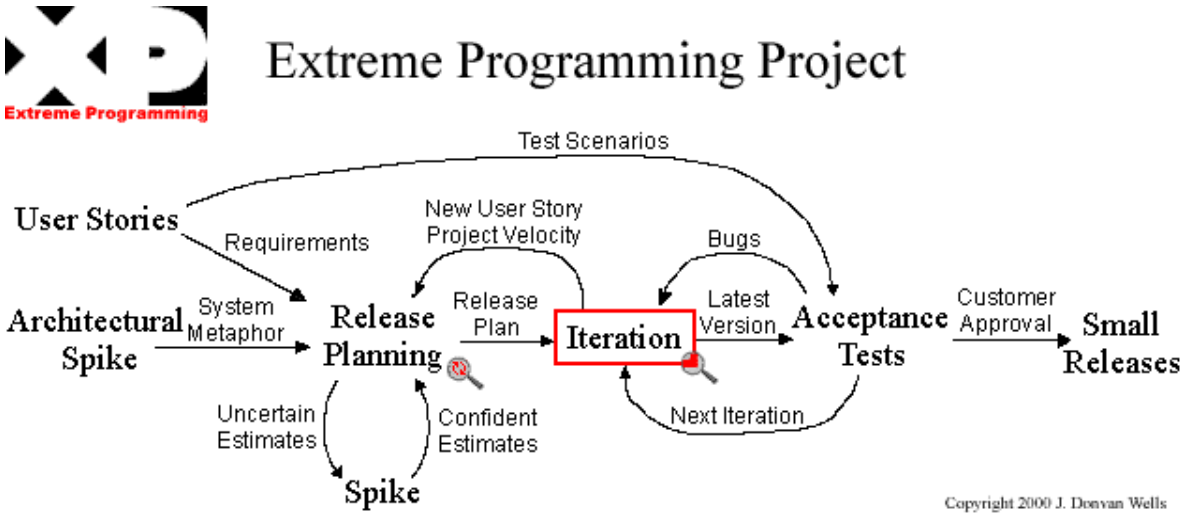
1) Планираща игра (програмистите оценяват усилието необходимо за изработването на дадена функционалност, след което клиента решава дали дадена функционалност е важно сега да бъде изработена)
2) Малки по обем промени до следващото работещо копие (small releases)
3) Клиентски тестове за приемственост на системата (Customer Acceptance Tests)
4) Прост Дизайн
5) Програмиране по двойки.
6) Разработка управлявана от тестването. (Test-driven development).
7) Преработка на кода (Refactoring)
8) Непрекъсната интеграция
9) Колективна собственост на кода
10) Дефиниране на стандарти за кодиране
11) Metaphor (именуването на класовете и обектите в системата)
12) Sustainable Pace (постоянно и непрекъснато темпо на работа)
13) 40 часова работна седмица

В XP, „клиента” работи заедно с екипа за да дефинира и зададе приоритети на обединения от функционалност, наричани още „User Stories”. Разработващия софтуера екип оценява, планира и доставя най-високо приоритетните „User Stories” във формата на работещ и тестван софтуер на базата на итерационния принцип. За да максимализираме продуктивността, XP практиките ни осигуряват олекотена система за управление на работата („framework”), която да ни подпомага в работата на екипа и да ни осигури висококачествен софтуер.

Жизнения цикъл при XP се състои от пет фази:

- Exploration (Изследване) – Потребителите пишат своите “story cards” (неща които да бъдат направени) които да бъдат включени в първия работещ продукт (release). Всяка карта с „потребителски истории” описва някаква функционалност която да бъде добавяна в системата. През това време екипа се запознава с програмните инструменти, практиките и технологията която ще използват в проекта. Технологията която ще се използва се тества и се изследват архитектурните възможности като се построява „прототип”(работещо копие на продукта/ системата, което служи само за демонстрация на възможностите и) на системата. Според запознатостта със системата отнема от две седмици до няколко месеца.
- Planning (Планиране) – установява се приоритета на потребителските истории. Програмистите определят колко е сложна за реализация всяка една потребителска история. Планирането отнема само няколко дена.
- Iterations to release planning (Итерации до пускане в експлоатация) – Тази фаза се състои от няколко итерации. Определя се приоритета на наслагването във времето на потребителските истории. Задачите се разбиват на итерации, чието имплементиране да отнема от една до четири седмици. Първата итерация създава система ползваща архитектурата на цялата система. Потребителят решава кои потребителски истории да бъдат избирани за всяка итерация. функционалните тестове се създават и изпълняват от потребителите в края на всяка итерация. След края на последната итерация системата е готова за влизане в следващата фаза „Пускане в експлоатация”.
- Productionizing (Пускане в експлоатация) – В тази фаза се извършва допълнително тестване и проверка за производителността на системата преди тя да бъде пусната в експлоатация. И в тази фаза нови решения или промени могат да бъдат взимани в предвид, и да връщат проекта на по предна фаза. Гореспоменатите идеи и предложения са документирани за последващото им използване в фазата на поддръжката.
- Maintenance (Поддръжка) – Предвидена за изпълняването на клиентски задачи за поддръжка и донастройка. Имплементирането на задачите от тази фаза не изискват целия човешки ресурс на екипа.
- Death (край на проекта) – Когато потребителя няма какви потребителски истории за имплементиране да предостави. До тази стъпка може да се достигне само когато стабилността и производителността на системата са на ниво. Системата е напълно документирана, и не са предвидени последващи промени в архитектурата, дизайна или кода и. До тази фаза може да се стигне преждевременно, ако работата по проекта бъде прекратена.

Ще опишем най-явния начин за използване на XP при стартирането на един проект:
Първо започваме да събираме „потребителските истории”(user stories) докато достигнем до идеята за разрешаването на проблемите при разработката на проекта (spike solutions). Тази фаза трае по-малко от месец. След което позовавайки се на Metaphor-а на системата, насрочваме среща за организирането на политиката по изкарването на нови версии на продукта - “release planning”. В тази среща участват разработчици , менажери и клиент. След което започва итеративната разработка на проекта предхождана от „iteration planning meeting”. Всяка една итерация трае от една до три седмици. Селектирани са определени потребителски истории за всяка една итерация от плана на разработка „release plan”, с определени от клиента приоритети. При проиграването на „тестовите сценарии” провалените тестове за приемственост (acceptance tests) от предишните итерации също са селектирани. След приключването на дадена итерация се измерва прогреса на проекта (project velocity) на база на стойностите от останалите итерации. По долу на фиг. 2.2.2.2 е дадена схема на софтуерния процес при XP:



Фигура 2.2.2.2: XP проект

Beck (1999) [11] дефинира следните роли и отговорности: Програмист, Клиент, Тетстер, Супервайзер следящ работата(Tracker), Отговорник за самия софтуерен процес (Coach), Консултант и Мениджър.

2.2.2.3. Crystal

Crystal [15] е една от най-олекотените и лесно адаптирани гъвкавата методологии. Crystal се състои всъщност от цяла фамилия Crystal методологии (Crystal Clear, Crystal Yellow, Crystal Orange, и т.н.), чиято уникалност е дефинирана от няколко фактора като размера на екипа, критичността на системата, приоритетите на самия проект. Фамилията Crystal адресира тезата че всеки един проект може да изисква споен набор от политики, практики и процеси спомагащи за посрещането на уникалните особености на проекта.

Някои от основните принципи на Crystal включват работа в екип, комуникация, простота както и честото донагласяне и подобряване на процеса на работа. Както другите гъвкави методологии Crystal презентира честото доставяне на работещ софтуер, висока въвлеченост на крайния потребител, високо ниво на адаптация, и премахването на бюрокрацията други разсейващите ни фактори.

Трудно е да се определят ролите при Crystal фамилиите тъй като едно лице може да изпълнява няколко роли. Според Cockburn в Crystal Clear ролите които изискват отделна личност за изпълнението им са: спонсор, главен програмист дизайнер, дизайнер програмист и потребител. Но и тези роли включват множество под-роли. Пак според него в Crystal Orange биват дефинирани и няколко нови роли: дизайнер на потребителския интерфейс, дизайнер на базата данни, експерт потребител, подобрител на техническата част, бизнес анализатор, архитект, ментор дизайнер, технически документатор, тестер, отговорник за подобряване на „преизползването" на фрагменти програмен код. Някои индивиди може да изпълняват повече от една роля.

Докато софтуерния процес при различните Crystal методологии се различава значително, то практиките, които използват могат да се посочат по-общо (за цялата фамилия):

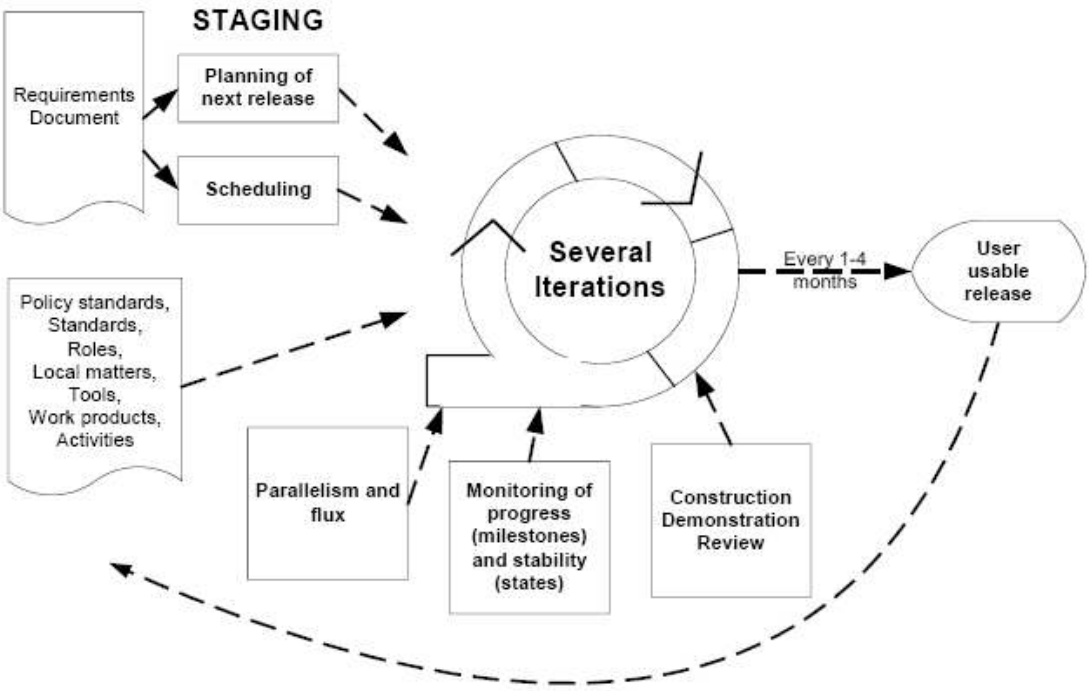
- Staging – Планирането на следващата итерация на системата.
- Revision and Review – Всяка „инкрементална стъпка" съдържа няколко итерации. А всяка итерация може да бъде описана от няколко вида активност (activities) – конструкция, демонстрация, и преразглеждате на целта на итерацията.
- Monitoring – Прогресът е следен на базата на стабилността на екипа и свършената от него работа.

- Parallelism and Flux – Според стабилността на екипите, много екипи да разработват максимално много паралелни задачи.
- Holistic diversity strategy – разделянето на големите екипи на многофункционални групи. Тоест един екип да може да има няколко специализации. Този подход позволява също и формирането на съвсем малки екипи със специфично ноу-хау с идеята да могат да комуникират и да се координират лесно с по-големи екипи.
- Methodology-tuning technique - подобряване на процеса на разработка. Със всяка следваща итерация екипите все по-добре разбират и използват софтуерния процес.
- User viewings – две потребителски гледни точки са предлагани винаги когато е възможно.

Ще разгледаме софтуерния процес на Crystal Orange за да опишем изготвянето на инкремент. Освен гореописаните практики Crystal Orange, както и Crystal Clear дефинират следните понятия:

1) Политика на използване на методологията (Policy standarts):
   - Редовно доставяне на еденица функционалност на клиента
   - Следене на прогреса по проекта на базата на редовното доставяне на еденица функционалност и брой на взетите важни решения
   - Директно въвличане на потребителя
   - Автоматични регресивни тестове на функционалността
   - Разглеждане на две потребителски гледни точки за всяка нова версия на продукта
2) Управление на работния продукт (Work product) – политика на разработка на версиите, ясен обектен модел, потребителски ръководства, test cases, …
3) Процедурите необходими за реализирането на проекта като такъв (Local matters).
4) Приложни програми (Tools)– за управление на версиите, тестовете, комуникацията, следенето на прогреса по проекта.
5) Стандарти (Standarts) – Нотация, вид на дизайна, форматиране, качество.

По долу фигура 2.2.2.3 илюстрира изготвянето на един инкремент използвайки софтуерния процес на Crystal Orange:



Фигура 2.2.2.3: Crystal Orange диаграма на софтуерния процес

Според “критичността на системата” Crystal дефинира различни димeнсии “C (Comfort), D (Discretionary money), E (Essential money), L (Life)”. А различните по вид Crystal методологии се отнасят за различни по обем проекти. Ако разполагаме с 6 човека – логичния избор би бил Crystal Clear, с 20 – Crystal Yellow, с 40 – Crystal Orange, с 80 или повече – Crystal Red. Повече информация относно Crystal можете да прочетете книгата на Alistair Cockburn „A Human-Powered Methodology for Small Teams”.

2.2.2.4. Dynamic Systems Development Method (DSDM)

Използването на DSMD [13] датира от 1994, когато DSDM се опитва да отговори на нуждите на индустрията за стандартна система за управление на съставянето на софтуера. За това до този момент се е грижела методологията RAD (Rapid Application Development). RAD става доста популярната в началото на деветдесетте. Но отношението на RAD към доставката на софтуер се разви по един доста неструктуриран начин. Вследствие на това бе създаден „DSDM консорциума”, който през 1994та се събра с цел дефиниране за индустрията на „бързата”(rapid) софтуерна разработка система от правила за управление. Оттогава методологията DSDM е еволюирала и узряла да ни даде изчерпателна фондация за планиране, менажиране, изпълнение, и скалиране на гъвкави и итеративни софтуерни проекти за разработка.

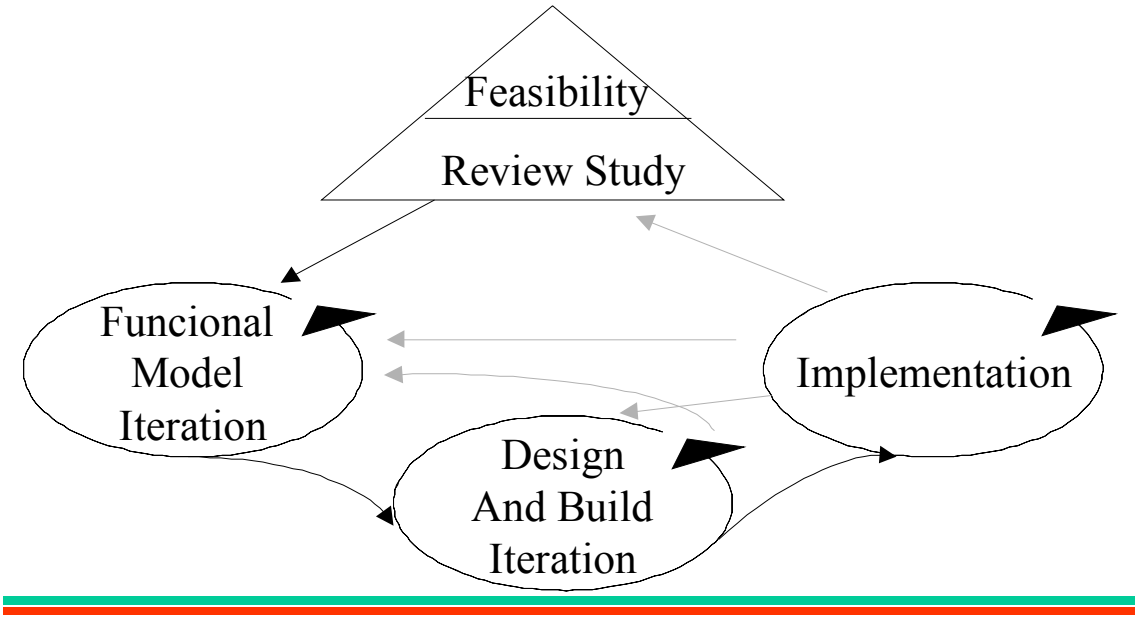
DSDM е базирана на девет ключови принципа които се въртят основно около бизнес нуждите, активната въвлеченост на потребителя, “упълномощените екипи”, честата доставка на работещ софтуер, интегрираното тестване и съдействието на всички участници в проекта. DSDM специално посочва "годността за бизнес използване" като главен критерий за доставка и приемане на дадена система, фокусирайки върху способността за 20 % от времето, необходимо за пълно инсталиране на системата, да бъде инсталирана такава част от нея, която ще позволи използване на 80 % от пълния й капацитет.

Софтуерния процес при DSDM се състои от пет фази:

- Изучаване на проекта (feasibility study) – Класифицира се самия проект и се изследва дали да се използва DSDM, определят се готовността на екипа за поемането на този проект и се прави оценка на риска. Два документа се изготвят. Документ за „Изпълнимост на проекта” и скициран план на проекта. Възможно е и да бъде изваден бърз прототип, ако технологията или бизнес изискванията не са много ясни. Тази фаза трае по-малко от месец.
- Анализ на бизнес изискванията и технологиите (business study) – Експерт потребители се събират заедно с екипа за да оценят по приоритет различните аспекти на проекта. Бизнес процесите и потребителските класове са описани в така наречения „бизнес среда дефиниции” документ. Идентификацията на „потребителските класове” спомага за въвличането на клиента в ранна фаза на проекта. В тази фаза се изготвят и документите „дефиниране на системна архитектура” (първата скица на архитектурен модел) и „общ план на прототипа”.
- Итерации на построяването на функционалния модел – Първата итеративна и инкрементална фаза. Построяваме прототипа. Доусъвършенстваме модела на анализ на базата на опита придобит при изграждането на прототипа. Прототипа не е необходимо да бъде тотално премахван от програмния код на проекта, би могъл да служи като основа за изграждане на проекта.
- Итерации на построяване на дизайна и изграждане на системата – тук се изгражда системата. Крайния резултат от тази фаза е изграждането на изтествана „система” покриваща поне минималните изисквания. Дизайна на системата и прототипа са преглеждани обстойно и от потребителя, като по нататъшната разработка е базирана и на коментарите на потребителя.
- Имплементация – При тази фаза освен крайното имплементиране на проекта, се пуска системата и в експлоатация. Вътрешното обучаването на потребителите се оставя на крайния потребител. Част от тази фаза се състои и в изготвянето на „потребителски наръчник” (user manual) и общ документ описващ проекта като цяло. Ако системата не удовлетворява всички изисквания трябва да се върнем и към по ранни фази за да ги удовлетворим.

Първите две фази са силно взаимнозависими и се извършват само веднъж. Докато последните три фази, при които се обобщава модела на разработка и се извършва самата разработка на проекта, са итеративни. „Времевите интервали” са предефинирани периоди от време. Всяка итерация трябва да приключва, когато свършва някой времеви интервал.

Времето отделено за всяка итерация е планирано предварително, както и резултатите от успешното приключване на итерацията са входна точка за започването на други итерации.(имаме наличие на планиране подобно на RUP (които няма да описваме, а само ще посочваме като „едната крайност" в изследваните от нас методологии.)). Продължителността на тези „времеви интервали" е от няколко дена до седмици.По долу на фиг. 2.2.2.4 е дадена схема на софтуерния процес на DSDM разделен на три подпроцеса:



Фигура 2.2.2.4: DSDM диаграма на софтуерния процес

Ролите в DSDM изпълнявани от потребители и разработчици са около 15. Stapleton (1997) описва следните основни роли:

Разрабочик и Опитен разработчик (senior developer), Технически координатор (дефинира системната архитектура, представлява също и „качествен контрол"), „Потребител посланик"(отговорника за уточняването на потребителската гледна точка във всичките и аспекти), „Потребител съветник"(потребител специализиран в използването на дадена функционалност, представящ неговата специфична гледна точка). Главен потребител(аналитик(Visionary). Отговорник за следването на главната идея на проекта, инициатор на проекта.) , „Спонсор изпълнител"(потребителя които финансира проекта. Най висша инстанция във взимането на решенията).

Няма да описваме Практиките, тъй като се дублират с тези на останалите гъвкави методологии.

Изискванията към софтуера са ясни с високо ниво на детайлност още когато проекта е в ранна фаза. „Преработката" е внедрена в процеса, и всички промени на ниво разработка са обратими. Изискванията са планирани и удовлетворявани в малки фиксирани интервали от време (fixed-length time-boxes), също известно като итерации. При DSDM са зададени приоритети на клиентските изисквания по MoSCoW правилата:

M – Трябва да има изисквания (Must have requirements).
S – Би трябвало да ги има ако е възможно (Should have if at all possible).
C – Би могло да ги имам, но не е критично (Could have but not critical).
W – Не точно сега, но евентуално по нататък.(Won't have this time, but potentially later).

Цялата критична работа трябва да бъде извършена в DSDM проекта. Важно е също и не всяко изискване в проекта да бъде критично. Във всяка една такава „времева кутийка" по-малко критичните неща са включвани и ако е необходимо, могат да бъдат отлагани за по-късна итерация.

2.2.2.5. Feature-Driven Development (FDD)

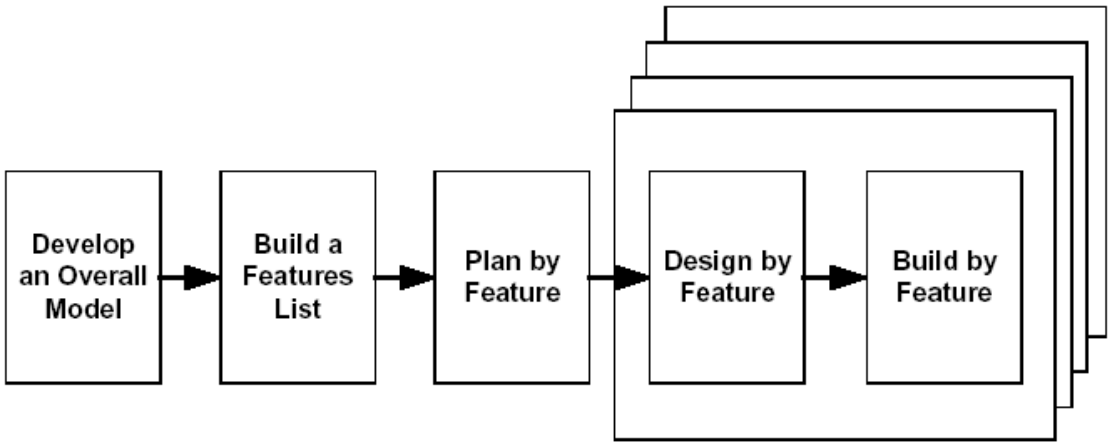
Идеите на FDD [12] са създадена и по късно свързана в една методология в началото от Jeff De Luca със съдействието на M.A. Rajashima, Lim Bak Wee, Paul Szego, Jon Kern and Stephen Palmer. FDD е модел базиран, базиращ се на къси итерации софтуерен процес, който започва с установяването на „формата на модела". След което се стъпва в серия от двуседмични итерации: „дизайн според функционалността(feature), изграждане според функционалността". Ново въвежданите парчета функционалност, са малки, „полезни и необходими, според клиента".

FDD прилага доста линеен софтуерен процес, итеративност се наблюдава само при дизайна и разработването на различната функционалност.

FDD се състои от пет под процеса:

- Разработването на общия модел – системата се моделира със съдействието на експертите по предметната област които предоставят на архитекта на системата така нареченото „walkthrough". Възможностите на системата от гледна точна на потребителя ("use cases") се дефинират. Разработчиците разделят предметната област на различни части и съставят общия модел на системата.
- Изготвянето на списъците с функционалност – на базата на списъка с изисквания, обектния модел и „the walkthrough" се изготвя списъка с функционалност, която се разпределя в различни групи (така наречените обединения на функционалност). Тези списъци от функционалност са преглеждани от потребителите и спонсора на системата за валидност и пълнота
- Планиране на функционалността – тук се задават приоритети и разпределят във времето различните парчета функционалност от приоритет и независимост (от останалите). След като опитните програмисти идентифицират и класифицират класовете функционалност те се задават за реализация на отделните разработчици (описаните по долу собственици на клас).
- Дизайн на Функционалност* -
- Изграждане на функционалност* -
    - (последните два процеса са циклични): Малка група от функционалност бива реализирана от собствениците на клас за всяка итерация (за от няколко дена до две седмици). Този итеративен процес включва задачи като инспекция на кода и дизайна, кодиране, тестване на модулите, интеграция. След като една итерация мине успешно изградената функционалност се включва в следващата версия на продукта.

Схемата на софтуерния процес е показана на фиг. 2.2.2.5:



Фигура 2.2.2.5: Софтуерен процес при FDD

Дизайнът на процеса на разработката при FDD като цяло представлява доставянето на клиента на горепосочените „ново въвеждана функционалност" следвайки осем практики:

1) Обектно моделиране в дадена област (Domain Object Modeling)
2) Разработка според функционалността (Developing by Feature)
3) Компонентно или класово притежание на кода (Component/Class Ownership)
4) Екипи разработващи функционалност. (Feature Teams)
5) Редовни инспекции на кода и екипите (Inspections)
6) Контрол на програмния кода на проекта (Configuration Management)
7) Редовно компилирана и тествана функционалност. Често компилиране до работещо копие. (Regular Builds)
8) Видимост на общия прогрес и текущите резултати. (Visibility of progress and results)

Palmer и Felsing (2002) класифицират FDD ролите и отговорностите в три категории:
1) Ключови роли
2) поддържащи роли
3) допълнителни роли.

Шестте ключови роли в FDD са мениджър на проект, главен архитект, лидер на екипа, опитен програмист, собственик на клас (неопитни програмисти реализиращи малки парчета функционалност), специалист по предметната област. Петте поддържащи роли са наблюдател на прогреса, експерт по дадена технология, контролиращ версиите на продукта, разработчик на малки програмни инструменти помагащи на разработчиците и системните администратори. Трите допълнителни роли са тестерите, инсталаторите и техническите документатори.

FDD препоръчва специфични програмистки практики като честите компилации на работеща функционалност и тази за притежанието на кода от компоненти или класове. FDD бележи доста добри резултати при по-големи екипи зареди лесната му и ясна „скалируемост". За разлика от другите гъвкави методологии, FDD описва специфични и доста кратки фази на работа, отнасящи се за различни „области от функционалност", които могат да бъдат „преминавани" от различни екипи. Тези фази могат да бъдат: обхождане на областта от знания, дизайн, кодиране, инспекция на кода, и обявяване на наличност на работещо копие. Първата практика в последно време започва да се развива и интерес към нея се проявява все повече и повече и извън FDD обществото. За повече информация можете да разгледате книгата на Eric Evans „Domain-Driven Design".

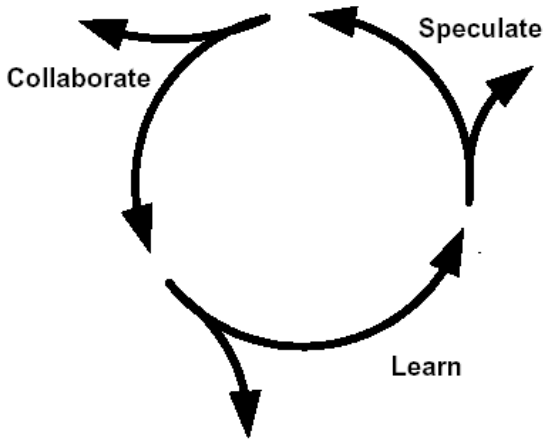
2.2.2.6. Adaptive Software Development (ASD)

ASD [14] методологията е разработена от James A. Highsmith, като преработка на предшественика си „Radical Software Development" създадена от съавторите Highsmith и S. Bayer и представена през 1994 г. ASD се фокусира върху изграждането на големи, комплексни системи. Методологията силно набляга на инкременталната и итеративна разработка, с непрекъснатото изготвяне на прототипи. Фундаменталната идея на ASD е „балансиране на ръба на хаос" – или предоставяне на „система за управление"(framework) необходима и достатъчна за да не позволи на проекта да изпадне в хаос, и достатъчно лека за да стимулира креативността и бързата обработката на спешни събития.

Софтуерния процес при ASD се поема от цикъл състоящ се от три фази: Спекулация (Speculate), Сътрудничество (Collaborate), Поучаване (Learn). Фазите са именувани така за да специфицират по-точно промяната във времето.

- Терминът „Спекулация" е използвана на мястото на „Планиране". Като под „план" се разбира сценарии където несигурността е главната слабост, от която се поражда провалът.
- Терминът „Сътрудничество" пък подчертава важността на добрата работа в екип, при разработването на една често променяща се система.
- „Поучаването" акцентира върху нуждата от изучаване и реагиране на грешките, и факта че изискванията може да се променят по време на разработката.

Схемата на софтуерния проецес (фиг. 2.2.2.6) ни показва една непрекъсната „обратна връзка". Усъвършенстването на комуникацията, подобряването на познанието и способността да се взимат правилни решения на всеки един участник в разработката, представляват тази „обратна връзка" целяща самоусъвършенстване на същите тези участници в проекта.



Фигура 2.2.2.6: Софтуерен процес при ASD

Софтуерния процес започва с „въвеждане”(Initiation), където се дефинира “мисията на проекта”. „Мисията” дава „курс” на проекта, и всички разработчици се придържат към него докато не изпълнят мисията. Едно от най-важните неща които трябва да се знаят при дефинирането на мисията е каква информация е необходимо да носи проекта. Три са важните аспекти при дефинирането на мисия: потребителската гледна точка, начален списък с данните на проекта и скица на спецификацията на продукта. Трите аспекта са описани подробно в “Highsmith 2000”.

ASD е по-скоро компонент-ориентирана отколкото задача-ориентирана методология. На практика се интересува повече от получените резултати (качествени резултати), отколкото от задачите и процеса използван за формирането на тези резултати. Тези резултати се постигат в „адаптивни цикли на разработка” , съставляващи фазата „Сътрудничество”, където няколко компонента могат да бъдат разработвани едновременно. Планирането на тези цикли е част от итеративния процес, като дефинициите на компонентите непрекъснато се изменят в отговор на получената нова информация или на изменените изисквания към компонента.

Фазата „Поучаване” се състои от непрекъснати прегледи на качеството, целящи демонстрация на функционалност, на софтуера разработван в тези цикли. За изготвянето на тези „прегледи” е необходимо да присъства група от експерти или така наречените „клиент фокусирани групи”. Когато разработката по проекта приключи, или когато излезем от цикъла: спекулация, сътрудничество, поучаване стигаме до крайния преглед на качеството и пускането на продукта в експлоатация. Изготвя се postmortem (крайните поуки от проекта, историята и развитието на проекта, архитектурата му ...) документ, като живо доказателство за изминатия път. Важните моменти които характеризират ASD могат да бъдат посочени:

1) „Мисия-управление на проекта” – При впускането в нов цикъл на разработка винаги се взима в предвид „мисията на проекта”.
2) „Компонентно базирана разработка” – разработването на софтуера се фокусира върху изграждането на работещ софтуер, или изработването на системата парче по парче.
3) Итеративност – Типичната “Waterfall” методология работи само в перфектно ясна и дефинирана среда. Самото впускане в реализацията на наглед най-доброто моментно решение предполага множество преразработване на една и съща функционалност. Но ASD е методология която движи проекта напред на базата на „проба грешка”. За намаляването на грешките опита от предишни проекти и високото ниво на самообучение са изключително важни.
4) „Времеви интервали” – слагането на подходящо фиксирани крайни срокове може да облекчи планирането и в тази методология. В най-ранната фаза се прави разпределянето на задачите и цялостното изготвяне на предварителния план.
5) „Толерантни към промени” – Тук по важно е да можеш да се адаптираш към промените отколкото да контролираш наличието им. За да изградят модифицируема система разработчиците трябва да оценят колко често се променят проблематичните компоненти.
6) „Риск-управляван проект” - Разработването на високорискови компоненти трябва да започне колкото е възможно по-бързо.

ASD предполага доста малко практики за „ден за ден” разработка. Highsmith посочва три практики: итеративна разработка, компонентно-базирано планиране, и прегледи на проекта от гледната точка на клиента. Но принципно при ASD е трудно да бъдат идентифицирани не само практиките но и ролите и отговорностите. Повечето роли, отговорности и практики в книгата на Highsmith са дадени като приемери за това какво би могло да се направи.

2.3. Сравнителен анализ на съществуващите гъвкави методологии по ключови особености

Първо ще категоризираме различните гъвкави методологии по зрялост обръщайки голямо внимание на това кога са били създадени.

DSDM и Scrum са представени още в началото на 90те. Но Scrum методологията е представяна все още в „доизграждаща се” фаза. Други гъвкави методологии представяни в „доизграждаща се” фаза са FDD, Crystal, PP и ASD. Но за разлика от Scrum, където просто научните разработки не са достатъчни, причината горепосочените методологии да бъдат причислени към тази категория е че са по-малко са използвани. AM е предложен през 2001 и затова може да се смята че е най младата от „доизграждащите се” гъвкави методологии. XP, RUP, OSS и DSDM от друга страна са методологии които са добре документирани и за които има достатъчно налична литература. Освен това всеки от тях има своя група поддръжници (активно общество), които ги доразвиват. Затова ще окачествим тези методологии като „активни”. Никоя от гореизброените методологии не се смята за остаряла.

В таблица 2.3 ще демонстрираме използването на „втория квази-формален метод” за сравнителен анализ описан в горната точка.

Таблица 2.3: *Сравнителен анализ на съществуващите гъвкави методологии по ключови особености*

Методология	Ключови особености	Специфични особености	Недостатъци
ASD	Адаптивна култура, сътрудничество, мисия – изграждането на компонент.	Създаване на екип от „близко свързани” (комуникацията е лице в лице) индивидуалисти.	ASD е повече концепция и култура отколкото софтуерна практика.
AM	Гъвкава култура, организирана поддръжка, по интензивна комуникация по-малко документация в сравнение с останалите гъвкави методологии, простота.	Прилагат се „гъвкавите” принципи и при моделирането.	Това е добра неосновна философия за дизайнерите. Но сама по себе си не е достатъчна.
Crystal	Фамилия от методи. Всеки Crystal method запазва същите основни принципи. Техники, роли, програмни инструменти, и стандартизирани подходи.	Възможността да избереш на удачния Crystal method, с оглед на това кой ни е критичния ресурс.	Все още твърде незрял за оценка. Crystal Clear се използва(засенчва останалите методи от фамилията).
DSDM	Подобен на RAD. При менажирането се ползват времеви блокчета. Упълномощени DSDM екипи. Активен консорциум управляващ развитието на методологията.	Първата гъвкава методология използваща прототипи. Интересни потребителски роли: „посланик”, „човек с цялостна визия за проекта”, „съветник".	Само въпросния членовете на „консорциума” имат допир до „материята” на прилагания метод.
XP	Клиентски ориентирана разработка. Прилага се	Refactoring – премоделиране на	Отдаване на малко внимание на „общия

	основно за проекти поемани от екипи от няколко човека. Ежедневен build.	системата за да подобрим производителността и улесни внасянето на бъдещите промени.	вид на проекта" и менажерските практики.
FDD	Пет-стъпкова методология. Разработка на принципа на добавянето на нова функционалност. (Feature-based) Много къси итерации: от няколко часа до 2 седмици.	Простота на нововъведенията. Дизайн и имплементация на системата като функционалност. (Обектно ориентирано)	FDD се фокусира само върху дизайна и имплементацията.
OSS	Географски отдалечена разработка от доброволци. Често определянето на авторството на проекта е по-голямо предизвикателство от комериализирането на продуктите.	Лицензионна политика. (Употреба и притежание на проекта). Кодът е свободно достъпен за всички участници.	*OSS не е методология сам по себе си*. А възможност за трансформиране на OSS принципите до разработка на комерсиален софтуер.
PP	Набляга на прагматизма. Теорията на програмирането не е толкова важна. Високо ниво на автоматизация във всички аспекти на програмирането.	Конкретни и емпирично проверени за валидност „сказки" (съвети, принципи).	Неподходящ за разработка на сложни системи. Фокусира се върху практиките все пак.
RUP	Завършена модел за разработка на софтуер. Включващ в себе си и поддържаните програмни инструменти. Набляга се на активността.	Бизнес моделиране. Създаване на цяла фамилия от средства за поддръжка.	Няма граници в обхвата на проектите. Липсва описание на това как да свържем и специфицираме нуждите ни.
Scrum	Независими, малки, самоорганизиращи се екипи. 30 дневен цикъл на разработка.	Измества парадигмата „дефиниран и преизползваем" към „новата Scrum разработка гледна точка"	Недетайлизирани acceptance и integration тестове.

ASD е най-абстрактната методология от гледна точка на софтуерните разработчици. AM, XP и прагматичното програмиране като цяло ни представят един доста практически ориентиран подход. Всички те съдържат достатъчно голям набор от емпирично проверени за валидност практики, които са се доказали и наложили с времето. Crystal фамилията гъвкави методологии е едно ясно формулирано предложение за дизайн на проекта управляван от принципа наложен от даден критичен ресурс или размера на проекта. DSDM се различава доста от останалите методологии като концепция поради необходимостта от използването на „прототипи". Тази гъвкава методология изкарва на преден план и няколко „роли" като „съветник", „посланик", „човек с цялостна визия на проекта" чужди за останалите гъвкави методологии. Основния подводен камък в DSDM е наличието на този DSDM консорциум които единствен има достъп до документите описващи аспектите и начина на прилагане на методологията. FDD не се опитва да ни даде общото(всичко в едно) решение на софтуерната разработка. А се фокусира в пет-стъпков модел на работа базиран на идентифициране, дизайн и имплементиране на функционалност (да не се бърка с Waterfall модела). И въпреки този пет стъпков модел, FDD за разлика от DSDM и RUP не покрива най-ранния етап на жизнения цикъл на софтуерната разработка: „Създаването на концепцията". (виж таблица 1.3.2.2). Не покрива и крайните стъпки в този жизнен цикъл: "Acceptance test" и „System in use". Менажирането на проекта при Scrum разчита на самоорганизиращите се независими екипи имплементиращи функционалност в накъсани интервали от време (един месец) или в така наречените „спринтове" (Sprints). Метода все още не ни предоставя достатъчно добре дефинирани integration и acceptance тестове.

Подобно на ASD, ***OSS*** *е повече философия на софтуерна разработка отколкото метод. В доста голяма степен ние ще изградим нашата гъвкава методология именно на базата на тази философия.* Специална особеност при OSS е лицензната практика, която изисква source кода на проекта да е достъпен за всички участници в проекта. Той може да бъде четен, модифициран, компилиран, и редистрибутиран свободно.

RUP не е гъвкава методология за софтуерна разработка, но би могла да бъде такава ако се модифицира. (използван е в сравнителния анализ единствено за „доизясняването на общата картинка"). RUP съдържа бизнес моделиращи практики подобни на тези в DSDM. DSDM подобно на RUP покрива изцяло жизнения цикъл на проект по отношение на процес и менажиране на проекта. (виж таблица 1.3.2.2).

2.4. Сравнителен анализ на съществуващите гъвкави методологии по жизнен цикъл

Гъвкавите методологии рядко обхващат всички фази от жизнения цикъл на софтуерна разработка. Таблица 2.4 показва кои фази на софтуерната разработка от кои гъвкави методологии се поддържат. Всяка методология е разделена на три блока.

Първия блок индицира дали методологията ни позволява да поддържаме мениджмънт на проекта във съответната фаза. (Ще го бележим със съкращението „менаж".) Втория блок ни показва дали „процеса" предложен от методологията за използване се покрива във съответната фаза. Ще го бележим със съкращението „процес". Третия блок идентифицира дали методологията застъпва практики и activities в различните фази от жизнения цикъл. Ще го бележим със съкращението „практики".

Таблица 2.4: *Сравнителен анализ на съществуващите гъвкави методологии по жизнен цикъл*

Фаза/ Методология	Блокове	Създаване на концепцията	Спецификация на изискванията	Дизайн	Кодиране	Модулни тестове	Интеграционни тестове	Системни тестове	Възприемчивост тестове.	Трасиране на системата по време на изп.
Adaptive software development	менаж.		X	X	X	X	X	X	X	
	процес		X	X	X	X	X	X	X	
	практики									
Agile modeling	менаж.									
	процес									
	практики		X	X	X					
Crystal family and metho-dologies.	менаж.			X	X	X	X	X		
	процес			X	X	X	X	X		
	практики									
Dynamic systems development	менаж.	X	X	X	X	X	X	X	X	X
	процес	X	X	X	X	X	X	X	X	X
	практики									
Extreme programming	менаж.									
	Процес		X	X	X	X	X	X		
	практики		X	X	X	X	X	X		
Feature-driven development	менаж.		X	X	X	X	X	X		
	процес		X	X	X	X	X	X		
	практики		X	X	X	X	X	X		
Open source software	менаж.		X	X	X	X	X	X	X	
	процес									
	практики		X	X			X	X	X	
Pragmatic programming	менаж.									
	процес									

	практики		X	X	X	X	X	X		
Rational unified process	менаж.	X	X	X	X	X	X	X	X	X
	процес	X	X	X	X	X	X	X	X	X
	практики	X	X	X	X	X	X	X	X	X
Scrum	менаж.		X	X	X	X	X	X		
	процес		X	X			X	X		
	практики		X	X			X	X		

Таблица 1.3.2.2 показва че различните гъвкави методологии са фокусирани над различни аспекти или фази от жизнения цикъл на софтуерната разработка. Освен това някои гъвкави методологии (като екстремното програмиране, прагматичното програмиране и гъвкавото моделиране) са фокусирани върху практиките, докато други като C Scrum са фокусирани върху менажирането на софтуерния проект. Някои методологии като гъвкавата методология DSDM и дадената за сравнение методология RUP ни дават пълно покритие над целия жизнен цикъл на софтуерна разработка. Повечето гъвкави методологии (като FDD например) започват директно от фазата „Спецификация на изискванията".

2.5. Общи белези на гъвкавите методологии

Ще представим няколко типа общи белези на гъвкавите методологии, характеризиращи ги като такива. Цитирани са автори на книги и публикации за гъвкавите методологии като Miller, HighSmith, Cockburn и Boehm. В следващата глава ще дефинираме нашия софтуерен процес позовавайки се и на тези характеристики.

Miller в книгата си "The Characteristics of Agile Software Processess" [18] ни дава няколко характеристики за софтуерен процес на една гъвкава методология от гледна точка на бързината на изготвяне на продукта:

1) Изменяемост на нивото на процеса на разработка
2) Итерации със къси цикли на разработка позволяващи бърза верификация и корекции.
3) Дължина на итерациите от една до шест седмици.
4) Пестеливостта в процеса на разработка премахва всички ненужни activities.
5) Адаптивност към риска от бързи промени във софтуера.
6) Инкрементален софтуерен процес който позволява внедряването на нова функционалност в малки стъпки.
7) Ориентиран към хората. Тоест гъвкавата методология набляга на хората пред софтуерния процес и технология.
8) Работен стил в духа на сътрудничество и комуникация.

В гъвкавите методологии според HighSmith и Cockburn [19] не се набляга толкова на това как да спрем промените дори в ранна фаза на разработка, а как да ги приемем и интегрираме в жизнения цикъл на проекта. В последствие започва да се твърди че гъвкавите методологии са направени за:

- Да изкарат първата версия на продукта (не става дума прототипа) в близките седмици, с цел получаването на бърз и обширен отговор от клиента - „feedback".
- Да се измислят елементарни решения, с идеята тяхното прилагане да бъде елементарно
- Непрекъснато подобряване на качеството на дизайна за да минимизираме нужните ресурси при следващата итерация
- Непрестанно тестване, за ранно и по евтино отстраняване на дефекти.

Според **Scott W. Ambler** [20] основните принципи които правят една методология гъвкава методология са:

1) Хората са най-важното
2) Колкото е възможно по малко документация
3) Комуникацията е ключов фактор
4) Програмните инструменти за моделиране не са толкова полезни колкото по принцип се смята

Най-интересния за „нашия софтуерен процес" анализ (таблично представен в табл. 2.5.) е този на Boehm, който анализира гъвкавите, процес-ориентираните методологии и интересуващия ни „Софтуер с отворен код" (Open Source Software - OSS). Той нарича процес-ориентираните методологии още план-базирани методологии. От таблицата се вижда че OSS стой между гъвкавите и процес-ориентираните методологии. Тъй като OSS е все още нещо сравнително ново в бизнес средите и съответно има много въпроси за изследване които чакат да бъдат анализ и отговор. Въпреки че всеобщата теза е че OSS би трябвало да се разглежда като още един вариант от множеството гъвкави методологии.

Таблица 2.5: *Общи белези на гъвкавите методологии*

Предмет на внимание	Гъвкави методологии	Софтуер с отворен код	План-базирани методи
Разработчици	Гъвкави, широки познания, наставническо и сътрудничество	Географски разхвърляни, сътрудничещи, гъвкави екипи с широки познания	План-ориентирани, адекватни умения, достъп до външни познания
Клиенти	Посветени на проекта. С широки познания, склонни към наставничество, Сътрудничещи и добре обясняващи. Упълномощени	Посветени на проекта. С широки познания. Сътрудничещи и упълномощени.	Достъп до сътрудничещи и добре обясняващи клиенти с широки познания
Изисквания към типа на софтуера.	Високо ниво на неопределеност. Бързи промени.	Високо ниво на неопределеност. Бързи промени. Общи права за притежание. Непрекъснато доизграждане и никога незавършване на софтуера	Ранно известен. Достатъчно стабилен.
Архитектура	Направена според текущите изисквания	Отворена архитектура. Направена според текущите изисквания.	Направена за бъдещи и предвидими изисквания в близко бъдеще.
Преработка на кода „Refactoring"	Не скъпа	Не скъпа	Скъпа
Размер	Малки екипи и продукти	Големи и разпилени екипи и малки продукти	Големи екипи и големи продукти
Ключова дума	Бързина	Справянето с проблема	Голяма увереност и сигурност.

2.6. Изводи за гъвкавите методологии.

Опитът показва че традиционните план-управлявани софтуерни методологии не се използват много на практика. Това твърдение обикновено се аргументира с факта че въпросните методологии са прекалено механистически, за да бъдат приложени в детайли.
Видимите аспекти на леките и гъвкавите методологии са простота и бързината. В средите на разработчиците общо взето разработчиците се концентрират основно върху най-необходимата функционалност (необходима за първата итерация) , бързото и внедряване, получаването на отговор от клиента (feedback) и респективно реагирането на получената информация. В следващата глава ще наблегнем повече на приликите и разликите между нашата нова гъвкава методология и съществуващите като се позоваваме на горепосочените анализи (особено на

анализа на Boehm, тъй като нашия софтуерен процес представлява гъвкава методология и обхваща разработката на софтуер с отворен код).

Като заключение можем да отговорим ясно на въпроса: „Какво прави един метод за разработка на софтуер гъвкав?”

Първо когато софтуерната разработка е инкрементална (малки софтуерни копия на продукта, кратки цикли на итерация), когато цари дух на сътрудничество (разработчици и клиенти работят заедно в близък контакт), имаме наличие на праволинейна работа (самия метод е лесен за научаване и промяна, и е достатъчно добре документиран), и не на последно място лесна адаптация (да можем да правим „горещите” промени в софтуера в процеса на разработка).

Описахме накратко интересуващите ни гъвкави методологии (тези от които сме взаимствали в една или друга степен за създаването на нашата нова гъвкава методология), като извадихме на преден план софтуерния им процес, ролите и отговорностите им, и практиките които се използват в тях. Или използвахме четвъртия от описаните методи за сравнение (виж т.1.2.3) на гъвкави методологии. След което противопоставихме всички по важни гъвкави методологии по няколко критерия. По дефиниция гъвкавите методологии следват обща политика. Опитахме се да вникнем в това върху какво се фокусира всяка една от изброените гъвкави методологии.

Всяка методология разполага със собствен речник и терминология. Това от което се нуждаят гъвкавите методологии е широкото им използване - най-добрия начин за възприемане. Но една от основните цели на „обществото на гъвкавите методологии” е да даде точната гъвкава методология в точния момент за точния тип проекти. Именно за това ние ще се предложим една такава гъвкава методология.

3. Предложение на нова гъвкава методология за проекти с отворен код

В глава 2 разгледахме и сравнихме съществуващите гъвкави методологии. В тази глава ще дефинираме нова гъвкава методология, като използваме направените от сравнението на съществуващите гъвкави методологии изводи и дефинирани понятия. Ще дефинираме една гъвкава методология специализирана в разработването на проекти използващи други проекти – проекти с отворен код. Малко гъвкави методологии могат да се пригодят за поемането на един нов проект в различна фаза на зрялост. Тази гъвкава методология предполага софтуерна разработка сходна като фази с компонентно базираната разработка на софтуер. Методологията предполага следване и обновяване на даден план на разработка подобно на план-базираните методологии. Освен че ще дефинираме , класифицираме и именуваме новопредложената методология ще докажем че темата на дипломната работа не представлява план-базирана методология.

Ще опишем методологията по използвания в предната точка начин за описване на една методология - чрез използването на *метаезик,* дефиниран основно от следните елементи:

- идея на методологията
- софтуерния процес
- роли и отговорности на участниците
- използваните практики

След като дефинираме методологията ще определим нейното място в света на гъвкавите методологии като:

- наблегнем на приложението на тази гъвкава методология
- обособим проектите върху които се фокусира
- опишем специалните и особености
- защитим твърдението си, че това е гъвкава методология

3.1. Софтуерен процес

Софтуерния процес при тази новопредложена гъвкава методология може да се опише в следните стъпки.

- търсенето, идентифицирането и адаптирането на софтуерен проект с отворен код към нуждите на проекта разработван от екипа работещ по АПОК методологията.
- „играта на два фронта”
    - От страната на разработването на софтуерния проект с отворен код, като част от обществото на разработчиците на ПОК (проекти с отворен код).
    - От гледна точка на специфичните нужди на нашия продукт.
- Интеграцията му в системата ни ако тя съществува.
- Пускането на изградената система в експлоатация и осигуряването на поддръжката й.

Самият софтуерен процес се дели на три („двата фронта” и връзката между тях):

- Проект с отворен код - Първата част от Гъвкавата методология се занимава с Работа с проекти с отворен код.
- Адаптация - Втората се занимава с адаптирането на софтуерния процес използван в изграждането на останалата част от системата към „поемането” на ПОК.
- Логистика - Третата част е осигуряването, инициирането и управлението на връзката между първите две части.

Макар „Логистиката”, породена от „Адаптацията”, да са по първични от „Проект с отворен код” в ретроспективен план, ще опишем първо именно него.

3.1.1. Проекти с отворен код

3.1.1.1 Идея на проекта с отворен код

Заражда се с възникването на интернет, където колеги могат свободно да споделят програмния си код. Това стимулира създаването на някакъв подход при изграждането на приложения от такъв тип. Първоначално създадените успешни подобни проекти са “Apache server”, “Perl programming language”, “Sendmail mail handler” and the “Linux operating system”.
Основната идея на ПОК (OSS – Open source software) е програмния код да бъде достъпен, модифицируем и разпространяван безплатно.
Feller и Fitzerald (2000) ни представят в три точки причините за съществуване на OSS:

1) Техническа: Необходим е ясен програмен код, бързи цикли на разработка, високи стандарти на качество, надеждност, стабилност, и повече отворени стандарти и платформи.
2) Икономическа: корпоративната нужда от това цената на софтуера и риска от използването му да са споделена инвестиция. (И други корпорации да го използват и да подпомагат за усъвършенстването му).
3) Социално-политическа: Изкушаването на отделния разработчик да направи нещо значимо на своя глава. Създаването на сдружения с „идеална цел”. (Общество причина за съществуването на което е необходимостта от изграждането на даден проект).

Повечето от по-известните проекти с отворен код са се фокусирали върху програмните инструменти за разработка или други „средства”, които са използвани от професионалисти (по-често експерти в дадена област), които също по някакъв начин са въвлечени в софтуерната разработка.

Друга интересна отличителна черта на ПОК е идеята един човек да играе ролята на разработчик и на клиент. ПОК не е компилация от добре дефинирани и публикувани практики за софтуерна разработка, описващи една софтуерна методология. И все пак обосновавайки се на лицензните му политики (license polices), може да се каже че ПОК е достатъчно добре описан като философия. А общата идея за сътрудничеството на много и разпръснати индивиди (разработчици), които да реализират малки и чести версии на проекта, може да ни даде представа какво представлява ПОК от практическа гледна точка.

3.1.1.2 Софтуерен процес на проект с отворен код

Според Cockburn (2002) [] разработката на проекти с отворен код се различава от тази при гъвкавата методология и във философски и в икономически и в организационен аспект, но все пак ПОК следва принципно същата главна линия и използва същите практики като другите гъвкави методологии. Примерно разработката при ПОК стартира със създаването на ранни и чести версии на продукта. Липсват много от традиционните механизми използвани за координирането на софтуерната разработка със създаване на план, системен дизайн и дефиниран софтуерен процес. Обикновено разработването на ПОК се състои от следните фази:

1) Откриване на проблема
2) Намирането на доброволци
3) Идентифициране на решението
4) Кодиране и тестване
5) Преглед на промените по кода (code change review)
6) Издаване на нова версия на продукта и документация
7) Управление на версиите.

И все пак, това са по-скоро стъпки в разработката, отколкото фази. По-интересното е как евентуално се извършва менажирането на софтуерната разработка. Mockus (2000) [21] дефинира следните твърдения:

1) Системата бива построявана от потенциално неограничен брой от доброволци
2) Работата по проекта не е разпределена предварително. Хората сами избират задачите с които да се справят. (тези които ги интересуват)

3) Не съществува категоричен системен дизайн
4) Няма план на проекта или разпределяне на задачите във времето.
5) Системата се разширява на малки стъпки чрез често изграждане на нови версии на проекта.
6) Тества се често и непрекъснато.

Според Feller и Fitzgerald (2000) [22] разработката на ПОК е организирана като масивна паралелна разработка на проект с доста положени усилия по отстраняването на грешките. Разработването на ПОК предполага децентрализирано, кооперативно и безплатно разпространение на софтуера от всеки един разработчик. Разработката на ПОК не включва никакви формализми или норми за писане на документация. И все пак „софтуерния процес" при ПОК включва в себе си разни „навици и привички" и „неписани закони", чието използване се е наложило с времето.

3.1.1.3 Роли и отговорности при проекти с отворен код

Така наречения „софтуерен процес" при ПОК изглежда доста освободен от дефиниции или голям безпорядък. И все пак той би трябвало да има някаква изградена структура, за да може да ПОК да се окажат толкова успешни начинания в последните години. Големи концерни като IBM, Apple, Oracle, Corel, Netscape, Intel и Ericsson не крият интереса си към тези проекти (Feller и Fitzgerald 2000). При някои по-големи ПОК софтуерните компании дори се решават да поемат ролята на „координатор" и „медиатор" или да обосноват някакво ръководство на проекта, инвестирайки немалко ресурси също така. По надолу в темата ще подкрепим твърдението с реален случай.

Може би най-големия феномен е наличието на „хранилище на проекти" (repository), организирано като глобално достъпен сайт, поддържащ форуми и други средства за свободна комуникация, контрол на версиите на проекта, система за следене и разрешаване на проблеми и грешки (bug tracking system), правила за организиране на менажирането на проекта. Най-известни подобни хранилища са „Sourceforge" и „Freshmeat". Проекта който ние ще разгледаме използва именно „Sourceforge" за свое хранилище. При липса или невъзможност за близка комуникация (лице в лице) необходимостта и важността на интернет комуникацията е очевидна. Процеса на разработката изисква добре функционираща система за контрол на версиите, която да може да се представи като „живата история" на проекта. При ПОК самия „софтуерен процес" се дефинира от програмните инструменти които се използват от основните участници в проекта. Силно твърдение което ще обосновем индуктивно, чрез примери.

Типичната структура (на разпределението на ролите) на разработката на ПОК според Gallivan (2001) е композирана от няколко нива доброволци:

1) Създатели на проекта – Онези които поемат отговорността за проекта и общо взето са написали първата версия на проекта.
2) Разработчици доброволци – тези които добавят функционалност и допринасят с програмен код за проекта. Могат да бъдат подразделени във:
   - Опитни разработчици, разработчици на ядрото на проекта с добра и доказана репутация при разработването на проекта.
   - Периферийни разработчици, добавящи функционалност.
   - Случайни и еднократно помагащи разработчици, разработчици които разрешават даден проблем и излизат от него. (Опитни разработчици с ниска репутация).
   - Разработчици които имат много добра репутация или са подпомогнали проекта с финансови средства. (Участници в проекта чието мнение тежи). Често най-активни в идеите за създаването и моделирането на нова функционалност.
3) Други личности, потребители които използват софтуера, тестват го, идентифицират програмни грешки и рапортуват за проблеми.
4) Потребители които имат засилен интерес към проекта участват в дискусии, но не кодират.

Тези нива са нестрого дефинирани „роли", които се изпълняват от дадени индивиди. Но често се случва в процеса на опознаване на проекта даден участник да започне да изпълнява дадена роля, след което да се прехвърли в изпълнението на друга роля и така до пълното опознаване на проекта. В случая на нашия разглеждан проект например се случи точно това.

Според Sharma (2002) ПОК при разработката са обикновено разделяни от главните архитекти и дизайнери на малки и по-лесно менажируеми задачи, които впоследствие се поемат от индивиди или групи от такива. Самите доброволци разработчици се разделят на малки групички или действат самостоятелно. Сами избират задачата или проблема който да разрешат, поемайки отговорността и залагайки името си за това. Правилното разделение на проекта на задачи е ключово за успешното му завършване. Тези задачи трябва да са и интересни казуси сами по себе си за да привлекат вниманието на програмисти доброволци. (Отделно ако проекта е интересен разработчиците са склонни да разрешават и не дотам сложни проблеми само и само да са част от проекта).

3.1.1.4 Практики при проекта с отворен код

За да започнеш или да придобиеш правото да се наречеш инициатор или собственик (founder) на даден ПОК трябва или да започнеш такъв от нулата, да наследиш това право от създателя на проекта, или доброволно да поемеш един умиращ проект с отворен код (Bergquist и Ljungberg 2001). Процесът на разработка на теория е продължителен а на практика безкраен. Като разработката плавно се изражда в поддръжка. Според Ghosh, Prakash, Mockus и Dempsey (2000) дори и стотици доброволци да участват в изграждането на даден ПОК, то основната част от работа се върши от мака група хора. Въпреки че те навярно са имали в предвид добавянето на функционалност, и работата на многото не кодиращи участници се вижда, подпомага разработчиците, доизяснява идеята, и е също важна част от разработката на ПОК. Sharma (2002) описва някои от централните организационни аспекти при разработката на ПОК.: разделението на работата, координиращия механизъм, допринасянето към проекта във всеки един аспект и начина на взимане на решения, дефинирането на границите на организацията и неформалната структура на проекта.

При описването на използвания от нас ПОК ще покажем примерни практики при разработването на един проект в Sourceforge хранилището. Ще стане ясно как именно „програмните инструменти" определят използваните практики при разработването на един ПОК.

3.1.1.5 Особености на проектите с отворен код

Lerner и Tirole (2001) разказват за опита си при изминаването на пътя на един разработчик на Linux, Perl и Sendmail проектите. Те описват разработването на тези проекти от усилията положени от отделния програмист за изграждането на дадена функционалност до тези положени от групи от индивидуалисти наброяващи хиляди.

Един от проблемите при разработването на ПОК, и едновременно с това и негова силна страна е, че изискванията към ПОК непрекъснато „еволюират"(стават все повече и все по-трудно изпълними), а проекта никога не достига финализираща фаза, докато всички разработчици не престанат да работят по проекта (Feller и Fitzgerald 2000). Доскоро седящия проблем с достъпа до интернет вече не е проблем. В днешно време глобалната зависимост от интернет комуникацията не е проблем.

Друг основен казус при ПОК е стабилизирането и комерсиализирането на кода до ниво приложение. Тъй като програмния код при ПОК само по себе си не е комерсиален продукт, бизнес изискванията в дадена предметна област задават косвено изискванията към продукта, до превръщането му в използваемо приложение. Често се прилага пакетния подход на разработване на продукти. Реализира се повече функционалност от колкото е необходима на крайния потребител, а той си избира колко от нея да използва. Примери за това са Redhat и SOT (Linux). Има случаи когато цялата бизнес идея се появява на белия свят вследствие на работа по ПОК. (Wall 2001).

От правна гледна точка ПОК винаги указват лицензната политика под която се разпространяват. В общия случай това е GPL - General Public License (можеш да разпространяваш и ползваш софтуерния продукт, но не и да го продаваш като част от друг продукт, без да разпространяваш и комерсиалния си продукт под същия лиценз) или LGPL (можеш и да разпространяваш и ползваш парчета функционалност от продукта и без да отваряш кода на своя комерсиален продукт).

Софтуерните компании показват все по голям и по голям интерес към изграждането на ПОК методология която да им помогне в работата. Srinarayan Sharma заедно с Vijayan Sugumaran и Balaji Rajagopalan (2002) решават да създадат система за управление (framework) за създаването на хибридни ПОК (използване на ПОК в комерсиален продукт). Целта на тази система е да се опита да помогне на компаниите да изградят своя комерсиален продукт на базата на ПОК. Според тях само някои компании са направили някои емпирични учения или целенасочени изследвания в тази насока и то за изграждането на тяхното собствено ноу-хау. (Без да споделят опита си, да се гордеят с това или да се опитат да дефинират софтуерен процес използваем и от останалите). Целта на дипломната работа е да дефинира именно такава софтуерна методология.

3.1.2. Адаптация

Ще опишем под-процеса занимаващ се с менажирането на разглеждания проект “SuperSeeder”, представляващ комерсиалния продукт направен от екипа към който принадлежи и дипломанта. Идеята на този под-процес е изградена емпирично по аналогия на съществуващите гъвкави методологии. Като от много от тях са взети някои заемки като практики, роли, дизайн и идеи.

Ако трябва да се опише или дисоциира „Адаптацията” би могло да се каже че тя е:

- С духа на ASD – Взаимства мотото на Адаптивната софтуерна разработка, да се държи проекта на ръба на хаоса. Да се научим не да ограничаваме хаоса, а да го контролираме. Да използваме такъв процес които ще ни позволи разработването на проект максимално гъвкав за настъпващата неясна функционалност, която ще му се добавя.
  - Създаваме начален план на проекта, поставяйки нестроги крайни срокове на базата на ресурсите с които разполагаме. Този план на проекта е управляван от риска или на преден план изкарваме най-малко рисковите промени първо, за да подпомогнем създаването на непрекъснато работещ софтуер. Този план не е окончателен и ще търпи доста промени, наложени от неясните елементи които ще ни вкара Логистиката.
  - Непрекъснатото прототипизиране, което ASD предлага, е начин да се направи това. Не знаем „Логистиката” какъв ПОК ще ни представи затова трябва да сме готови да поемем и доста трудни проекти.
  - Да създадем доста гъвкав интерфейс за връзка с ПОК. Когато е ясен избора на ПОК за „Логистиката”, да се създаде новия прототип, когато познаваме в детайли ПОК и сме направили дизайн на модификациите. (при наличие на сложна предметна област е възможно да е необходим и трети прототип който да ни даде последен поглед върху цялостната картинка). Освен прототипизирането се индексира коя функционалност е възможно да се променя често, или къде ни е необходима тази гъвкавост.
  - Идеята за наличие на „мисия на проекта”, на базата на която се гради най-ранния прототип , също е заемка от ASD.
- С Доуточняване на процеса – Подобно на консорциума при DSDM, където се прави и проучване дали да се използва въобще DSDM, и при тази „разнородна среда” е необходим екип от софтуерни инженери занимаващи се със софтуерния процес.
  - Тъй като в доста голяма степен програмните инструменти и неписаните традиции обособяват софтуерния процес при ПОК, то нашия екип от софтуерни инженери трябва да се погрижи да избере най-удачните програмни инструменти, съобразявайки се и с традициите на ПОК и със „Адаптацията”.
  - Друг динамичен елемент при нашата методология са използваните практики. За това кои от тях да от тях е удачно да използваме също е отговорност на този екип. Софтуерните инженери, които са запознати със съществуващите практики, избират такива на базата на твърди факти като ресурсите с които разполага проекта и случайни фактори като спешната необходимост от промяна.

При ПОК разгледахме ролите и отговорностите в обществото на ПОК. В следващата точка ще разгледаме „ролите и отговорностите” на под-процесите Адаптация, Логистика и ПОК , който са разпределени между участниците в екипа разработващ „SuperSeeder” проекта.

3.1.3. Логистика

Този под-процес се нарича така заради логистичната подкрепа която оказва в комуникацията между „адаптацията” и ПОК. При този под-процес имаме само комуникация и анализ.

- Осигуряването на връзката – свързано с поддържането на комуникацията между ПОК обществото и участниците в „Адаптацията”.
  - От една страна участниците в „Адаптацията” предоставят „чисти клиентски изисквания” към участници в „Логистиката” запознати с предметната област на ПОК. В следващия момент те се оказват клиенти на „пазара от продукти” и участниците в „логистиката” се явяват техни консултанти.
  - От гледна точка на ПОК, участниците в „Логистиката” се оказват няколко типа участници от „разработчици доброволци(идеолози или финансови поддръжници)” до „участници в дискусиите”.
- Управлението на връзката – участниците в „Логистиката” които:
  - анализират потребностите на прототипите на „адаптацията” и започват „играта на два фронта”: оценяват както сложността на добавяне на дадена функционалност в ПОК така и бизнес нуждите на прототипа на „адаптацията” (могат да кажат какво ПОК може да даде още на комерсиалния ни продукт).
  - Те са в тясна връзка с ПОК обществото и отговарят за поддръжката, комуникацията инсталацията на модификациите и тяхната приемственост от ПОК. Важен принцип при „логистиката” е да не се обособяваме или отделяме модифицирания си ПОК в отделна версия (различна от последната в ПОК обществото) докато това не се налага. Именно този принцип издига „приемствеността на нашите промени” наравно с останалите activities.
  - Следят за промените по нашия комерсиален продукт и за тези в ПОК. И се стараят да синхронизират версиите им.
- Инициирането на връзката - представлява първата фаза в логистиката и пораждането на под-процеса на ПОК. Циклично може да се раздели на три стъпки:
  - Търсене – на подходящ ПОК, отговарящ максимално на архитектурните критерии и нуждите на прототипа ни (или на мисията на продукта).
  - Анализ на намерения ПОК по вторични критерии като развитие, зрялост.
  - Презентация на намерения ПОК или повтаряне на цикъла до намирането на подходящия ПОК.

Може да се забележи, че на места биха се дублирали ролите на логистиката и тези на ПОК. Въпреки че в един момент и участници в логистиката и участници в ПОК изпълняват подобни роли в ПОК обществото, то те го правят в различни аспекти. Участниците от „Логистиката” следи за нуждите на „Адаптацията” и всяко нейно участие в ПОК обществото има за цел да удовлетвори нуждите на „Адаптацията”. Докато участниците в ПОК изпълнявайки най-често само една роля (тази на разработчик) се отдалечават от комерсиалния ни продукт и се сливат с ПОК обществото. Приоритетно изпълняват изискванията предоставени им от логистиката, а през останалото време удовлетворяват нуждите на самия ПОК. В следващата подточка ще изясним различните роли и отговорности, както тяхното сливане и миграция.

3.1.4. Практики в Адаптацията и ПОК

Практиките използвани при разработката на проекти според АПОК методологията се делят на основни и допълнителни според нуждата необходимостта от прилагането им. А Основните практики се делят между двата под-процеса Адаптация и ПОК:

- Избор на основни практики
  - Много от практиките са наложени от традициите на ПОК, такива са:

- Колективна собственост на кода
- Непрекъснато тестване и интеграция
- Честото изваждането на нови версии на проекта с малки промени
- Преструктурирането на системата (премахване на дублиращи се елементи, опростяване и подобряване на гъвкавостта).
- Стандартите на кодиране наложени от ПОК.
- Unit testing – За всеки фрагмент влагана функционалност се извършва се минава по класовете на еквивалентност, обикновено чрез използването на специално написани за целта програмни инструменти.

▪ От друга страна „Адаптацията" в опитите си да управлява хаоса внесен от „Логистиката", налага някои практики които целят наличието на една централизирана система:
  - История на продукта (съхранявана в използваните програмни инструменти, от системата за менажиране до "wiki" системата).
  - Ежеседмичен „спринт" – При прототипизирането „Адаптацията" се нуждаят от време в което да реализират системата както са я разбрали до този момент. Логистиката в тясна връзка с ПОК се нуждае от време за да направи нужните анализи. А при реализация на нова функционалност ПОК се нуждае от време което да прекара само сред ПОК обществото. Спринта сам по себе си е процедура на самоорганизиране на екипа разработчици, адаптирайки се към изменящите се клиентски изисквания, време, човешки ресурс, бюджет, знания и технология.
  - Спринтово планиране – Срещите между участниците са необходими за да може разработчиците да изложат интересни факти (които е възможно да инициират нова функционалност) или проблеми по реализирането на идеите си. И да предложат модела за реализирането на следващата функционалност. Или нестрого следвайки плана наложен от „Адаптацията", моделираме, кодираме, тестваме и налагаме промени в плана ако е необходимо. Тези срещи подпомагат и за добиване на знания у участниците за предметните области на „Адаптацията" и на ПОК (много често те са различни) и са жизнено важни за дооформянето на плана на проекта.
  - Оценяване на усилията – В ранна фаза неясната предметна област на ПОК, за „Адаптацията" и обратно, често създават пречки като поставяне на нереални срокове първоначалния план. Спринтовете и историята на продукта подпомагат за откриването на проблема „кое кара проекта да се бави" или „защо разработчиците от ПОК или Адаптацията реализират своите модификации доста по-рано отколкото се очаква".
  - Играта на планиране – Тъй като разработчиците, вече навлезли дълбоко в своята си предметна област, моделират и кодират модификацията, най-вече те могат да кажат реалистично колко време би отнела. Или промените в предварителния план се внасят на базата на техните твърдения.
  - Прост дизайн и често преструктуриране до постигането му – Разработчиците ползващи този подход създават очаквания в останалите които не е добре да променят. Честите промени обезсмислят сложните архитектури. („Сложната" архитектура е налична в ПОК обикновено).
  - Визуално моделиране – На всеки спринт на който се обсъжда моделирането на нова функционалност разработчика представя модел на модификацията. „Общият език" между участниците в дискусията е UML. А крайния модел се съхранява в историята на продукта.
  - Създаването на прототипи - изисква максимално бързо и елементарното им моделиране и кодиране. Създават се два типа

прототипи. „Първичен” – frontend дизайн на продукта и „Зрял” – frontend които симулира работен режим на целия проект.

- Избор на допълнителни практики – Познатите ни гъвкави методологии дефинират достатъчно на брой практики, които даже се припокриват на места. Често дори е излишен труда хвърлен по измислянето на нова практика с цел оригиналност. Описаните дотук практики се използват по подразбиране в тази методология. Но екипа от софтуерни инженери може да реши да наложи и други, такива които повишават даден критерии за сметка на друг. Често е необходимо сигурността и качеството да се повишат за сметка на значителното увеличение на важни ресурси като време, пари и хора. Тези промени в методологията се изразяват най-вече в промяна на програмните инструменти или вкарването на нови практики като преглед на кода, следене за правилността на изпълнение на зададените правила (monitoring), верификация на софтуера и други.

3.1.5. Схема на софтуерния процес

Софтуерните инженери дефинират и поддържат рамка на софтуерния процес, разделена в три под-процеса които описахме поотделно. Тук ще опишем взаимодействието между тях.(фиг. 3.1.5)

Цялостният софтуерен процес се разделя на две фази:

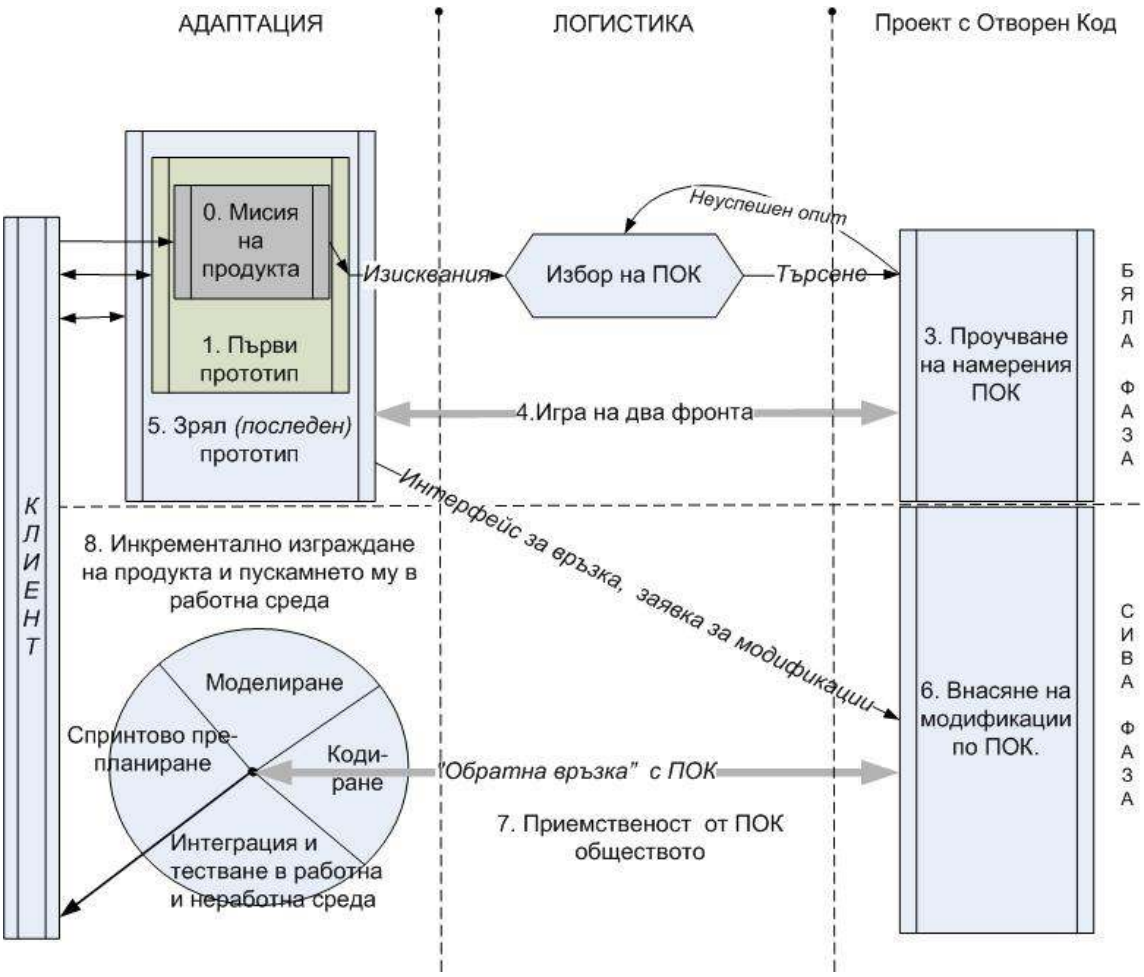
- Сива фаза
  - за Адаптацията фазата, в която се доизяснява предметната област.
  - за Логистиката фазата, в която се изследва(т) ПОК.
  - за третия под-процес (ПОК) в тази фаза възприемаме ПОК като “grey-box” или проучваме даден проект в дълбочина, без да участваме като разработчици. (откъдето идва и името на фазата).
- Бяла фаза
  - за Адаптацията фазата, в която започва производствения цикъл.
  - за Логистиката фазата, в която се управлява зрелия цикъл на разработка и при ПОК и Адаптацията.
  - За под-процеса ПОК фазата, в която участваме като разработчици в проекта с отворен код. (Макар и в сивата фаза да разполагаме с целия наличен код, тогава сме се фокусирали върху интерфейса който ни предоставя ПОК, докато в тази фаза ни интересува по-скоро програмния код, който седи зад тези интерфейси).

Софтуерният процес може да се опише постъпково. В началото клиентът ни възлага поръчката за изграждането на даден продукт. След което ние придобиваме идеята за мисията на продукта. Съставяме прототип с цел трансформирането на идеята до ясни изисквания. Всички участници си доизясняват идеята. Започва да се търси решение на ниско ниво, разрешаващо сложен проблем в чужда нам предметна област. След като намерим такъв ПОК, проучваме го и съставяме крайния зрял прототип. Представяме го на клиента и съставяме неокончателен план и приблизителен краен срок за изпълнението на поръчката. Вече имаме както конкретни задачи за разработка към ПОК, така и интерфейс за връзка с комерсиалния продукт. Започваме производствения цикъл по внасянето на модификациите в ПОК, спринтовото препланиране (предизвикано от неочаквани изненади от ПОК по време на срещите. Възможно е и клиента да се намеси), моделирането, кодирането, тестването на модификациите, интеграцията, тестване в работна среда, обратната връзка както с клиента така и с ПОК.

От гледна точка на ПОК обществото внасяме промените по проекта съобразявайки се със наложените от ПОК обществото стандарти и традиции. След като успешно изтестваме модификациите си, търсим приемственост като следваме основния принцип: *Стараем се нашите промени да станат част от проекта, и да не се разграничаваме от него докато е възможно.*

От схемата (фигура 3.1.5) се вижда че клиента има допир само до единия под-процес - Адаптацията.

Тъй като ПОК няма “death phase” (проекта се доразвива докато има участници по него), то е много вероятно и Адаптацията никога да не свърши. Дори когато минимален човешки ресурс изпълнява ролите отредени от осма точка от софтуерния процес, изродени в поддръжка, имаме наличие на активност по проекта.



Фигура 3.1.5: Схема на софтуерния процес

3.2. Роли и отговорности

Интересното при тази гъвкава методология е че може няколко отместени една спрямо друга „точки" от софтуерния процес да се поемат от една роля. Всеизвестна практика е прескачането на хора от една роля в друга след обучение. Границите на под-процеса не се явяват пречка за миграцията на хората от една роля в друга, точно обратното тази гъвкава методология стимулира това. Ще дефинираме 11 роли:

Клиент
Поръчител на проекта, често самия спонсор на проекта, доставя на разработчиците „клиентски истории" и извършва потребителски функционални тестове. Решава кога изискванията към продукта са задоволени. Също определя и приоритети в имплементирането на изискванията си.

Бизнес анализатор и мениджър на проекта
Събира и сглобява клиентските изисквания, познава бизнес материята. Също така управлява ресурсите налични за проекта. Трасира и оценява усилията положени от участниците в Адаптацията и следи прогреса на всяка една итерация, дали крайната и цел е достижима с

ограниченията наложени от наличните ресурси. При наличие на проблем уведомява за него софтуерните инженери.

Главен архитект
Опитен програмист отговорен за целия дизайн на комерсиалния ни продукт (архитект на Адаптацията). Проявява се и като лидер на екипа на Адаптацията и негов ментор.

Адаптация разработчик
Участва в моделирането, кодира, изготвя unit тестове, документира функционалност от цикъла на разработка на комерсиалния ни продукт. Също така във сивата фаза, при прототипизирането , се явява „собственик на клас функционалност" и вкарва демонстративна функционалност в обявяваните от главния архитект класове и интерфейси (stubs).

Интегратор, инсталатор и системен администратор
Конфигурира и разрешава хардуерни проблеми, мрежови проблеми, проблеми със сървъра , и други проблеми свързани със средата за разработка. Също отговорен за конвертирането на данните, инсталирането на новите версии на продукта.

Софтуерен инженер
Отговорник за софтуерния процес, прилагането му и модифицирането му когато е необходимо да се подобри начина на работа. Също отговорник за създаването на документна база и използването на контрол на версиите.

Лидер на ПОК екип
Опитен разработчик отговорен за дизайна на ПОК модификациите. Отговаря също така и за ръководенето на малки екипи от ПОК разработчици. Идентифицира класовете функционалност от предоставения му списък с изисквания.

ПОК разработчик
Под ръководството на ПОК лидера си моделират, кодират тестват и документират текущо прикачения им клас функционалност. В сивата фаза се занимават с разработването на the unit tests.

Тестер
Проверява дали системата отговаря на клиентските изисквания. Извършва и чисто функционални тестове също на системата в процеса на разработка. Подпомага извършването на unit тестовете за ПОК , а по-късно тества функционалност от гледна точка на Адаптацията.

Логист
Реализира комуникацията между Адаптацията и ПОК. Менажер на ПОК разработката. Подобно на бизнес анализатора трасира и оценява усилията положени от участниците в ПОК и следи прогреса на всяка една итерация, дали крайната и цел е достижима с ограниченията наложени от наличните ресурси. При наличие на проблем уведомява за него софтуерните инженери.

Инженер по знание
Експерт в предметната област на ПОК проекта. Помага на Логиста при търсенето и изследването на ПОК и на лидера на ПОК екип в идентифицирането на функционалност. Консултира ПОК екипите по въпроси свързани с разрешаването на специфични проблеми. Често при несложните проекти тази роля бива изпълнявана от Логиста.

В тази гъвкава методология ролите са дисоциирани дотолкова че да е необходим за всяка роля поне един човек поемащ само нея. Но е възможно при проекти с малък обхват някои роли да се слеят в една (Така ролите Интегратор, инсталатор и системен администратор са се слели в една обща роля). Често и ролите „Логист" и „Лидер на ПОК екип", в зависимост от обхвата на проекта, са сливани в една роля.

3.4. АПОК – Адаптиране на проект с отворен код

За да можем да защитим тезата си че това предложение за гъвкава методология е гъвкава методология, ще обобщим съставящите я фрагменти. Първо нашата идея и принципи за методология са сходни с тези на останалите гъвкавите методологии. Второ, дефинирания софтуерен процес предполага „цикличен" подход използван в останалите гъвкавите методологии. Трето използваните практики с основната си част са взаимствани от останалите гъвкави методологии. Или по всички критерии можем да обявим нашата нова методология за гъвкава.

След като дефинирахме софтуерния процес, практиките, принципите ролите и отговорностите на тази гъвкава методология можем да и дадем име според идеята и. Сглобявайки името от под-процесите можем да поднесем ясно идеята за приложението на тази гъвкава методология. „Адаптиране на проект с отворен код" предполага откриването и адаптирането на даден външен компонент (ПОК) към нашите нужди. Под-процесът 'Логистика" не е загатнат в името поради нуждата от изграждането на ясна и недетайлизирана концепция. (начина на управление на „външния компонент" си е вътрешен проблем на софтуерния екип, които не се упоменаваме в името на гъвкавата методология).

В следващата глава от дипломната работа ще опишем изграждането на примерен проект, следващ АПОК гъвкавата методология.

4. Избор на проект с отворен код(изследване) и приложение на софтуерния процес

Ще демонстрираме използването на АПОК методологията за разработването на един продукт предполагащ ядро и изнесен приложен интерфейс (API). Тази гъвкава методология бе използвана и при разработването на проекта използващ XBT Client – SuperSeeder и при последвалото разработване на mplayer frontend за Windows - MOP (Необходимо бе да се модифицира функционалността и master-slave протокола на mplayer ПОК за да можем да напишем MFC mplayer frontend). От опита в използването на методологията при разработването на горепосочените проекти можем да обявим тази методология за една от най-подходящите и в случаи на разработване на продукти използващи middleware софтуер, за клиент-сървър архитектури и за разпределени среди. За екипа от които е част и дипломанта тази гъвкава методология се оказа най-подходящия избор на методология за разработката на middleware софтуер (MOP) и за разработката на клиент-сървър архитектури (разглеждания в темата на дипломната работа проект - SuperSeeder).

Ще опишем разработката на този проект следвайки схемата на софтуерния процес на АПОК, ще опишем използваните практики, програмни инструменти, особеностите и ограниченията при разработката на този продукт. Няма да изложим UML диаграми и план на проекта извадени от системите използвани при разработката, за да не нарушим авторските права над продукта.

Поради наличните ограничения дипломантът като участник в разработката на проекта SuperSeeder изпълнява няколко роли (описани подробно в трета глава):

1) Логист – едноличен ръководител на под-процеса Логистика.
2) ПОК лидер на екип – ръководи себе си и един разработчик.
3) ПОК разработчик – разработчик внедрен в ПОК обществото.
4) Инженер по знание – единствена инстанция по въпросите свързани с битторент предметната област.
5) Софтуерен инженер – Наред със още един участник (Менажер на Адаптацията) дипломантът изпълнява тази роля.

4.1. Иницииране на търсенето на ПОК

Имаме поръчка за разработването на web-управляван битторент клиент. Ще опишем постъпково намирането на подходящия ПОК.

1) Изясняваме си с клиента бизнес изискванията.

2) Разработваме дизайна (като „stub” или форми зад които стои функционалност само на идея) на web-базираната част. Показваме предварителния външен дизайн на клиента за да доизясним бизнес изискванията.

3) Създаваме прототип на “front-end”. И поставяме бизнес изискванията за backend-а.

4) Започваме търсенето на битторент клиент ПОК по бизнес съображения и архитектурни такива.

4.2. Избор на подходящия ПОК

След като имаме бизнес изисквания, които трябва да удовлетворим и изисквания за надеждност и гъвкавост започваме да търсим ПОК с най-подходящата архитектура с приоритет пред ПОК удовлетворяващ най-точно бизнес изискванията ни.

4.2.1. Предметна област на ПОК.

Преди да опишем намерения ПОК, ще изясним накратко предметната област на въпросния ПОК.

Става дума за мрежово приложение свързващо се с други мрежови приложения с които обменя данни по определен протокол. Протоколът е битторент (bittorent), а всяко едно приложение реализира един или повече пийрове (peer) за няколко различни „торента" (единица за цялост на данните). Оттам и типа на „мрежовия проект": peer-to-peer (децентрализиран обмен на информация между потребители). Системата за обмен на данни се дели на две части битторент клиенти и тракер. Тракерът представлява един контейнер (база данни) съхраняващ информация за отделните пийрове и статусът на данните които се обменят. Също предполага и удобен интерфейс за визуализиране на данните намерени от тази машина за търсене из собствената и база данни. Данните на всеки един торент са разделени на парчета които се споделят от клиентите заинтересованите от торента потребители. Един торент може да бъде доставян на други потребители или да бъде теглен от тях или и двете едновременно. (когато имаш няколко парчета от торента които разпространяваш и искаш да се сдобиеш с останалите парчета от него). Разпространяването се нарича „seeding" (посяваш данни във интересуващите се) , а тегленето "leeching" (взимаш данни). Всеки един клиент заинтересован от въпросния торент реализира един пийр. С други думи всеки един клиент може да „притежава" много торенти всеки от които може да „притежава" много пийрове. (Където притежава е начин за упоменаване на дъщерно съществувание без значение от статуса на „притежаваните").

4.2.2. Мисия на продукта

В нашият случай клиента разполага с тракер, а всеки един от потребителите на системата му има битторент клиент. Мисията на продукта е симулирането и управлението на битторент клиент на потребителите на продукта от собственика на тракера (въпросния клиент). Мисията на продукта на ниво разработка е реализираното разпределена система за съхранение на данни. Начина на опериране с данните в тази разпределена система за съхранение на данни зависи от възможностите на използвания битторент клиент.

От програмна гледна точка проблема при битторент протокола е че е необходимо наличието на работещ тракер и достатъчен брой сийдъри на достатъчен брой торенти за да предизвика интерес от останалите потребители. Реализирането на една работеща система без външната намеса на потребителите премахва времето необходимо за набирането на скорост на определен тракер.

Друго приложение на този продукт е да замени локалните сървъри в частните мрежи, и гъвкавостта им при отварянето на достъпа към данните за глобалната мрежа (Просто репликираме базата данни на локалния тракер към тази на по-големия външен тракер). В по-мащабен план идеята на този продукт е да даде популярност на използването на разпределените файлови системи.

4.2.3. Избор на архитектура

За оценката на архитектурата отговаря Логиста, подпомаган от Инженерите по знание.

4.2.3.1. Опит с CTorrent

В нашия случай ние първо се спряхме на Ctorrent ПОК. Основното му предимството бе простото и разпаралелируемо ядро. Самото му ядро предполагаше инкапсулация на ниво торент. А ние имаме нужда да реализираме клиент (поддържащ обработката на торентите и техните състояния едновременно). В този случай когато първо решихме да се позовем на разучаването на този ПОК по отношение на бизнес изискванията сгрешихме. Изхабихме немалко време докато разберем че архитектурата на този ПОК не издържа на натоварването на което трябва да се подложи цялата система. Или не отговаря на основно архитектурно изискване: надеждност.

Опита ни за разпареляване на ниво нишка и на ниво процес се оказа неуспешен тъй като не отговаряше на това изискване. Необходимо ни бе не ядро което да може да се разпарарели на ниво нишка или процес, а ядро с вътрешно разпарарелияване реализирано чрез state machine.

4.2.3.1. XBT архитектура

Самия XBT (eXtended BitTorrent) клиент представлява сървър за нашето комерсиално приложение – „web frontend”.
Backend(the server part) - ядрото на клиента предоставя на този web frontend специфично API.

- Той не само че е мулти-торент ориентиран, но и предполага лесното написване на мулти-клиент архитектура управлявана от един web-frontend. (Много добра дисоциация и идентификация на различните backends, разположени на една или повече машини).

- Използва persistent data storage – или временни външни смущения които принудило биха рестартирали ядрото не са проблем.

- Вътрешно реализирана state-machine с богат набор от интерфейси за управление като
  - Конзолен – изключително удобен за провеждане на unit тестовете.
  - Графичен – предоставящ ни богат набор от външна логика която да използваме. Използването на този интерфейс предполага използването на backend-а като библиотеки необходими ни за изграждането на един графичен клиент. Благодарение на наличната функционалност на този графичен клиент ние имаме широкия поглед към реализирането на по-детайлни изисквания. Сравняваме функционалността на другите клиенти спрямо тази на нашия графичен клиент с цел цялостно развитие на проекта под напора на конкуренцията. (Много от новата функционалност в разширяването на web интерфейса е аналогична на тази в графичния клиент. А нас ни интересува именно web интерфейса)
  - Web интерфейс – стандартен bencoded протокол удобен за скриптовите езици като използвания PHP.
- Целия проект е написан на език от ниско ниво (C++) , което предполага бързодействие и безотказност. Освен това целия проект е междуплатформен (междуплатформеността е реализирана чрез условни дефиниции за компилация на различните платформи).

- И не на последно място функционалността е клас-ориентирана или както възможностите на C++ предполагат цялата архитектура е базирана на обектно-ориентирано програмиране. Минималния overhead внасян от използването на обекти е поносим, тъй като основната тежест в производителността пада върху дисковите операции.(Трансфера на всяка една единица медия данни в крайна сметка опира до дискова операция).

Изброеното описание на този ПОК ни кара да изберем него пред всички разглеждани ПОК като твърде елементарния CTorrent и твърде ресурсоемкия Azureus.

4.2.4. Проучване на XBT client

Успешното изпълнение на този етап е съпроводен от доста участници в проекта. Освен Логиста и инженерите по знание като главна движеща сила, е необходим опита и на системния администратор и тестера. В крайния доклад бива замесен и мениджъра на проекта, като отговорник за бизнес изискванията на клиента.

4.2.4.1. Тест за надеждност и производителност

Горепосоченото описание са също и изводи от едно некратко самостоятелно тестване на всеки един от интересуващите ни ПОК. XBTC (eXtended BitTorrent client) бидейки междуплатформен предполага изследване за най-добрата платформа за работа. Избираме Линукс дистрибуцията – Debian.
Чрез подходяща конфигурация успяваме да увеличим броя на проблемните максимално използвани файлови дескриптори, създаваме различни инстанции за различните XBT клиенти, и администрирайки машината да разпределим bandwidth-а между XBT клиенти.След това тестваме производителността на клиента за следните постановки:
- при натоварен и продължителен обмен на медия данни (трансфер на данни)
- при бързи промени на състоянията на данните (динамични торенти, пийрове, медия данни)
- при голямо количество торенти и пийрове
- комбинация от трите горепосочени постановки

За разлика от опита с CTorrent където първо изследвахме бизнес възможностите на проекта, а след това архитектурата тук изследването бе в правилния ред.

4.2.4.2. Анализ на бизнес възможностите на проекта

След като задоволихме приоритетно „архитектурното" изискване, започваме да изследваме доколко съществуващата бизнес логика отговаря точно на бизнес нуждите ни. (Ако под направеното „изследване на ПОК на база на бизнес изискванията" разбираме общото разглеждане на бизнес възможностите, то при по задълбочения анализ трябва да детайлизираме кои от тях и доколко ще се използват и модифицират и какви нови възможности трябва да бъдат добавяни).

Ако достатъчно много изисквания са задоволени и/или нямаме по-добра налична алтернатива на този ПОК решаваме че него ще използваме в нашия проект. Ако твърде слабо задоволява бизнес изискванията един проект, то ще е необходима много работа за постигането и. Трябва да си направим сметката дали е рентабилно да се захващаме със всеки един специфичен ПОК.

4.2.4.3. Създаване на спецификация на проекта

Рядко ПОК имат спецификация и почти никога документация. Първата крачка в изнасянето на част от управлението на един ПОК е създаването на спецификация на ПОК (за ПОК от по високо ниво или въпросните с по-голям обхват е много вероятно да е необходима и документация, но този случай не е разгледан в примера с този проект).

При създаването на спецификацията започваме да използваме програмните инструменти, чието използване ще продължи по времето на целия живот на проекта. Работата на софтуерните инженери започва да дава отражение на работния процес като цяло. Самата спецификация описана в следващата точка спазва точен протокол на описание на функционалността описан също в wiki системата.

4.3. Софтуерен процес – Програмни инструменти и политика на използване

Тук се намесват и софтуерните инженери. В проекта тяхната работа по дефинирането на програмни инструменти (freeware или други ПОК) може да се обобщи хронологично:

1) Трябва да се предостави система за следене на клиентските изисквания, която да може да пази история на дискусиите и предложенията, да ги класифицира и да зададе приоритети на задачите в тях. В началото за целта бе ползван локален форум, но с еволюирането на проекта се установи че „wiki" системата е много по-добра за целта. Използвана бе "tiki-wiki" а по-късно и "media-wiki".

2) Wiki системата може да се ползва успешно за съхранение на вътрешни документи като процедури дефиниращи работния процес, спецификации (като тази която се създава за ПОК в 3.1.4.3) и други вътрешни правила. Освен възможността за съхранение на история на дискусията тази система предлага и възможност за преглед на промените в дадени документи – changelog.

3) На участниците в проекта, които се занимават с разработка и на техните мениджъри е необходима прозрачна система за съставяне на „план на проекта". Такъв тип система е online project management системата. Софтуерните инженери се спряха на "dotproject" – ПОК предлагащ доста добра компилация от визия и възможности. По-късно се наложи комерсиалната „celoxis" заради по-добрите възможности при определяне на приоритета на дадени задачи и под-задачност. Впоследствие се наложи интегрирано решение като "trac" (ПОК обединяващ в себе си SVN, wiki-system, bug tracking system, project management

system), или отново се премина на ПОК. За менажирането на този проект бе използван dotproject.

4) По време на разработката е необходима система за контрол на версиите и политика за разклоняване (branching) и именуване на клоновете версии. Софтуерните инженери продложиха SVN (subversion) като технология и конзолния SVN клиент като най-удобен за част от разработчиците. За друга част от разработчиците бе разработена собствена web обвивка на SVN, поддържаща web интерфейс за опериране с SVN.

5) След което идва нуждата от bug tracking system. Която в последната фаза (при изграждането на продукта става централната система, съдържаща препратки към останалите системи. (във всяка една програмна грешка се посочва за коя точно версия става дума, препратка към wiki ако е необходимо, и описание на задачата от project management системата често дори в заглавието на програмната грешка). Използваната система bugzilla също представлява ПОК.

Софтуерните инженери работещи по АПОК методологията предпочитат да налагат програмни инструменти които пак за ПОК. *Или разработваме проект стараейки се да използваме само ПОК.* В wiki системата се описват и по-общи данни като бизнес организацията на фирмата (презентационен сайт в wiki системата), ролите и отговорностите свързани с отделни личности за даден проект или кои участници какво отношение имат към даден проект. Освен гореописаните практики дефиниращи софтуерния процес са необходими специфични процедури процедурите и избраната политика за работа с горепосочените програмни инструменти. Използваните програмни инструменти до голяма степен налагат „правилата" в един софтуерен процес.

4.4. Комуникация между бизнес средата и ПОК обществото

След като Логистиката е проучила проекта, изяснила си е идеята му идва ред на двупосочната комуникация между „Адаптация" и ПОК обществото.

- Първо описваме на участниците в „Адаптацията" (и по специално на главния архитект) кои бизнес изисквания са вече реализирани и до колко имплементацията на всеки един от тях задоволява изискванията ни.

- След това бе обяснено на ПОК обществото каква функционалност ни е необходима и до колко е удачна нейната имплементация. Тази която е удачно да се имплементира е приоритетна. За останалата се води задълбочена дискусия относно смисъла и приложението и. Ако тази нова функционалност изисквана от нашето бизнес приложение е извън целите на ПОК се разграничаваме от ПОК в отделно дърво (правим наша отделна версия на ПОК разграничаваща се от тази в глобалното хранилище на проекти, в случая Sourceforge).

- При разработването на отделна функционалност в бизнес приложението ни (web frontend) демонстрираме нуждите си за съответстваща функционалност от страна на ПОК. И се стараем да получим „приемственост" на идеите си от страна на ПОК обществото за този проект.

4.5. Прототипизиране. Ползите от прилагането му

В под-процеса Адаптация в сивата фаза след извеждането на мисията на продукта се създават два прототипа. Единия „първичен" имащ за цел да изясни отношенията ни с клиента, а другият „зрял" имащ за цел да изясни отношенията между Адаптация и ПОК.

След като на мениджъра на проекта, поемащ ролята и на отговорник за клиентските изисквания, е изяснил с клиента визията си за проекта заедно с Главният архитект създават първичния прототип. В този проект първичния прототип представляваше един дизайн на web

сайт с идея за стояща зад него функционалност. Мениджъра се проявява и като бизнес анализатор на клиентските изисквания, а главния архитект като web дизайнер. За разработването на зрелия прототип мениджъра на проекта решава какъв минимален човешки ресурс би бил необходим за разработването на целия комерсиален продукт. Докато тази преценка за ПОК и Логистиката я прави Логиста, за човешкия ресурс в Адаптацията е отговорен само Мениджъра на проекта. Адаптацията се разширява с достатъчен брой разработчици които след като дочакат първите резултати от Логистиката ще се впуснат в разработката на зрелия прототип под ръководството на главния архитект. При едно обектно-ориентирано програмиране, като това застъпено в нашия проект, главния архитект моделира класовете и интерфейсите, а тяхната имплементация оставя на разработчиците.

Съществената полза от прототипизирането за Адаптация разработчиците е, че те трупат опит като разработчици в тази сфера, сработват се като екип, запознават се с проекта и със софтуерния процес. Клиента с първичния прототип вижда визуално своите идеи пречупени през погледа на бизнес анализатора ни. А зрелия прототип демонстрира на клиента една симулация на работата на реалния му продукт. Въпреки наложените ограничения по отношение на ресурси финализирането на сивата фаза за този проект ни отне един месец време, което е напълно допустим срок. А изкарването на стабилна версия на продукта ни отне още два месеца (бялата фаза).

4.6. Планиране и моделиране

В тази подточка ще разгледаме прилагането на практиките: „спринтово планиране", „спринт" и „визуално моделиране".

Сприновете са затворени интервали от време през които се изкарват нови версии на продукта. Или един производствен цикъл се изпълнява без намесата на клиента.

В сивата фаза основните вътрешни срещи са между:

- Логиста и Менажера на проекта - синхронизират непрекъснато плановете си с цел смаляване на критичния път.
- Логиста и Инженера по знание – даване на качествени и количествени характеристики на плана на ПОК.
- Главния архитект и Логиста – изясняват технически въпроси свързани с комуникацията между ПОК и Адаптацията.
- Главния архитект и Менажера на проекта – създават първоначалния дизайн на проекта.

Онлайн комуникацията е достатъчна при създаването на прототипите и "the unit tests". Tiki-wiki е достатъчно добра система за описването на задания към разработчици по дефиниран (пак там) протокол за описание. В wiki системата се записват и резултатите от дискусиите или запазва се цялата история на проекта.

В бялата фаза основните срещи са между:

- Главния архитект и Адаптация разработчиците - всеки един разработчик заявява кой клас функционалност ще реализира и сам определя времето необходимо му за имплементацията му. Моделирайки техния клас функционалност Адаптация разработчиците се намесват и в съставянето на неокончателния план на проекта. Генерират подзадачи и планират времето за тяхното изпълнение (модифицират плана в project management системата ни - dotproject) стараейки се да се вписват в крайния срок за изпълнение на задачата.
- Логиста и ПОК разработчиците – подобно на горепосочения случай, самоинициативата е движещата сила в срещите за самоорганизиране на екипите.

Всеки един разработчик (ПОК или Адаптация разработчик) освен че заявява кой клас функционалност ще разработва, рапортува евентуалните проблеми свързани с разработката също така и моделира визуално на черната дъска чрез UML начина на разрешаване на проблема. Участниците в дискусията трябва да са запознати с езика на комуникацията – UML.
Докато при срещите в сивата фаза се чертаят основно „the use cases", sequence и collaboration диаграми (още Communication and Activity diagrams) то при срещите в бялата фаза се чертаят основно class diagrams и state machine diagrams (още, макар и само веднъж и Object диаграми).
Резултатните диаграми претърпели дискусия също се описват в wiki системата.

4.7. Бяла фаза и преход към нея

4.7.1 Критичен път.

Със завършването на сивата фаза разполагаме със изготвен приблизителен план на проекта разделен на задачи и вписани в него времеви ограничения. Разполагаме с разработен интерфейс за връзка между frontend-а и backend-а. Във въпросния план може да отбележим две паралелни разработки от страна на ПОК и от страна на Адаптацията. Едната от двете разработка несъмнено лежи на критичния път, което ни показва една от основните задачи на менажера на проекта и на логиста които трябва така да оптимизират ресурсите на Адаптацията и на ПОК че да скъсят максимално критичния път.

4.7.2. Промяна на отговорностите

При прехода от едната фаза в другата много от участниците променят своята работа към сходна такава. Всяка роля предполага дадена отговорност, но към някои по-обобщени роли са прикачени няколко отговорности. Именно тази работна промяна ще опишем.
Във бялата фаза Бизнес анализатора трябва да се проявява повече като менажер на проекта.
Главния архитект при една изяснена архитектура започва да се проявява като ментор и лидер на екипа от разработчици на адаптацията.

До този момент (сивата фаза) подпомагащия успешното протичане на unit тестовете (на всяка една взимана в предвид функционалност на ПОК) тестер в бялата фаза бива натоварен да тества версията на продукта при всяка итерация.

В бялата фаза също Логиста поема ролята и на лидер на екип от разработчици на ПОК. Докато във сивата фаза ПОК разработчиците са изработвали само unit тестове в бялата фаза започват да моделират, кодират и тестват нововъведената от тях функционалност.

4.7.3. Итеративна разработка

В тази подточка ще разгледаме прилагането на доста от практиките при разработването на този проект по АПОК методологията.

За нуждите от изграждането на нашия проект идентифицираме три главни итерации:

1) Подобряване на протокола за front-end<->back-end комуникация, „Error handling and logging”.
2) Добавяне на нова функционалност.
3) Вътрешен “peer-to-peer firewall”.

- Изпълнението на всяка една итерация за Адаптацията е свързано с :
    - така нареченото спринтово препланиране в което главния архитект, адаптация разработчиците които ще разработва дадената функционалност и менажера на проекта се събират за да обсъдят планираната итерация, да се разпределят класовете функционалност между адаптация разработчиците и те да демонстрират на колегите си избрания модел за реализиране на итерацията. Тези две срещи (Една за планирането и една за моделирането) се извършват на така наречените sprit meetings.
    - След което Адаптация разработчиците кодират функционалността си и в тесен контакт с тестерите подготвят въведената функционалност за интеграция.
    - Системните администратори настройват работните машини (сървъра за разработка и клиентската машина) , новата функционалност бива интегрирана в системата и инсталирана на машината за разработка, където след финален тест, бива качена на клиентската машина.

- Изпълнението на всяка една итерация за ПОК е свързано с :

- Дискусия с ПОК обществото за нуждата от тази функционалност.
- Разработването на отделна версия на ПОК и обединяването и със текущата и в рамките на нашия проект и в рамките на ПОК обществото. Принципа за „приемственост на промените" е основна движеща сила при ПОК.
- Обновяване на промените базирани на външна разработка (разработчици от ПОК обществото) в нашата работната версия на ПОК. (В случая ръчна синхронизация на версиите на системите за контрол на версиите, спазвайки дефинирана от нас политика за контрол на версиите). Тъй като имаме колективна собственост на кода всеки външен потребител може да оправи проблеми свързани с правилния начин на работа на целия ПОК и ние сме длъжни да се съобразяваме с това. Кода на ПОК непрекъснато се преструктурира от наши и външни разработчици.
- Изграждане и минаване на "the unit tests" за въведената функционалност. Повечето от "the unit tests" са написани още във сивата фаза, в бялата се дописват само тези които са свързани с тестването на новата функционалност. ПОК принципа за непрекъснатата интеграция и тестване е широко застъпен с цел да разполагаме с работеща версия на ПОК непрекъснато.

4.8. Ограничения и адаптация към тях

Често проектите имат или ограничения или във финансовите ресурси или във времевия ресурс или в човешкия ресурс. Основното ограничение на този проект бе във човешкия ресурс. Затова се наложи някои участници да поемат повече от една роля. Наложилото се сливане на роли удвои работата на някои индивиди и срока за изпълнение на поръчката съответно.

Един индивид изпълняваше ролята на Логист и на Инженер по знание и на ПОК лидер и на Софтуерен инженер. Друг участник изпълняваше ролята и на главен архитект и на адаптация разработчик и на софтуерен инженер. Трети участник изпълняваше ролята и на бизнес анализатор и на системен администратор, инсталатор и интегратор и на софтуерен инженер. Друг участник бе и адаптация разработчик и тестер. Клиента бе представен от отделна личност. Доста от участниците в ПОК под-процеса бяха индивиди от ПОК обществото и играеха ролята и на ПОК разработчици.

В последната точка от тази глава ще опишем скица на суровата схема на АПОК процеса при разработването на комерсиалния ни продукт използващ модифициран XBT Client.

4.9. Бъдещо развитие на XBT ПОК

Както бе изяснено, работата по един ПОК не приключва докато има активни участници по него. В случая е запланувана архитектурна промяна по XBT клиента, целяща подобряване на производителността на проекта при върхови натоварвания. Става дума за заменянето на остарелия, но универсален начин (имплементирано за всякакви операционни системи) , за организиране на мрежови проекти:

- на базата на циклещата по всички мрежови гнезда (sockets) функционалност – "select()"
- със новия събитийно ориентиран начин на обхождане на мрежовите гнезда – epoll (наследника на poll).

Разликата между двата подхода може да се опише като обиколка на домовете от един пощальон. В първия случай пощальона чука на вратата на всяка къща и разпитва собственика дали има писмо за него. Докато във втория случай в пощенската служба са генерирани адресите които трябва да обиколи пощальона.

При по големи натоварвания разликата в производителността между двата подхода е значителна. Остарялата идея на select-а води до голям overhead (ненужна загуба на производителност).

Първият проблем на новия архитектурен подход е, че не е имплементиран за операционната система Windows, но би могъл да се използва чрез „условно компилиране" за Linux OS.

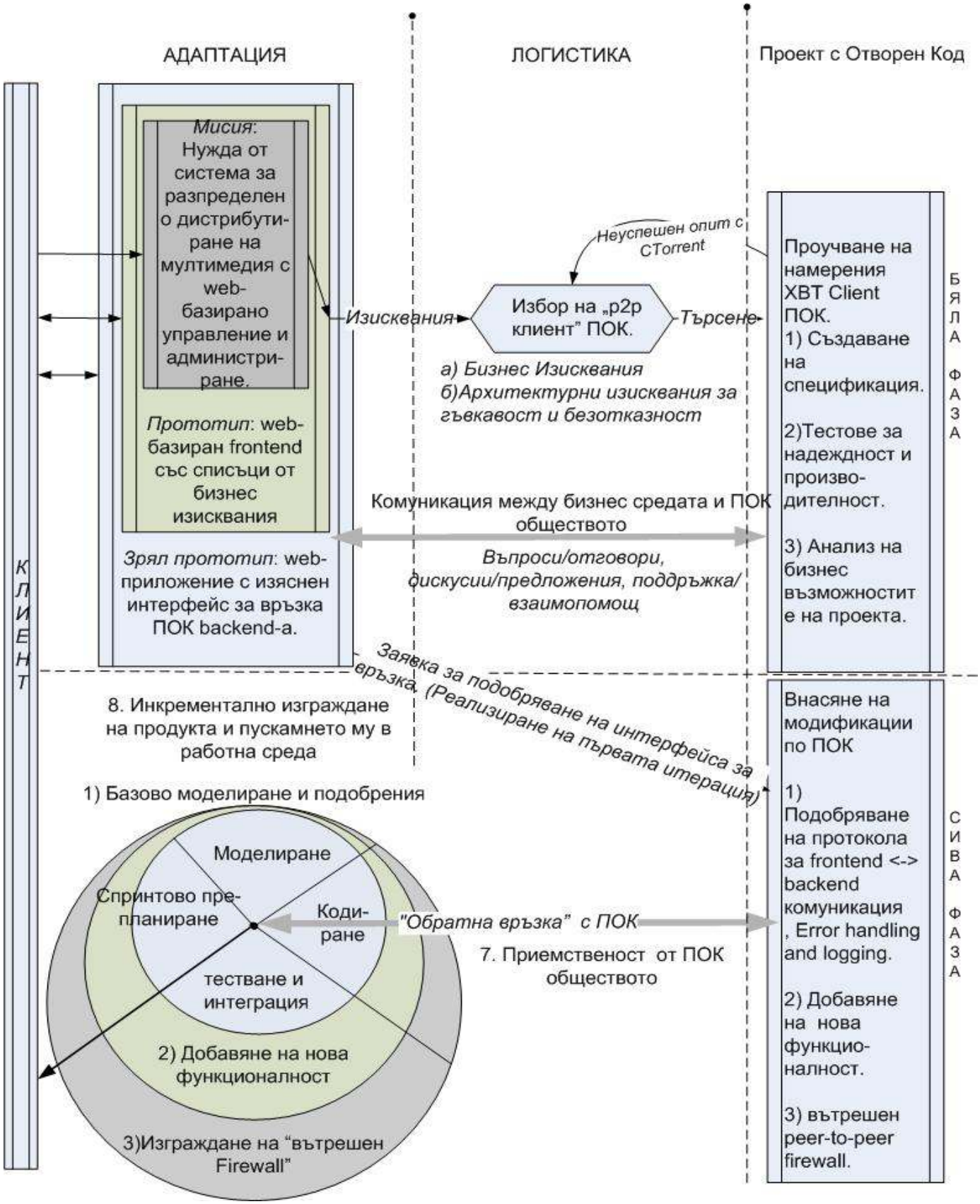
Вторият проблем бяха големите мащаби на модификацията. Внасянето на подобна модификация би ни струвала значително повече човекочаса отколкото компанията от която е част и дипломанта може да си позволи.

В следствие на горните два проблема тази модификация не бе одобрена от създателя на XBT ПОК, и бе оставена за имплементиране в бъдещето.

4.10. Сурова схема на процеса на разработка на проекта

Ще скицираме реалното приложение на софтуерния процес на АПОК за разработването на комерсиалния продукт използващ XBT Client ПОК. Скицата (фигура 4.10) описва постъпково целия изминат път до създаването на продукта.

Фигура 4.10: Сурова схема на процеса на разработка на проекта



5. Описание на добавената функционалност

Ще опишем функционалността на всяка една итерация, необходимостта от нея и пътя до достигането и. Няма да излагаме програмен код на web frontend-а поради законите за защита на авторски права. Но ще приложим пачът (patch) писан от дипломанта обновяващ програмния код на ПОК до необходимата ни функционалност. Заедно със този пач ще приложим и спецификацията описваща промените.

5.1. Описание на добавената функционалност

- Подобряване на frontend-backend протокола за комуникация

Комуникацията между front-end-а и back-end-а се осъществява чрез bencoded protocol, позволяващ конструкции като речници, листинги, целочислени и стрингови данни ...

Такъв тип протокол (bencoded) съдържа в себе си големината на разменяната команда и е лесен за обработка от скриптови езици като PHP, който разполага с библиотеки за обработка на такъв тип протокол. Именно затова в XBT client-а бе избран такъв тип протокол. Проблема в съществуващата комуникация беше че при изпращане на каквато и да е команда от страна на front-end-а не се получава отговор за успешно/ неуспешно изпълнение на командата. Нашия front-end не можеше да гарантира безотказност. Затова се наложи да се реализира handshaking механизъм. Или винаги да получаваме отговор за резултата от изпълнението на дадена команда.

Това ни навежда на мисълта за следващата модификация („error handling and logging”). Двете модификации са реализирани в една итерация поради високото ниво на преплитане на функционалност.

- Обработка и съхранение на грешки

Необходимо бе да дефинираме типове грешки, с които back-end-а да може да нотифицира front-end-а че изпълнението на дадена команда не е протекло успешно. И съответно от страната на web frontend се взимат мерки за прихващането и обработката на тази грешка с цел постигане на безотказност на системата. Освен разпращането на грешка в реално време реализирано горепосочената функционалност бе необходимо да се реализират и още два механизма за отчитане на грешки:

Първия механизъм бе опашка от съобщения за грешка реализиран като контейнер съдържащ код на грешката, съобщение за грешката и класификация на грешката. Тази опашка се обработва от frontend-а, позволявайки му да пази история на грешките, да ги визуализира и обработва независимо от времето на възникването на аварията.

Вторият механизъм бе записването на грешки в логове на машината на която се намира back-end-а. Той е полезен за откриване не на грешки от неправилен начин на работа като горепосочените механизми а за откриване на грешки и неточности при разработката. Този механизъм разполага с два режима на работа: „release” и „debug”, който се прилагат след условна компилация, минимизиращи записите в логовете или максимализиращи ги.

- Добавена минорна функционалност и оправени проблеми в проекта

В процеса на работа с този ПОК и ние както останалите разработчици от ПОК обществото сме се натъквали на не една дребна грешка или неточност. Намирането на такива грешки често и алгоритъма за тяхното отстраняване често е пращано директно на собственика на този ПОК посредством форума на въпросния ПОК.

Когато се опитваме да въведем дадена функционалност отваряме нова тема за дискусия във форума на този ПОК. Излагаме доводите си за необходимостта от този ПОК, изслушваме останалите участници в дискусията и предлагаме решението си като пач.
Понякога нашите идеи са били оборвани и са ни били показвани грешките в идеите ни, но немалко от нашите идеи за добавяне на нова функционалност или сливане на съществуваща е била приемана също от XBT ПОК обществото. Пример за такива идеи са дадените в приложението с програмния код идеи за сливане на “remove torrent” и “close torrent” командите в една по обща с въвеждането и режим на подаване. Пример за добавена функционалност е добавянето на командата „space left” уведомяваща ни за оставащото дисково пространство на машината на която се намира backend-а на XBT.

От изложената разработка по този ПОК бяха идентифицирани две итерации. Тази функционалност за разлика от долу описаната имаше пълна приемственост.

- internall peer-to-peer firewall

Същинската функционалност, която бе необходимо да бъде въведена е реализирането на вътрешна защита на правата за достъп до ресурсите на нашия битторент клиент. Необходимо ни бе да ограничаваме не само отделни потребители а да раздаваме различни права на достъп на различни видове мрежи в глобалното интернет пространство. Идеята за ограничаването на потребителите на даден доставчик на интернет на вътрешно ниво, раздаване на право на потребителите от дадена локална мрежа само да извличат данни, но да не могат и да качват такива в системата (както и обратното) могат да бъдат реализирани и на вътрешно ниво.

Ако подобна функционалност бъде изпълнявана от външни инструменти за ограничаване на достъпа , като iptables, ipchains, trafic shaper и други, системата няма да издържи на натоварването. Ще се предизвика напълно излишно натоварване на системата (overhead) което ще я извади от правилния и начин на работа. Всички заявки за данни или информация за състоянието им, известни още като анонси (announces), ще продължават да бъдат правени и спирани на външно ниво, което ще доведе до допълнително натоварване на машината ни.
Реализирането на вътрешен peer-to-peer firewall разрешава проблема ни. В приложението със спецификацията на проекта ще опишем дефинирания интерфейс за управление на този firewall.

Тази функционалност не бе приета от ПОК обществото. След задълбочена лична дискусия със откривателя на проекта, открихме идеологични противоречия, които наложиха разграничаването на нашия модифициран ПОК в отделна версия на този ПОК. Идеологически създателя на проекта не желаеше неговия проект да се превърне в удобна за администриране разпределена файлова система служеща за вероятното споделяне на пиратски данни.

5.2 Спецификация. Приложение

Спецификацията е дадена във формат описан в нашата wiki система. Под формата на метаезик (Потребителска заявка, Въпрос, Действие, Отговор, Решение) дефинираме спецификацията както следва:

- Локация на пача и скрипта за тестването му

- patch.txt – съдържа пач версията на модификациите на XBTC проекта от версия 0.67 до нашата разграничена от дървото версии версия на XBTC проекта.

- XBTCLient.php и xbt_cli.php – съответно са APIто използвано от web frontend-а и конзолно приложение за трасиране и тестване на XBTC.

PATCH:
http://xbt.hit.bg/patch.txt

PHP CLASS (XBTCLient.php) and PHP xbt_cli (xbt_cli.php) and a INSTAL HOWTO for xbt pear module:
http://xbt.hit.bg/xbt_cli-1.0.tar.gz

- Описание на изисквания за модификацията „internal peer-to-peer firewall”

Всяка една модификация е описана под форма на
- Изискания – разбито на точки описание на изискваната функционалност
- Нотация – описание на метаезика използван за описване на всяка една точка функционалност.
- Решение – самото описание на еденица функционалност от тази модификация

==========
The first feature described bellow is the "internal peer-to-peer firewall":
==========

-=::XBT Backend Communication Protocol::=-

{maketoc}

!I.Request
In order to achive the required behavior of the "Super Seeder" machines
The basic idea is to develop some kind of system, that will make possible to allow all connections to and from the XBT Backend if the peer is still in leeching stage. Once seeding commences:
1) all connection to the backend for that peer (torrent), that are not made from an "ALLOWED" ip will be refuzed.
2) all connection to the backend for that peer (torrent), that are made from an "DENY"(denied) ip will be refuzed.
3) A global rule "net filter leeching" defines what happened with the torents in leeching state (will they be filtered or not)
4) A global rule "net filter seeding" defines what happened with the torents in seeding state (will they be filtered or not)
5) all connection to the backend for that peer (torrent), that are made from an ip in the net which :
5.1) seeding is filtered, will be blocked while "seeding".
5.2) leeching is filtered, will be blocked while "leeching".
5.3) seeding and leeching are filtered , will be blocked in "both" cases.
The behavior cannot be achieved otherwise, because the rules must apply once the TCP connection is made, and the firewall decision depends on the actual torrent and state of the torrent.

If you have two net filters, the less descriptive one will be applied last. If you have two filters with the same "net" the filter, which net mask is lower will be applied last.
(If an IP address is matched in more than one rule, the rule with the greatest net mask will be applied)

!-II.Notation
All of the mechanisms specified in this document are described in both prose and an augmented))Backus-Naur((Form (BNF) similar to that used by [http://www.w3.org/Protocols/rfc2616/rfc2616-sec2.html#sec2|RFC 2616]

===TAGS===
~~RED:UNKNOWN~~ : The Protocol is incomplete, and something needs to be entered at this spot

===Command notation===
"<Description of what the command does>"
- Question:
<Description of the command to send>
<key> => {<type>[<options>]} // <Value explanation>
...
- Response:
<The responce of the backend>

<key> => {<type>[<options>]} // <Value explanation>
...

!III.Solution
Those options are implemented via set/get options:

!!"get options: Fetch client option"
- Question:
action => get options
- Response:
net filter leeching => {int 0|1} // 0:Allow, 1:Deny.
net filter seeding => {int 0|1} // 0:Allow, 1:Deny.
net filter => //The structure(dictonary) which is filled with a list of rules, ips and nmask in the backend
  rule => {int} 0|1 //enumerated: (-1: error(unititialized)), 0: allow, 1: deny
  mode => {int 0|1} //Optional. defines what state of the torents the rule will filter (Seeding, Leeching). If the key is missing, the rule is applied for both seeding and leeching torrents
  net => {long int} //In Socket style format long ip.
  net mask => {int} //The bit mask used over the ip mentioned above.Example 234.23.0.0/24. (the last number(24) is the nmask).

!!"set options: Set client options"
- Question:
action => set options
net filter leeching => {int 0|1} // 0:Allow, 1:Deny.
net filter seeding => {int 0|1} // 0:Allow, 1:Deny.
net filter => //Pass the structure(dictonary) which is filled with a list of rules, ips and nmask.
  rule => {int} 0|1 //enumerated: (-1: error(unititialized)), 0: allow, 1: deny
  mode => {int 0|1} //Optional. defines what state of the torents the rule will filter (0=Seeding, 1=Leeching). If the key is missing, the rule is applied for both seeding and leeching torrents
  net => {long int} //In Socket style format long ip.
  net mask => {int} //The bit mask used over the ip mentioned above.Example 234.23.0.0/24. (the last number(24) is the nmask).
- Response:
// 0(empty dictonary): no error, "Dictonary": error/warning
warning|error(type of the error) => {string} //description of the error

---
!!"add net filter: Add a new net filter rule"
- Question:
action => add net filter
//Adds a new record into the structure(dictonary) which is filled with a list of rules, ips and nmask.
rule => {int} 0|1 //enumerated: (-1: error(unititialized)), 0: allow, 1: deny
mode => {int 0|1} //Optional. defines what state of the torents the rule will filter (0=Seeding, 1=Leeching). If the key is missing, the rule is applied for both seeding and leeching torrents
net => {long int} //In Socket style format long ip.
net mask => {int} //The bit mask used over the ip mentioned above.Example 234.23.0.0/24. (the last number(24) is the nmask).
- Response:
// 0(empty dictonary): no error, "Dictonary": error/warning
warning|error(type of the error) => {string} //description of the error
---
!!"remove net filter: Remove a net filter rule"
- Question:
action => remove net filter
// we do not enter an rule becouse you must not add a different rule for the same ip and nmask.
net => {long int} //In Socket style format long ip.
net mask => //The bit mask. (look above).
mode => {int 0|1} //Optional. defines what state of the torents the rule will filter (0=Seeding, 1=Leeching). If the key is missing, buth Seeding and Leeching rules are removed

- Response:
// 0(empty dictonary): no error, "Dictonary": error/warning
warning|error(type of the error) => {string} //description of the error

- Описание на едно от минорните изисквания – разширената „close command”

Команди като close command, space left, … са реализирани на втората итерация и са обединени в една модификация. Ще опишем само “close command” по гореописания метаезик.

==========
The second major feature is an extension of the close command. The idea is to prevent an external intervation (by shell script) to a mashine who is leeching/seeding.
==========

!I.Request
In order to make the torrent client self-sufficient, the close command needs to allow the user, to remove the data and/or the torrent metafile on close. The command must remain failsafe. Either the whole command will be executed correctly, or the command will fails and will have NO effect on the backend!

Here's how the new close torrent command showld look like

!!"close torrent: Close the torrent."
- Question:
action => close torrent
hash => {string} //20 bytes torrent info hash
remove torrent => {int 0|1} // Optional. If missing, assumed 0. Delete the metainfo file
remove data => {int 0|1} // Optional. If missing, assumed 0. Delete the data from Completes/Incompletes
- Response:
//0(empty dictonary): no error , dictonary: error
=> {string warning|error}//type of the error
=> {string} // The describtion of the error/warning
=> {string}//list of the files which proccessing is a problem. Followed by the error code (returned by the system calls).

- Описание на комуникационния протокол на XBTC – backend to frontend

Комуникационния протоколо представлява размяна на команди между backend и frontend. Структурата на тази точка е както следва:

- Нотация – описание на използвания метаезик за описание на протокола. Предостаена е препратка към Backus-Naur начина в отговор на въпроса защо е използван такъв формат при дефинирането на метаезика.

- Абривиатура на използваните съкращения – превод от понятията от тематични описания до програмни такива.

- Команди – описание на отделните команди.

==========
Here is the description of the XBT Communication Protocol. (the first step to make an PHP class encapsulating the backend<->frontend communiaction logic.). Using the class is demonstrated in the xbt_cli.
==========

-=::XBT Backend Communication Protocol::=-

{maketoc}

!-I.Notaion

All of the mechanisms specified in this document are described in both prose and an augmented))Backus-Naur((Form (BNF) similar to that used by [http://www.w3.org/Protocols/rfc2616/rfc2616-sec2.html#sec2|RFC 2616]

===TAGS===
~~RED:UNKNOWN~~ : The Protocol is incomplete, and something needs to be entered at this spot

===Command notation===
"<Description of what the command does>"
- Question:
<Description of the command to send>
<key> => {<type>[<options>]} // <Value explanation>
...
- Response:
<The responce of the backend>
<key> => {<type>[<options>]} // <Value explanation>
...

!II.Messages

All messages sent to the backend use a standart TCP/IP socket.
On every client request, the server sends one responce.

^
request = message-length "@" message
response = message-length "@" message

message-length = 4OCTET ; The length of the message + 1 presented in unsigned long (always 32 bit, big endian byte order)
message = 1*CHAR ; Bencoded string^

From now on the request messages will be called a QUESTION and the response messages will be called RESPONSE

!III.Commands

!!"open torrent: Open a torrent for leaching/seeding and start the transfer"
- Question:
action => open torrent
torrent => {string} // bencoded torrent source
- Response:
{int 0|1|2} //1-successful.1 - error.unable to open the file. 2 - error.unable to parse the file.

!!"close torrent: Close the torrent."
- Question:
action => close torrent
hash => {string} //20 bytes torrent info hash
- Response:
 {int 0|1} //0-successful.1-error.

!!"get status: Fetch server status"
- Question:
action => get status

- Response:
files =>
  {string} //20 bytes torrent info hash =>
    complete => {int} //The number of seeders having on open TCP connection with the backend for the torrent. (the current session only)
    complete total => {int} //The total number of all seeders for the torrent (from all session made). This value is updated when you make an announce, and the tracker includes this value in the responce. (At the scrape)
    down rate => {int} //download bitrate (in bytes per second).
    events => //The structure(dictonary) which is filled with events in the backend, it contains the "message" field (string) which represents the current event followed by the "time" field (when the event emerg, occures(string)).
      level => {int 0|1|2|3|4|5|6|7} // 0:emergency,1:alert,2:critical error, 3:error, 4:warn, 5:notice, 6:info, 7:debug.
      message => {string} //The description of the event.
      time => {long int} //In "UNIX timestamp".
    incomplete => {int} //The number of leechers having on open TCP connection with the backend for the torrent.(the current session only).
    incomplete total => {int} //The total number of leechers for the torrent. (from all session made). This value is updated when you make an announce, and the tracker includes this value in the responce. (at the scrape)
    left => {int} //Bytes left to download (0 = seeding)
    name => {string} //The filename (including the directory name) of the torrent (i.e. "/store/Incompletes/RealPlayer10GOLD.bin")
    priority => {int 1|0|-1) //'high priority' = 1 , 'normal priority' = 0 , 'low priority' = -1.
    size => {int} // Total size of the torrent in bytes.
    state => {int 0|1|2|3|4|5} //'queued' = 0 ,'hashing' = 1, 'running' = 2 , 'paused' = 3 , 'stopped' = 4, 'unknown' = 5. State "unknown" should be impossible!
    total downloaded => {int} //The total number of bytes downloaded.
    total uploaded => {int} //The total number of bytes uploaded.
    up rate => {int} //upload bitrate (in bytes per second).
    started at => {long int} //The "UNIX timestamp", when the torrent is started.(0 - means that the torrent is not started yet)
    completed at => {long int} //The "UNIX timestamp", when the torrent is completed.(0- means that the torrent is not completed yet).
version => {string} // Version of the beckend.
space left => {string} // the numbers of disk space left on the device, in bytes.

!!"set priority: Set the priotity for the torrent"
- Question:
action => set priority
hash => {string} //20 bytes torrent infohash
priority => {int -1|0|1} //1 is the highest
- Response:
{int 0|1} //0-successful.1- error.

!!"set state: Set the state for the torrent"
- Question:
action => set state
hash => {string} //20 bytes torrent info hash
state => {int 0|2|3|4} // 0 - queued, 1 - hashing, 2 - started, 3 - paused, 4 - stopped.
- Response:
{int 0|1} //0-successful.1- error.

!!"get options: Fetch client option"
- Question:
action => get options
- Response:
admin port => {int} // The port the xbt backend communicted with the frontend(s)

completes dir => {string} // The folder where the completed torrent files are stored
incompletes dir => {string} // The folder where the non-completed torrent files are stored
peer limit => {int} // The maximum number of peers the client can open (0 = unlimited)
peer port => {int} //The port for communication with other peers
seeding ratio => {int} //Look above.
torrent limit => {int} //The number that represent the maximum open torrents allowed restriction. No matter in which state they are.
torrents dir => {string} // The folder where the .torrent files will be stored
tracker port => {int} // The tracker UDP Port to use
upload rate => {int} // The upload rate limit in bytes per second (0 = unlimted)
upload slots => {int} // Maximum upload slots availabel (0 = unlimited)
user agent => {string} // Force a user agent header when connectiong to the tracker (empty = Default)

!!"set options: Set client options"
- Question:
action => set options
completes dir => {string} //Path to completes dir.
incompletes dir => {string} //Path to incompletes dir.
peer limit => {int} //Maximum number of peers allowed.
peer port => {int} //The port to use for the peers.
seeding ratio => {int 0-100} //The "complete total"(total uploaded data) / "size"(size of the torrent data). This option sets the seeding ratio at which the seeding will stop (the torrent will be automatically set to state stopped). Only numbers from zero to 100 are correct input.(0=Disable the feature).
torrent limit => {int} //The number that represent the maximum open (see open torrent action) torrents allowed restriction. No matter in which state they are. When the limit is reached, and you execute another open torrent action, whe "open torrent" command just will not be executed. (the checks for this must be made before you pass those action.).
torrents dir => {string} // The directory whe the torrents will be stored
tracker port => {int} // The tracker UDP Port to use
upload rate => {int} // The maximum upload rate in bytes per second (0 = unlimited).
upload slots => {int} // The maximum upload slots available (0 = unlimited)
user agent => {string} // Force a user agent header when connectiong to the tracker (empty string = Default)
// Any combination of the above fields is possible! The only necessary field is "action"
- Response:
{int 0} //No responce

!IV. Torrent life (states explained)
Commands: open torrent, close torrent, set state
States: 0 - queued, 1 - hashing, 2 - started, 3 - paused, 4 - stopped
When we pass "open torrent" command (we pass the error checks, and the torrent is successfully opened) the torrent goes in hashing state. An announce has been made, and if we don't recieve a socket error the torrent goes in started mode. Announce has been made.
When we pass "close torrent" command, the torrent goes in stopped state. The torrent data is cleared and make an announce.
When you pause a torrent you leave all the connections made opened. This means that connection identificator , transaction identificator, peer identicator and torent data such as uploaded, left, ports will be saved. The difference between stopped and paused torrent is that: in closed state those data is cleared (only "incomplete total" and "complete total" are saved to the so called local DB).

!V.Extending the protocol
Actually some of the clients, like "XBT Client", has much more options. But this is so, becouse it uses the backend libraries directly. This is not discussed here, couse this method do not use the standart universal request handling from the backend.
We are interested to extend the communication protocol, so any client can communicate with the backend.
We list here the current futures of the backend, many of them at now can be setted only via backend configuration. But if we extend the handle mechanism of a backend we can handle them. We must explain what stand behind the described blank strings("left","message",e.t.c.). Any of those blank strings is represened by a value (blank_string,value).

Overall: All types of actions:
=> close torrent
=> get options
=> get status
=> open torrent
=> set options
=> set priority
=> set state

All bittorent strings, that can be eventually setted or getted via "set ..." or "get ...". Interesting obversation: "message" value and "time" value are nested in the "event" value.
We shall explain the UNEXPLAINED strings before:
bts_announce = "announce"; //announce url.
bts_announce_list = "announce-list"; // the list of those announce urls.
bts_failure_reason = "failure reason"; // the dictonary filled with error codes if they occures.
bts_info = "info"; // (dictonary) that embeds the:
  name => {string}//explained before.
  pieces => {string} //explained before.
  piece lenght => {int} //explained before
  files => //explained before.
  path => {string} //explained before
  merkle hash => //explained above
  lenth => //explained before
bts_interval = "interval"; //The Non-Default announce interval in ms.
bts_ipa = "ip"; // The ip of the other client, to wich the peer has been made.
bts_merkle_hash = "merkle hash"; // search in google for merkle hash tree magic.
bts_path = "path"; // paths to the files in the torrent file.
bts_peer_id = "peer id"; //peer identifiers.
bts_piece_length = "piece length"; // the length of piece of data (in bits) that is transfered between to clients.
bts_pieces = "pieces"; // the measurement of data.
bts_torrent = "torrent"; // the main structure(dictonary) that embeds:
   announce list => //explained before
   announce => //explained before
   info => //explained before

- Примери за използването на xbt_cli и настройки на XBTC

Наред с останалите клиент на XBTC xbt_cli бе широко използван при реализирането на the unit tests. Негова особеност е, че той предполага използването на протокола за комуникация между backend и frontend на най-ниско ниво, без допълнителна транслация.
В тази част на приложението са описани и по важните програмни настройки на XBTC (hardcoded или опции взимани в предвид при условна компилация).

==========
Also the "xbt_cli" is posted. This client was also a very capable testing tool, becouse it uses an php code as an in parameter.
==========
Examples:

xbt_cli "array('action'=> 'get status')"

You can also update this xbt_cli to use xbt pear module, and even then work in the old way by passing "-r" as the first parameter.

xbt_cli -r "array('action'=> 'get options')"

xbt_cli -r "array('action'=> 'set options', 'net filter'=>array(array('rule'=>1, 'mode'=> 0, 'net'=>ip2long('82.103.65.181'), 'net mask'=>31)))"

xbt_cli -r "array('action'=> 'close torrent', 'hash'=> pack('H*', '7722af45bd9c22579b0f6bbc0a0371023722d0ce'), 'remove data'=>1, 'remove torrent'=>1)"

xbt_cli -a 'add net filter' -x "'net' =>-1062731776 , 'net mask'=>16, 'rule'=>1, 'mode'=>1"

//for a litle help how to use the cli via the pear module, just type the following:
xbt_cli --help

PS: the xbt class (XBTCLient.php) uses the this documentation as an assumption. So if you don't patch the "xbt client" you may not be able to use the whole class's functionallity.

==========
Some minor changes
==========
- Added an option to log the events in the log file. (just pass the proper parameter to the compiler -D<EVENTLOG|EVENT|NDEBUG|NETFILTERLOG>). //those events are filling the STL containers non-stop. And if you don't kill the backend, they will never be erased. (While debug the backend with the gdb, i saw how those STL calls are the main reason for the crashing of the backend)
- "update chokes" now happen 10 times rarely(100 ms). "update states" now happen 0 times rarely(150 ms). "save state" now happen 10 times rarely(600 ms). Checking the remote links and "running the scheduler" times are also increased 10 times. //i 'm trying to decrease read/write system calls.
- Some commands return code changed, to improve the error handling. "space left" command added. Just not to use the shell scripts to understand that. In my case i want everything to be handled by the backend.

6. Анализ на получените резултати и препоръки

Ще обобщим начина на излагането на тази разработка, и причините той да е именно такъв:

1) Разгледахме съществуващите гъвкави методологии, използвайки същия метаезик с който ще представим нашата нова гъвкава методология.
2) Сравнихме съществуващите гъвкави методологии, наблягайки на отличителните признаците и специфични критериите на някои от тях, използвани в нашата гъвкава методология.
3) Използвайки метаезика дефиниран при разглеждането на съществуващите гъвкави методологии и описаните понятия описани по подробно във втората точка, изложихме нашата нова гъвкава методология
4) Показахме, че нашата методология е гъвкава и я именуваме – АПОК (Адаптация на проекти с отворен код)
5) Разгледахме един проект разработван с използването на АПОК методологията. Показваме приложение на гъвкавата методология (Case study).
6) В тази последна глава ще покажем мястото на АПОК методологията в дървото на гъвкавите методологии. Ще изясним в кои случаи използването и е подходящо и в кои не.

В тази глава първо ще покажем какво е мястото на АПОК. След което ще въведем нов критерии за избор на ПОК – „перспективност". Ще покажем алтернативни начини на прилагане на АПОК и ще обобщим плюсовете и минусите от прилагането на АПОК.

6.1. Мястото на АПОК в света на гъвкавите методологии

В демонстрираното използване на АПОК, показахме че АПОК може да се прилага при разработването на проекти с малък бюджет и обхват. Но дали тази гъвкава методология е приложима и за по големи случаи?

В разгледания случай имахме само един екип разработчици за целия под-процес „ПОК". Логистът изпълняваше и ролята на лидер на ПОК екип. При ПОК с по-голям обхват със сигурност освен че няма да е възможно няколко роли да се изпълняват от един участник, ще се наложи и обособяването на няколко ПОК екипа, или няколко Адаптация екипа. Ако ползваме в нашия продукт повече от един ПОК може да се наложи и да имаме повече от един Логист. Или АПОК дава възможност за разпареляване на работата и малък и при голям обхват. Използването на АПОК би могло да е правилния избор и за проекти с нисък бюджет и за проекти с кратък срок за изпълнение. По принцип ползването на ПОК предполага нисък бюджет (нямаме възможност да разработим свои компоненти а се опитваме да пригодим чужди такива за нашите нужди). Прилагането на АПОК спестява и доста голямото време отделено за разработването на въпросните компоненти (наместо това използваме ПОК модули които за нас са разработени в някаква степен).

Логичният въпрос е, кога не е подходящо използването на АПОК?

Когато имаме проект с голям бюджет и първия приоритет ни е качеството на продукта. Използвайки ПОК, ние след сивата фаза можем да гарантираме определено ниво на качество, можем да го повишаваме участвайки в разработката на въпросния ПОК, но главния проблем е че заварваме дадено „ниво на качеството" на този ПОК.

Именно с това ограничение се бори АПОК – „качеството". Една от основните неявни задачи на сивата фаза на АПОК е отсяването на качествените ПОК и констатацията на качеството на въпросния ПОК.

6.2. Класификация на наличните ПОК

Освен по бизнес изисквания и архитектурни съображения е необходимо да класифицираме ПОК подходящи за нашия проект и по така наречените „вторични критерии за избор на ПОК" като развитие, зрялост и популярност. Ще разгледаме популярността отделно, а зрялостта и нивото на развитие на един ПОК ще обобщим в характеристиката „перспективност".

6.2.1. Популярност на един ПОК

Популярността характеристика, която не е свързана с обхвата на проекта, а със заинтересоваността към него и от участниците в този ПОК, и от потребителите му (и от тези които го ползват за бизнес цели, като нас).

- Заинтересованост от страна на участниците в ПОК - Често в зависимост от популярността на предметната област на проекта (описано по долу като „благоприятна постановка") се влияе активността на участниците в ПОК обществото (за този проект). От друга страна по голямата активност при един ПОК спрямо сходни такива се обуславя с по-доброто качество на въпросния проект. Често избора на ПОК се влияе от активността на участниците в него.

- Заинтересованост от страна на потребителите - По широката популярност на даден ПОК сред дадена потребителска среда също показва качество в дадения ПОК. По дългото използване от потребителите на този ПОК е довело до отстраняване на повече грешки и неточности в него, тази класификация до голяма степен е проявление на описания като вторичен критерии за избор на ПОК – „зрялост на ПОК".

- Заинтересованост от бизнес съображения – по-рядко явление, но и по-показателно за нивото на качеството на дадения ПОК. Наличието на такъв тип заинтересованост предполага че някой вече е използвал този ПОК за сходни с нашите цели и на базата на неговия опит можем до голяма степен да си облекчим работата в сивата фаза.

6.2.2. Перспективност на един ПОК

Един ПОК може да се намира в няколко фази на развитие. Може да е:

- алфа версия – Реализирана идея. Неизтествана версия, трудно използваема без прилагане на сериозни модификации.
- бета версия – Версия която е в процес на тестване. Функционалността и се доизгражда.
- официална версия – Изградена версия на проекта. Разполага с „пълната" (според създателите на проекта) функционалност необходима на използващите продукта лица. Функционалността на такава версия на продукта по-рядко търпи промени.

Различните типове версиите номерират с различен порядък. Алфа версията се номерира като 0.0N, бета версията като 0.N, а официалните версии като N.

Официалната версия на един ПОК много трудно може да се направлява към даден път на развитие от участници като нас, а по алфа версиите е необходима много работа до изграждането на необходимия ни ПОК, компромисния вариант се явяват бета версиите.
Бета версията е доста удобна за използване тъй като не предполага толкова много работа по модифицирането на ПОК , както при алфа версиите, и толкова ниска приемственост на промените както при официалните версии.

Използвания ПОК, XBT Client, бе заварен като бета версия 0.53 и бе разработван до бета версия 0.79.

Друг елемент определящ перспективността на даден ПОК е състоянието в което се намира:

- Дали е възможно да бъде финансиран от външни институции (например фонд на Европейския съюз)
- До каква степен е възможно в бъдеще да бъдат откупени правата на собственост от участниците му от софтуерна компания.

- Дали развитието на даден ПОК не е подпомагано от особености свързани с него, като „осиротяването" на въпросния проект от фалирала софтуерна компания. Софтуерната компания не може да поддържа повече въпросния продукт, отваря програмния код на и завещава проекта на ПОК обществото с идея за по нататъшно развитие на проекта. Пример за такъв тип проект е Interbase проекта, който се трансформира във Firebird ПОК.

Този критерии е извън областта на изследванията на дипломанта при изграждането на този проект, но не и извън областта на изследвания на АПОК.

6.3. Алтернативни начини на прилагането на АПОК

Вече изяснихме че използването на ПОК е подходящо при разработката на middleware или при разпределени архитектури като клиент-сървър. Сивата фаза на АПОК предполага комуникация чрез прототипизация на бъдещия продукт и задълбочено изследване на външни проекти. Но възможно ли е този външен проект вместо да е ПОК да е:

- друг вътрешен проект
- външен проект разработван от друга компания

Ако нашият продукт е разработван от екип (част от който сме и ние) на дадена софтуерна компания то е доста вероятно да използваме като външни модули за нашия проект проекти разработвани от други екипи. Или проекта на другия екип се оказва външен проект за нашия екип и вътрешен за нашата компания. След като заместим ПОК със този „друг вътрешен проект" можем спокойно да прилагаме АПОК за разработката на нашия проект, съобразявайки се вместо с традициите и неписаните правила на ПОК обществото, с описания начин на работа на колегите от другия екип. Много е вероятно нашия мениджър да поеме и ролята на Логист, а хора от другия екип да бъдат заети като Инженери по знание в нашия екип. Условните „ПОК разработчици' и лидера на техния екип респективно ще си останат същите хора. Добра практика би била размяната на един разработчик между двата екипа, или внедряването на такива в другия екип.

При втория вариант (външен проект разработван от друга компания с пълна поддръжка от тяхна страна) ролите в под-процесите Адаптация и Логистика са доста по-ясно дисоциирани. Силната заинтересованост за задоволяването на нуждите на нашия проект от страна на другата компания до голяма степен обезсмисля създаването на наша вътрешна структура работеща по ПОК под-процеса. Тъй като нашата компания се явява клиент на другата компания то нашия Логист би се оказал в ролята на Клиент за тях. И в този случай заемаме външен за нашия екип Инженер по знание, в случая от лице от другата компания.

И в двата алтернативни начина на прилагане на АПОК имаме един допълнителен положителен фактор - гаранция за качеството на външния проект. Първата стъпка в сивата фаза – „Изследването" се свежда до изследването на един на точно определен проект. Заместването на ПОК обществото с изградените структури на другия проект е приемлива условност за прилагането на АПОК.

Съществува и трети вариант при който имаме най-голямо доближаване до описания в темата начин на прилагане на АПОК. Когато външна компания разработила даден проект ни даде пълни права за комерсиалното му използване. Или да имаме право да модифицираме програмния код на проекта и да може да се обръщаме и към компанията собственик на модула за необходимост от модификации, срещу заплащане. В този случай имаме ясно следване на софтуерния процес на АПОК без условности. Разликата е в нивото на комуникацията – имаме финансови взаимодействия със външно общество вместо идеологически.

6.4. Подходящи среди за прилагането на АПОК

6.4.1. Сравнение с гъвкавата методологията Google

В тази дипломна работа бе разгледано прилагането на АПОК от малка компания. Ще изложим изводите на Steve Yegee от прилагането на подобна гъвкава методология в една голяма компания – Google – една от най-големите компании за разработка на софтуер. Ще сравним приликите във използваните методологии за да извадим общите благоприятни предпоставки за използването на АПОК.

Управляван от краен срок?

Най-голямо впечатление на Steve Yegee прави, че крайните срокове за разработката на дадени вътрешни проекти в Google са нестрого дефинирани (често се говори че дадена функционалност сигурно ще бъде готова след три седмици, проекта трябва да е готов до първото тримесечие на следващата година ...). Софтуерната индустрия принципно не търпи разработване на проекти, които не са пришпорвани от крайни срокове. Според Yegee единствено в университетите, ПОК и Google имаме разработка на проекти неосакатявана от крайни срокове.

При АПОК има план на проекта но той е нестрог, условните крайни срокове в него непрекъснато се обновяват от почти всички участници, и ни служи за насока за правилното следване на мисията на проекта. От друга гледна точка нашия продукт може да се разглежда и като придатък на ПОК. Следователно по този показател двете методологии доста си приличат.

Миграция на индивиди

В Google разработчиците могат да се местят в различни екипи когато си пожелаят, без има негативни последствия.

АПОК даже поощрява миграцията на индивиди между различните роли. Основна предпоставка за внедряването на разработчици в един ПОК екип.

Странични занимания

В Google разработчиците са поощрявани да работят в 20% от работното си време над каквото си пожелаят, без това да е свързано с техния главен проект.

Проектите разработвани по АПОК предполагат внедряването на много външни идеи, при разработката на разглеждания проект емпирично е установено че страничните занимания често допринасят индиректно за проекта. (Освен че помагат за по-добрата атмосфера на работа).

Нормално ниво на срещите

В Google повечето разработчици имат по-малко от 3 „срещи” на седмица с колегите и ръководството.

АПОК е базирана на спринтове които също предполагат твърдо но ниско ниво на срещите – често един път на седмица.

Тиха работна среда

В Google всеки разработчик е фокусиран в работата си и работната среда е тиха.

При АПОК всички използвани системи са достъпни online, цялата комуникация е налична в тях. АПОК се грижи за това лице в лице комуникацията между колеги е необходима единствено на дискусиите („срещите”).

Ненатоварване на разработчиците

Дори и в тежки за проекта времена разработчиците не работят извънредно, освен ако не пожелаят, не са притеснени, обядват редовно.

Проектите разработвани по АПОК предполагат компромис с едно ограничение вече – качеството. Ясно е че, не може да направим добър продукт при наличие на няколко ограничения. Затова и разработчиците работещи по АПОК не са пресирани по отношение на времето. Когато дадена работа се отклонява силно от очакванията софтуерните инженери се грижат да установят проблема и да помогнат на разработчика.

И посочените интересни според Yegee практики има две по които работата в Google и тази по АПОК се различават:

Кодиращи мениджъри

Според Yegee в Google има „някакъв вид мениджъри”, но те програмират през половината от времето и са по скоро ментори.

При децентрализирания начин на управление на проекта при АПОК подобна роля имат по-скоро лидера на ПОК екип и Архитекта на проекта.

Невидими системи за следене на времето

В Google разработчиците не са притеснявани от време-следящи системи и графики за развитието на проекта.

По тази практика силно се различава начина на работа с този при използването на АПОК, където разработчика не е просто ресурс, търсещ сам най-доброто си приложение, а участник в менажирането на проекта. Всеки един разработчик не само сам моделира и кодира функционалността която си е избрал да разработва, но и има властта да промени крайния срок съгласувано с мениджъра му.

Или единствената съществена разлика между двата процеса е в начина на управление на плана на проекта, в случая на Google е централизиран, а при прилагането на АПОК – децентрализиран.

6.4.2. Благоприятни постановки за прилагането на АПОК

Тези „постановки" са външни за софтуерния процес фактори.
Ще опишем три благоприятни постановки, налични за нашия проект. Първите две са извод от направеното по-горе сравнение с прилагането на подобна гъвкава методология в различен мащаб. Едната зависеща от политиката на софтуерната ни компания другата от типа на клиента. След като обосновахме сходството между гъвкавата методология на Google и АПОК, ще опишем общите благоприятни постановки за прилагане на АПОК:

- Динамична атмосфера – На всички участници в проекта се дава доста голяма свобода.(Както Steve Yegee бе определил работната среда в Google като „джунгла", така и ние смеем да твърдим че при АПОК всеки разработчик сам определя какво да прави и как да прави). Свободата да се самоуправляват, да взимат решения и да поемат отговорност. За разлика от другите методологии където индивидуалната свобода трябва да се заслужи при тези гъвкави методологии участниците разполагат с нея, и се стараят да оправдаят гласуваното им доверие.
- Вътрешен клиент – както Google така и фирмата поръчител на проекта разработват вътрешни проекти, разликата е в мащабите. Наличието на вътрешен клиент е най-добрия вариант, но при наличието на един отдаден на проекта клиент резултатите от прилагането на АПОК не са по-лоши.

Друга благоприятна постановка, може би най-съществената, е:

- популярността на предметната област на проекта. Това би привлякло интереса на ПОК обществото. И съответно ще разполагаме с по-богат избор от ПОК които да изследваме. Ограничението „качество" на избрания ПОК ще представлява по-малък проблем (по-видни специалисти ще разработват интересни проекти като избрания от нас ПОК).

Тези благоприятни постановки са опитно изведени след успешното разработване на даден проект по АПОК. Колкото повече се прилага АПОК толкова повече благоприятни постановки ще бъдат извеждани.

6.4.3. Развиване на ПОК обществото чрез прилагането на АПОК

Не на последно място използването на АПОК може да се предпочете от някои участници в проекта като разработчиците участвали в ПОК по идеологични съображения. АПОК се стреми винаги когато е възможно да предпочете ПОК решението на всеки един проблем, което развива специфични ПОК и участниците в тях. От друга страна АПОК допринася за подобряването на традициите и неписаните правила на ПОК като цяло. Използването на АПОК ни насочва към:

Разработване на комерсиален продукт използвайки ПОК, по АПОК методологията която системно използва други ПОК.

7. Заключение

В тази дипломна работа предложихме гъвкавата методология АПОК. Разгледахме примерния проект „SuperSeeder", разработван по АПОК методологията от екип софтуерни разработчици от който е част и дипломанта. Детайлно бе изложена ролята на дипломанта в проекта по модифициране и интегриране на проекта с отворен код "XBT Client".

За изпълнението на първата задача „*Сравнителен анализ със съществуващите гъвкави методологии*" избрахме по-популярните гъвкави методологии и тези от които сме взаимствали за направата на АПОК. Изводите от изпълнението на първата задача биха могли за да се използват като базово знание за дефинирането и на други гъвкави методологии. На дипломанта тези изводи му послужиха при изпълняването на втората задача „*Предложение на нова гъвкава методология за проекти с отворен код".*

При изпълнението на втората задача дипломанта дефинира и обособи новопредложената гъвкава методология като такава и я класифицира в света на гъвкавите методологии. Позовавайки се на вече съществуващите гъвкави методологии дипломанта успя да определи новопредложената методология като гъвкава. Именуваме я Адаптация на проекти с отворен код.

При изпълнението на третата задача „*Избор на проект (изследване) с отворен код и приложение на софтуерния процес*" дипломанта избра проекта с отворен код XBT Client, постъпково следва софтуерния процес дефиниран от АПОК (използва предложените практики и с оглед на наложените ограничения разпределі дефинираните роли между участниците в проекта). Главната предпоставка за изпълнението на задачата бе доброто познаване на предментната област на използвания ПОК – битторент протокола който бе представен в началото на дипломната работа. Друга подзадача която дипломанта изпълни в голяма степен бе успешно да се интегрира в ПОК обществото от the SourceForge community. При изпълнението на тази задача дипломанта представи един успешен модел за прилагане на АПОК. Всяко бъдещо използване на АПОК би могло да подобри предложения модел на прилагане или да предложи алтернативен такъв.

Доброто познаване на предметната област на XBT Client и съдействието на ПОК обществото бяха и главните предпоставки за реализирането на четвъртата (програмната) задача „*Описание на добавената функционалност*". Път за бъдещо развитие на XBT Client бе предложен в подточката Бъдещо развитие на XBT ПОК.

Като заключение можем да посочим анализа на получените резултати, и определянето на мястото на АПОК в света на гъвкавите методологии и предложенията за алтернативно приложение и перспективите и за развитие, които стоят пред тази млада гъвкава методология. Най-добрия начин за подобряване на АПОК методологията е нейното използване.

8. Литература

[1] Публикация в интернет: Софтуерни методологии който могат да се охарактеризират като гъвкави:
Agile software development methods

[2] Ресурси на VersionOne Agile Project Management Tool: Кратък преглед на по популярните гъвкави методологии:
http://www.versionone.net/Resources/AgileMethodologies.asp
http://www.agilejournal.com/component/option,com_magazine/func,show_article/id,74/

[3] Статия в Интернет: Въведение в гъвкавите методологии:
http://en.wikipedia.org/wiki/Agile_software_development

[4] Статия в Интернет: Битторент технологията
BitTorrent - Wikipedia, the free encyclopedia
BitTorrent Protocol Specification

[5] Публикация в Интернет: Дали есенцията на разработването на проекти с отворен код не е гъвкава методология:
http://agile.vtt.fi/docs/publications/2003/2003_oss.pdf

[6] Статия в Интернет: Избирането на правилната методология за разработването на даден проект:
http://www.agiledata.org/essays/differentStrategies.html

[7] Ресурсите свързани с интересуващия ни проект:
http://xbtt.sourceforge.net/

[8] Статия в интернет: Добрия и лошия начин за работа с гъвкави методологии, Steeve Yegge представя софтуерния процес в Google:
http://steve-yegge.blogspot.com/2006/09/good-agile-bad-agile_27.html

[9] Статия в интернет: Как проект с отворен код става финансиран от правителствена организация:
http://codespeak.net/pypy/extradoc/talk/22c3/agility.pdf

[10] Статия в интернет: Какво представлява Scrum:
What is SCRUM? - The Code Project - Application Design

[11] Статия в интернет: Какво представлява XP:
Extreme Programming

[12] Статия в интернет: Какво представлява FDD:
http://www.featuredrivendevelopment.com/

[13] Статия в интернет: Какво представлява DSDM:
http://www.dsdm.org/

[14] Статия в интернет: Какво представлява ASD:
http://www.adaptivesd.com/

[15] Статия в интернет: Какво представлява Crystal:
http://alistair.cockburn.us/crystal

[16] Книгата на Barry Boehm и Richard Turner „Balancing Agility and Discipline”
Balancing Agility and Discipline

[17] Книгата на Song & Osterweil 1991 “Comparing Design Methodologies through Process Modeling”
Comparing Design Methodologies through Process Modeling

[18] Книга на Miller “The Characteristics of Agile Software Processess”
The Characteristics of Agile Software Processes

[19] Статия на HighSmith и Cockburn
Light Methods become Agile Methods

[20] Статия на Scott W. Ambler
Agile Model Driven Development

[21] Публикация на Mockus 2000
1 Open Source Software Development, Innovation, and Coordination ...

[22] Книга на B.Feller & J.Fitdgeranld 2000
A FRAMEWORK ANALYSIS OF THE OPEN SOURCE DEVELOPMENT PARADIGM

9. Приложение 1

Целия добавен програмен код е даден под формата на пач.

```
diff -u1 -r diffold/BT Test/bt_admin_link.cpp diffnew/BT Test/bt_admin_link.cpp
--- diffold/BT Test/bt_admin_link.cpp      2006-02-10 16:43:55.000000000 -0100
+++ diffnew/BT Test/bt_admin_link.cpp 2006-02-10 16:34:29.000000000 -0100
@@ -61,4 +61,12 @@
                        return 1;
-               if (0 && m_server->time() - m_ctime > 60)
+               if (0 && m_server->time() - m_ctime > 1000)
+               {
+#ifdef EVENT
+                       alert(Calert::debug, "Admin: Timeout");
+#endif
+#ifdef EVENTLOG
+                       event_logger().event(Cbt_logger::debug, "Admin: Timeout");
+#endif
                        return 1;
+               }
                return m_close;
@@ -74,4 +82,18 @@
                                if (e == WSAEWOULDBLOCK)
+                               {
+#ifdef EVENT
+                                       alert(Calert::debug, "Admin: recv WSAEWOULDBLOCK?" +
Csocket::error2a(e));
+#endif
+#ifdef EVENTLOG
+                                       event_logger().event(Cbt_logger::debug, "Admin: recv
WSAEWOULDBLOCK?" + Csocket::error2a(e));
+#endif
                                        return 0;
+                               }
+#ifdef EVENT
                                alert(Calert::debug, "Admin: recv failed: " + Csocket::error2a(e));
+#endif
+#ifdef EVENTLOG
+                               event_logger().event(Cbt_logger::debug, "Admin: recv failed: " +
Csocket::error2a(e));
+#endif
+
                                return 1;
@@ -93,4 +115,17 @@
                                if (e == WSAEWOULDBLOCK)
+                               {
+#ifdef EVENT
+                                       alert(Calert::debug, "Admin: send WSAEWOULDBLOCK?" +
Csocket::error2a(e));
+#endif
+#ifdef EVENTLOG
+                                       event_logger().event(Cbt_logger::debug, "Admin: send
WSAEWOULDBLOCK?" + Csocket::error2a(e));
+#endif
                                        return 0;
+                               }
+#ifdef EVENT
```

```
                                alert(Calert::debug, "Admin: send failed: " + Csocket::error2a(e));
+#endif
+#ifdef EVENTLOG
+                               event_logger().event(Cbt_logger::debug, "Admin: send failed: " +
Csocket::error2a(e));
+#endif
                                return 1;
@@ -140 +175,8 @@
 }
+
+Cbt_logger& Cbt_admin_link::event_logger()
+{
+        return m_server->event_logger();
+}
+
+
diff -u1 -r diffold/BT Test/bt_admin_link.h diffnew/BT Test/bt_admin_link.h
--- diffold/BT Test/bt_admin_link.h         2006-02-10 16:43:55.000000000 -0100
+++ diffnew/BT Test/bt_admin_link.h     2006-02-10 16:34:29.000000000 -0100
@@ -10,3 +10,3 @@
 #include "socket.h"
-
+#include "bt_logger.h"
 class Cserver;
@@ -16,2 +16,3 @@
 public:
+        Cbt_logger& event_logger();
         void alert(Calert::t_level, const string&);
diff -u1 -r diffold/BT Test/bt_file.cpp diffnew/BT Test/bt_file.cpp
--- diffold/BT Test/bt_file.cpp     2006-02-10 16:43:55.000000000 -0100
+++ diffnew/BT Test/bt_file.cpp 2006-02-10 16:34:29.000000000 -0100
@@ -33,2 +33,3 @@
         m_validate = true;
+        m_alter_mode_request = true;
 }
@@ -133,5 +134,5 @@

-void Cbt_file::t_sub_file::erase(const string& parent_name)
+int Cbt_file::t_sub_file::erase(const string& parent_name)
 {
-        ::unlink((parent_name + m_name).c_str());
+        return ::unlink((parent_name + m_name).c_str());
 }
@@ -261,4 +262,6 @@

-void Cbt_file::erase()
+string Cbt_file::erase()
 {
+        string undeleted;
+        int i_unlink;
 #ifdef WIN32
@@ -275,5 +278,14 @@
         for (t_sub_files::iterator i = m_sub_files.begin(); i != m_sub_files.end(); i++)
-                i->erase(m_name);
+        {       //Cbt_file::t_sub_file* p_i = &*i;
+                //if(p_i)
+                i_unlink = 0;
+                if(i_unlink = i->erase(m_name))
+                        if(i_unlink != -1)
+                                undeleted += i->m_name + ' ' + n(i_unlink) + '\t';
```

```
+		}
		if (m_sub_files.size() != 1)
-			unlink(m_name.c_str());
+			if(i_unlink = rmpath(m_name))
+				undeleted += m_name + ' ' + n(i_unlink);
+		if(undeleted == "") return "";
+		return undeleted;
 #endif
@@ -288,3 +300,8 @@
			m_seeding_ratio_reached_at = time();
+#ifdef EVENT
			alert(Calert(Calert::notice, "Seeding ratio reached"));
+#endif
+#ifdef EVENTLOG
+			event_logger().event(Cbt_logger::notice, "Seeding ratio reached");
+#endif
			close();
@@ -368,2 +385,3 @@
		peer.m_local_link = false;
+		peer.m_alter_peer_request = true;
		peer.m_state = 4;
@@ -424,4 +442,12 @@
				else
+				{
+#ifdef EVENT
+					alert(Calert(Calert::error, "File " + native_slashes(m_name + i-
>name()) + ": open failed"));
-			}
+#endif
+#ifdef EVENTLOG
+					event_logger().event(Cbt_logger::error, "File " +
native_slashes(m_name + i->name()) + ": open failed");
+#endif
+				}
+
+			}
			int cb_write = min(size, i->offset() + i->size() - offset);
@@ -429,3 +455,8 @@
			{
+#ifdef EVENT
+				alert(Calert(Calert::error, "Piece " + n(a) + ": write failed"));
+#endif
+#ifdef EVENTLOG
+				event_logger().event(Cbt_logger::error, "Piece " + n(a) + ": write failed");
+#endif
				m_state = s_paused;
@@ -446,3 +477,8 @@
			mc_rejected_pieces++;
+#ifdef EVENT
			alert(Calert(Calert::warn, "Piece " + n(a) + ": invalid"));
+#endif
+#ifdef EVENTLOG
+			event_logger().event(Cbt_logger::warn, "Piece " + n(a) + ": invalid");
+#endif
			logger().invalid(m_info_hash, false, a);
@@ -453,2 +489,3 @@
		{
+			m_alter_mode_request = true;
			m_completed_at = time();
@@ -467,3 +504,3 @@
```

```
                                        i->close();
-                                       i->open(m_name, O_RDONLY);
+                                       i->open(m_name, O_RDWR);
                                }
@@ -486,6 +523,13 @@
                        for (t_sub_files::iterator i = m_sub_files.begin(); i != m_sub_files.end(); i++)
-                               i->open(m_name, O_RDONLY);
+                               i->open(m_name, O_RDWR);
                }
                if (server()->log_piece_valid())
+               {
+#ifdef EVENT
                        alert(Calert(Calert::debug, "Piece " + n(a) + ": valid"));
+#endif
+#ifdef EVENTLOG
+                       event_logger().event(Cbt_logger::debug, "Piece " + n(a) + ": valid");
+#endif
+               }
                logger().valid(m_info_hash, false, a);
@@ -923,2 +967,7 @@

+Cbt_logger& Cbt_file::event_logger()
+{
+       return server()->event_logger();
+}
+
 bool Cbt_file::begin_mode() const
diff -u1 -r diffold/BT Test/bt_file_data.cpp diffnew/BT Test/bt_file_data.cpp
--- diffold/BT Test/bt_file_data.cpp    2006-02-10 16:43:55.000000000 -0100
+++ diffnew/BT Test/bt_file_data.cpp    2006-02-10 16:34:29.000000000 -0100
@@ -5,3 +5,3 @@
 {
-       m_allow_end_mode = false;
+       m_allow_end_mode = true;
        m_end_mode = false;
diff -u1 -r diffold/BT Test/bt_file_data.h diffnew/BT Test/bt_file_data.h
--- diffold/BT Test/bt_file_data.h      2006-02-10 16:43:55.000000000 -0100
+++ diffnew/BT Test/bt_file_data.h      2006-02-10 16:34:29.000000000 -0100
@@ -50,2 +50,3 @@
        string m_name;
+       string m_peer_id;
        t_state m_state;
diff -u1 -r diffold/BT Test/bt_file.h diffnew/BT Test/bt_file.h
--- diffold/BT Test/bt_file.h   2006-02-10 16:43:55.000000000 -0100
+++ diffnew/BT Test/bt_file.h   2006-02-10 16:34:29.000000000 -0100
@@ -44,2 +44,3 @@
        Cbt_logger& logger();
+       Cbt_logger& event_logger();
        string get_hashes(__int64 offset, int c) const;
@@ -66,3 +67,3 @@
        void close();
-       void erase();
+       string erase();
        void open();
@@ -113,3 +114,3 @@
                        void dump(Cstream_writer&) const;
-                       void erase(const string& parent_name);
+                       int erase(const string& parent_name);
                        bool open(const string& parent_name, int oflag);
@@ -209,3 +210,3 @@
```

```
        Cvirtual_binary m_info;
-
+
        Cdata_counter m_down_counter;
@@ -218,2 +219,3 @@
        Cserver* m_server;
+       bool m_alter_mode_request;
 };
diff -u1 -r diffold/BT Test/bt_peer_link.cpp diffnew/BT Test/bt_peer_link.cpp
--- diffold/BT Test/bt_peer_link.cpp        2006-02-10 16:43:55.000000000 -0100
+++ diffnew/BT Test/bt_peer_link.cpp    2006-02-10 16:34:29.000000000 -0100
@@ -16,2 +16,3 @@
        m_state = 1;
+       m_alter_peer_request = true;
 }
@@ -36,3 +37,8 @@
                {
+#ifdef EVENT
                        alert(Calert::debug, "Peer: connect failed: " +
Csocket::error2a(WSAGetLastError()));
+#endif
+#ifdef EVENTLOG
+                       event_logger().event(Cbt_logger::debug, "Peer: connect failed: " +
Csocket::error2a(WSAGetLastError()));
+#endif
                        close();
@@ -71,3 +77,8 @@
                                {
+#ifdef EVENT
                                        alert(Calert::debug, "Peer: connect failed: " +
Csocket::error2a(WSAGetLastError()));
+#endif
+#ifdef EVENTLOG
+                                       event_logger().event(Cbt_logger::debug, "Peer: connect
failed: " + Csocket::error2a(WSAGetLastError()));
+#endif
                                        return 1;
@@ -77,3 +88,10 @@
                        if (server()->log_peer_connect_failures())
+                       {
+#ifdef EVENT
                                alert(Calert::debug, "Peer: connect failed: " + Csocket::error2a(e));
+#endif
+#ifdef EVENTLOG
+                               event_logger().event(Cbt_logger::debug, "Peer: connect failed: " +
Csocket::error2a(WSAGetLastError()));
+#endif
+                       }
                        return 1;
@@ -188,3 +206,11 @@
                if (server()->log_peer_connection_closures())
+               {
+#ifdef EVENT
                        alert(Calert::debug, "Peer: seeder to seeder link closed");
+#endif
+#ifdef EVENTLOG
+                       event_logger().event(Cbt_logger::debug, "Peer: seeder to seeder link closed");
+#endif
+
+               }
```

```
                return 1;
@@ -245,3 +271,10 @@
                        if (server()->log_peer_recv_failures())
+                       {
+#ifdef EVENT
                                alert(Calert::debug, "Peer: recv failed: " + Csocket::error2a(e));
+#endif
+#ifdef EVENTLOG
+                               event_logger().event(Cbt_logger::debug, "Peer: recv failed: " +
Csocket::error2a(e));
+#endif
+                       }
                        return 1;
@@ -259,3 +292,10 @@
        if (server()->log_peer_connection_closures())
+       {
+#ifdef EVENT
                alert(Calert::debug, m_local_link ? "Peer: local link closed" : "Peer: remote link
closed");
+#endif
+#ifdef EVENTLOG
+               event_logger().event(Cbt_logger::debug, m_local_link ? "Peer: local link closed" :
"Peer: remote link closed");
+#endif
+       }
        return 1;
@@ -297,3 +337,10 @@
                        if (server()->log_peer_send_failures())
+                       {
+#ifdef EVENT
                                alert(Calert::debug, "Peer: send failed: " + Csocket::error2a(e));
+#endif
+#ifdef EVENTLOG
+                               event_logger().event(Cbt_logger::debug, "Peer: send failed: " +
Csocket::error2a(e));
+#endif
+                       }
                        return 1;
@@ -388,3 +435,8 @@
        {
+#ifdef EVENT
                alert(Calert::warn, "Peer: handshake failed");
+#endif
+#ifdef EVENTLOG
+               event_logger().event(Cbt_logger::warn, "Peer: handshake failed");
+#endif
                return 1;
@@ -653,3 +705,8 @@
        {
+#ifdef EVENT
                alert(Calert::warn, "No matching request found, piece: " + n(piece) + ", offset: " +
n(offset) + ", size: " + b2a(size, "b") + " (" + peer_id2a(m_remote_peer_id) + ")");
+#endif
+#ifdef EVENTLOG
+               event_logger().event(Cbt_logger::warn, "No matching request found, piece: " +
n(piece) + ", offset: " + n(offset) + ", size: " + b2a(size, "b") + " (" + peer_id2a(m_remote_peer_id) +
")");
+#endif
                return 1;
@@ -674,3 +731,8 @@
```

```
		{
+#ifdef EVENT
			alert(Calert::warn, "Chunk " + n(offset >> 15) + ": invalid");
+#endif
+#ifdef EVENTLOG
+			event_logger().event(Cbt_logger::warn, "Chunk " + n(offset >> 15) + ": invalid");
+#endif
			return;
@@ -693,7 +755,28 @@
		if (o % piece.cb_sub_piece())
+		{
+#ifdef EVENT
			alert(Calert::debug, "Piece " + n(a) + ", offset " + n(o % m_f->mcb_piece) + ", size: " +
b2a(cb_s) + ": invalid offset (" + peer_id2a(m_remote_peer_id) + ")");
+#endif
+#ifdef EVENTLOG
+			event_logger().event(Cbt_logger::debug, "Piece " + n(a) + ", offset " + n(o % m_f-
>mcb_piece) + ", size: " + b2a(cb_s) + ": invalid offset (" + peer_id2a(m_remote_peer_id) + ")");
+#endif
+		}
		else if (cb_s != piece.cb_sub_piece(b))
+		{
+#ifdef EVENT
			alert(Calert::debug, "Piece " + n(a) + ", chunk " + n(b) + ", size: " + b2a(cb_s) + ":
invalid size (" + peer_id2a(m_remote_peer_id) + ")");
+#endif
+#ifdef EVENTLOG
+			 event_logger().event(Cbt_logger::debug, "Piece " + n(a) + ", chunk " + n(b) + ", size: "
+ b2a(cb_s) + ": invalid size (" + peer_id2a(m_remote_peer_id) + ")");
+#endif
+		}
		else
+		{
+#ifdef EVENT
			alert(Calert::debug, "Piece " + n(a) + ", chunk " + n(b) + ", latency: " + n(latency) + " s:
rejected (" + peer_id2a(m_remote_peer_id) + ")");
+#endif
+#ifdef EVENTLOG
+			event_logger().event(Cbt_logger::debug, "Piece " + n(a) + ", chunk " + n(b) + ",
latency: " + n(latency) + " s: rejected (" + peer_id2a(m_remote_peer_id) + ")");
+#endif
+		}
		return 1;
@@ -787,3 +870,8 @@
		case bti_get_peers:
+#ifdef EVENT
			alert(Calert::debug, "Peer: get_peers");
+#endif
+#ifdef EVENTLOG
+			event_logger().event(Cbt_logger::debug, "Peer: get_peers");
+#endif
			if (r_end - r >= 2 && time() - m_peers_stime > 300)
@@ -792,3 +880,8 @@
		case bti_peers:
+#ifdef EVENT
			alert(Calert::debug, "Peer: " + n((r_end - r - 2) / 6) + " peers");
+#endif
+#ifdef EVENTLOG
+			event_logger().event(Cbt_logger::debug, "Peer: " + n((r_end - r - 2) / 6) + " peers");
+#endif
```

```
                        if (r_end - r >= 2 && time() - m_get_peers_stime < 60)
@@ -890 +983,6 @@
 }
+
+Cbt_logger& Cbt_peer_link::event_logger()
+{
+       return server()->event_logger();
+}
diff -u1 -r diffold/BT Test/bt_peer_link.h diffnew/BT Test/bt_peer_link.h
--- diffold/BT Test/bt_peer_link.h          2006-02-10 16:43:55.000000000 -0100
+++ diffnew/BT Test/bt_peer_link.h      2006-02-10 16:34:29.000000000 -0100
@@ -31,2 +31,3 @@
        Cbt_logger& logger();
+       Cbt_logger& event_logger();
        void clear_local_requests();
@@ -149,2 +150,3 @@
        bool m_can_send;
+       bool m_alter_peer_request;
 private:
diff -u1 -r diffold/BT Test/bt_tracker_link.cpp diffnew/BT Test/bt_tracker_link.cpp
--- diffold/BT Test/bt_tracker_link.cpp     2006-02-10 16:43:55.000000000 -0100
+++ diffnew/BT Test/bt_tracker_link.cpp2006-02-10 16:34:29.000000000 -0100
@@ -46,3 +46,8 @@
                        case Cbt_tracker_url::tp_http:
+#ifdef EVENT
                                f.alert(Calert(Calert::info, "Tracker: URL: http://" + m_url.m_host + ':' +
n(m_url.m_port) + m_url.m_path));
+#endif
+#ifdef EVENTLOG
+                               event_logger(f).event(Cbt_logger::info, "Tracker: URL: http://" + m_url.m_host
+ ':' + n(m_url.m_port) + m_url.m_path);
+#endif
                                m_announce_time = f.m_server->time() + (300 << mc_attempts++);
@@ -52,3 +57,8 @@
                        case Cbt_tracker_url::tp_udp:
+#ifdef EVENT
                                f.alert(Calert(Calert::info, "Tracker: URL: udp://" + m_url.m_host + ':' +
n(m_url.m_port)));
+#endif
+#ifdef EVENTLOG
+                               event_logger(f).event(Cbt_logger::info, "Tracker: URL: udp://" + m_url.m_host
+ ':' + n(m_url.m_port));
+#endif
                                m_announce_time = f.m_server->time() + (60 << mc_attempts++);
@@ -64,3 +74,8 @@
                                {
+#ifdef EVENT
                                        f.alert(Calert(Calert::error, "Tracker: gethostbyname failed"));
+#endif
+#ifdef EVENTLOG
+                                       event_logger(f).event(Cbt_logger::error, "Tracker: gethostbyname
failed");
+#endif
                                        close(f);
@@ -70,3 +85,8 @@
                                        return 0;
+#ifdef EVENT
                                f.alert(Calert(Calert::info, "Tracker: IPA: " + Csocket::inet_ntoa(h)));
+#endif
+#ifdef EVENTLOG
```

```
+                       event_logger(f).event(Cbt_logger::info, "Tracker: IPA: " +
Csocket::inet_ntoa(h));
+#endif
                }
@@ -87,3 +107,8 @@
                        }
+#ifdef EVENT
                        f.alert(Calert(Calert::debug, "Tracker: UDP: connect send"));
+#endif
+#ifdef EVENTLOG
+                       event_logger(f).event(Cbt_logger::debug, "Tracker: UDP: connect send");
+#endif
                        m_connect_send = f.m_server->time();
@@ -125,3 +150,8 @@
                        m_s.getsockopt(SOL_SOCKET, SO_ERROR, e);
+#ifdef EVENT
                        f.alert(Calert(Calert::error, "Tracker: HTTP: connect failed: " +
Csocket::error2a(e)));
+#endif
+#ifdef EVENTLOG
+                       event_logger(f).event(Cbt_logger::error, "Tracker: HTTP: connect failed: " +
Csocket::error2a(e));
+#endif
                        close(f);
@@ -139,3 +169,8 @@
                                        {
+#ifdef EVENT
                                                f.alert(Calert(Calert::warn, "Tracker: HTTP: recv failed:
" + Csocket::error2a(e)));
+#endif
+#ifdef EVENTLOG
+                                               event_logger(f).event(Cbt_logger::warn, "Tracker:
HTTP: recv failed: " + Csocket::error2a(e));
+#endif
                                                close(f);
@@ -194,3 +229,8 @@
                                }
+#ifdef EVENT
                                f.alert(Calert(Calert::debug, "Tracker: UDP: announce send"));
+#endif
+#ifdef EVENTLOG
+                               event_logger(f).event(Cbt_logger::debug, "Tracker: UDP: announce
send");
+#endif
                                m_announce_send = f.m_server->time();
@@ -219,3 +259,8 @@
                                        mc_attempts = 0;
+#ifdef EVENT
                                        f.alert(Calert(Calert::info, "Tracker: " + n((r - utoa_size) / 6) + "
peers (" + n(r) + " bytes)"));
+#endif
+#ifdef EVENTLOG
+                                       event_logger(f).event(Cbt_logger::info, "Tracker: " + n((r -
utoa_size) / 6) + " peers (" + n(r) + " bytes)");
+#endif
                                        for (int o = utoa_size; o + 6 <= r; o += 6)
@@ -227,3 +272,8 @@
                                {
+#ifdef EVENT
```

```
                                        f.alert(Calert(Calert::error, "Tracker: failure reason: " + string(d
+ utoe_size, r - utoe_size)));
+#endif
+#ifdef EVENTLOG
+                                       event_logger(f).event(Cbt_logger::error, "Tracker: failure
reason: " + string(d + utoe_size, r - utoe_size));
+#endif
                                        close(f);
@@ -247,3 +297,8 @@
                                {
+#ifdef EVENT
                                    f.alert(Calert(Calert::error, "Tracker: HTTP error: " + n(http_result)));
+#endif
+#ifdef EVENTLOG
+                                   event_logger(f).event(Cbt_logger::error, "Tracker: HTTP error: " +
n(http_result));
+#endif
                                    return 1;
@@ -258,3 +313,8 @@
                                        {
+#ifdef EVENT
                                            f.alert(Calert(Calert::error, "Tracker: bdecode failed"));
+#endif
+#ifdef EVENTLOG
+                                           event_logger(f).event(Cbt_logger::error, "Tracker:
bdecode failed");
+#endif
                                            return 1;
@@ -270,3 +330,8 @@
                                                const Cbvalue::t_list& peers =
v.d(bts_peers).l();
+#ifdef EVENT
                                                f.alert(Calert(Calert::info, "Tracker: " +
n(peers.size()) + " peers (" + n(d.size()) + " bytes)"));
+#endif
+#ifdef EVENTLOG
+                                               event_logger(f).event(Cbt_logger::info,
"Tracker: " + n(peers.size()) + " peers (" + n(d.size()) + " bytes)");
+#endif
                                                for (Cbvalue::t_list::const_iterator i =
peers.begin(); i != peers.end(); i++)
@@ -277,3 +342,8 @@
                                                string peers = v.d(bts_peers).s();
+#ifdef EVENT
                                                f.alert(Calert(Calert::info, "Tracker: " +
n(peers.size() / 6) + " peers (" + n(d.size()) + " bytes)"));
+#endif
+#ifdef EVENTLOG
+                                               event_logger(f).event(Cbt_logger::info,
"Tracker: " + n(peers.size() / 6) + " peers (" + n(d.size()) + " bytes)");
+#endif
                                                for (const char* r = peers.c_str(); r + 6 <=
peers.c_str() + peers.length(); r += 6)
@@ -283,3 +353,8 @@
                                        }
-                                       f.alert(Calert(Calert::error, "Tracker: failure reason: " +
v.d(bts_failure_reason).s()));
+#ifdef EVENT
+                                       f.alert(Calert(Calert::error, "Tracker: failure reason2: " +
v.d(bts_failure_reason).s()));
```

```
+#endif
+#ifdef EVENTLOG
+                                                event_logger(f).event(Cbt_logger::error, "Tracker: failure
reason2: " + v.d(bts_failure_reason).s());
+#endif
                                                 return 1;
@@ -290,3 +365,8 @@
         }
+#ifdef EVENT
         f.alert(Calert(Calert::error, "Tracker: Invalid HTTP output"));
+#endif
+#ifdef EVENTLOG
+        event_logger(f).event(Cbt_logger::error, "Tracker: Invalid HTTP output");
+#endif
         return 1;
@@ -367 +447,6 @@
 }
+
+Cbt_logger& Cbt_tracker_link::event_logger(Cbt_file& f)
+{
+        return f.m_server->event_logger();
+}
diff -u1 -r diffold/BT Test/bt_tracker_link.h diffnew/BT Test/bt_tracker_link.h
--- diffold/BT Test/bt_tracker_link.h       2006-02-10 16:43:55.000000000 -0100
+++ diffnew/BT Test/bt_tracker_link.h   2006-02-10 16:34:29.000000000 -0100
@@ -10,2 +10,3 @@
 #include "stream_writer.h"
+#include "bt_logger.h"

@@ -32,2 +33,3 @@
         void post_select(Cbt_file& f, fd_set* fd_read_set, fd_set* fd_write_set, fd_set* fd_except_set);
+        Cbt_logger& event_logger(Cbt_file& f);
         Cbt_tracker_link();
diff -u1 -r diffold/BT Test/config.cpp diffnew/BT Test/config.cpp
--- diffold/BT Test/config.cpp      2006-02-10 16:43:55.000000000 -0100
+++ diffnew/BT Test/config.cpp 2006-02-10 16:34:29.000000000 -0100
@@ -4,2 +4,4 @@
 #include "bvalue.h"
+#include "netfilter.h"
+#include "server.h"

@@ -21,3 +23,3 @@
         m_seeding_ratio = 0;
-        m_send_stop_event = false;
+        m_send_stop_event = true;
         m_torrent_limit = 0;
@@ -28,4 +30,6 @@
         m_upload_rate = 0;
-        m_upload_slots = 8;
+        m_upload_slots = 64;
         m_upnp = true;
+        m_net_filter_seeding = false;
+        m_net_filter_leeching = false;
 #ifdef WIN32
@@ -38,6 +42,6 @@
         {
-                m_completes_dir = home + "/XBT/Completes";
-                m_local_app_data_dir = home + "/XBT";
-                m_incompletes_dir = home + "/XBT/Incompletes";
-                m_torrents_dir = home + "/XBT/Torrents";
```

```
+                       m_completes_dir = "/store/Completes";
+                       m_local_app_data_dir = "/store";
+                       m_incompletes_dir = "/store/Incompletes";
+                       m_torrents_dir = "/store/Torrents";
        }
@@ -89,2 +93,6 @@
                        m_user_agent = v.d("user_agent").s();
+       if (v.d_has("net filter seeding"))
+                       m_net_filter_seeding = v.d("net filter seeding").i();
+      if (v.d_has("net filter leeching"))
+                       m_net_filter_leeching = v.d("net filter leeching").i();
        return *this;
@@ -116,2 +124,4 @@
        v.d("user_agent", m_user_agent);
+       v.d("net filter seeding", m_net_filter_seeding);
+       v.d("net filter leeching", m_net_filter_leeching);
        return v;
diff -u1 -r diffold/BT Test/config.h diffnew/BT Test/config.h
--- diffold/BT Test/config.h        2006-02-10 16:43:55.000000000 -0100
+++ diffnew/BT Test/config.h    2006-02-10 16:34:29.000000000 -0100
@@ -2,3 +2,3 @@
 #define AFX_CONFIG_H__CE8DA4C3_CDFC_46F3_A22E_ECCC9EAFD1DC__INCLUDED_
-
+#include "netfilter.h"
 #if _MSC_VER > 1000
@@ -12,2 +12,3 @@
 public:
+       typedef vector<Cbvalue> t_list;
        Cconfig();
@@ -24,2 +25,4 @@
        bool m_upnp;
+       bool m_net_filter_seeding;
+       bool m_net_filter_leeching;
        int m_admin_port;
Only in diffold/BT Test: CVS
diff -u1 -r diffold/BT Test/http_link.cpp diffnew/BT Test/http_link.cpp
--- diffold/BT Test/http_link.cpp  2006-02-10 16:43:55.000000000 -0100
+++ diffnew/BT Test/http_link.cpp       2006-02-10 16:34:29.000000000 -0100
@@ -34,3 +34,8 @@
                        m_s.getsockopt(SOL_SOCKET, SO_ERROR, e);
+#ifdef EVENT
                        alert(Calert::error, "connect failed: " + Csocket::error2a(e));
+#endif
+#ifdef EVENTLOG
+                       event_logger().event(Cbt_logger::error, "connect failed: " + Csocket::error2a(e));
+#endif
                        return 1;
@@ -53,3 +58,8 @@
                                        return 0;
+#ifdef EVENT
                                alert(Calert::debug, "recv failed: " + Csocket::error2a(e));
+#endif
+#ifdef EVENTLOG
+                               event_logger().event(Cbt_logger::debug, "recv failed: " + Csocket::error2a(e));
+#endif
                                return 1;
@@ -73,3 +83,8 @@
                                        return 0;
+#ifdef EVENT
                                alert(Calert::debug, "send failed: " + Csocket::error2a(e));
```

```
+#endif
+#ifdef EVENTLOG
+                               event_logger().event(Cbt_logger::debug, "send failed: " +
Csocket::error2a(e));
+#endif
                                return 1;
@@ -112 +127,6 @@
 }
+Cbt_logger& Chttp_link::event_logger()
+{
+       return m_server->event_logger();
+}
+
diff -u1 -r diffold/BT Test/http_link.h diffnew/BT Test/http_link.h
--- diffold/BT Test/http_link.h     2006-02-10 16:43:55.000000000 -0100
+++ diffnew/BT Test/http_link.h 2006-02-10 16:34:29.000000000 -0100
@@ -11,2 +11,3 @@
 #include "socket.h"
+#include "bt_logger.h"

@@ -17,2 +18,3 @@
 public:
+       Cbt_logger& event_logger();
        void close();
diff -u1 -r diffold/BT Test/make.sh diffnew/BT Test/make.sh
--- diffold/BT Test/make.sh         2006-02-10 16:43:55.000000000 -0100
+++ diffnew/BT Test/make.sh     2006-02-10 16:34:29.000000000 -0100
@@ -1,2 +1,4 @@
+#!/bin/sh
 clear
-g++ -DNDEBUG -I ../misc -I . -O3 -lz -o xbt_client_backend *.cpp ../misc/*.cpp && strip
xbt_client_backend
+g++ -g3 -pg -DEVENTLOG -DNETFILTERLOG -I ../misc -I . -lz -o xbt_client_backend *.cpp
../misc/*.cpp
+#strip xbt_client_backend
diff -u1 -r diffold/BT Test/server.cpp diffnew/BT Test/server.cpp
--- diffold/BT Test/server.cpp      2006-02-10 16:43:55.000000000 -0100
+++ diffnew/BT Test/server.cpp 2006-02-10 16:34:29.000000000 -0100
@@ -6,2 +6,3 @@
 #include <signal.h>
+#include "netfilter.h"
 #include "bt_strings.h"
@@ -94,3 +95,7 @@
 #ifndef NDEBUG
-       m_logger.open("/temp/bt_logger.txt");
+       m_logger.open("/tmp/bt_logger.txt");
+#endif
+#ifdef EVENTLOG
+       m_event_logger.open("/tmp/bt_event_logger.txt");
+       m_netfilter_logger.open("/tmp/bt_firewall_logger.txt");
 #endif
@@ -130,2 +135,14 @@

+void Cserver::net_filter_seeding(bool v)
+{
+       m_config.m_net_filter_seeding = v;
+       (&m_seeding_netfilters)->m_alter_netfilters_request = true;
+}
+
+void Cserver::net_filter_leeching(bool v)
```

```
+{
+	m_config.m_net_filter_leeching = v;
+	(&m_leeching_netfilters)->m_alter_netfilters_request = true;
+}
+
 void Cserver::tracker_port(int v)
@@ -150,2 +167,37 @@

+
+int Cserver::pop_netfilter(int mode) const
+{
+	if(mode == 0 || mode == 2)
+	{
+		Cnetfilters* p_seeding_netfilters = const_cast<Cnetfilters*>(&m_seeding_netfilters);
+		p_seeding_netfilters->clear();
+		//for (Cnetfilters::const_iterator j = m_seeding_netfilters.begin(); j !=
m_seeding_netfilters.end(); j++) {
+		//	p_seeding_netfilters->net_erase(*&*j);
+	}
+	if(mode == 1 || mode == 2)
+	{
+		Cnetfilters* p_leeching_netfilters = const_cast<Cnetfilters*>(&m_leeching_netfilters);
+		p_leeching_netfilters->clear();
+	}
+	return 0;
+}
+int Cserver::pop_netfilter(int mode, int net, int nmask)
+{
+	Cnetfilter* f = find_netfilter(mode,net,nmask);
+	  if (!f)
+		return 1;
+	  if(mode == 0)
+		m_seeding_netfilters.net_erase(*f);
+	  else if(mode == 1)
+		m_leeching_netfilters.net_erase(*f);
+	  else if(mode == 2)
+	  {
+		m_seeding_netfilters.net_erase(*f);
+		m_leeching_netfilters.net_erase(*f);
+	  }
+	  return 0;
+}
+
+
 #ifdef WIN32
@@ -182,3 +234,10 @@
	  if (FAILED(hr))
+	  {
+#ifdef EVENT
+		alert(Calert(Calert::warn, "Server", "CoInitialize failed: " + hex_encode(8, hr)));
+#endif
+#ifdef EVENTLOG
+		event_logger().event(Cbt_logger::warn, "Server", "CoInitialize failed: " +
hex_encode(8, hr));
+#endif
+	  }
	  IStaticPortMappingCollection* static_port_mapping_collection = NULL;
@@ -189,3 +248,10 @@
		if (FAILED(hr) || !upnp_nat)
+		{
```

```
+#ifdef EVENT
                                alert(Calert(Calert::warn, "UPnP NAT", "CoCreateInstance failed: " +
hex_encode(8, hr)));
+#endif
+#ifdef EVENTLOG
+                               event_logger().event(Cbt_logger::warn,"UPnP NAT", "CoCreateInstance
failed: " + hex_encode(8, hr));
+#endif
+                       }
                        else
@@ -195,3 +261,10 @@
                                if (FAILED(hr) || !static_port_mapping_collection)
+                               {
+#ifdef EVENT
                                        alert(Calert(Calert::warn, "UPnP NAT",
"get_StaticPortMappingCollection failed: " + hex_encode(8, hr)));
+#endif
+#ifdef EVENTLOG
+                                       event_logger().event(Cbt_logger::warn, "UPnP NAT",
"get_StaticPortMappingCollection failed: " + hex_encode(8, hr));
+#endif
+                               }
                        }
@@ -207,6 +280,13 @@
                        if (la.open(SOCK_STREAM) == INVALID_SOCKET)
+                       {
+#ifdef EVENT
                                alert(Calert(Calert::error, "Server", "socket failed: " +
Csocket::error2a(WSAGetLastError())));
+#endif
+#ifdef EVENTLOG
+                               event_logger().event(Cbt_logger::error, "Server", "socket failed: " +
Csocket::error2a(WSAGetLastError()));
+#endif
+                       }
                        else
                        {
-                               while (admin_port() < 0x10000 && la.setsockopt(SOL_SOCKET,
SO_REUSEADDR, true), la.bind(htonl(INADDR_LOOPBACK), htons(admin_port())) &&
WSAGetLastError() == WSAEADDRINUSE)
+                               while (admin_port() < 0x10000 && la.setsockopt(SOL_SOCKET,
SO_REUSEADDR, true), la.bind(htonl(INADDR_ANY), htons(admin_port())) && WSAGetLastError()
== WSAEADDRINUSE)
                                        m_admin_port++;
@@ -214,3 +294,8 @@
                                {
+#ifdef EVENT
                                        alert(Calert(Calert::error, "Server", "listen failed: " +
Csocket::error2a(WSAGetLastError())));
+#endif
+#ifdef EVENTLOG
+                                       event_logger().event(Cbt_logger::error, "Server", "listen failed: " +
Csocket::error2a(WSAGetLastError()));
+#endif
                                        la.close();
@@ -222,3 +307,10 @@
                        if (l.open(SOCK_STREAM) == INVALID_SOCKET)
+                       {
+#ifdef EVENT
```

```
                        alert(Calert(Calert::error, "Server", "socket failed: " +
Csocket::error2a(WSAGetLastError())));
+#endif
+#ifdef EVENTLOG
+                       event_logger().event(Cbt_logger::error, "Server", "socket failed: " +
Csocket::error2a(WSAGetLastError()));
+#endif
+                   }
                    else
@@ -245,3 +337,8 @@
                                        {
+#ifdef EVENT
                                                alert(Calert(Calert::warn, "UPnP NAT",
"static_port_mapping_collection->Add failed failed: " + hex_encode(8, hr)));
+#endif
+#ifdef EVENTLOG
+                                               event_logger().event(Cbt_logger::warn, "UPnP NAT",
"static_port_mapping_collection->Add failed failed: " + hex_encode(8, hr));
+#endif
                                                break;
@@ -253,6 +350,16 @@
                                        {
+#ifdef EVENT
                                                alert(Calert(Calert::warn, "UPnP NAT",
"static_port_mapping->get_ExternalIPAddress failed: " + hex_encode(8, hr)));
+#endif
+#ifdef EVENTLOG
+                                               event_logger().event(Cbt_logger::warn, "UPnP NAT",
"static_port_mapping->get_ExternalIPAddress failed: " + hex_encode(8, hr));
+#endif
                                                break;
                                        }
+#ifdef EVENT
                                        alert(Calert(Calert::info, "UPnP NAT", "External IPA: " +
wchar_to_mbyte(bstrExternalIPA)));
+#endif
+#ifdef EVENTLOG
+                                       event_logger().event(Cbt_logger::warn, "External IPA: " +
wchar_to_mbyte(bstrExternalIPA));
+#endif
                                        SysFreeString(bstrExternalIPA);
@@ -261,3 +368,8 @@
                                {
+#ifdef EVENT
                                        alert(Calert(Calert::warn, "UPnP NAT", e.what()));
+#endif
+#ifdef EVENTLOG
+                                       event_logger().event(Cbt_logger::warn, "UPnP NAT",
e.what());
+#endif
                                }
@@ -268,3 +380,8 @@
                        {
+#ifdef EVENT
                                alert(Calert(Calert::error, "Server", "listen failed: " +
Csocket::error2a(WSAGetLastError())));
+#endif
+#ifdef EVENTLOG
+                               event_logger().event(Cbt_logger::error, "Server", "listen failed: " +
Csocket::error2a(WSAGetLastError()));
```

```
+#endif
                                        l.close();
@@ -276,3 +393,10 @@
                if (lt.open(SOCK_DGRAM) == INVALID_SOCKET)
+               {
+#ifdef EVENT
                                        alert(Calert(Calert::error, "Server", "socket failed: " +
Csocket::error2a(WSAGetLastError())));
+#endif
+#ifdef EVENTLOG
+                                       event_logger().event(Cbt_logger::error, "Server", "socket failed: " +
Csocket::error2a(WSAGetLastError()));
+#endif
+               }
                else
@@ -295,2 +419,3 @@
        m_tracker_accounts.load(Cvirtual_binary(trackers_fname()));
+       write_rules(Cvirtual_binary(netfilters_fname()));
        clean_scheduler();
@@ -299,3 +424,10 @@
        if (daemon(true, false))
+       {
+#ifdef EVENT
                alert(Calert(Calert::error, "Server", "daemon failed: " + n(errno)));
+#endif
+#ifdef EVENTLOG
+               event_logger().event(Cbt_logger::error, "Server", "daemon failed: " + n(errno));
+#endif
+       }
        ofstream(g_pid_fname) << getpid() << endl;
@@ -329,3 +461,3 @@
                                        else if (s.open(SOCK_STREAM) != INVALID_SOCKET
-                                               && !s.bind(htonl(INADDR_LOOPBACK),
htons(m_config.m_admin_port))
+                                               && !s.bind(htonl(INADDR_ANY), htons(m_config.m_admin_port))
                                                && !s.listen())
@@ -398,3 +530,8 @@
                {
+#ifdef EVENT
                                        alert(Calert(Calert::error, "Server", "select failed: " +
Csocket::error2a(WSAGetLastError())));
+#endif
+#ifdef EVENTLOG
+                                       event_logger().event(Cbt_logger::error, "Server", "select failed: " +
Csocket::error2a(WSAGetLastError()));
+#endif
                                        continue;
@@ -420,3 +557,3 @@
                lock();
-               if (l != INVALID_SOCKET && FD_ISSET(l, &fd_read_set))
+               if (la != INVALID_SOCKET && FD_ISSET(la, &fd_read_set))
                {
@@ -426,3 +563,3 @@
                                        socklen_t cb_a = sizeof(sockaddr_in);
-                                       Csocket s = accept(l, reinterpret_cast<sockaddr*>(&a), &cb_a);
+                                       Csocket s = accept(la, reinterpret_cast<sockaddr*>(&a), &cb_a);
                                        if (s == SOCKET_ERROR)
@@ -430,10 +567,24 @@
                                                if (WSAGetLastError() != WSAEWOULDBLOCK)
+                                               {
```

```
+#ifdef EVENT
                                                alert(Calert(Calert::error, "Server", "accept failed: " +
Csocket::error2a(WSAGetLastError())));
+#endif
+#ifdef EVENTLOG
+                                               event_logger().event(Cbt_logger::error, "Server",
"accept failed: " + Csocket::error2a(WSAGetLastError()));
+#endif
+                                               }
                                                break;
                                        }
-                                       else if (!block_list_has(a.sin_addr.s_addr))
+                                       else
                                        {
                                                if (s.blocking(false))
+                                               {
+#ifdef EVENT
                                                alert(Calert(Calert::error, "Server", "ioctlsocket failed: "
+ Csocket::error2a(WSAGetLastError())));
-                                               m_links.push_back(Cbt_link(this, a, s));
+#endif
+#ifdef EVENTLOG
+                                               event_logger().event(Cbt_logger::error, "Server",
"ioctlsocket failed: " + Csocket::error2a(WSAGetLastError()));
+#endif
+                                               }
+                                               m_admins.push_back(Cbt_admin_link(this, a, s));
                                        }
@@ -441,3 +592,3 @@
                        }
-                       if (la != INVALID_SOCKET && FD_ISSET(la, &fd_read_set))
+                       if (l != INVALID_SOCKET && FD_ISSET(l, &fd_read_set))
                        {
@@ -447,3 +598,3 @@
                                socklen_t cb_a = sizeof(sockaddr_in);
-                               Csocket s = accept(la, reinterpret_cast<sockaddr*>(&a), &cb_a);
+                               Csocket s = accept(l, reinterpret_cast<sockaddr*>(&a), &cb_a);
                                if (s == SOCKET_ERROR)
@@ -451,10 +602,24 @@
                                        if (WSAGetLastError() != WSAEWOULDBLOCK)
+                                       {
+#ifdef EVENT
                                                alert(Calert(Calert::error, "Server", "accept failed: " +
Csocket::error2a(WSAGetLastError())));
+#endif
+#ifdef EVENTLOG
+                                               event_logger().event(Cbt_logger::error, "Server",
"accept failed: " + Csocket::error2a(WSAGetLastError()));
+#endif
+                                               }
                                                break;
                                        }
-                                       else
+                                       else if(!block_list_has(a.sin_addr.s_addr))
+                                       {
-                                               if (s.blocking(false))
-                                               alert(Calert(Calert::error, "Server", "ioctlsocket failed: "
+ Csocket::error2a(WSAGetLastError())));
-                                               m_admins.push_back(Cbt_admin_link(this, a, s));
+                                               if(s.blocking(false))
```

```
+                                       {
+#ifdef EVENT
+                                               alert(Calert(Calert::error, "Server", "ioctlsocket failed: "
+ Csocket::error2a(WSAGetLastError())));
+#endif
+#ifdef EVENTLOG
+                                               event_logger().event(Cbt_logger::error, "Server",
"ioctlsocket failed: " + Csocket::error2a(WSAGetLastError()));
+#endif
+                                       }
+                                       m_links.push_back(Cbt_link(this, a, s));
                                }
@@ -462,2 +627,3 @@
                }
+
                if (lt != INVALID_SOCKET && FD_ISSET(lt, &fd_read_set))
@@ -477,11 +643,11 @@
                }
-               if (time() - m_update_chokes_time > 10)
+               if (time() - m_update_chokes_time > 100)
                        update_chokes();
-               else if (time() - m_update_states_time > 15)
+               else if (time() - m_update_states_time > 150)
                        update_states();
-               else if (time() - m_save_state_time > 60)
+               else if (time() - m_save_state_time > 600)
                        save_state(true).save(state_fname());
-               else if (time() - m_run_scheduler_time > 60)
+               else if (time() - m_run_scheduler_time > 600)
                        run_scheduler();
-               else if (time() - m_check_remote_links_time > 900)
+               else if (time() - m_check_remote_links_time > 1900)
                        check_remote_links();
@@ -565,5 +731,104 @@
                        continue;
+               bool b_am = i->m_alter_mode_request;
                for (Cbt_file::t_peers::iterator j = i->m_peers.begin(); j != i->m_peers.end(); )
-               {
+               {
                        int q = INT_MAX;
+                       int default_rule = false;
+                       bool b_ap = j->m_alter_peer_request, b_asn =
m_seeding_netfilters.m_alter_netfilters_request, b_aln =
m_leeching_netfilters.m_alter_netfilters_request;
+                       if(b_am || b_ap || b_asn)
+                       {
+#ifdef EVENT
+                                       //i->alert(Calert(Calert::debug, n(i->m_alter_mode_request) + n(j-
>m_alter_peer_request) +n(m_seeding_netfilters.m_alter_netfilters_request),  n((j-
>m_a.sin_addr.s_addr)) + i->m_info_hash));
+#endif
+#ifdef NETFILTERLOG
+                                       //netfilter_logger().firewall((string)(Csocket::inet_ntoa(j-
>m_a.sin_addr.s_addr)), (string)(i->m_info_hash), i->m_completed_at*100 +  i-
>m_alter_mode_request*10 + j->m_alter_peer_request,
m_seeding_netfilters.m_alter_netfilters_request);
+#endif
+                                       j->m_alter_peer_request = false;
+                                       default_rule = true;
```

```
+                                       for (Cnetfilters::const_iterator k = m_seeding_netfilters.begin(); k !=
m_seeding_netfilters.end(); k++)
+                                       {
+                                               if(!i->m_left && (k->mode() == Cnetfilter::seeding || k->mode()
== Cnetfilter::both))
+                                               {
+                                       //int s_ip = ntohl(j->m_a.sin_addr.s_addr);
+                                                       //int s_mask = ((1 << k->nmask()) - 1) << (32 - k-
>nmask());
+                                                       //int s_knet = k->net();
+                                                       //netfilter_logger().print(s_ip,s_mask,s_knet, k->rule());
+                                                       if(((ntohl(j->m_a.sin_addr.s_addr) & ((1 << k-
>nmask()) - 1) << (32 - k->nmask())) == (k->net() & ((1 << k->nmask()) - 1) << (32 - k->nmask()))))
+                                                       {
+                                                               if(k->rule() == Cnetfilter::deny)
+                                                               {
+#ifdef EVENT
+                                                                       i->alert(Calert(Calert::debug, n(ntohl(j-
>m_a.sin_addr.s_addr)), hex_encode(i->m_info_hash) + n(k->mode()) + n(k->rule())));
+#endif
+#ifdef NETFILTERLOG
+
        netfilter_logger().firewall((string)(Csocket::inet_ntoa(j->m_a.sin_addr.s_addr)), (string)(i-
>m_info_hash), b_am, b_ap, b_asn, k->mode(), k->rule());
+#endif
+                                                                       i->peer_disconnect(j-
>m_a.sin_addr.s_addr);
+                                                                       goto next_peer;
+                                                               }
+                                                               else
+                                                               {
+
        netfilter_logger().firewall((string)(Csocket::inet_ntoa(j->m_a.sin_addr.s_addr)), (string)(i-
>m_info_hash), b_am, b_ap, b_asn, k->mode(), k->rule());
+                                                                       goto allow_peer;
+                                                               }
+                                                       }
+                                               }
+                                       }
+                               }
+                       if(b_am || b_ap || b_aln)
+                               {
+                                       //i->alert(Calert(Calert::debug, n(i->m_alter_mode_request) + n(j-
>m_alter_peer_request) +n(m_leeching_netfilters.m_alter_netfilters_request), n(j-
>m_a.sin_addr.s_addr) + i->m_info_hash ));
+                                       j->m_alter_peer_request = false;
+                                       default_rule = true;
+                                       for (Cnetfilters::const_iterator k = m_leeching_netfilters.begin(); k !=
m_leeching_netfilters.end(); k++)
+                                       {
+                                               if(i->m_left && (k->mode() == Cnetfilter::leeching || k->mode()
== Cnetfilter::both))
+                                               {
+                                                       //int l_ip = j->m_a.sin_addr.s_addr;
+                                                       //int l_mask = ((1 << k->nmask()) - 1) << (32 - k-
>nmask());
+                                                       //int l_knet = k->net();
+                                                       //netfilter_logger().print(l_ip,l_mask,l_knet, k->rule());
+                                                       if(((ntohl(j->m_a.sin_addr.s_addr) & ((1 << k-
>nmask()) - 1) << (32 - k->nmask())) == (k->net() & ((1 << k->nmask()) - 1) << (32 - k->nmask()))))
```

```
+                                                       {
+                                                               if(k->rule() == Cnetfilter::deny)
+                                                               {
+#ifdef EVENT
+                                                                       i->alert(Calert(Calert::debug, n(ntohl(j-
>m_a.sin_addr.s_addr)), hex_encode(i->m_info_hash) + n(k->mode()) + n(k->rule())));
+#endif
+#ifdef NETFILTERLOG
+
        netfilter_logger().firewall((string)(Csocket::inet_ntoa(j->m_a.sin_addr.s_addr)), (string)(i-
>m_info_hash), b_am, b_ap, b_aln, k->mode(), k->rule());
+#endif
+                                                                       i->peer_disconnect(j-
>m_a.sin_addr.s_addr);
+                                                                       goto next_peer;
+                                                               }
+                                                               else
+                                                               {
+
        netfilter_logger().firewall((string)(Csocket::inet_ntoa(j->m_a.sin_addr.s_addr)), (string)(i-
>m_info_hash), b_am, b_ap, b_aln, k->mode(), k->rule());
+                                                                       goto allow_peer;
+                                                               }
+
+                                                       }
+                                               }
+                                       }
+                               }
+                               if(default_rule)
+                               {
+                                       if(((config().m_net_filter_seeding) && !i->m_left) ||
((config().m_net_filter_leeching) && i->m_left))
+                                       {
+#ifdef EVENT
+                                               i->alert(Calert(Calert::debug, n(ntohl(j-
>m_a.sin_addr.s_addr)), hex_encode(i->m_info_hash)  + n(i->m_left)));
+#endif
+#ifdef NETFILTERLOG
+                                               netfilter_logger().firewall((string)(Csocket::inet_ntoa(j-
>m_a.sin_addr.s_addr)), (string)(i->m_info_hash), b_am, b_ap, (b_asn || b_aln), i->m_left, false);
+#endif
+                                               i->peer_disconnect(j->m_a.sin_addr.s_addr);
+                                               j++;
+                                               continue;
+                                       }
+                                       else
+#ifdef NETFILTERLOG
+                                               netfilter_logger().firewall((string)(Csocket::inet_ntoa(j-
>m_a.sin_addr.s_addr)), (string)(i->m_info_hash) , b_am, b_ap, (b_asn || b_aln), i->m_left, true);
+#endif
+                               }
+allow_peer:                    ;
                                if (m_config.m_upload_rate)
@@ -579,6 +844,11 @@
                                }
-                              else
-                                      j++;
+                              else
+                              {
+next_peer:                             j++;
```

```
+                               }
                        }
+                       i->m_alter_mode_request = false;
                }
+               (&m_seeding_netfilters)->m_alter_netfilters_request = false;
+               (&m_leeching_netfilters)->m_alter_netfilters_request = false;
                if (!m_config.m_upload_rate)
@@ -957,5 +1227,7 @@

-int Cserver::close(const string& id, bool erase)
+Cbvalue Cserver::close(const string& id, bool erase, bool erase_torrent)
 {
                Clock l(m_cs);
+               Cbvalue d;
+               int rmdir_err = 0;
                for (t_files::iterator i = m_files.begin(); i != m_files.end(); i++)
@@ -964,10 +1236,32 @@
                                continue;
-                       i->close();
-                       if (erase)
-                               i->erase();
-                       m_files.erase(i);
+                       i->close();
+                       if (erase)
+                       {
+                               string s_err;
+                               s_err = i->erase();
+                               if(s_err != "")
+                                       d.l(Cbvalue().d(bts_warning, (string)"Close torrent action failed:
Cannot remove torrent directories").d(bts_code, -1).d(bts_files, s_err));
+                               //if(rmdir_err = rmpath(i->m_name))
+                               //      d.l(Cbvalue().d(bts_warning, (string)"Close torrent action failed:
Cannot remove torrent directories").d(bts_code, -1).d(bts_files, s_err));
+                       }
+                       if (erase_torrent)
+                       {
+                               int a = i->m_name.find_last_of("/");
+                               if (a == string::npos)
+                               {
+                                       d.l(Cbvalue().d(bts_error, (string)"Close torrent action failed: Internal
error, Cannot remove torrent").d(bts_code, -5));
+                               }
+                               string removed = torrents_dir() + i->m_name.substr(a) + ' ' + hex_encode(i-
>m_info_hash) + ".torrent";
+                               if(rmdir_err = unlink(removed.c_str()))
+                               {
+                                       d.l(Cbvalue().d(bts_warning, (string)"Close torrent action failed:
Cannot remove torrent").d(bts_code, rmdir_err).d(bts_files, removed));
+                               }
+                       }
+                       m_files.erase(i);
                        save_state(true).save(state_fname());
-                       return 0;
+
+                       return d;
                }
-               return 1;
+               d.l(Cbvalue().d(bts_warning, (string)"Close torrent action failed: Invalid info hash").d(bts_code,
1));
+               return d;
```

```
 }
@@ -1034,2 +1328,7 @@

+string Cserver::netfilters_fname() const
+{
+	return local_app_data_dir() + "/netfilters.bin";
+}
+
 void Cserver::alert(const Calert& v)
@@ -1039,2 +1338,11 @@

+void Cserver::seeding_netfilter(const Cnetfilter& v)
+{
+	m_seeding_netfilters.push_back(v);
+}
+void Cserver::leeching_netfilter(const Cnetfilter& v)
+{
+	m_leeching_netfilters.push_back(v);
+}
+
 void Cserver::update_chokes()
@@ -1204,7 +1512,19 @@
 		Cbvalue d;
+		__int64 cb_err = 0;
 		string action = s.d(bts_action).s();
-		if (action == bts_close_torrent)
-			close(s.d(bts_hash).s(), false);
-		else if (action == bts_erase_torrent)
-			close(s.d(bts_hash).s(), true);
+		if (action == bts_close_torrent) {
+			string s_close_err;
+			if((s.d_has(bts_remove_torrent)) && (s.d_has(bts_remove_data))) {
+				return close(s.d(bts_hash).s(), s.d(bts_remove_data).i(),
s.d(bts_remove_torrent).i());
+			}
+			else if (s.d_has(bts_remove_torrent)) {
+				return close(s.d(bts_hash).s(), 0, s.d(bts_remove_torrent).i());
+			}
+			else if (s.d_has(bts_remove_data)) {
+				return close(s.d(bts_hash).s(), s.d(bts_remove_data).i());
+			}
+			else if (!(s.d_has(bts_remove_torrent)) && !(s.d_has(bts_remove_data))) {
+				return close(s.d(bts_hash).s());
+			}
+		}
 		else if (action == bts_get_options)
@@ -1213,2 +1533,4 @@
 			d.d(bts_peer_port, peer_port());
+			d.d(bts_net_filter_seeding, net_filter_seeding());
+			d.d(bts_net_filter_leeching, net_filter_leeching());
 			d.d(bts_tracker_port, tracker_port());
@@ -1223,2 +1545,11 @@
 			d.d(bts_torrents_dir, native_slashes(torrents_dir()));
+
+			Cbvalue rulenetnmask(Cbvalue::vt_list);
+			for (Cnetfilters::const_iterator j = m_seeding_netfilters.begin(); j !=
m_seeding_netfilters.end(); j++)
+				rulenetnmask.l(Cbvalue().d(bts_rule, j->rule()).d(bts_mode, j-
>mode()).d(bts_net, j->net()).d(bts_net_mask, j->nmask()));
```

```
+                   for (Cnetfilters::const_iterator k = m_leeching_netfilters.begin(); k !=
m_leeching_netfilters.end(); k++)
+                           if(k->mode() != Cnetfilter::both) //just a litle hack not to showeing the same
records
+                                   rulenetnmask.l(Cbvalue().d(bts_rule, k->rule()).d(bts_mode, k-
>mode()).d(bts_net, k->net()).d(bts_net_mask, k->nmask()));
+                   d.d(bts_net_filter, rulenetnmask);
+
           }
@@ -1226,8 +1557,8 @@
           {
-                   Cbvalue files(Cbvalue::vt_dictionary);
+                   Cbvalue files(Cbvalue::vt_list);
                    for (t_files::const_iterator i = m_files.begin(); i != m_files.end(); i++)
                    {
-                           Cbvalue events;
+                           Cbvalue events(Cbvalue::vt_list);
                            for (Calerts::const_reverse_iterator j = i->m_alerts.rbegin(); j != i-
>m_alerts.rend(); j++)
-                                   events.l(Cbvalue().d(bts_time, j->time()).d(bts_message, j-
>message()));
+                                   events.l(Cbvalue().d(bts_time, j->time()).d(bts_level, j-
>level()).d(bts_message, j->message()));
                            Cbvalue file;
@@ -1252,6 +1583,13 @@
                    d.d(bts_files, files);
+                   d.d(bts_space_left, space_left());
                    d.d(bts_version, xbt_version2a(version()));
           }
-          else if (action == bts_open_torrent)
-                   open(Cvirtual_binary(s.d(bts_torrent).s().c_str(), s.d(bts_torrent).s().size()), "");
+          else if (action == bts_open_torrent) {
+                   if( cb_err = open(Cvirtual_binary(s.d(bts_torrent).s().c_str(), s.d(bts_torrent).s().size()),
"")) {
+                    if(cb_err == 1) d.l(Cbvalue().d(bts_error, (string)"Open Torrent action failed: Cannot
initialize the virtual binary").d(bts_code, cb_err));
+                           if(cb_err == 2) d.l(Cbvalue().d(bts_error, (string)"Open Torrent action failed:
Duplicate info hash").d(bts_code, cb_err));
+                           return d;
+
+                   }
+          }
           else if (action == bts_set_options)
@@ -1260,2 +1598,6 @@
                           peer_port(s.d(bts_peer_port).i());
+                   if (s.d_has(bts_net_filter_seeding))
+                           net_filter_seeding(s.d(bts_net_filter_seeding).i());
+               if (s.d_has(bts_net_filter_leeching))
+                           net_filter_leeching(s.d(bts_net_filter_leeching).i());
                    if (s.d_has(bts_tracker_port))
@@ -1280,7 +1622,35 @@
                           torrents_dir(s.d(bts_torrents_dir).s());
+                   if (s.d_has(bts_net_filter))
+                           return net_filter(s);
+          }
+          else if (action == bts_add_net_filter) {
+                   return add_net_filter(s);
+          }
+          else if (action == bts_remove_net_filter) {
+                   return remove_net_filter(s);
```

```
+		}
+		else if (action == bts_set_priority) {
+			if( cb_err = file_priority(s.d(bts_hash).s(), s.d(bts_priority).i())) {
+				d.l(Cbvalue().d(bts_error, (string)"Changing torrent's priority failed: Cannot find
the specified torrent").d(bts_code, cb_err));
+				return d;
+
+			}
		}
-		else if (action == bts_set_priority)
-			file_priority(s.d(bts_hash).s(), s.d(bts_priority).i());
-		else if (action == bts_set_state)
-			file_state(s.d(bts_hash).s(), static_cast<Cbt_file::t_state>(s.d(bts_state).i()));
+		else if (action == bts_set_state) {
+			if( cb_err = file_state(s.d(bts_hash).s(),
static_cast<Cbt_file::t_state>(s.d(bts_state).i()))) {
+			 if(cb_err == 1) d.l(Cbvalue().d(bts_error, (string)"Changing torrent's state failed:
Cannot find the specified torrent").d(bts_code, cb_err));
+				else d.l(Cbvalue().d(bts_error, (string)"Unknown state error").d(bts_code,
cb_err));
+				return d;
+			}
+		}
+		else if(action != "")
+		{
+			 d.l(Cbvalue().d(bts_error, (string)"Wrong action").d(bts_code, 1));
+			 return d;
+		}
+		else if(action == "")
+		{
+			d.l(Cbvalue().d(bts_error, (string)"Missing action").d(bts_code, 1));
+			return d;
+		}
		return d;
@@ -1320,4 +1690,10 @@
			return;
+#ifdef EVENT
		alert(Calert(Calert::warn, "Did you forget to open a port in your firewall or router?"));
		alert(Calert(Calert::warn, n(c_local_links) + " local links, but no remote links have been
established."));
+#endif
+#ifdef EVENTLOG
+		event_logger().event(Cbt_logger::warn, "Did you forget to open a port in your firewall or
router?");
+		event_logger().event(Cbt_logger::warn, n(c_local_links) + " local links, but no remote links
have been established.");
+#endif
 }
@@ -1382 +1758,197 @@
 }
+
+Cnetfilter* Cserver::find_netfilter(int mode, int net, int nmask)
+{
+		if(mode == 0 || mode == 2)
+			for (Cnetfilters::iterator i = m_seeding_netfilters.begin(); i !=
m_seeding_netfilters.end(); i++)
+				if ((i->net() == net) && (i->nmask() == nmask)) {
+					int int_mode = -1;
+				  switch(i->mode()) {
+					case Cnetfilter::seeding: int_mode = 0; break;
```

```
+                                               case Cnetfilter::leeching: int_mode = 1; break;
+                                               default: int_mode = 2;  break;
+                                       }
+                                       if(int_mode == mode || mode == 2)
+                                               return &*i;
+                               }
+       if(mode == 1 || mode == 2)
+                       for (Cnetfilters::iterator i = m_leeching_netfilters.begin(); i !=
m_leeching_netfilters.end(); i++)
+                               if ((i->net() == net) && (i->nmask() == nmask)) {
+                                       int int_mode = -1;
+                                       switch(i->mode()) {
+                                               case Cnetfilter::seeding: int_mode = 0; break;
+                                               case Cnetfilter::leeching: int_mode = 1; break;
+                                               default: int_mode = 2; break;
+                                       }
+                                       if(int_mode == mode || mode == 2)
+                                               return &*i;
+                               }
+       return NULL;
+}
+
+void Cserver::write_rules(const Cbvalue& v)
+{
+       for(t_list::const_iterator iter = v.l().begin(); iter != v.l().end(); iter++)
+                       if ((*iter).d_has(bts_net))
+                               if ((*iter).d_has(bts_net_mask))
+                               {
+                                       Cnetfilter::t_rule rule = Cnetfilter::error;
+                                       switch((*iter).d(bts_rule).i()) {
+                                               case 0: rule = Cnetfilter::allow; break;
+                                               case 1: rule = Cnetfilter::deny; break;
+                                       }
+                                       Cnetfilter::t_mode mode = Cnetfilter::none;
+                                       switch((*iter).d(bts_mode).i()) {
+                                               case 0: mode = Cnetfilter::seeding;
+                                                       seeding_netfilter(Cnetfilter(rule, mode,
(*iter).d(bts_net).i(), (*iter).d(bts_net_mask).i()));
+                                                       break;
+                                               case 1: mode = Cnetfilter::leeching;
+                                                       leeching_netfilter(Cnetfilter(rule, mode,
(*iter).d(bts_net).i(), (*iter).d(bts_net_mask).i()));
+                                                       break;
+                                               case 2: mode = Cnetfilter::both;
+                                                       seeding_netfilter(Cnetfilter(rule, Cnetfilter::both,
(*iter).d(bts_net).i(), (*iter).d(bts_net_mask).i()));
+                                                       leeching_netfilter(Cnetfilter(rule, Cnetfilter::both,
(*iter).d(bts_net).i(), (*iter).d(bts_net_mask).i()));
+                                                       break;
+                                       }
+                               }
+}
+
+
+void Cserver::read_rules() const
+{
+       Cbvalue write_netfilters(Cbvalue::vt_list);
+       for (Cnetfilters::const_iterator j = m_seeding_netfilters.begin(); j != m_seeding_netfilters.end();
j++)
```

```
+                   write_netfilters.l(Cbvalue().d(bts_rule, j->rule()).d(bts_mode, j->mode()).d(bts_net, j-
>net()).d(bts_net_mask, j->nmask()));
+           for (Cnetfilters::const_iterator k = m_leeching_netfilters.begin(); k !=
m_leeching_netfilters.end(); k++)
+                   write_netfilters.l(Cbvalue().d(bts_rule, k->rule()).d(bts_mode, k->mode()).d(bts_net, k-
>net()).d(bts_net_mask, k->nmask()));
+           write_netfilters.read().save(netfilters_fname());
+}
+
+Cbvalue Cserver::net_filter(const Cbvalue& s)
+{
+           Cbvalue d;
+           pop_netfilter(2);
+           int c = 1;
+           int has_seeding = 0, has_leeching = 0;
+           for(t_list::const_iterator iter = s.d(bts_net_filter).l().begin(); iter != s.d(bts_net_filter).l().end();
iter++, c++)       {

+                   if ((*iter).d_has(bts_net) && (*iter).d_has(bts_net_mask))                  {

+                           Cnetfilter::t_rule rule = Cnetfilter::error;
+                           if ((*iter).d_has(bts_rule)) {
+                                   switch((*iter).d(bts_rule).i()) {
+                                           case 0: rule = Cnetfilter::allow;
+                                                   break;
+                                           case 1: rule = Cnetfilter::deny;
+                                                   break;
+                                   }
+                           }
+                           if(!s.d_has(bts_mode)) const_cast<Cbvalue*>(&s)-
>d(bts_mode,static_cast<int>(Cnetfilter::both));
+                           Cnetfilter::t_mode mode = Cnetfilter::none;
+                           switch((*iter).d(bts_mode).i()) {
+                                   case 0: mode = Cnetfilter::seeding;
+                                           seeding_netfilter(Cnetfilter(rule, mode, (*iter).d(bts_net).i(),
(*iter).d(bts_net_mask).i()));
+                                           has_seeding = 1;
+                                           break;
+                                   case 1: mode = Cnetfilter::leeching;
+                                           leeching_netfilter(Cnetfilter(rule, mode, (*iter).d(bts_net).i(),
(*iter).d(bts_net_mask).i()));
+                                           has_leeching = 1;
+                                           break;
+                                   case 2: mode = Cnetfilter::both;
+                                           seeding_netfilter(Cnetfilter(rule, Cnetfilter::both,
(*iter).d(bts_net).i(), (*iter).d(bts_net_mask).i()));
+                                           leeching_netfilter(Cnetfilter(rule, Cnetfilter::both,
(*iter).d(bts_net).i(), (*iter).d(bts_net_mask).i()));
+                                           has_leeching = has_seeding = 1;
+                                           break;
+                           }
+                           if((*iter).d(bts_mode).i() < 0 || (*iter).d(bts_mode).i() > 2)
+                                   d.l(Cbvalue().d(bts_warning, (string)"netfilter failed: Incorrect
mode").d(bts_code, c));
+                   }
+                   else
+                           d.l(Cbvalue().d(bts_warning, (string)"netfilter failed: Incorrect net or net
mask").d(bts_code, c));
+           }
+           const_cast<Cnetfilters*>(&m_seeding_netfilters)->sort();
```

```
+	const_cast<Cnetfilters*>(&m_leeching_netfilters)->sort();
+	read_rules();
+	if(has_seeding)
+		(&m_seeding_netfilters)->m_alter_netfilters_request = true;
+	if(has_leeching)
+		(&m_leeching_netfilters)->m_alter_netfilters_request = true;
+	return d;
+}
+
+Cbvalue Cserver::add_net_filter(const Cbvalue& s)
+{
+	Cbvalue d;
+	if (s.d_has(bts_net) && s.d_has(bts_net_mask)) {
+		//int net = (*((s.d("add net filter").l()).begin())).d("net").i();
+	    int net = s.d(bts_net).i();
+		int nmask = s.d(bts_net_mask).i();
+		Cnetfilter::t_rule rule = Cnetfilter::error;
+		if (s.d_has(bts_rule)) {
+			switch(s.d(bts_rule).i()) {
+				case 0: rule = Cnetfilter::allow;
+					break;
+				case 1: rule = Cnetfilter::deny;
+					break;
+			}
+		}
+		if(find_netfilter(s.d(bts_mode).i(),net,nmask))
+		{
+			d.l(Cbvalue().d(bts_error, (string)"Add net filter failed: The specified netfilter
already exists").d(bts_code, 2));
+			return d;
+		}
+		if(!s.d_has(bts_mode)) const_cast<Cbvalue*>(&s)-
>d(bts_mode,static_cast<int>(Cnetfilter::both));
+		if(rule == Cnetfilter::error || (static_cast<Cnetfilter::t_mode>(s.d(bts_mode).i())) ==
Cnetfilter::none || net > INT_MAX || net < 0 || nmask > 32 || nmask < 0)
+		{
+			d.l(Cbvalue().d(bts_error, (string)"Add net filter failed: Incorent
input").d(bts_code, 1));
+			return d;
+		}
+		Cnetfilter::t_mode mode = Cnetfilter::none;
+		switch(s.d(bts_mode).i()) {
+			case 0: mode = Cnetfilter::seeding;
+				seeding_netfilter(Cnetfilter(rule, mode, net, nmask));
+				break;
+			case 1: mode = Cnetfilter::leeching;
+				leeching_netfilter(Cnetfilter(rule, mode, net, nmask));
+				break;
+			case 2: mode = Cnetfilter::both;
+				seeding_netfilter(Cnetfilter(rule, Cnetfilter::both, net, nmask));
+				leeching_netfilter(Cnetfilter(rule, Cnetfilter::both, net, nmask));
+				break;
+		}
+		if(s.d(bts_mode).i() < 0 || s.d(bts_mode).i() > 2) {
+
+			d.l(Cbvalue().d(bts_error, (string)"Add netfilter failed: Incorrect
mode").d(bts_code, 1));
+			return d;
+		}
+		const_cast<Cnetfilters*>(&m_seeding_netfilters)->sort();
```

```
+                       const_cast<Cnetfilters*>(&m_leeching_netfilters)->sort();
+                       read_rules();
+                       if(s.d(bts_mode).i() == 0 || s.d(bts_mode).i() == 2)
+                               (&m_seeding_netfilters)->m_alter_netfilters_request = true;
+                       if(s.d(bts_mode).i() == 1 || s.d(bts_mode).i() == 2)
+                               (&m_leeching_netfilters)->m_alter_netfilters_request = true;
+               }
+               return d;
+}
+
+Cbvalue Cserver::remove_net_filter(const Cbvalue& s)
+{
+               Cbvalue d;
+               __int64 cb_err = 0;
+               if(s.d_has(bts_net) && s.d_has(bts_net_mask)) {
+                       int rem_net = 0, rem_net_mask = 0, rem_mode = 2;
+                       if(s.d_has(bts_mode)) rem_mode = s.d(bts_mode).i();
+                       rem_net = s.d(bts_net).i();
+                       rem_net_mask = s.d(bts_net_mask).i();
+                       if(cb_err = pop_netfilter(rem_mode, rem_net, rem_net_mask)) {
+                               d.l(Cbvalue().d(bts_error, (string)"Remove net filter failed: Cannot remove the
specified netfilter").d(bts_code, cb_err));
+                               return d;
+                       }
+                       read_rules();
+                       if(rem_mode == 0 || rem_mode == 2)
+                       (&m_seeding_netfilters)->m_alter_netfilters_request = true;
+                       if(rem_mode == 1 || rem_mode == 2)
+                               (&m_leeching_netfilters)->m_alter_netfilters_request = true;
+               }
+               return d;
+}
+
diff -u1 -r diffold/BT Test/server.h diffnew/BT Test/server.h
--- diffold/BT Test/server.h        2006-02-10 16:43:55.000000000 -0100
+++ diffnew/BT Test/server.h    2006-02-10 16:34:29.000000000 -0100
@@ -21,2 +21,3 @@
 #include "version_check_handler.h"
+#include "netfilter.h"

@@ -34,3 +35,5 @@

+               typedef vector<Cbvalue> t_list;
                Cbt_file* find_torrent(const string&);
+               Cnetfilter* find_netfilter(int mode, int net, int nmask);
                Cbvalue admin_request(const Cbvalue& s);
@@ -48,3 +51,6 @@
                int announce(const string& id);
-               int close(const string& id, bool erase = false);
+               Cbvalue close(const string& id, bool erase = false, bool erase_torrent = false);
+               Cbvalue net_filter(const Cbvalue& s);
+               Cbvalue add_net_filter(const Cbvalue& s);
+               Cbvalue remove_net_filter(const Cbvalue& s);
                int file_priority(const string&, int priority);
@@ -69,2 +75,3 @@
                string trackers_fname() const;
+               string netfilters_fname() const;
                string user_agent() const;
@@ -72,2 +79,4 @@
                void alert(const Calert&);
```

```
+         void seeding_netfilter(const Cnetfilter&);
+         void leeching_netfilter(const Cnetfilter&);
          void clean_scheduler();
@@ -87,2 +96,4 @@
          void send_stop_event(bool);
+         void net_filter_seeding(bool v);
+         void net_filter_leeching(bool v);
          void set_block_list(const Cxif_key&);
@@ -98,2 +109,4 @@
          void torrents_dir(const string&);
+         int pop_netfilter(int mode, int net, int nmask);
+         int pop_netfilter(int mode) const;
          void tracker_port(int);
@@ -106,2 +119,4 @@
          void user_agent(const string&);
+         void write_rules(const Cbvalue&);
+         void read_rules() const;

@@ -182,2 +197,12 @@
          }
+
+         Cbt_logger& event_logger()
+         {
+                  return m_event_logger;
+         }
+
+         Cbt_logger& netfilter_logger()
+         {
+                  return m_netfilter_logger;
+         }

@@ -257,3 +282,13 @@
          }
+
+         int net_filter_seeding() const
+         {
+                  return m_config.m_net_filter_seeding;
+         }

+       int net_filter_leeching() const
+         {
+                  return m_config.m_net_filter_leeching;
+         }
+
          int upload_rate() const
@@ -272,2 +307,4 @@
          Calerts m_alerts;
+         Cnetfilters m_seeding_netfilters;
+         Cnetfilters m_leeching_netfilters;
          Cblock_list m_block_list;
@@ -276,3 +313,5 @@
          t_links m_links;
+         Cbt_logger m_event_logger;
          Cbt_logger m_logger;
+         Cbt_logger m_netfilter_logger;
          Cbt_tracker_accounts m_tracker_accounts;
diff -u1 -r diffold/BT Test/stdafx.h diffnew/BT Test/stdafx.h
--- diffold/BT Test/stdafx.h        2006-02-10 16:43:55.000000000 -0100
+++ diffnew/BT Test/stdafx.h    2006-02-10 16:34:29.000000000 -0100
@@ -22,4 +22,6 @@
```

```
+//__FD_SETSIZE definition should be changed in the /usr/include/bits/typesizes.h also!
+#define __FD_SETSIZE 8096
 #ifdef WIN32
-#define FD_SETSIZE 1024
+#define FD_SETSIZE 8096

diff -u1 -r diffold/BT Test/version_check_handler.cpp diffnew/BT Test/version_check_handler.cpp
--- diffold/BT Test/version_check_handler.cpp    2006-02-10 16:43:55.000000000 -0100
+++ diffnew/BT Test/version_check_handler.cpp           2006-02-10 16:34:29.000000000 -0100
@@ -23,6 +23,25 @@
          if (status_code != 200)
+         {
+#ifdef EVENT
                   alert(Calert(Calert::error, "HTTP error: " + n(status_code)));
+#endif
+#ifdef EVENTLOG
+                  event_logger().event(Cbt_logger::error, "HTTP error: " + n(status_code));
+#endif
+         }
          m_version = atoi(get_message_body(response).c_str());
          if (m_version > m_server.version())
+         {
+#ifdef EVENT
                   alert(Calert(Calert::info, xbt_version2a(m_version) + " is now available!"));
+#endif
+#ifdef EVENTLOG
+                  event_logger().event(Cbt_logger::info, xbt_version2a(m_version) + " is now
available!");
+#endif
+         }
 }
+Cbt_logger& Cversion_check_handler::event_logger()
+{
+         return m_server.event_logger();
+}
+
diff -u1 -r diffold/BT Test/version_check_handler.h diffnew/BT Test/version_check_handler.h
--- diffold/BT Test/version_check_handler.h       2006-02-10 16:43:55.000000000 -0100
+++ diffnew/BT Test/version_check_handler.h   2006-02-10 16:34:29.000000000 -0100
@@ -8,2 +8,3 @@
 #include "http_response_handler.h"
+#include "bt_logger.h"

@@ -17,2 +18,4 @@
          Cversion_check_handler(Cserver&);
+         Cbt_logger& event_logger();
+
 private:
Only in diffnew/BT Test: xbt_client_backend
Only in diffnew/BT Test: xbt_client_backend.pid
diff -u1 -r diffold/misc/bt_logger.cpp diffnew/misc/bt_logger.cpp
--- diffold/misc/bt_logger.cpp     2006-02-10 16:44:06.000000000 -0100
+++ diffnew/misc/bt_logger.cpp 2006-02-10 16:44:57.000000000 -0100
@@ -58 +58,27 @@
 }
+
+void Cbt_logger::firewall(const string& ip, const string& info_hash, bool alter_mode_request, bool
alter_peer_request, bool alter_netfilters_request, int left, bool decision)
+{
```

```
+		m_os << time2a(time(NULL)) << "\tIP: " << ip << "\tinfo_hash: " << hex_encode(info_hash) <<
"\ttorrent mode: " << (left ? 'l':'s')  << "\t: " << (left ? "Default leeching rule":"Default seeding rule") << '\t'
<< (alter_mode_request ? "event: Changed mode\t" : "") << (alter_peer_request ? "event: New
connection\t" : "") << (alter_netfilters_request ? "event: Changed FW ruleset\t" : "") << "Decision: " <<
(decision ? 'A':'D') << endl;
+}
+
+void Cbt_logger::firewall(const string& ip, const string& info_hash, bool alter_mode_request, bool
alter_peer_request, bool alter_netfilters_request, int mode, int rule)
+{
+		m_os << time2a(time(NULL)) << "\tIP: " <<  ip << "\tinfo_hash: " << hex_encode(info_hash) <<
"\ttorrent mode: " << (mode ? (mode == 2 ?'b':'s'):'l') <<  "\trule_hit: " << (rule ? 'D':'A') << '\t' <<
(alter_mode_request ? "event: Changed Mode\t" : "") << (alter_peer_request ? "event: New
connection\t" : "") << (alter_netfilters_request ? "event: Changed FW ruleset\t" : "")  << "\tDecision: "
<< (rule ? 'D':'A') << endl;
+}
+
+void Cbt_logger::event(t_level level, const string& message)
+{
+		m_os << time2a(time(NULL)) << '\t' <<  level << "\t-\t" << message << endl;
+}
+
+void Cbt_logger::event(t_level level, const string& source, const string& message)
+{
+		m_os << time2a(time(NULL)) << '\t' <<  source << ":\t" << message << endl;
+}
+
+void Cbt_logger::print(int ip, int mask, int net, int rule)
+{
+	m_os << time2a(time(NULL)) << "\tIP:" <<  ip  << "\tmask:" << mask << "\tnet:" << net <<
"\trule:" << rule << endl;
+}
+
diff -u1 -r diffold/misc/bt_logger.h diffnew/misc/bt_logger.h
--- diffold/misc/bt_logger.h	2006-02-10 16:44:06.000000000 -0100
+++ diffnew/misc/bt_logger.h	2006-02-10 16:44:57.000000000 -0100
@@ -10,2 +10,14 @@
 public:
+	enum t_level
+	  {
+		emerg,
+		alert,
+		crit,
+		error,
+		warn,
+		notice,
+		info,
+		debug,
+	  };
+
		void choke(const string& file, const string& peer, bool remote, bool v);
@@ -17,2 +29,7 @@
		void valid(const string& file, bool remote, int piece);
+		void firewall(const string& ip, const string& info_hash, bool alter_mode_request, bool
alter_peer_request, bool alter_netfilters_request, int mode, int rule);
+		void firewall(const string& ip, const string& info_hash, bool alter_mode_request, bool
alter_peer_request, bool alter_netfilters_request, int left, bool decision);
+		void event(t_level level, const string& message);
+		void event(t_level level, const string& source, const string& message);
+		void print(int ip, int mask, int net, int rule);
```

```
	Cbt_logger();
diff -u1 -r diffold/misc/bt_misc.cpp diffnew/misc/bt_misc.cpp
--- diffold/misc/bt_misc.cpp	2006-02-10 16:44:06.000000000 -0100
+++ diffnew/misc/bt_misc.cpp	2006-02-10 16:44:57.000000000 -0100
@@ -2,4 +2,6 @@
 #include "bt_misc.h"
-
+#include <sys/vfs.h>
+#include "../BT Test/server.h"
 #ifndef WIN32
+#include <sys/statvfs.h>
 #include <sys/stat.h>
@@ -7,2 +9,6 @@
 #include "socket.h"
+#include <dirent.h>
+#include <sys/types.h>
+#include <sys/stat.h>
+#define bufsize 255

@@ -339,2 +345,44 @@

+int rmpath(const string& v)
+{
+	DIR *dirh;
+	struct dirent *dirp;
+	struct stat statbuf;
+	char pathname[bufsize];
+
+	if ((dirh = opendir(v.c_str())) == NULL)
+	{
+		//perror("opendir");
+		return 2;
+	}
+	for (dirp = readdir(dirh); dirp != NULL; dirp = readdir(dirh))
+	{
+		if (strcmp(".",dirp->d_name) == 0 || strcmp("..",dirp->d_name) == 0)
+			continue;
+		if (strcmp("lost+found",dirp->d_name) == 0)
+			continue;
+		sprintf(pathname,"%s/%s",v.c_str(),dirp->d_name);
+		if (lstat(const_cast<char*>(pathname),&statbuf) == -1)
+		{
+			//perror("stat");
+			continue;
+		}
+		//if (S_ISREG(statbuf.st_mode))
+		//	unlink(const_cast<char*>(pathname));
+		if (S_ISDIR(statbuf.st_mode))
+			if(rmdir(const_cast<char*>(pathname)))
+				rmpath(const_cast<char*>(pathname));
+		//if (S_ISLNK(statbuf.st_mode))
+		//{
+		//	//bzero(linkname,bufsize);
+		//	//readlink(pathname,linkname,bufsize);
+		//	unlink(const_cast<char*>(pathname));
+		//}
+		//printf("The mode of %s is %o\n\n",pathname,statbuf.st_mode & 07777);
+	}
+	closedir(dirh);
+	return rmdir(v.c_str());
```

```
+}
+
+
 int hms2i(int h, int m, int s)
@@ -349,2 +397,10 @@

+string space_left()
+{
+	struct statvfs64 tStatFs;
+	statvfs64("/store",&tStatFs);
+	//printf("Mbytes left[%d]\n",tStatFs.f_bavail*tStatFs.f_bsize/(1024*1024));
+	return n(tStatFs.f_bavail*tStatFs.f_bsize);
+}
+
 #ifdef BSD
diff -u1 -r diffold/misc/bt_misc.h diffnew/misc/bt_misc.h
--- diffold/misc/bt_misc.h		2006-02-10 16:44:06.000000000 -0100
+++ diffnew/misc/bt_misc.h	2006-02-10 16:44:57.000000000 -0100
@@ -23,2 +23,3 @@
 int mkpath(const string&);
+int rmpath(const string& v);
 string n(__int64);
@@ -35,2 +36,3 @@
 string xbt_version2a(int);
+string space_left();

diff -u1 -r diffold/misc/bt_strings.h diffnew/misc/bt_strings.h
--- diffold/misc/bt_strings.h	2006-02-10 16:44:06.000000000 -0100
+++ diffnew/misc/bt_strings.h	2006-02-10 16:44:57.000000000 -0100
@@ -39,2 +39,3 @@
 const string bts_announce_list = "announce-list";
+const string bts_level = "level";
 const string bts_close_torrent = "close torrent";
@@ -104,3 +105,19 @@
 const string bts_version = "version";
+const string bts_space_left = "space left";
 const string bts_wait_time = "access denied, wait time in effect";
+const string bts_net_filter = "net filter";
+const string bts_net = "net";
+const string bts_rule = "rule";
+const string bts_mode = "mode";
+const string bts_net_mask = "net mask";
+const string bts_remove_net_filter = "remove net filter";
+const string bts_add_net_filter = "add net filter";
+const string bts_clear_net_filters = "clear net filters";
+const string bts_net_filter_seeding = "net filter seeding";
+const string bts_net_filter_leeching = "net filter leeching";
+const string bts_remove_data = "remove data";
+const string bts_remove_torrent = "remove torrent";
+const string bts_error = "error";
+const string bts_warning = "warning";
+const string bts_code = "code";

diff -u1 -r diffold/misc/netfilter.cpp diffnew/misc/netfilter.cpp
--- diffold/misc/netfilter.cpp	2006-02-10 16:48:42.000000000 -0100
+++ diffnew/misc/netfilter.cpp	2006-02-10 16:44:57.000000000 -0100
@@ -0,0 +1,37 @@
+#include "stdafx.h"
+#include "netfilter.h"
+
```

```
+#include "bt_misc.h"
+
+#ifdef _DEBUG
+#undef THIS_FILE
+static char THIS_FILE[]=__FILE__;
+#define new DEBUG_NEW
+#endif
+
+//////////////////////////////////////////////////////////////////////
+// Construction/Destruction
+//////////////////////////////////////////////////////////////////////
+
+//int Cnetfilter::pre_dump() const
+//{
+//	return m_net.size() + m_nmask.size() + 1;
+//}
+
+Cnetfilters::Cnetfilters()
+{
+	m_alter_netfilters_request = true;
+}
+
+Cnetfilters::~Cnetfilters()
+{
+	m_alter_netfilters_request = false;
+}
+
+void Cnetfilter::dump(Cstream_writer& w) const
+{
+	w.write_int(4, m_rule);
+	w.write_int(4, m_mode);
+	w.write_int(4, m_net);
+	w.write_int(4, m_nmask);
+}
diff -u1 -r diffold/misc/netfilter.h diffnew/misc/netfilter.h
--- diffold/misc/netfilter.h	2006-02-10 16:48:46.000000000 -0100
+++ diffnew/misc/netfilter.h	2006-02-10 16:44:57.000000000 -0100
@@ -0,0 +1,131 @@
+#if !defined(AFX_IPLIST_H__FE59568B_F1B9_45F4_9148_369A1454BC33__INCLUDED_)
+#define AFX_IPLIST_H__FE59568B_F1B9_45F4_9148_369A1454BC33__INCLUDED_
+
+#if _MSC_VER > 1000
+#pragma once
+#endif // _MSC_VER > 1000
+
+#include <list>
+#include "stream_writer.h"
+
+class Cnetfilter
+{
+public:
+	enum t_rule
+	{
+		error = -1,
+		allow,
+		deny,
+	};
+
+	enum t_mode
+	{
```

```
+                   none = -1,
+                   seeding,
+                   leeching,
+                   both,
+         };
+
+         int nmask() const
+         {
+                   return m_nmask;
+         }
+
+         t_rule rule() const
+         {
+                   return m_rule;
+         }
+
+         t_mode mode() const
+         {
+                   return m_mode;
+         }
+
+         int net() const
+         {
+                   return m_net;
+         }
+
+         void nmask(int nmask) {
+                   m_nmask = nmask;
+         }
+
+       void mode(t_mode mode)
+         {
+                   m_mode = mode;
+         }
+
+         void net(int net)
+         {
+                   m_net = net;
+         }
+
+         Cnetfilter(t_rule r = Cnetfilter::allow, t_mode mode = Cnetfilter::seeding,  int net = 0, int nmask
= 0)
+         {
+                   m_rule = r;
+                   m_mode = mode;
+                   m_net = net;
+                   m_nmask = nmask;
+         }
+       /*Cnetfilter(int net, int nmask)
+       {
+             m_rule = Cnetfilter::allow;
+                   m_mode = Cnetfilter::seeding;
+             m_net = net;
+             m_nmask = nmask;
+       }*/
+
+       int operator==(const Cnetfilter& v) const
+         {
+                   if( this->m_net != v.m_net) return 0;
+                   if( this->m_nmask != v.m_nmask) return 0;
```

```
+                       return 1;
+               }
+
+               int operator<(const Cnetfilter& v) const
+               {
+                       if( this->m_nmask < v.m_nmask) return 0;
+                       return 1;
+               }
+
+
+
+               //int pre_dump() const;
+               void dump(Cstream_writer&) const;
+private:
+               int m_net;
+               int m_nmask;
+               t_mode m_mode; // 0,1,2
+               t_rule m_rule;
+};
+
+class Cnetfilters: public list<Cnetfilter>
+{
+public:
+
+               bool m_alter_netfilters_request;
+               Cnetfilters();
+               ~Cnetfilters();
+               void push_back(const value_type& v)
+               {
+                       list<value_type>::push_back(v);
+                       while (size() > 250)
+                               erase(begin());
+               }
+
+               void net_erase(const value_type& v)
+               {
+                       list<value_type>::remove(v);
+               }
+
+               void pop_back()
+               {
+                       list<value_type>::pop_back();
+               }
+               void net_sort(const value_type& v)
+               {
+                       list<value_type>::sort();
+               }
+};
+
+#endif // !defined(AFX_IPLIST_H__FE59568B_F1B9_45F4_9148_369A1454BC33__INCLUDED_)
```