ВЕЛИКОТЪРНОВСКИ УНИВЕРСИТЕТ
“СВ. СВ. КИРИЛ И МЕТОДИЙ”

ФАКУЛТЕТ “МАТЕМАТИКА И ИНФОРМАТИКА”
КАТЕДРА „КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ”

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”

Професионално направление: 4.6 “Информатика и компютърни науки”

Научен ръководител: **доц. д-р Христо Тужаров**

Велико Търново
2013

Дисертационният труд е обсъден и насочен за защита на заседание на катедра "Компютърни системи и технологии" към Факултет „Математика и информатика" на ВТУ"Св. св. Кирил и Методий".

Публичната защита на дисертационния труд ще се състои на заседание на научно жури на …… г., от …… часа, в зала …… на ВТУ "Св. св. Кирил и Методий". Материалите по защитата са на разположение на интересуващите се в Централната университетска библиотека на ВТУ "Св. св. Кирил и Методий".

ВЪВЕДЕНИЕ

АКТУАЛНОСТ

Управлението на версиите на софтуерните продукти заема важно място в областта на софтуерното инженерство. 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. Настоящата дисертация е опит да се увеличи нивото на автоматизация на дейностите в софтуерното производство чрез подобряване модела на данните, чрез използване на йерархично композирани работни пространства, чрез подобряване на проследимостта на промените и др.

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 при управление на версиите;
- ускоряване анализа на влиянието на промените над системата;
- усъвършенстване политиката на управление на знанията в компаниите;
- подходящ инструментариум, подпомагащ обсъждането на 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 модели за управление на софтуерни версии в среда, базирана на йерархично композирани 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 версии 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 изработи методологична рамка за създаване на софтуерни продукти в среда с йерархично композирани работни пространства.
6. Да се усъвършенства дейността при създаване на софтуерни продукти в следствие на използване на разработените модели.

СТРУКТУРА НА ДИСЕРТАЦИОННИЯ ТРУД

Дисертацията се състои от увод, три глави, заключение, използвана литература, три приложения, сред които прототип на DVD.

В Първа глава е направен преглед 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 начина на тяхното съхраняване в репозиторито с версии. Разгледана е темата за съвместната работа на сътрудниците, където е наблегнато на работните пространства като средство за осъществяване на кооперираност. За постигане 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 аналитично 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. Направените изводи в края на главата разкриват перспективите от използването на разработените модели.

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

Разработката на прототипа е извършена самостоятелно. Апробацията е извършена на семинари на катедра „Компютърни системи и технологии" на Великотърновски университет „Св. св. Кирил и Методий", научни конференции от национален и международен мащаб.

СЪДЪРЖАНИЕ НА ДИСЕРТАЦИОННИЯ ТРУД

ГЛАВА 1. УПРАВЛЕНИЕ НА ВЕРСИИТЕ ПРИ СЪЗДАВАНЕТО НА СОФТУЕРНИ СИСТЕМИ

В първа глава са разгледани съществуващите модели в областта на управлението на софтуерни версии. В резултат на направения обзор 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 системи 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 на състояния на версионизираните обекти предполага проста реализация и по-висока скорост на системата. Това го превръща в подходящ за използване при създаването на модела на версионаризиран обект в частта съхраняване.

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. Установено е, че йерархично композираните работни пространства служат като инструмент, който, от една страна, осигурява автономна работа, а от друга, дава възможност за кооперирane на работата между участниците в процеса по създаване на софтуерни продукти. Моделите на йерархично композиране на пространства притежават добър потенциал за по-нататъшно развитие.

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.

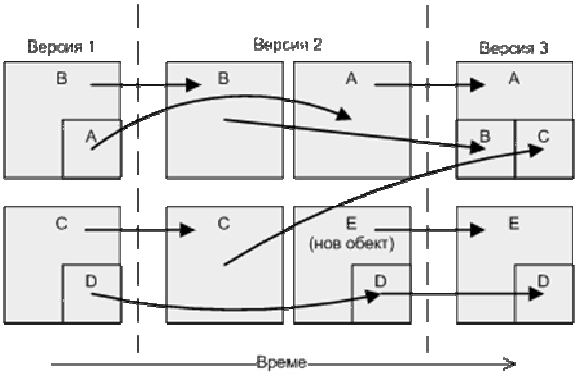
ГЛАВА 2. МОДЕЛИ ЗА УПРАВЛЕНИЕ НА ВЕРСИИ В СРЕДА С ЙЕРАРХИЧНА КОМПОЗИЦИЯ НА РАБОТНИ ПРОСТРАНСТВА

2.1. Модел на версионизиран обект

Водещите автори (Конради и Вестфехтел) в областта на управлението на версиите определят версионизираните обекти като съставени от две части – състояния на обекта (версии) и граф на версиите. Граф на версиите е такъв граф, чиито върхове са отделните състояния (версии) на обекта, а ребрата – логическата последователност на създаване на версиите.

Основна характеристика на модела на версионизиран обект е той да предоставя възможност да се определи нивото на детайлизираност (гранулираност) на данните. Това предполага необходимостта от композиране на обектите, а също така и от въвеждането на следната дефиниция:

Дефиниция 1. *Съставен обект* ще наричаме обект, който е съставен от други обекти посредством композиции.
Дефиниция 2. Под *композиция* ще се разбира същността, определяща връзката между *суперобект* и *подобект*. Един съставен обект може да бъде *суперобект* на една или повече композиции, т.е. да е съставен от един или повече *подобекти*.



Фиг. 1 Пример за промени на съставността на обектите

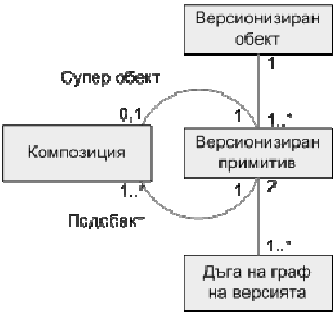
Същността **версионизиран обект** е необходимо да притежава само и единствено уникален и непроменяем номер, който е удачно да се използва и като първичен ключ за същността. Версиите на един обект може да се разглеждат като негови примитиви (**версионизирани примитиви)**, чиито основни атрибути са:

- Глобален номер на версията.
- Номер на версионизиран обект, с който дадената версия е свързана.
- Уникален номер на версия в рамките на обекта.
- Наименуване на обекта. Така полученият модел става по-пълноценен, елиминирайки недостатъка, свързан с преименуването на обектите (файловете) при системи като CVS, SVN, Git, Metcury и др.
- Съдържание на обекта – 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 *суперобекта* с версиите на неговите *подобекти*. Атрибутите на същността са: глобален номер на *суперобекта*; глобален номер на *подобекта.*

За нуждите на отчетността и проследимостта на промените моделът следва да се разшири и да включи граф на версиите. В ER моделите е прието графовата структура да се моделира от две същности – същност на възлите и същност на дъгите. 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. 2 ER модел на версионизиран обект

2.1.1. Версионизиране на съставен обект

Тук са представени 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.

Дефиниция 3. Съставен обект от ред **0**, т.е. прост обект, ще наричаме такъв обект, за който няма асоциирани *подобекти*. Съставен обект от ред **N** ще наричаме такъв обект, за който най-големият ред на асоцииран *подобект* е равен на **N-1**.

$$R_{\text{обект}} = \begin{cases} 0, & \sum \text{под} - \text{обекти} = 0 \\ N, \max\left(R_{\text{под}-\text{обект}}\right) = N-1 \end{cases}$$

Степен на гранулираност на обект ще се нарича редът на обекта.

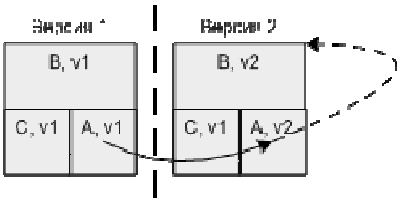
Следствие 1. Един *подобект* сам по себе си може да се явява съставен обект от други обекти, като по този начин да се създаде композиция от съставни обекти.

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

Правило 1. В дадена композиция от съставни обекти, обект може да присъства най-много един път.

Правило 2. Един обект може да присъства най-много в една композиция от обекти.

Следствие 2. Промяна на версията на даден *подобект* за даден *суперобект*, не влияе на версиите на другите *подобекти*, съставящи същия *суперобект* (Фиг. 3).



Фиг. 3 При промяна във версията на един подобект не се променя версията на съседните подобекти

Следствие 3. Версия на даден съставен обект е видима в дадено работно пространство само и единствено, когато всички версии на съставящите го *подобекти* са видими в съответното работно пространство.

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

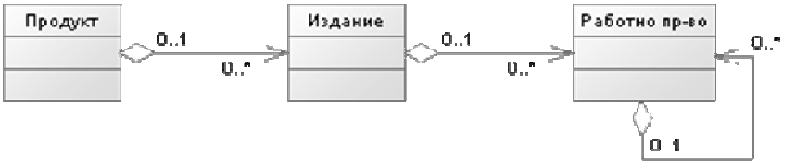
2.2.1. Модел на йерархично композирани работни пространства

Дефиниция 4. Продукт се нарича обект на материалното или нематериалното производство, който след своето създаване може да бъде размножен и разпространяван сред клиентите.

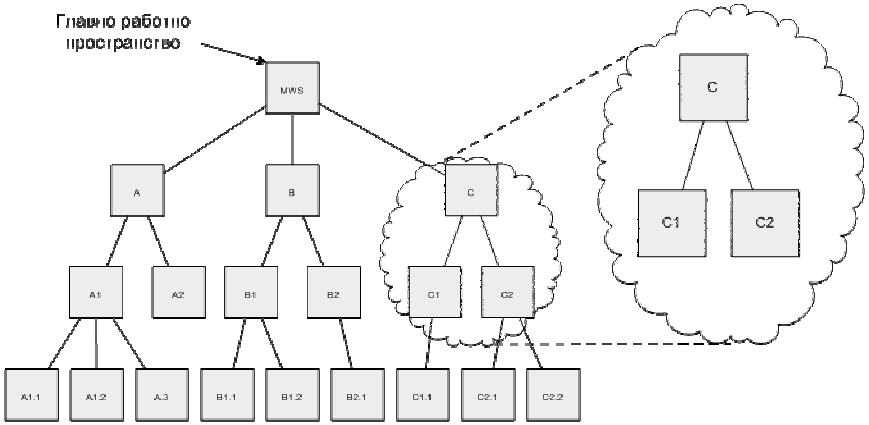
Дефиниция 5. Издание на продукт се нарича определена фиксирана негова версия, която е преминала определени количества проверки и отговаря на определени критерии за качество, безопасност и др. Само издания на продукта се разпространяват сред клиентите. Версии, които не представляват издание, се наричат в практиката работни версии.

Дефиниция 6. Работно пространство се нарича място, където се извършват определени дейности по създаването на версия на продукт.

Дефиниция 7. Главно работно пространство се нарича работно пространство, в което се извършва окончателната комплектация и подготовка на издание на продукта.



Фиг. 4 Клас диаграма на модел продукт-издание-работно пространств



Фиг. 5 Примерна йерархична композиция на пространства

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

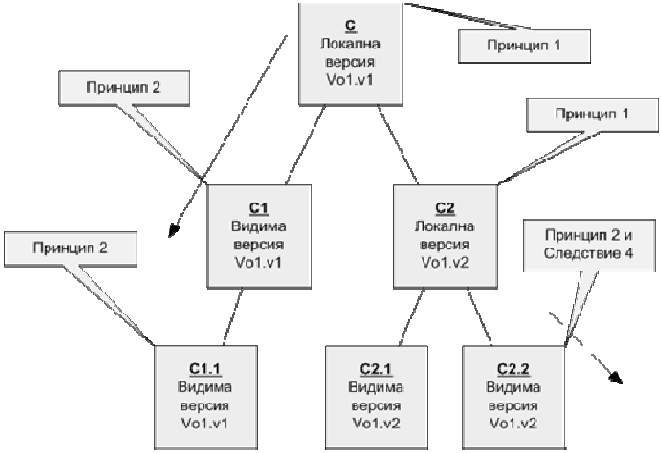
Дефиниция 8. *Локална версия на версионизиран обект* е версия, която е асоциирана с конкретно работно пространство.
Дефиниция 9. Под *видима версия на версионизиран обект* за дадено работно пространство се разбира такава версия на обекта, с която потребителят може да работи.

За разпространение на версиите на не-локалните обекти са въведени следните принципи на видимост:

Принцип. 1. *Локалната версия на версионизиран обект* за дадено работно пространств се явява *видима версия* на този обект в същото работно пространство, въпреки наличието на други локални версии в родителските пространства.
Принцип. 2. *Локалната версия* на обект от дадено работно пространство се вижда рекурсивно във всички дъщерни пространства, освен ако няма друга локална версия в тях.

От изложените принципи може да се изведат следствията:

Следствие 4. Във всяко работно пространство, където обектите нямат локална версия, те са представени с тяхна версия, намираща се в най-близкото родителско работно пространство.
Следствие 5. 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. 6 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 нелокален 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.

Дефиниция 10. Нека разгледаме един версионизиран обект и две негови версии X и Y. Ако съществува път в графа на версии на обекта от версия X до версия Y, то версия Y се нарича *производна версия* на версия X, а версия на X – предшестваща версия Y.

Дефиниция 11. Нека разгледаме един версионизиран обект и две негови версии X и Y. Ако не съществува път в графа на версиите за обекта от версия X до версия Y, то двете версии се наричат *паралелни* или *непроизводни версии*.

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.

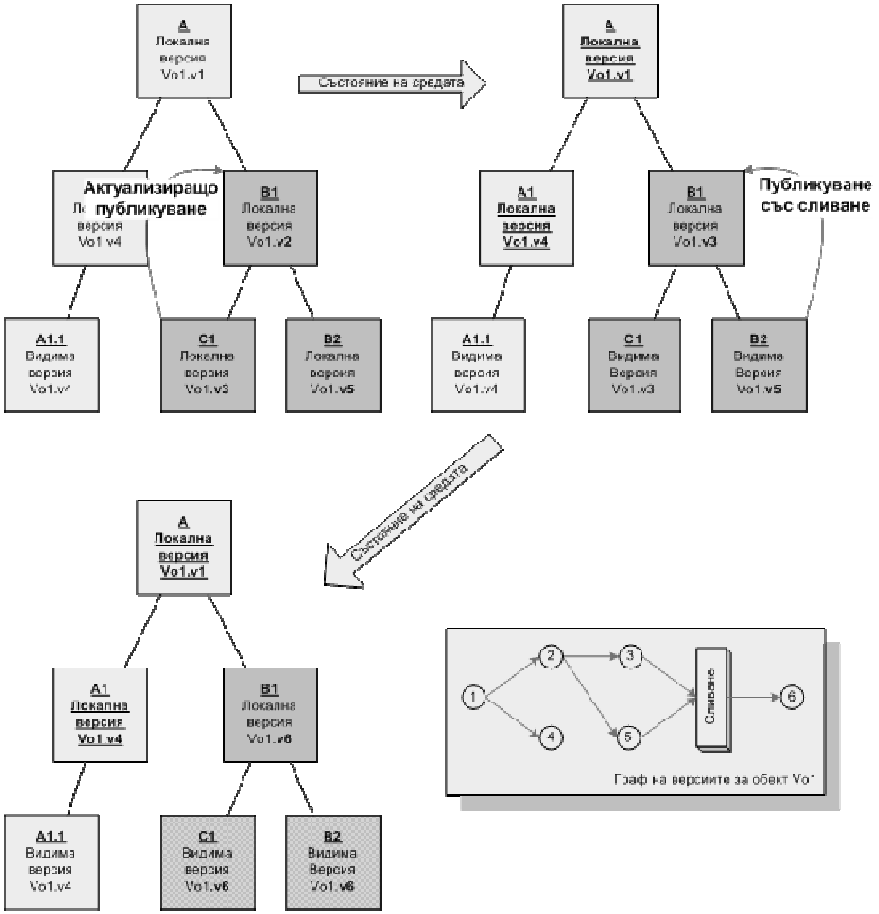
Простото публикуване на версия е възможно, когато в родителското работно пространство не съществува локална версия на публикувания обект.

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. 7), 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 **Следствие 4** от модела на видимост на версионизираните обекти.



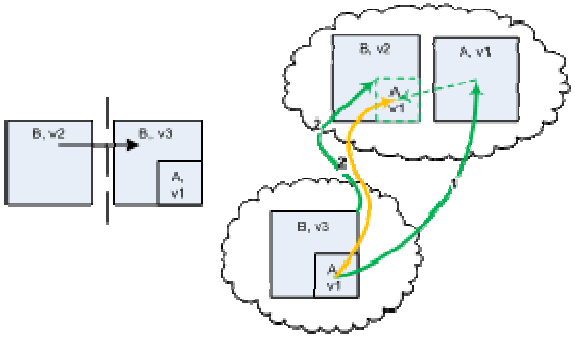
Фиг. 7 Публикуване със сливане

2.3.3. Транзакции над съставни обекти

Нека имаме следната ситуация: локална версия на обекта **B** в родителското работно пространство и негова видима версия в текущото работно пространство. В текущото работно пространство се създава *подобект* **A** за обекта **B** (Фиг. 8). При публикуването версията

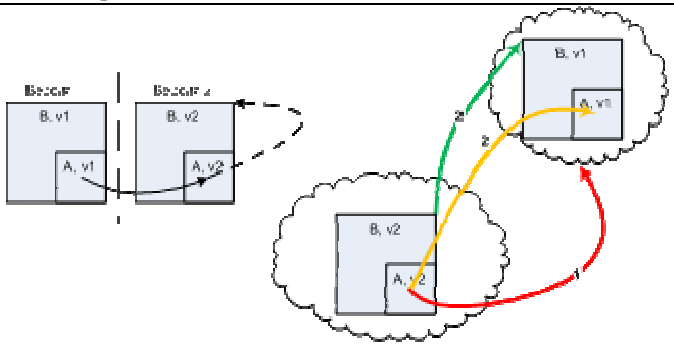
на *подобекта* **A** е възможно да не води до промяна във версията на обект **B** в родителското работно пространство. Въпреки това при последващо публикуване версията на обекта **B** заедно с неговите композиции, в родителското работно пространство ще доведе до автоматично обновяване (в рамките на работното пространство) на композиционната схема на обектите (Фиг. 8 – 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 *подобект* (Фиг. 8 – стрелките с №2).



Фиг. 8 Новосъздаен подобект към суперобект

Правило 3. Публикуването на версия на локален съставен обект следва да се извършва в комплект с всички локални версии на неговите *подобекти*, които имат различна версия в родителското работно пространство (Фиг. 9 – зелената и жълтата стрелки с №2).



Фиг. 9 Индиректна променена версия на суперобект, породена от нова версия на подобект

Правило 4. Публикуването на версия на обект, който притежава предишна версия, явяваща се *подобект* 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.
Правило 5. Отказът от локална версия на съставен обект следва да се извършва заедно с рекурсивен отказ от локална версия на всички негови *подобекти*.

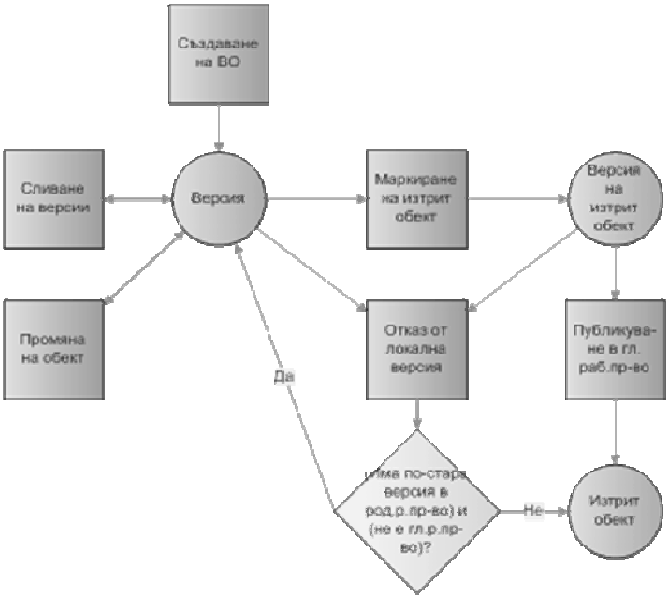
2.3.4. Класификация на транзакциите над версионизирани обекти

В дисертацията е направена класификация на транзакциите над версионизирани обекти, която има следния вид:

1. Транзакции в рамките на едно работно пространство.
   - 1.1. Прости обекти.
     - 1.1.1. Създаване на версионизиран обект.
     - 1.1.2. Маркиране на версия.
     - 1.1.3. Актуализация на *нелокален* версионизиран обект.
     - 1.1.4. Маркиране на изтрито състояние.
   - 1.2. Съставни обекти.
     - 1.2.1. Включване на обект в композицията на съставен обект.
     - 1.2.2. Автоматично регистриране на индиректна нова версия на съставен обект, породена от нова версия на *подобект*.
     - 1.2.3. Изваждане на *подобект* от композицията на съставен обект.
2. Транзакции между две работни пространства.
   - 2.1. Прости обекти.
     - 2.1.1. Просто публикуване.
     - 2.1.2. Актуализиращо публикуване.
     - 2.1.3. Публикуване със сливане.
     - 2.1.4. Отказ от локална версия.
   - 2.2. Съставни обекти.
     - 2.2.1. Публикуване на съставен обект.
     - 2.2.2. Публикуване на обект, изваден от композицията на съставен обект.
     - 2.2.3. 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. 10 е представен модел на жизнения цикъл на версионизиран обект чрез използването на диаграма на поредица от събития (event-driven chain). Моделът включва следните етапи: създаване на версионизиран обект; създаване на версия на обект (промяна 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. 10 Диаграма на жизнен цикъл на версионизиран обект

На Фиг. 10 са използвани следните означения: кръглите елементи представляват състояния на обекта в неговият жизнен цикъл, а квадратите – транзакциите по промяна на обекта

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

В текущия параграф е направено адаптиране на метод за проследимост на промени, базиран на събития, за разработените модели на среда с йерархична композиция на работни пространства. Адаптацията представлява допълнение, свързващо модела на работните пространства с този на версионизиран обект.

2.4.1. Работни задачи и работни пространства

Под *работна задача (work item)* ще се използва разширение на определението, което дава Хелминг в своя труд[1] – Работна задача е работата, която следва да се извърши.

Дефиниция 12. Работна задача се нарича съвкупността от дейности, която следва да се извърши.
Дефиниция 13. Следствие, породено от причина, ще се нарича наборът от промени над обекти в резултат от изпълнението на работна задача.

Работните задачи са основно средство за определяне и разпределение на работата между членовете на екипа. 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.

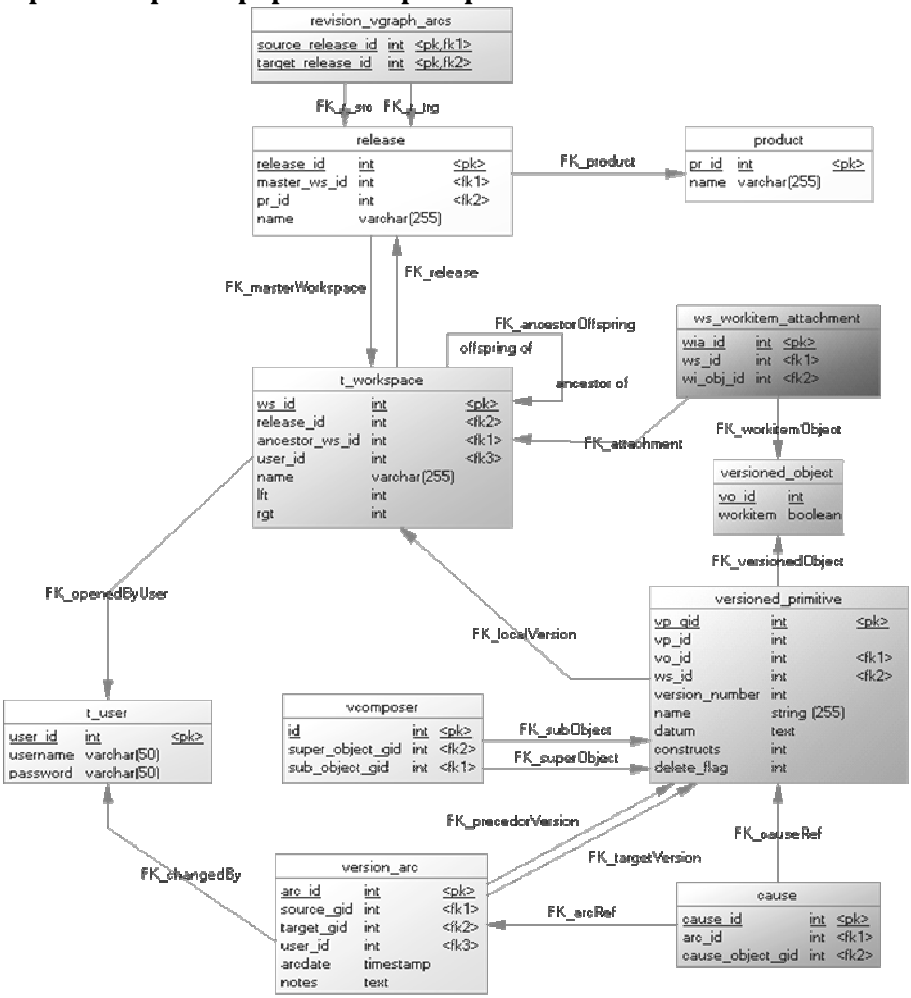
Процесът по настройване на средата представлява представлява ръчен процесс, който се състои от следните две стъпки:

1. Определяне на даден версионизиран обект като работна задача. 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 асоциирането (активирането) на необходимите работните задачи към работното пространство. В рамките на тази стъпка потребителят избира по кои работни задачи възнамерява да работи. Стъпката следва да се извършва от съответния участник в процеса по създаване на софтуерния 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).

[1] 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, 2009,

2.4.2. Модел на данните на система за управление на версията чрез йерархични пространства



Фиг. 11 ER модел на данните

2.5. Методологична рамка за използване на разработените модели

В параграфа е представена методологична рамка, която включва следните елементи: подготовка на средата; създаване и определяне на изискванията към 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, която съответства на избраната методология и подход на разработване.

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 установи причината за тяхното изменение, да се върнем към по-стара версия, както и да се намали рискът от изгубване на знания.

Под изпълнение на задачите 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 до неговата видимост в сестринските работни пространства. Именно публикуването представлява механизъм за споделяне на обектите, съответно и на сглобяване на крайната версия на продукта. Когато едно изискване се одобри, т.е. по него е достигнат консенсус между участниците в проекта, то може да се публикува в главното работно пространство, като става видимо за всички участници в проекта.

В параграфа са разгледани процесите на създаване на нова функционалност, а също така и процесите 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 определя 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). Концептуалният 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 транзакциите над версионизирани обекти. Използвайки модела на транзакциите, е направен опит за създаване на модел на жизнен цикъл на версионизиран обект. Получените резултати са 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 версиите, чрез използването на йерархично композирани работни пространства. Въведени 2 принципа, 13 дефиниции, 5 правила и 5 следствия. В Приложение 2 е представена онтологична диаграма на понятията в предметната област.

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.

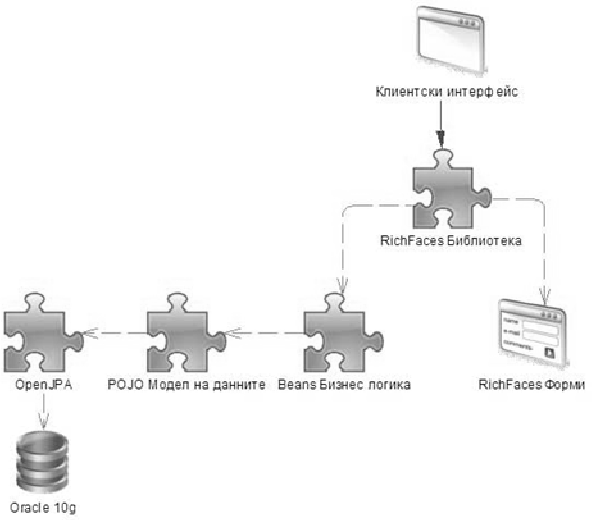
ГЛАВА 3. ИЗСЛЕДВАНЕ НА ПРИЛОЖИМОСТТА НА МОДЕЛИТЕ

3.1. Възможности за реализиране на моделите

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

3.2. Разработка на прототип, реализиращ представените модели

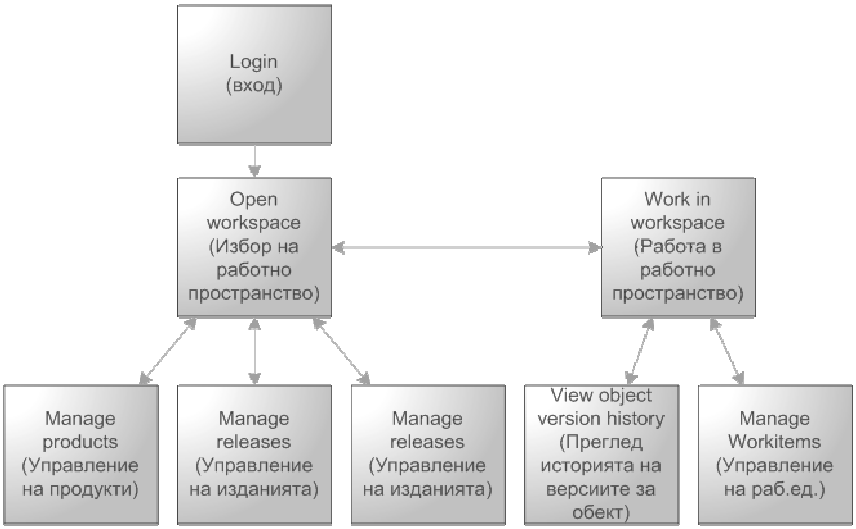
Тук е представен работният процес на разработка на прототипа, както и използваните инструменти: Eclipse IDE, Ant и Hudson. Разгледан е архитектурният модел на прототипа – Фиг. 12, организацията на класовете.



Фиг. 12 Архитектура на система-прототип Versia

Представен е навигационен модел, който помага да се разбере начинът на работа в системата, както и пътят за достигането до определена функционална точка.

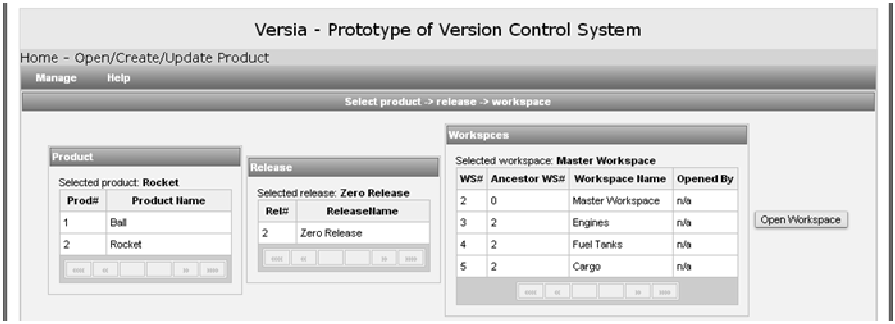
След като се избере и се отвори дадено работно пространство, системата зарежда екранът на работното пространство - Фиг. 15. Системата предоставя информация относно разположението на версиите



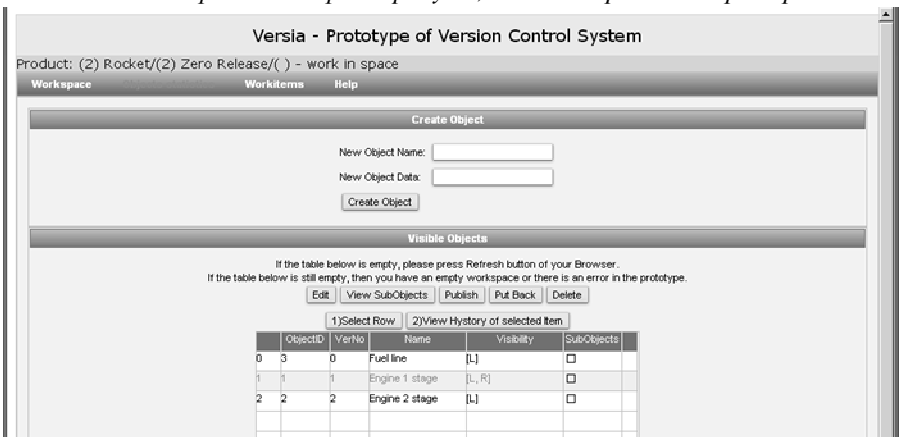
Фиг. 13 Навигационен модел на прототипа

на всеки обект, който е видим в пространството. Информацията се показва в колона Visibility, като се използват следните означения:

- R (Release) – версия на обекта съществува в главното работно пространство.
- P (Parent) – съществува версия в родителско пространство.
- L (Local) – версията, която се вижда е локална.
- C (Child) – съществува версия в дъщерно пространство.
- O (Other) – съществува версия в друг клон от йерархията на пространства.
- D (Deleted) – видимата версия се явява маркер на изтрито състояние.



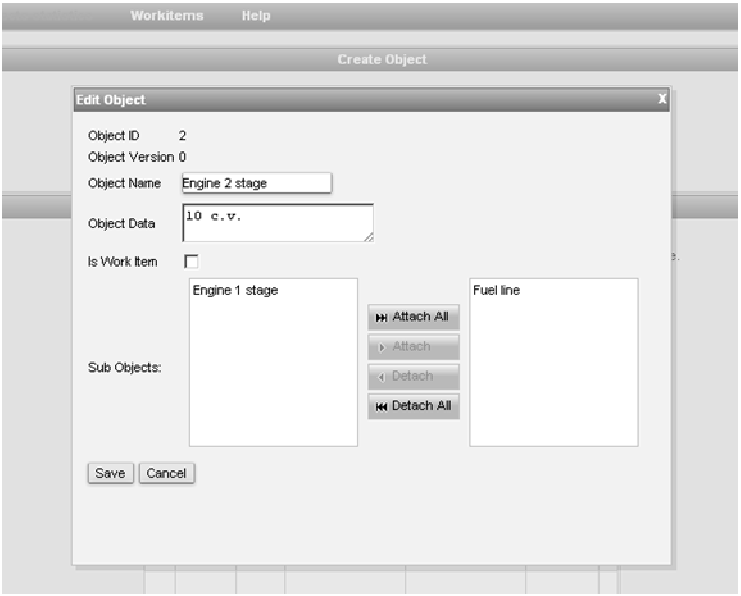
Фиг. 14 Екран за избор на продукт, издание и работно пространство



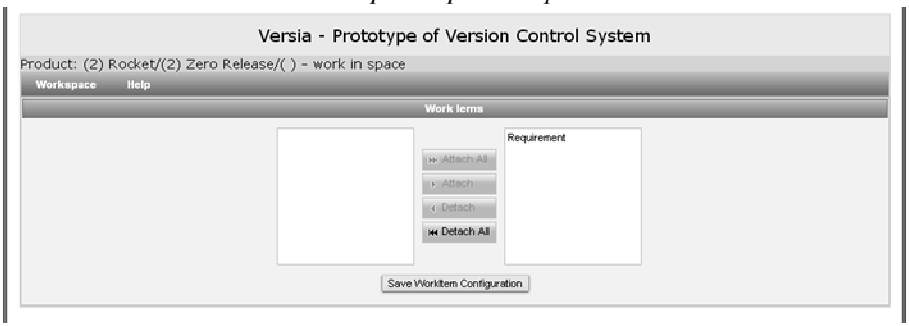
Фиг. 15 Версионизирани обекти в работно пространство

При редактиране на обект (Фиг. 16), той може да се отбележи като работна задача. Пак там се предоставя възможност да се добавят или премахнат *подобекти*.

На екрана за настройване на активните работни задачи потребителят има възможност да определи над кои задачи работи. На Фиг. 18 потребителят има възможност да проследи историята на един обект. Там той вижда версиите, предшестващи избраната, както и кои обекти – работни задачи са били активни в момента на промяната.



Фиг. 16 Екран за редактиране на обект



Фиг. 17 Екран за настройване на активните работни задачи

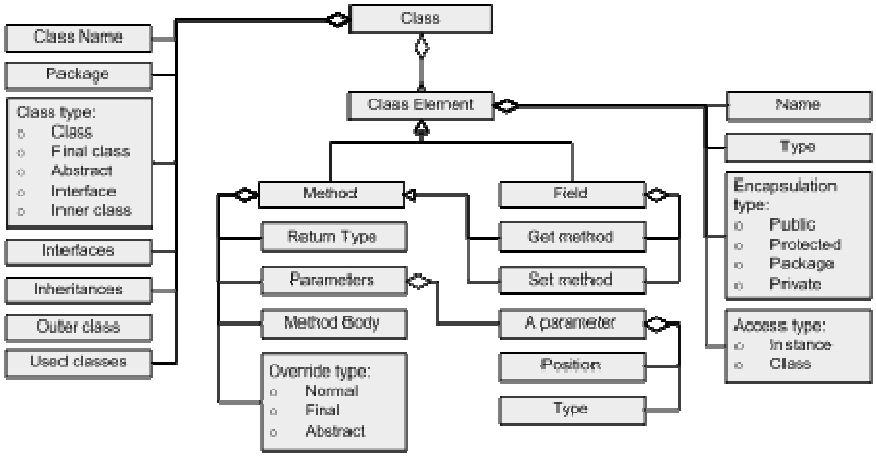
Фиг. 18 Екран показващ историята на един обект и причините за неговата промяна

Отделна точка от параграфа заема тази, в която са представени основните алгоритми на прототипа. Особен интерес представляват алгоритмите за създаване на нова версия на обект, този за публикуване на версионизиран обект. Алгоритмите са допълнени с диаграми на последователностите.

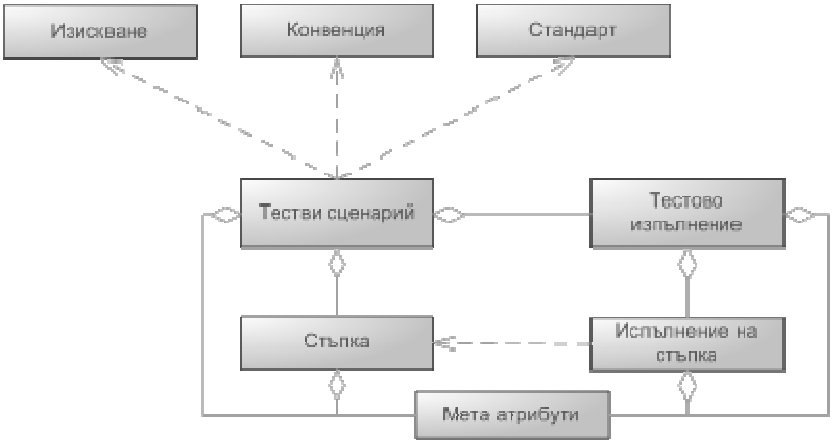
3.3. Примерни модели за композиране на версионизирани обекти

Демонстрират се възможностите на ER модела на версионизиран обект, като се акцентира на подобренията в сравнение с файловия модел за версионизиране. Представени са модели на версионизирани обекти, които предоставят възможности за свободно определяне на степента на тяхната гранулираност. Това дава 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.

Създадени са два примерни модела на декомпозиране на версионизирани обекти – модел на на същността „клас" (Фиг. 19) и модел на обектите в областта на тестирането (Фиг. 20).



Фиг. 19 Примерен модел на композиране на версионизирани обекти за същността клас



Фиг. 20 Примерен модел на композиране на версионизирани обекти за областта на тестрането

3.4. Сравнителен анализ от използването на моделите в прототипа

Направен е опит за експериментално-теоретично демонстрация на предимствата от използване на разработените модели. В таблицата по-долу е представен резултатът от експеримента, което позволява сравнение на основните моменти.

Фаза	**Съществуващи технологии**	**Предложени модели**
Регистриране на изисквания.	Ръчно регистриране в Trac	Регистрация като версионизиран обект
Разработка на архитектурата, изходния код и тестовите сценарии	Създаване на отделени файлове.	1. Избор на изискване, като работна задача. 2. Създаване като отделни версионизирани обекти (автоматично свързани към изисквания).
Регистриране на арх-рата и изходния код в с-ма за упр. на версии	1. Ръчво указване на № изискване в коментара към публикуването.	Публикуване в главно работно пространство.

Фаза	Съществуващи технологии	Предложени модели
	2. Ръчна смяна на статуса на изискването в Trac.	
Тестиране и определяне на причината на дефекта	1. Създаване на дефект в Trac, и ръчно свързване с изискване.	1. Избор на изискване, като работна задача. 2. Създаване на дефект (автоматично свързан към изискване)
Отстраняване на дефект	1. Ръчно намиране на мястото на дефекта – в изходния код или в изскване. 2. Отстраняване на дефекта. 3. Публикуване на корекцията с указване № на дефекта. 4. Ръчна смяна статуса на дефекта.	1. Генериране на справка и намиране мястото на дефекта. 2. Избор на дефекта, като работна задача. 3. Отстраняване на дефекта. 4. Публикуване в главно работно пространство.

Детайлният анализ от реализациите показва следното:

- Сега съществуващите практики предполагат използването на отделни системи за управление на заявките/изискванията и система за контрол на версиите. Това изисква допълнителни усилия по синхронизация на данните между системите от страна на екипа, участващ в проекта. Този факт неминуемо води до повишаване на риска от човешка грешка в рамките на проекта, а също така увеличава натоварването на сътрудниците често с несвойствени за тях дейности. Използването на единна система, позволяваща както да се управлява изискванията, тестовете, така и да се управлява версията на изходния код, води до намаляването на този риск.
- Етап от експерименталната задача, включващ промяна на изискване към системата, демонстрира в пълна сила

предимствата от разработените модели, с цел подпомагане и автоматизиране работата на участниците в процеса на създаване на софтуерни продукти. Те биват разтоварени от необходимостта да преглеждат ръчно цялостната архитектура и изходния код, за да определят местата за локална промяна.

ЗАКЛЮЧЕНИЕ
НАУЧНИ И НАУЧНО-ПРИЛОЖНИ ПРИНОСИ

1. В резултат на изследване на основните въпроси и анализ на дадената предметна област, са формулирани нерешените проблеми в съвременните системи за управление на версии.
2. Създаден е авторски модел на версионизиран обект, който позволява да се определи степента на гранулираност на данните.
3. Предложен е авторски модел на среда с йерархично композирани работни пространства и са определени правилата за управление на версия на обекти в тази среда. Това позволява всички участници в процеса да работи изолирано, както и да се кооперират по определени задачи или направления.
4. Направена е адаптация на метод за проследимост на промени, базиран на събития, за среда с йерархично композирани работни пространства, който осигурява по-добри възможности за анализ на промените.
5. Определена е терминологията в областта на версионизирането с използването на йерархично композирани работни пространства.
6. Предложена е методологична рамка за използване на разработените модели. Направен е сравнителен анализ между използването на съществуващите инструменти и разработените модели. Анализът показва увеличаване на степента на автоматизация на част от дейностите при създаване на софтуерни продукти.
7. Реализиран е функционален прототип на система за управление на версии, с помощта на който е направена апробация на разработените модели.

ПУБЛИКАЦИИ, СВЪРЗАНИ С ДИСЕРТАЦИОННИЯ ТРУД

1. Jotov, Vl. An investigation on the approaches for version control systems. In: Proceedings of the Bulgarian International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing. CompSysTech'08, Gabrovo, 12-13 June, 2008, pp.V.11-1 – V.11-3, ISBN: 978-954-9641-52-3.
2. Jotov, Vl. Transaction over versioned objects in hierarchical workspace environment. In: Proceedings of the International Conference on Electronics, Computers and Artificial Intelligence – ECAI" 09, 3-5 July, Pitesti, Romania, 2009, pp.119-122, ISSN – 1843 – 2115.
3. Jotov, Vl., Towards a model of versioning domain. In: 2009 5th Central and Eastern European Software Engineering Conference in Russia (CEE – SECR), Moscow, 28-29 October, 2009, pp. 272-275, ISBN: 978-1-4244-5665-9.
4. Йотов, Вл., Модел на данните в система за контрол на версии, базирана на йерархични работни прстранства. В сборник доклади: Научна конференция с международно участие „25 години Педагогически факултет.. Велико Търново, 6-7 ноември 2009 г.", Велико Търново, 2010, с. 465-467, ISBN: 978-954-400-422-4.
5. Jotov, Vl., Adaptation of Event-Based Traceability Method for Environment with Hierarchal Composed Workspaces. In Proceedings: John Atanassov Celebration Days. International Conference Automatics and Inforatics'10, Sofia, October 3-7, 2010, pp. I-269 – 1 - 272, ISSN: 1313-1850.