|  |  |
| --- | --- |
|  | ВЕЛИКОТЪРНОВСКИ УНИВЕРСИТЕТ  “СВ. СВ. КИРИЛ И МЕТОДИЙ” |

Владимир Йорданов Йотов

Модели, базирани на йерархични композиции от пространства, за управление на софтуерни версии

Автореферат

на ДИСЕРТАЦИЯ

за присъждане на образователна и научна степен “Доктор”

по научна специалност: 01.01.12 “Информатика”

Научен ръководител: доц. д-р Хр. Тужаров

Велико Търново  
2012

# Въведение

## Актуалност

Управлението на версията на софтуерните продукти заема важно място в областта на софтуерното инженерство [22, 71, 74 – стр. 746, 90]. Въпреки наличието на разработени модели, научно-приложната област предоставя възможности за търсене на решения за постигане на по-висока ефективност на работния процес. Модерните гъвкави методологии предлагат един по-свободен начин на развитие на софтуерните продукти. Те предполагат използването на специалисти от много високо ниво, които познават разработвания продукт в детайли. Създаването и задържането на такива специалисти представлява предизвикателство пред ръководството на всяка една компания. Настоящата дисертация може да се разглежда като опит да се предостави възможност за снижаване на рисковете от използването на такива кадри и въвеждането на инструментариум за автоматизация при използването на гъвкавите методологии.

## Обект и методология на изследването

Обект на изследване в дисертацията са моделите и методите в управлението на версии чрез използването на йерархично композирани работни пространства за постигане на по-ефективен подход на нейното управление, ускоряване анализа на влиянието на промените над системата, усъвършенстване политиката на управление на знания в компаниите и инструмент за обсъждане финансовите аспекти на проектите.

Методологията на изследването включва следните подходи: евристичен анализ на предизвикателствата, стоящи пред съществуващите модели в научно-приложната област; сравнителен анализ на използваните модели и методи и определяне на нови идеи; търсене, изследване и ефективно развитие на модели и методи за управление на версия и повишаване ефективността на процеса на създаване и поддържане на софтуерните продукти.

## Цели и задачи дисертацията

**Цел:** Да се изследват, създадат и развият на модели за управление на софтуерни версии в среда, базирана на йерархично композирани на работни пространства.

Във връзка с основните цели се поставят следните задачи:

1. Да се създаде модел на версионизиран обект, осигуряващ максимална гъвкавост при определяне степента на гранулираност на данните в съчетание с простота и универсалност.
2. Да се създаде модел на среда с йерархично композирани работни пространства, както и да се определят правилата за управление на версия на обекти в тази среда.
3. Да се адаптира метод за проследимост на промени, базиран на събития, за среда с модел на йерархично композирани работни пространства.
4. Да се определи терминологията в областта на версионизирането с използването на йерархично композирани работни пространства.
5. Да се създаде методологична рамка за създаване на софтуерни продукти в среда с йерархично композирани работни пространства.
6. Да се докаже повишаване на ефективността при разработването на софтуерни продукти при разработените модели.

## Структура на дисертационния труд

Дисертацията се състои от увод, три глави, заключение, използвана литература, едно приложение и прототип.

**В Първа глава** е направен обзор на моделите в областта на управлението на версиите. Разгледани са мястото, целите и задачите на управлението на версии в рамките на разработването и поддържането на софтуерни продукти. Направен е обзор на съществуващите модели на версионизираните обекти и на начинът на тяхното съхраняване в репозиторито с версии. Отделно е направен обзор на темата за съвместната работа на сътрудниците, където е наблегнато на работните пространства като средство за осъществяването на кооперираност. За постигане пълнота на пълнота в обзора са сравнени методите за проследимост на промените. Главата завършва с определяне на изводите, формиране на целта и задачите на дисертацията, които следва да бъдат решени във втора и трета глава.

**Във Втора глава** са представени теоретичните модели за управление на версия в среда с йерархично композирани работни пространства. Моделите са допълнени с авторска методологична рамка за тяхното ефективно използване. Във формулираните в края на главата изводи са посочени предимствата на разработените модели.

**Трета глава** съдържа аналитичено обоснован избор на средства за реализиране на програмен прототип на система, реализираща теоретичните модели. Представени са описания на авторска алгоритмична реализация на по-важните моменти от прототипа. В главата е направена теоретично-експериментална сравнителна симулация на разработка на програмен продукт с и без използване на разработения прототип при гъвкава методология за разработване. Направените изводи в края на главата разкриват переспективите от използването на разработените модели.

В заключението е обобщение получените резултати. Формулирани са основните резултати в рамката на дисертацията. Посочени са някои актуални задачи, които могат да бъдат естествено продължение на настоящото изследване.

Разработката и апробацията на резултатите са извършени самостоятелно, като регулярно са представяни в катедра „Компютърни технологии” на Великотърновски университет „Св. св. Кирил и Методий”.

# Публикации, свързани с дисертационния труд

1. Jotov, V. An investigation on the approaches for version control systems. In Proceedings of the 9th international Conference on Computer Systems and Technologies and Workshop For PhD Students in Computing (Gabrovo, Bulgaria, June 12 - 13, 2008). B. Rachev and A. Smrikarov, Eds. CompSysTech '08, vol. 374. ACM, New York, NY, V.11-1. DOI= http://doi.acm.org/10.1145/1500879.1500959
2. Jotov, Vl., Transaction over Versioned Objects in Hierarchical Workspace Environment, ECAI09, 2009, Pitesti, Romania.
3. Jotov, Vl., Towards a model of versioning domain, 5th Central and Eastern European Software Engineering Conference in Russia, Moscow, 28-29 Oct. 2009, ISBN 978-1-4244-5665-9, http://dx.doi.org/10.1109/CEE-SECR.2009.5501147
4. Йотов, Вл., Модел на данните в система за контрол на версии, базирана на йерархични работни пространства, 6 – 7. 11. 2009г, Велико Търново.
5. Jotov, Vl., Adaptation of Event-Based Traceability Method for Environment with Hierarchal Composed Workspaces, In Proceedings of International Conference Automatics and Informatics’10, Sofia, 3 – 7 October 2010.

# Съдържание на дисертационния труд

## Глава 1. Управление на версията при създаването на софтуерни системи

В първа глава са разгледани съществуващите модели в областта на управлението на софтуерни версии. В резултат направеният обзор в се налагат следните изводи:

1. Системите за управление и контрол на версии представляват задължителен инфраструктурен инструмент в съвременното софтуерно производство. Може да се каже, че тези модели играят водеща роля в процеса на създаване на софтуерни продукти.
2. Експонирани и систематизирани са различни модели за представяне и съхраняване на версионизирани обекти. Изтъкнати са предимствата и недостатъците на разгледаните модели. Определена е липсата на адекватно ниво на поддръжка на версионизиране на различни нива на абстракция на системите. Тази необходимост може да се трансформира в изискването версионизираният обект да предоставя възможност за определяне на различно ниво на неговата степен на гранулираност.
3. Направен е анализ на различни подходи за съхраняване на версионизирани обекти. Подходът за съхраняване състояния на версионизираните обекти, предполага проста реализация и по-висока скорост на системата. Това го превръща в атрактивен кандидат за реализацията на прототипа.
4. Показани са предизвикателствата, стоящи пред съвместната работа над един продукт. Определена е възможността от научно изследване на моделите и механизмите, използвани в областта на йерархично композираните работни пространства [35 – стр. 406]. Установено е, че йерархично композираните работни пространства служат като инструмент за осигуряване на автономна работа. Това се допълва с възможността за коопериране на работата между участниците в процеса по създаване на софтуерни продукти.
5. Направен е анализ на темата за проследимост на промените. Представени са различните видове проследяващи връзки, както и методите за получаването им. Идентифицирана е липсата на инструменти, предоставящи адекватно ниво за създаване и управление на връзки на проследимост [112].

## Глава 2. Модели за управление на версии в среда с йерархична композиция на работни пространства

### 2.1. Модел на версионизиран обект

Водещите автори в областта на управлението и контрола на версиите [30, 84] определят версионизираните обекти като съставени от две части – състояния на обекта (версии) и граф на версиите. Под граф на версиите се разбира такъв граф, чиито върхове представляват отделните състояния (версии) на обекта, а ребрата съответстват на логическата последователност на създаване на версиите.

Първата нова характеристика, която следва да притежава един модел на версионизиран обект, е той да предоставя възможност да се определи нивото на детайлизираност, т.е. на гранулираност, която самият модел следва да поддържа.



Фиг. 26[[1]](#footnote-2) Пример за промяни на съставността на обектите

1. Съставен обект ще наричаме обект, който е съставен от други обекти (версии на обекти) посредством композиции.
2. Под композиция ще се разбира същността, определяща връзката между супер-обект и под-обект. Един съставен обект може да бъде супер-обект на една или повече композиции, т.е. да е съставен от един или повече под-обекти.

Същността **версионизиран обект** е необходимо да притежава само и единствено уникален и непроменяем номер, който е удачно да се използва и като първичен ключ за същността. Версиите на един обект може да се разглеждат като негови примитиви (**версионизирани примитиви)**, чиито основни атрибути са:

* Глобален номер на версията.
* Номер на версионизиран обект, с който дадената версия е свързан.
* Уникален номер на версия в рамките на обекта.
* Наименование на обекта. Така полученият модел става по-пълноценен, елиминирайки недостатъка, свързан с преименоването на обектите (файловете) при системи като CVS, SVN, Git, Metcury и др. [28, 40, 49, 61, 64, 75].
* Съдържание на обекта, включващо данните в съответната версията на обекта.

Версионизираният примитив се определя еднозначно посредством уникалната двойка **номер на версионизиран обект** и **номер на версия.**

За нуждите на версионизиране на съставни обекти, следва да се дефинира допълнителна същност - „Композиция на версионизирани примитиви” (накратко композиция), която еднозначно свързва версията на супер обекта с версиите на неговите под-обекти. Атрибутите на същността са: глобален номер на супер-обекта; глобален номер на под-обекта.

За нуждите на отчетността и проследимостта на промените, така създаденият модел следва да се разшири с цел да се осигури поддържка на функционал за граф на верссите. В ER моделите е прието графовата структура да се моделира от две същности – същност на възлите и същност на дъгите [86]. Тук дъгите на графа (показващи прехода от една верися в друга) следва да притежават следните атрибути: Номер на дъгата (първичен ключ); глобален номер на изходната версия; глобален номер на целевата версия; Потребител, извършил промяната; Дата и час на промяната; Допълнителни данни относно промяната.



Фиг. 27 ER модел на версионизиран обект

#### 2.1.1. Версионизиране на съставен версионизиран обект

Настоящата точка има за цел да представи особеностите при управлението на версия на съставни обекти от първи ред. Базирайки се на тях, ще се определи процесът на версионизиране на съставни обекти от ред N.

1. Съставен обект от ред **0** ще наричаме такъв обект, за който няма асоциирани под-обекти, т.е. фактически той не се явява съставен обект. Съставен обект от ред **N** ще наричаме такъв обект, за който най-големият ред на асоцииран под-обект е равен на **N-1**.
2. Един под-обект сам по себе си може да се явява съставен обект от други обекти, като по този начин да се създаде суперпозиция от съставни обекти.

Една от основните задачи, която стои пред настоящия научно-приложен труд, е да не се усложнява без необходимост тук създадените модели. Изхождайки от това, както и от факта на липсваща практическа необходимост, при построяването на суперпозиция от съставни обекти следва да въведем следващите ограничаващи правила:

1. В дадена суперпозиция от съставни обекти, обект може да присъства най-много един път.
2. Един обект може да присъства най-много в една суперпозиция от обекти.
3. Промяна на версията на даден под-обект за даден супер-обект, не влиае на версиите на другите под-обекти, съставящи същия супер-обект (Фиг. 32).



Фиг. 32 При промяна във версията на един под-обект, не се променя версията на съседните под-обекти

1. Версия на даден съставен обект е видима в дадено работно пространство само и единствено, когато всички версии на съставящите го под-обекти са видими в съответното работно пространство.

### 2.2. Йерархично композирани работни пространства. Модел на видимост на версионизирани обекти

#### 2.2.1. Модел на йерархично композирани работни пространства

1. Продукт се нарича обект на материалното или нематериалното производство, който след своето създаване може да бъде размножен и разпространяван сред клиентите.
2. Издание на продукт се нарича определена фиксирана негова версия, която е преминала определени количества проверки и отговаря на определени критерии за качество, безопасност и др. Само издания на продукта се разпространяват сред клиентите. Версии, които не представляват издание, се наричат в практиката работни версии.
3. Работно пространство се нарича място, където се извършват определени дейности по създаването на версия на продукт.
4. Главно работно пространство се нарича работно пространство, в което се извършва окончателната сборка и подготовка на издание на продукта.



Фиг. 33 Клас диаграма на модел продукт-издение-работно пространств



Фиг. 34 Примерна йерархична композиция на пространства

#### 2.2.2. Модел на видимост на версионизирани обекти в среда с йерархично композиране на работни пространства

Под видима версия на версионизиран обект ще се разбира версията или липсата на версия за обекта, с която потребителят (разработчикът) работи в рамките на определено пространство, което от своя страна се нарича локално работно пространство. Под локална версия на версионизиран обект за дадено работно пространство, ще се разбира такава негова версия, която е асоциирана с работното пространство.

Принципи на видимост:

1. Ако даден версионизиран обект има версия в рамките на дадено пространство (локална версия), то в това пространство се вижда само тази версия на обекта, въпреки наличието на други версии в родителските пространства.
2. Локалната версия на обект от дадено работно пространство се вижда рекурсивно във всички под-пространства, освен ако няма дефинирана друга локална версия в тях.

От изложените принципи можем да изведем следствията:

1. Във всяко работно пространство, където обектите нямат локална версия, те са представени с тяхна версия, намираща се в най-близкото родителско работно пространство.
2. Ако за дадено работно пространство обектът няма версия в нито едно родителско работно пространство, то той не се вижда в първоначално избраното работно пространство.



Фиг. 35 Разпределение на версиите на версионизиран обект съгласно принципите на видимост

### 2.3. Транзакции над версионизиран обекти

#### 2.3.1. Транзакции над версионизиран обект в рамките на едно работно пространство

Разгледани са следните транзакции: създаване на версионизиран обект; актуализиране на не локален версионизиран обект; създаване на маркер на състояние (state-mark) на версионизиран обект, маркер на състояние изтрит обект и отказ от маркер на състояние.

След създаването си обектите притежават първоначална (нулева) версия.

Създаването на маркер на състояние представлява транзакция, при която се създава нова версия на даден версионизиран обект.

Като обратна транзакция за създаване на състояние може да се квалифицира тази по отказ от маркер на състояние. Чрез нея в представения модел последното състояние се освобождава, а текущата локална версия на обекта става версията, предхождаща отказаната.

Актуализирането на нелокален версионизиран обект представлява сложна транзакция, която се състои от следните стъпки:

* Извличане на предишната видима за работното пространство версия на обекта.
* Създаване на локална версия на обекта в текущото работно пространство.
* Създаване на релация на версиите (дъга в графа на версиите).

Изтриването на даден обект е възможно чрез транзакция за създаване на т.нар. маркер за изтрит обект.

#### 2.3.2. Транзакции над версионизиран обект между две работни пространства

Транзакциите между две работни пространства може да се разделят на две групи – публикуване на версия на обект и отказ от локална версия. Преди да се разгледат е необходимо да се въведат термините „производна” и „паралелна” (непроизводна) версия на обект.

1. Нека разгледаме един версионизиран обект и две негови версии X и Y. Ако съществува път в графа на версии на обекта от версия Х до версия Y, то версия Y се явява производна версия на версия Х, а версия на Х – предшестваща версия Y.
2. Нека разгледаме един версионизиран обект и две негови версии X и Y. Ако не съществува път в графа на версиите за обекта от версия Х до версия Y, то двете версии се явяват паралелни или непроизводни.

Под публикуване на версия на обект ще се разбира поредицата от действия, необходими за привеждане локалната версия на обекта от текущото работно пространство в локална версия в родителското работно пространство.

Простото публикуване на версия е при ситуация, когато в родитилското работно пространство не съществува локална версия на публикувания обект.

При актуализиращото публикуване не се извършва сливане между двете версии, понеже производната версия представлява еволюционно продължение на предшестващата версия.За транзакцията на актуализиращо публикуване е характерно, че при нея е необходимо да бъдат изпълнени едновременно следните условия:

* В родителското работно пространство съществува локална версия на обекта, който се публикува.
* Версията на обекта, който се публикува, се явява производна на версията му в родителското работно пространство.

Когато версията на обекта, която се публикува в родителското работно пространство, се явява паралелна спрямо намиращата се там локална версия (Фиг. 38), тогава следва двете версии да се слеят. Като резултат на сливането се получава нова версия на обекта.

Транзакцията по отказ от локална версия се явява обратна на транзакциите по публикуване на версия. Тя включва само една стъпка: премахване на локалната версия на обекта от работното пространство. При премахването сработват механизмите от от модела на видимост на версионизираните обекти.



Фиг. 38 Публикуване със сливане

#### 2.3.3. Транзакции над съставни обекти

В ситуацията, когато имаме локална версия на обекта В в родителското работно пространство и негова видима версия в текущото работно пространство. В текущото работно пространство се създава под-обект А за обекта В (Фиг. 39). При публикуването версията на под-обекта А е възможно да не води до промяна във версията на обект В в родителското работно пространство. Въпреки това при последващо публикуване версията на обекта В заедно с неговите композиции, в родителското работно пространство ще доведе до автоматично обновяване (в рамките на работното пространство) на композиционната схема на обектите (Фиг. 39 – зелената пунктирана стрелка). Това е продиктувано от факта, че информацията относно организацията на съставния обект следва да се разглежда като неделима част от него.

При публикуване на новата версия на съставния обект B,v3 води до изискването това да се извърши в комплект с версията на новосъздадения под-обект (Фиг. 39 – стрелките с №2).



Фиг. Новосъздаен под-обект към супер-обект

1. Публикуването на версия на локален съставен обект следва да се извършва в комплект с всички локални версии на неговите под-обекти, които имат различна версия в родителското работно пространство (Фиг. 40 – зелената и жълтата стрелки с №2).



Фиг. Индиректна променена версия на супер-обект, породена от нова версия на под-обект

1. Публикуването на версия на обект, който притежава предишна версия, явяваща се под-обект на съставен обект в родителското работно пространство на текущото работно пространство, следва да се извършва едновременно с публикуването на локалната версия на съответния съставен обект.
2. Отказът от локална версия на съставен обект следва да се извършва заедно с рекурсивен отказ от локална версия на всички негови под-обекти.

#### 2.3.4. Класификация на транзакциите над версионизирани обекти

1. Транзакции в рамките на едно работно пространство
   1. Прости обекти
      1. Създаване на версионизиран обект
      2. Маркиране на версия
      3. Актуализация на не локален версионизиран обект
      4. Маркиране на изтрито състояние
   2. Съставни обекти
      1. Включване на обект в композицията на съставен обект.
      2. Автоматично регистриране на индиректна нова версия на съставен-обект, породена от нова версия на под-обект.
      3. Изваждане на под-обект от композицията на съставен обект
2. Транзакции между две работни пространства
   1. Прости обекти
      1. Просто публикуване
      2. Актуализиращо публикуване
      3. Публикуване със сливане
      4. Отказ от локална версия
   2. Съставни обекти
      1. Публикуване на съставен обект
      2. Публикуване на обект, изваден от композицията на съставен обект.
      3. Отказ от локална версия

#### 2.3.5. Жизнен цикъл на версионизиран обект

За описване на жизнен цикъл на версионизиран обект, са използвани представените в 2.3.4 транзакции. На Фиг. 43 е представен модел на жизнения цикъл на версионизиран обект чрез използванета на диаграма на поредица от събития (event-driven chain). Моделът включва следните етапи: създаване на версионизиран обект; създаване на версия на обект; сливане на версии на обект; маркиране на обект като изтрит; отказ от локална версия на обект; публикуване на версия на обект, означен като изтрит, в главно работно пространство.



Фиг. 43 Диаграма на състоянията на версионизиран обект

На са използвани следните означения: зелените кръгли елементи представляват стабилните състояния на обекта, а червените квадрати – транзакциите по промяна на обекта

### 2.4. Проследимост на промените в среда с йерархична композиция на работни пространства

В текущия параграф е направен опит за адаптиране и интегриране на метода в разглежданата среда с йерархична композиция на работни пространства. Моделирането на метода за проследяване на промените може да се разглежда като допълнително звено, свързващо модела на работните пространства с този на версинизиран обект, като се запази възможността за гъвкаво дефиниране нивото на гранулираност на обектите, с които се работи.

### 2.4.1. Работни единици и работни пространства

Под работната единица ще се използва разширение на определението, което дава Хелминг в своя труд [42] – Работна единица е работата, която следва да се извърши.

1. Работна единица се нарича съвкупността от дейности (работата), определена от дадена причина, която следва да се извърши.
2. Следствие, породено от причина, ще се нарича наборът от промени над обекти в резултат от изпълнението на работна единица.

Работните единици са основно средство за определяне и разпределение на задачите. При адаптацията на метода на проследимост, базиран на събития, са определени следните два етапа:

* Настройване на средата за генериране на проследяващи събития.
* Прихващане на събития за осъществена промяна над обект и създаване на проследяващи връзки.

Процесът по настройване на текущите работни единици включва следните стъпки:

1. Определяне на даден версионизиран обект като работна единица. Тази стъпка предполага да се извършва от мениджъра на задачите за съответното ниво на детайлизация на задачата, или от самия инициатор на задачите в рамките на проекта.
2. Подготвяне на работното пространсто за автоматично генериране на проследяващи връзки. Същността на стъпката се състои в активирането или деактивирането на работните единици към работното пространство. В рамките на тази стъпка потребителят избира по кои работни единици възнамерява да работи. Стъпката следва да се извършва от съответния участник в процеса по създаване на софтуерният продукт.

След изпълнението на втората стъпка системата е способна автоматично да прихваща събития по промяна на обектите и да създава проследяващи връзки (причинно-следствени връзки).

### 2.4.2. Модели на данните на система за управление на версията чрез йерархични пространства



Фиг. 46 ER модел на данните

### 2.5. Методологична рамка за създаване и поддържане на софтуерни продукти чрез използване на йерахично композирани работни простанства

В параграфа е представена методологична рамка, която включва следните елементи: подготовка на средата; създаване и определяне на задачите; изпълнение на задачите; публикуване на изпълнените задачи и сглобяване на крайния продукт.

Под подготовка на средата следва да се разбира процесът на определяне йерархичната архитектура от работни пространства, който следва определянето на методологията и подхода на разработване. При създаване на софтуерни продукти важен етап е този на определяне на изискванията.

В рамките на представената тук методология изискванията следва да се създадат под формата на версионизирани обекти. Това позволява да се проследи тяхното изменение, да се сравнят две техни версии, да се върнем към по-стара версия, както и да се намали риска от изгубване на знания.

Под изпълнение на задачи следва да се разбира същинският процес на създаване на софтуерния продукт. Резултатът от изпълнението на една задача може да представлява последваща задача, която разглежда първоначалната в по-големи детайли, с по-голяма прецизност. Така например създаването на архитектура на софтуерния продукт, както и на тестовите сценарии, може да се разглежда като задачи, продиктувани от изискванията, чиито краен резултат представлява задача съответно за разработването на продукта, така и за провеждането на тестовете, гарантиращи качеството на крайния продукт.

Публикуването следва да се разглежда като средство за интегриране на отделните компоненти на продукта. От модела на видимост на обектите следва, че публикуването на обект в по-горно работно пространство води неговата видимост в сестринските работни пространства. Именно публикуването представлява механизъм за споделяне обектите, съответно и на сглобяване на крайната версия на продукта. Когато едно изискване се одобри, т.е. по него е достигнат консенсус между учасниците в проекта, то може да се публикува в главното работно пространство и то става видимо за всички участници в проекта.

В параграфа са разгледани процесите на създаване на нова функционалност, а също така и този на промяна на съществуваща функционалност чрез използване на разработените модели посредством методологичната рамка

### 2.6. Изводи

От направените теоретични разработки в настоящата глава могат да бъдат направени следните изводи:

1. Предложен е гъвкав модел на версионизиран обект, предоставящ възможността свободно да се определи нивото на гранулираност на обектите. Това предполага неговата приложимост при различни практически задачи, за които е необходимо по-гъвкаво ниво на абстракции, отколкото може да се достигне при използването на файлове. Теоретичният модел е докладван на международната конференция „Central & Eastern European Software Engineering Conference”, Москва (2009). Практическият ER модел е представен в Научна конференция с международно участие "25 Години Педагогически Факултет", Велико Търново (2009).
2. Разработен е модел на среда с йерархично композиране на работни пространства, включващ модел на данните за тази среда. Определени са теоретичните правила, както и самите транзакции. Използвайки модела на транзакциите, е направен опит за създаване на модел на жизнен цикъл на версионизиран обект. Получените резултати бяха докладвани на международната конференция „Electronics, Computers and Artificial Intelligence” в Питещи, Румъния (2010).
3. Предложена е адаптация на метод за проследимост на промените, базиран на събития. При адаптацията на метода се използва композираността на пространствата и на обектите като механизъм за разбиване на големи задачи на по-малки и тяхното решаване. Това позволява в пълнота да се обхванат обектите и връзките между тях в процеса на създаване на софтуерни продукти. Резултатите от изследването са докладвани на международната конференция „Автоматика и информатика’10” в София (2010).
4. Предложена е българска адаптация на терминологията в областта на управлението на версии. Съвместно с адаптацията в настоящото научно-приложно изследване са представени нови научни термини и понятия в областта на управлението на версии, чрез използването на йерархично композирани работни пространства, като са въведени 2 принципа, 12 дефиниции, 4 правила и 5 следствия.
5. Представена е методологична рамка за използване на тук разработените модели и методи в практиката. Методологичната рамка е разработена във формата на стандартни работни процеси. Разгледани са конкретни теоретични примери на използването им в процесите на създаване и на поддръжка на софтуерни продукти..

## Глава 3. Изследване приложимостта на моделите

### 3.1. Възможности за реализиране на моделите

В параграфа са сравнени възможностите за реализация, използвайки различни технологии и платформи. От направеният анализ е избрана платформата J2EE (Java 2 Enterprice Edition), като платформа за реализация слоя на бизнес логиката. При изборът на технологиите за реализация са избрани: FSF и библиотеката RickFaces за изграждане на визуалната част на стистемата; JPA и библиотеката OpenJPA като ORM инструмент и средство за достъп до данните. Системата Oracle e избрана като платформа с най-добри възможности за слоя на данните.

### 3.2. Разработка на прототип на система за управление на версии

В параграфа е представен работният процес при разработката на прототипа, както и използваните инструменти: Eclipse IDE, Ant и Hudson. Разгледан е архитектурният модел на прототипа – Фиг. 56, организацията на класовете, както и навигационният модел на прототипа – Фиг. 57.

Алгоритмичната реализация на прототипа



Фиг. 56 Архитектура на система-прототип Versia

В параграфа важно място заема пункта за алгоритмичната реализация на прототипа. В него са представени алгоритмите за създаване на нова версия на обект, този за публикуване на версионизиран обект, и не на последно място – реализацията на регистъра на отворените работни пространства. Алгоритмите са допълнени с диаграми на последователностите.



Фиг. 57 Навигационна диаграма на прототипа

### 3.3. Примерни модели за композиране на версионизирани обекти

В параграфа се демонстрират възможностите на ER модела на версионизиран обект, като се акцентира на подобренията в сравнение с файловия модел за версионизиране [68, 69]. В резултат на извършения анализ на недостатъците на файловия модел, тук са представени модели на версионизирани обекти, който предоставя възможности за свободно определяне на степента на неговата гранулираност. Това предоставя възможност за значително подобряване на версионизирането както в областта на обектно-ориентираното програмиране, управлението качеството на софтуера и управлението на изискванията, така и на свързаността между елементите.

В параграфа са представени два модела на декомпозиране на версионизирани обекти – модел на на същността „клас” (Фиг. 61) и модел на обектите в областта на тестирането (Фиг. 62).



Фиг. 61 Примерен модел на композиране на версионизирани обекти за същността клас



Фиг. 62 Примерен модел на композиране на версионизирани обекти за областта на тестрането

### 3.4. Сравнителен анализ на предимствата на моделите в прототипа

В представеният параграф е направен опит за експериментално-теоретично доказване предимствата от използване на разработените модели. За тази цел

...

## Заключение

...

## Използвана литература

[1] Андрейчин, Л., Л. Георгиев, Ст. Илчев, Н. Костов, Ив. Леков, Ст. Стойков, Цв. Тодоров, Д. Попов, Български тълковен речник, Наука и изкуство, София, 2008.

[2] Белладжио Д., Т. Миллиган, Разработка програмного обеспечения: управление изменениями, ДМК Пресс, Москва, 2009,.

[3] Екел Брус, Да мислим на JAVA, том 1, СофтПрес, София, 2001.

[4] Коуберн Ал. Каждому проекту своя методология, 2005, http://www.citforum.ru/SE/project/meth\_per\_project/ (посетен през март 2011)

[5] Наков, Св., и колектив, Въведение в програмирането с Java, Фабер, Велико Търново, 2008.

[6] Наков, Св., и колектив, Въведение в програмирането със C#, Фабер, Велико Търново, 2011.

[7] Наков, Св., Интернет програмиране с Java, Фабер, Велико Търново, 2004, ISBN 954-775-305-3.

[8] Манева Н., А. Ескенази, Софтуерни Технологии, ИК Анубис, София, 2001.

[9] Фейсон Т., Borland C++ Обектно-ориентирано програмиране - Част І, Нисофт, София, 1994.

[10] Хемраджани, А., Гибкая разработка приложений на Java с помощю Spring, Hibernate и Eclipse, ООО "И.Д.Вилямс", Москва, 2008.

[11] Ambler, S. W., Pr. J. Sadalage,Refactoring Databases: Evolutionary Database Design, Addison Wesley Professional, 2006.

[12] Ammann, P., J. Offutt, Introduction to software testing, Cambridge University Press, 2008.

[13] Amza, C., Cox, A. L., and Zwaenepoel, W. Distributed versioning: consistent replication for scaling back-end databases of dynamic content web sites. In Proceedings of Middleware '03: Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware (Rio de Janeiro, Brazil, June 16 - 20, 2003), pp. 282–304, Springer-Verlag New York, Inc., New York, 2003.

[14] Andrews, J., Feature: No More Free BitKeeper, http://kerneltrap.org/node/4966, 2005, (посетен през януари 2012).

[15] Apache OpenJPA Project, http://openjpa.apache.org/ (посетен през март 2012).

[16] Arvin, Tr.,Comparison of different SQL implementations, 2011, http://troels.arvin.dk/db/rdbms/ (посетен през март 2012)

[17] Asun ion, H. U. Towards practical software traceability. In Companion of the 30th international Conference on Software Engineering (Leipzig, Germany, May 10 - 18, 2008) pp.1023-1026, ICSE Companion '08. ACM, New York, NY, 2008, DOI= http://doi.acm.org/10.1145/1370175. 1370228.

[18] Azad, A. Implementing Electronic Document and Record Management Systems. Auerbach Publications, Books24x7, 2008 http://common.books24x7.com/book/id\_26435/book.asp (посетен през март 2012)

[19] Barcis, R., JavaServer Faces (JSF) vs Struts, 2004, http://websphere.sys-con.com/node/46516 (посетен през декември 2011)

[20] Binkley, D., Horwitz, S., and Reps, T. Program integration for languages with procedure calls. ACM Trans. Softw. Eng. Methodol., Vol.4, no. 1, January 1995, pp. 3-35, 1995, DOI= http://doi.acm.org/10.1145/201055. 201056

[21] Boehm, B., A. Egyed, Kwan J., Port D., Shah A., Madach R., Using the WinWin Spiral Model: Case Study, Computer, Vol. 31, no. 7, pp. 33–44, July 1998.

[22] Bourque, P., R. Dupuis, A. Abran, J.W.Moore, L. Tripp, The Guide to the Software Engineering Boody of Knowledge, 1999, IEEE Software Vol. 16, no. 6, November/December 1999.

[23] Brown, A., Dart, S., Feiler, P., Wallnau, K.. The state of automated configuration management. Tech. Rep. CMU/SEI-ATR-91, Software Engineering Inst., Carnegie Mellon Univ., Pittsburgh, Pa., September 1991.

[24] Buffa, M. and Gandon, F. 2006. SweetWiki: semantic web enabled technologies in Wiki. In Proceedings of the 2006 international Symposium on Wikis (Odense, Denmark, August 21 - 23, 2006), pp 69–78, WikiSym '06. ACM, New York, 2006, DOI= http://doi.acm.org/10.1145/1149453.1149469.

[25] Cleland-Huang, J. 2005. Toward improved traceability of non-functional requirements. In Proceedings of the 3rd international Workshop on Traceability in Emerging Forms of Software Engineering (Long Beach, California, November 08 - 08, 2005), pp.14-19, TEFSE '05. ACM, New York, 2005, DOI= http://doi.acm.org/10.1145/1107656.1107660

[26] Cleland-Huang, J., Settimi, R., BenKhadra, O., Berezhanskaya, E., and Christina, S. 2005. Goal-centric traceability for managing non-functional requirements. In Proceedings of the 27th international Conference on Software Engineering (St. Louis, MO, USA, May 15 - 21, 2005), pp. 362–371, ICSE '05. ACM, New York, 2005, DOI= http://doi.acm.org/10.1145/1062455.1062525

[27] Clemm, G., Amsden, J., Ellison, T., Kaler, C., and Whitehead, J. Versioning Extensions to WebDAV (Web Distributed Authoring and Versioning). RFC. RFC Editor, 2002.

[28] Collins-Sussman B., Fitzpatrick, B. W., Pilato C. M., Version Control with Subversion, book compiled from Revision 10945, 2008, http://svnbook.red-bean.com/en/1.0/index.html (посетен през март 2009)

[29] Converse, T., Joyce P., PHP Bible, 2nd Edition, Wiley Publishing, 2002.

[30] Conradi, R. and Westfechtel, B. 1998. Version models for software configuration management. ACM Comput. Surv., Vol. 30, no. 2, pp. 232–282, June 1998, DOI= http://doi.acm.org/10.1145/280277.280280.

[31] Cruz, José R.C. , Branching Out with Git, MacTech Magazine, Vol. 26, no. 04, http://www.mactech.com/articles/mactech/Vol.26/26.04/ BranchingOutWithGit/index.html (посетен през януари 2012).

[32] Dart, S. 1991. Concepts in configuration management systems. In Proceedings of the 3rd international Workshop on Software Configuration Management (Trondheim, Norway, June 12 - 14, 1991), pp. 1-18, P. H. Feiler, Ed. ACM, New York, 1991, DOI= http://doi.acm.org/10.1145/111062.111063.

[33] Doray, A., Beginning Apache Struts: From Novice to Professional, Apress, 2006.

[34] Drupal - Open Source CMS, http://drupal.org/ (посетен през януари 2012).

[35] Estublier, J., Leblang, D., Hoek, A., Conradi, R., Clemm, G., Tichy, W., and Wiborg-Weber, D., Impact of software engineering research on the practice of software configuration management. ACM Trans. Softw. Eng. Methodol., Vol. 14, no. 4, pp. 383-430, October 2005, DOI= http://doi.acm.org/ 10.1145/1101815.1101817.

[36] Estublier, J. Software configuration management: a roadmap. In Proceedings of the Conference on the Future of Software Engineering (Limerick, Ireland, June 04 - 11, 2000), pp. 279-289, ICSE '00. ACM Press, New York, 2000, DOI= http://doi.acm.org/10.1145/336512.336576

[37] Feiler, P. P., Configuration Management Models in Commercial Environments, Technical Report, CMU/SEI-91-TR-007, 1991.

[38] Feiler, P. P. Software process support through software configuration management. In Proceedings of the 5th international Software Process Workshop on Experience with Software Process Models (Kennebunkport, Maine, United States, October 10 - 13, 1989), pp. 58-60, International Software Process Workshop. IEEE Computer Society Press, Los Alamitos, CA, 1990.

[39] Ferguson, Jeff, C# Bible, John Wiley & Sons, 2002.

[40] Git - Fast Version Control System, http://git-scm.com/ (посетен през януари 2012).

[41] Gilmore, W. J., Beginning PHP and MySQL: From Novice to Professional, Apress, 2010.

[42] Helming, J., Koegel, M., and Naughton, H. 2009. Towards traceability from project management to system models. In Proceedings of the 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering - Volume 00 (May 18 - 18, 2009), pp. 11-15, International Conference on Software Engineering. IEEE Computer Society, Washington, DC, 2009, DOI= http://dx.doi.org/10.1109/TEFSE.2009.5069576.

[43] Hillyer, M., Managing Hierarchical Data in MySQL, 2005, http://dev.mysql.com/tech-resources/articles/hierarchical-data.html (посетен през май 2010).

[44] Horwitz, S., J. Prins, and Th. Reps.. Integrating noninterfering versions of programs. ACM Trans. Program. Lang. Syst. Vol. 11, no. 3, pp. 345-387, July 1989, http://doi.acm.org/10.1145/65979.65980.

[45] Hudson Continuous Integration, http://hudson-ci.org/ (посетен през януари 2012).

[46] IIBA, A Giude to the Business Analysis Body of Knowledge (Version 2.0) BABOK, 2009.

[47] Jain, P. et al. , J2EE Professional Projects, Premier Press, 2002.

[48] Jarke, M. 1998. Requirements tracing. Commun. ACM, Vol. 41, no. 12, pp. 32-36. December 1998, DOI= http://doi.acm.org/10.1145/290133. 290145.

[49] Jones, M. T., Version control for Linux, 2006, http://www.ibm.com/ developerworks/linux/library/l-vercon/, (посетен през февруари 2009).

[50] Joomla!, http://www.joomla.org/ (посетен през февруари 2012).

[51] Katz, M., Practical RichFaces, Apress, 2008.

[52] Katz, R. H., A database approach for managing VLSI design data. In Proceedings of the 19th Conference on Design Automation Annual ACM IEEE Design Automation Conference, pp. 274-282, IEEE Press, Piscataway, NJ, 1982 (G279)

[53] Klimmer M.,The Mega Project Mandate, Transforming Government, pp. 25-32, 2008.

[54] Kögel, M., Towards software configuration management for unified models. In Proceedings of the 2008 international Workshop on Comparison and Versioning of Software Models (Leipzig, Germany, May 17 - 17, 2008), pp. 19-24., CVSM '08. ACM, New York, 2008, DOI= http://doi.acm.org/10.1145/1370152.1370158

[55] Kögel, M., Helming, J., and Seyboth, S., Operation-based conflict detection and resolution. In Proceedings of the 2009 ICSE Workshop on Comparison and Versioning of Software Models (May 17 - 17, 2009), pp. 43-48, International Conference on Software Engineering. IEEE Computer Society, Washington, DC, 2009, DOI= http://dx.doi.org/10.1109/CVSM.2009.5071721 (G361)

[56] Krahn, R., Ingalls, D., Hirschfeld, R., Lincke, J., and Palacz, K., Lively Wiki a development environment for creating and sharing active web content. In Proceedings of the 5th international Symposium on Wikis and Open Collaboration (Orlando, Florida, October 25 - 27, 2009), pp. 1-10,WikiSym '09. ACM, New York, NY, 2009, DOI= http://doi.acm.org/10.1145/1641309.1641324

[57] Lee, B. G., Chang, K. H., and Narayanan, N. H., An integrated approach to version control management in computer supported collaborative writing. In Proceedings of the 36th Annual Southeast Regional Conference ACM-SE 36, pp. 34-43, ACM Press, New York, NY, 1998, DOI= http://doi.acm.org/10.1145/275295.275302

[58] Linwood, J., Minter, D., Pro Hibernate 3, 2008.

[59] Mann, Kito D., Interview with Manfred Geiler, 2004 http://www.jsfcentral.com/articles/geiler-04-04.html (посетен през януари 2012).

[60] Marcus, A. and Maletic, J. I.. Recovering documentation-to-source-code traceability links using latent semantic indexing. In Proceedings of the 25th international Conference on Software Engineering (Portland, Oregon, May 03 - 10, 2003), pp. 125-135, International Conference on Software Engineering. IEEE Computer Society, Washington, DC, 2003.

[61] Mercurial SCM, http://mercurial.selenic.com/ (посетен през януари 2012).

[62] Microsoft Sharepoint, http://sharepoint.microsoft.com/ (посетен през юни 2010).

[63] Morris, J. C., DistriWiki:: a distributed peer-to-peer wiki network. In Proceedings of the 2007 international Symposium on Wikis (Montreal, Quebec, Canada, October 21 - 25, 2007), pp. 69-74, WikiSym '07. ACM, New York, NY, 2007 DOI= http://doi.acm.org/10.1145/1296951.1296959

[64] Morse, T., CVS, Linux J., vol. 1996, no. 21, page 3, January 1996.

[65] Morse, T., CVS: Version Control Beyond RCS, Linux Journal, Jan 01, 1996, http://www.linuxjournal.com/article/1118 (посетен през юни 2011).

[66] Munson, J. P. and Dewan, P., A flexible object merging framework. In Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work (Chapel Hill, North Carolina, United States, October 22 - 26, 1994), pp. 231-242, CSCW '94. ACM, New York, NY, 1994, DOI= http://doi.acm.org/10.1145/192844.193016

[67] Nakov, Sv., 05. Source Control Systems, 2011, http://kaltura.ludost.net/p/101/sp/10100/download/entry\_id/0\_vqc9ye7z/relocate/05.%20Source%20Control%20Systems%20(Source).mp4 (посетен през януари 2012).

[68] Nguyen, T. N., Munson, E. V., Boyland, J. T., and Thao, C. 2005. An infrastructure for development of object-oriented, multi-level configuration management services. In Proceedings of the 27th international Conference on Software Engineering (St. Louis, MO, USA, May 15 - 21, 2005), pp. 215-224, ICSE '05. ACM, New York, 2005, DOI= http://doi.acm.org/10.1145/1062455. 1062504

[69] Nguyen, T. N., Munson, E. V., and Boyland, J. T. 2004. Object-oriented, structural software configuration management. In Companion To the 19th Annual ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages, and Applications (Vancouver, BC, CANADA, October 24 - 28, 2004), pp. 35-36, OOPSLA '04. ACM, New York, 2004, DOI= http://doi.acm.org/10.1145/1028664.1028684

[70] Nitin, K. L., Ananya S., Mahalakshmi K., and S. Sangeetha, iBATIS, Hibernate, and JPA: Which is right for you?,JavaWorld.com, 07/15/08, http://www.javaworld.com/javaworld/jw-07-2008/jw-07-orm-comparison.html (посетен през юли 2010).

[71] O’Donovan, Brian, RCS Handbook, 1992, http://www.burlingtontelecom.net/~ashawley/rcs/rcs\_handbook.html (посетен през март 2011).

[72] Official RCS Homepage, http://www.cs.purdue.edu/homes/ trinkle/RCS/ (посетен през март 2011).

[73] Posner, J., Block, J., CASEVision™/ClearCase Concepts Guide, 1994, *http://techpubs.sgi.com/library/dynaweb\_docs/0620/SGI\_Developer/* books/ClrC\_CG/sgi\_html/index.html (посетен през септември 2011).

[74] Pressman, R. S., Software Engineering: A Practitioner's Approach, McGraw-Hill Professional, 2005.

[75] Price, Derek R., CVS—concurrent versions system v1.11.22, http://ximbiot.com/cvs/manual/cvs-1.11.22/cvs.html, 2006 (посетен през април 2009).

[76] Puntikov, N. I., Volodin, M. A., and Kolesnikov, A. A.. AVCS: the APL version control system. In Proceedings of the international Conference on Applied Programming Languages (San Antonio, Texas, United States, June 04 - 08, 1995), pp. 154-161, M. Griffiths and D. Whitehouse, Eds. APL '95. ACM Press, New York, 1995, DOI= http://doi.acm.org/10.1145/206913.206995

[77] Ramesh, B., Matthias J., Towards Reference Models for Requirements Traceability, 1999.

[78] Rochkind, M. J., The Source Code Control System. In IEEE Transactions on Software Engineering SE, Vol. 1, no. 4, pp. 364–370, December 1975.

[79] Royce W.W., Managing the Development of Large Software Systems, Proceedings of IEEE WESCON, pp. 328-338, 1970.

[80] Ruparelia, N. B. 2010. The history of version control. SIGSOFT Softw. Eng. Notes, Vol. 35, no. 1, pp.5-9, January 2010, DOI= http://doi.acm.org/10.1145/1668862.1668876

[81] Schmetzer, J., Introduction to Ant, http://www.exubero.com/ant/ antintro-s5.html (посетен през декември 2011).

[82] Schmidt, M., Wenzel, S., Kehrer, T., and Kelter, U. History-based merging of models. In Proceedings of the 2009 ICSE Workshop on Comparison and Versioning of Software Models (May 17 - 17, 2009), pp. 13-18. International Conference on Software Engineering. IEEE Computer Society, Washington, 2009, DOI= http://dx.doi.org/10.1109/CVSM.2009.5071716

[83] Schwaber, K., Agile Project Management with Scrum, Microsoft Press, 2004.

[84] Slein, J. A., Vitali, F., Whitehead, E. J., and Durand, D. G. 1997. Requirements for distributed authoring and versioning on the World Wide Web. StandardView, Vol. 5, no. 1, pp. 17-24, March 1997, DOI= http://doi.acm.org/10.1145/253452.253474

[85] Silva, M., Gedye, D., Katz, R., and Newton, R.. Protection and versioning for OCT. In Proceedings of the 26th ACM/IEEE Conference on Design Automation (Las Vegas, Nevada, United States, June 25 - 28, 1989), pp. 264-269, DAC '89. ACM, New York, 1989. DOI= http://doi.acm.org/10.1145/74382.74427

[86] Stephens, S. M. , Johan Rung , Xavier Lopez, X.: Graph data representation in oracle database 10g: Case studies in Life science, IEEE Data Eng. Bull, vol. 27, pages 61-67, 2004.

[87] SUN SHIPS JDK 1.1 -- JAVABEANS INCLUDED, http://web.archive.org/web/20060706192215/http://www.sun.com/smi/Press/sunflash/1997-02/sunflash.970219.0001.xml (посетен през март 2012).

[88] Sun WorkShop TeamWare User's Guide, http://docs.sun.com/source/806-3573/TeamWareTOC.html (посетен през ‎юли ‎2010).

[89] Sun WorkShop TeamWare http://caligari.dartmouth.edu/doc/ solaris-forte/tw-help/TWHelp.html (посетен през ‎юли ‎2010).

[90] The 1998 ACM Computing Classification System, http://www.acm.org/about/class/1998 (посетен през ‎юли ‎2009).

[91] The Java Community Process, JSR 19: Enterprise JavaBeansTM 2.0, 2002, http://www.jcp.org/en/jsr/detail?id=19 (посетен през март 2012).

[92] The Java Community Process, JSR 220: Enterprise JavaBeansTM 3.0, 2006, http://www.jcp.org/en/jsr/detail?id=220 (посетен през март 2012).

[93] The JSF Matrix (reborn) http://www.jsfmatrix.net/ (посетен през март 2012).

[94] Tiako, P. F., Lindquist, T., and Gruhn, V,. Process support for distributed team-based software development workshop. SIGSOFT Softw. Eng. Notes, Vol 26, no. 6, pp. 31-33, November 2001, DOI= http://doi.acm.org/10.1145/505532.505539

[95] Tichy, W. F., RCS—a system for version control. Softw. Pract. Exper., Vol. 15, no. 7, pp. 637-654. July 1985, DOI= http://dx.doi.org/10.1002/spe.4380150703 (G148)

[96] Terrence R., MSSql vs MySQL vs Oracle, Stored Procedures, and Code Generation, 2007, http://www.numtopia.com/terry/blog/archives/2007/11/ mssql\_vs\_mysql\_vs\_oracle\_stored\_procedures\_and\_cod.cfm (посетен през декември 2011).

[97] Troelsen, A., Pro C# 2010 and the .NET 4 Platform, Fourth Edition, Apress, 2010.

[98] Turvey, St., Duelling databases: Four apps tested, ZDNet.com.au , 2005,December 23rd, http://www.zdnet.com.au/duelling-databases-four-apps-tested-139226455.htm (посетен през януари 2012).

[99] WebDAV Resources, http://www.webdav.org/ (посетен през март 2010).

[100] Weiss, C., Premraj, R., Zimmermann, T., and Zeller, A., How Long Will It Take to Fix This Bug?, In Proceedings of the Fourth international Workshop on Mining Software Repositories (May 20 - 26, 2007), p. 1, International Conference on Software Engineering. IEEE Computer Society, Washington, DC, 2007, http://dx.doi.org/10.1109/MSR.2007.13

[101] Westfechtel, B., Structure-oriented merging of revisions of software documents. In Proceedings of the 3rd international Workshop on Software Configuration Management (Trondheim, Norway, June 12 - 14, 1991), pp. 68-79, P. H. Feiler, Ed. ACM, New York, NY, 1991, DOI= http://doi.acm.org/10.1145/111062.111071

[102] Whitehead, E. J., Design spaces for link and structure versioning. In Proceedings of the Twelfth ACM Conference on Hypertext and Hypermedia (Århus, none, Denmark, August 14 - 18, 2001), pp. 195-204, HYPERTEXT '01. ACM Press, New York, NY, 2001, DOI= http://doi.acm.org/10.1145/504216.504265

[103] Wiegers, Karl E., Software Requirements, Second Edition, Microsoft Press, 2003.

[104] Wikipedia contributors, "Comparison of web application frameworks," Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/w/index.php?title=Comparison\_of\_web\_application\_frameworks&oldid=465118984 (посетен през декември 2011).

[105] Wikipedia contributors, "Comparison of relational database management systems", http://en.wikipedia.org/w/index.php?title= Comparison\_of\_relational\_database\_management\_systems&oldid=465424149, (посетен през декември 2011).

[106] Wikipedia contributors, “Document management system”, Wikipedia, The Free Encyclopedia, http://en.wikipedia.org/w/index.php? title=Document\_management\_system&oldid=358652743 (посетен през април 2010).

[107] Wikipedia contributors, "Java Server Daces", http://en.wikipwdia.org/w/index.php?title=JavaServer\_Faces&oldid=45747157 (посетен през декември 2011).

[108] Winter, V., Harvey S. , Mansour Z. , Prasanna R. A., Aspect traceability through invertible weaving, In Early Aspects Workshop at AOSD’06, 2006.

[109] WordPress › Blog Tool, Publishing Platform, and CMS, http://wordpress.org/ (посетен през януари 2012).

[110] Wu, Q., Pu, C., and Irani, D., Cosmos: a Wiki data management system. In Proceedings of the 5th international Symposium on Wikis and Open Collaboration (Orlando, Florida, October 25 - 27, 2009), pp. 1-2, WikiSym '09. ACM, New York, 2009,DOI=http://doi.acm.org/10.1145/1641309.1641343

[111] Zeller, A., Snelting, G., Unified versioning through feature logic. ACM Trans. Softw. Eng. Methodol., vol. 6, no. 4, pp. 398-441, October 1997, DOI= http://doi.acm.org/10.1145/261640.261654

[112] Zhang, Y. , Witte R., Rilling J., Haarslev V., An Ontology-based Approach for Traceability Recovery, 3rd International Workshop on Metamodels, Schemas, Grammars, and Ontologies for Reverse Engineering (ATEM 2006), pp. 36-43, 2006.

1. Номерацията на фигурите в автореферата е представена така, както е в самата дисертация. [↑](#footnote-ref-2)