**8. Ділення кодом і контроль версій**

Наслідком програмування в світі живих об’єктів радше ніж в світі текстових файлів і редакторів є те, що тобі потрібно зробити щось явне, щоб експортувати програму з твого **образу Фаро.** Старий спосіб полягав у створенні “fileout” чи “change set”, які по своїй суті є закодованими текстовими файлами, що імпортуються в іншу систему. У Фаро бажаний спосіб передачі коду полягає в зберіганні пакетів і ділення ними через репозиторії на сервері. Аж до 5 версії Фаро це досягалося за рахунок програми під назвою “Monticello”, що є досить потужним і ефективним механізмом роботи, особливо коли йдеться про роботу в команді. В майбутньому Фаро надасть офіційну підтримку для програми контролю версій під назвою Git та можливість зберігати код на серверах таких , як GitHub, BitBucket тощо.

Ми дали короткий огляд Monticello, програми для створення пакетів у Фаро, в розділі «Перша програма». Однак, Monticello має набагато більше властивостей ніж було там зазначено там. Тому що Monticello управляє пакетами, перед тим як перейти до заглиблення в саму програму, дуже важливо є спочатку вияснити, що таке пакет.

**8.1 Пакети: групи класів і методів.**

Ми зазначали раніше (в параграфі «Перша програма») , що пакети це більшою чи меншою мірою набір класів та методів. Зараз же ми достеменно дослідимо цей об’єкт. Система пакетів є досить простою, легкий спосіб організації Фаро програмного коду (source code), що використовує просту ковенцію відносно назв.

Давайте проілюструємо це на прикладі. Припустимо, що ви розробляєте фреймворк для роботи з реляційними базами даних за допомогою Фаро. Ви вирішили назвати ваш фреймворк «PharoLink», і створили кілька класів таких, як OracleConnection, MySQLConnection, PostgresConnection, та DBTable, DBRow, DBQuery і тд. Всі ці класи є розміщені в середині пакету під назвою «PharoLink». Однак, не весь ваш код буде поміщений в ці класи. Наприклад, ви можете також мати набір методів, щоб конвертувати об’єкти в SQL дружній формат (Object>>#asSQL, String>>#asSQL, or Date>>#asSQL.).

Ці методи належать до того ж пакету, що й класи. Але, очевидно, що клас Obect не належить вашому пакету! Отже, вам потрібно мати можливість покласти певні методи в пакет, навіть якщо решта класу є в іншому пакеті.

Дана проблема вирішується шляхом розміщення цих методів в протокол (Object, String, Date і так далі) під назвою \*PharoLink (примітьте астеріск на початку назви). Протоколи \*PharoLink додаються в пакет PharoLink. Щоб бути точним, ось правила згідно яких відбувається створення пакету.

Пакет під назвою Foo містить :

1. Всі визначення класів, які є в пакеті Foo.
2. Всі методи з будь якого класу в протоколі під назвою \*Foo чи \*foo (При здійсненні синтаксичної перевірки не грає роли чи початкова літера є маленькою чи великою), або починається з \*Foo- або \*foo.
3. Всі методи в класах в пакеті Foo крім тих методів, які є в протоколах ім’я, яких розпочинається \*: класи пакету Foo також можуть бути розширені іншими пакетами.

З даних правил випливає, що кожне визначення класу і методу належить лише одному пакету.

Доступ до пакетів

Клас RPackage представляє власне пакети. Клас RPackageOrganizer імплементує сінглетона, який містить усі пакети. Наступні вирази є прикладами запитів, що ви можете виконати.

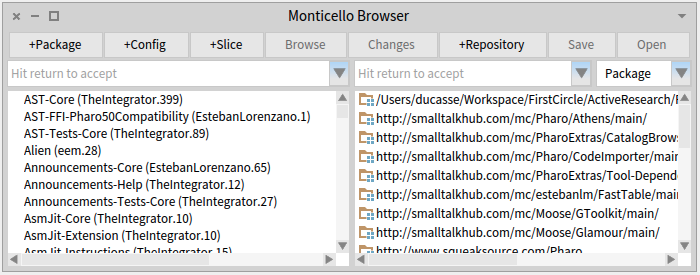
(RPackageOrganizer default packageNamed: 'AST-Core') definedClassNames

(RPackageOrganizer default packageNamed: 'AST-Core') extendedClasses

**8.2 Основи Monticello**

Коли ви відкриєте браузер Monticello, ви побачите дві панелі зі списками і ряд з кнопками, як на рисунку 8.1. Ліва панель містить списки усіх пакетів, які були завантажені в образ, що ви використовуєте; конкретна версія пакету показана в дужках після імені.

Права панель містить всі репозиторії із програмним кодом про яких знає Monticello, зазвичай через те що він вже завантажував код з них. Якщо ви виберете пакет в лівій панелі, списки в правій панелі будуть відфільтровані, щоб показати лише ті репозиторії, які містять дану версію вибраного пакету.

рис. 8.1 Браузер Monticello

Кеш пакетів

Один з репозиторіїв є директорія під назвою «package-cache», яка є підпапкою директорії, в якій виконується ваш образ системи. Коли ви завантажуєте код з чи записуєте код на віддалений репозиторій, копія пакету також зберігається в цій папці. Це може бути корисним, якщо інтернет мережа є недоступна, а вам потрібний даний пакет. Також, якщо вам дали Monticello (.mcz) файл напряму, наприклад через цифрову почту, найбільш зручний спосіб доступу до нього є помістити в package-cache директорію.

Додавання репозиторіїв

Для того щоб додати репозиторій до списку, клікніть на +Repository, і оберіть вид репозиторію з випадаючого списку. Давайте додамо HTTP репозиторій.

**Зробити** : Відкрийте Monticello, клікніть на +Repository і оберіть HTTP. Редагуйте діалог, щоб прочитати:

MCHttpRepository

location: ‘http://squeaksource.com/PharoByExample’

user: ‘ ’

password: ‘ ’

Натисніть кнопку Open, щоб відкрити браузер в цьому репозиторію. Ви повинні побачити щось схоже до рисунку 8.2. Зліва буде список всіх пакетів в репозиторії. Якщо ви виберете один, тоді панель справа покаже всі версії вибраного пакету в цьому репозиторії.

Перегляд версій.

Якщо ви виберете певну версію, ви можете *Browse it* ( переглянути її без завантаження на вашу систему), *Load it* (Завантажити її), чи подивитися на зміни (Changes), які будуть зроблені у вашому образі при завантаженні даного пакету. Ви також можете зробити Копію (Copy) даної версії пакета, яку ви можете потім записати до іншого репозиторію.

Як ви бачите, назви версій містять імена пакетів, імена авторів версії, і номер версії. Назва версії є також назвою файлу в репозиторії. Ніколи не змінюйте ці імена, коректність здійснення операцій Monticello залежить від них! Файли версій Monticello є просто zip архівами, і якщо вам цікаво ви можете скористатися відповідними засобами, щоб розпакувати і подивитися на їхній вміст, але найкращий спосіб це використовувати власне браузер Monticello/

Створення пакету.

Для того щоб створити пакет з допомогою Monticello, ви маєте зробити 2 речей: написати якийсь код , і сказати Monticello про це.

**Зробити:** Створіть пакет під назвою PBE, покладіть пару класів туди, як зображено на рис 8.3. Також створіть метод в існуючому класі, наприклад Object, і покладіть його в той самий пакет, що і класи. (див. рисунок 8.4).

Віддача пакету

Коли ви додаєте пакет через браузер класів, Monticello автоматично додає пакет до свого списку. РВЕ вже є в списку пакетів. Даний пакет в списку буде позначений зіркою, щоб показати, що образ ще не був записаний до жодного репозиторію (Про такий кажуть, що він брудний).

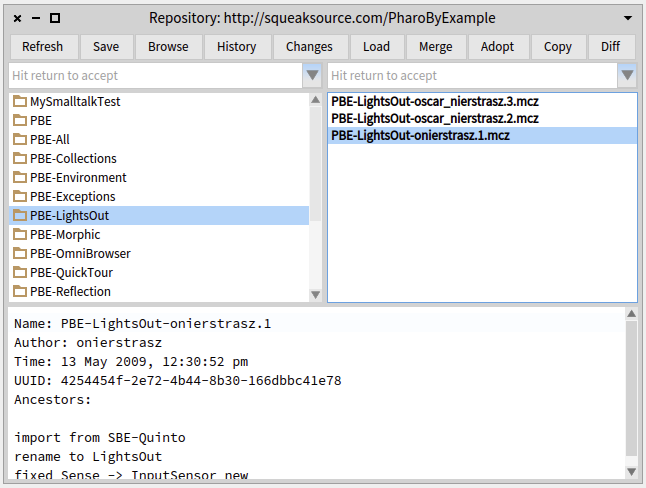


рис. 8.2 Браузер репозиторію.

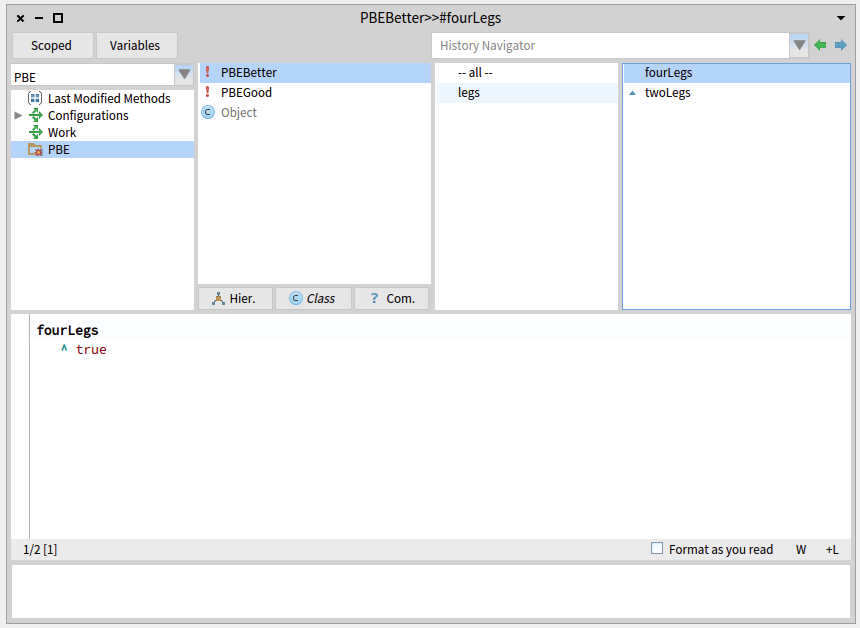


Рис 8.3 Два класи і один метод розширення в пакеті РВЕ.

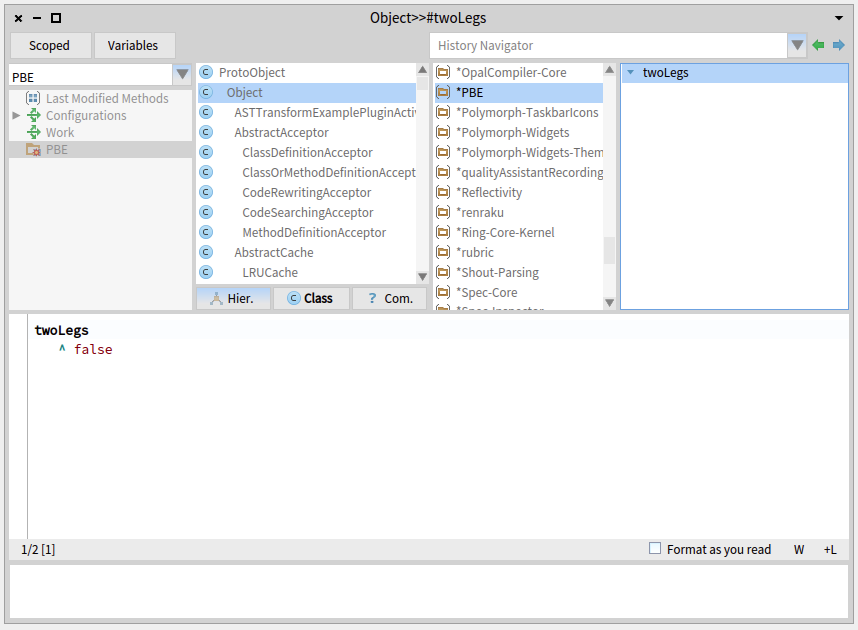


Рис 8.4 Метод розширення в пакеті РВЕ.

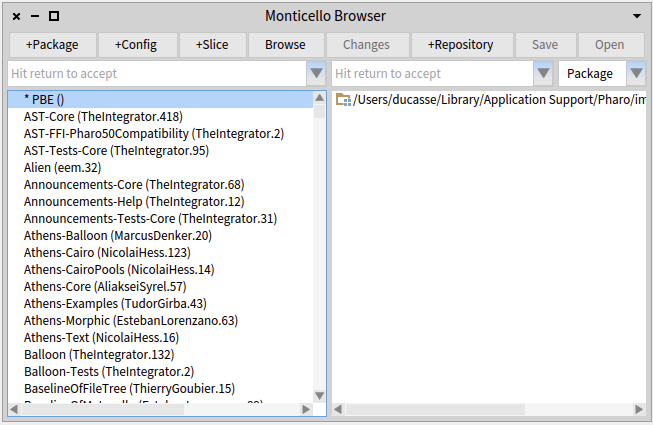


Рис 8.5 Поки що незбережений РВН пакет.

Напочатку, лише репозиторій асоційований з цим пакетом буде вашим кешом для пакету, як показано на рисунку 8.5. Це нормально: ви все ще можете зберегти код, що спричинить до його запису в кеш пакетів. Просто натисніть Save, як показано на рисунку 8.6; якщо ви приймете повідомлення, то Monticello збереже ваш пакет. Для відображення даного факту, астеріск буде забраний і буде доданий номер версії.

Здійснення змін

Якщо ви потім зробите зміни в пакет – скажемо, добавите метод до якогось класу – то астеріск знову появиться, показуючи що є незбережені зміни. Коли ви відкриєте браузер репозиторію в кеші пакетів, ви можете вибрати збережену версію, та використати Changes (Зміни) та інші кнопки. Ви можете, звісно , зберегти нову версію в репозиторій також. Тільки ви оновите (Refresh) вигляд репозиторію, то отримаєте зображення схоже до рисунку 8.7.

Для того щоб зберегти новий пакет до репозиторію іншого ніж кеш пакетів, ви повинні спершу переконатися, що Monticello знає про даний репозиторій, при потребі додавши його. Тоді ви можете використати копіювання (Copy) в браузері package-cache репозиторію, та обрати репозиторій, в який пакет потрібно скопіювати. Ви також можете асоціювати певну директорію з пакетом, клікнувши правою кнопкою миші на репозиторії і вибравши «add to package…» та обравши відповідний пакет. Як тільки пакет знає про репозиторій, ви можете зберігати нову версію, обираючи репозиторій і пакет в браузері Monticello, і клікаючи Save. Звичайно, вам потрібно мати дозвіл для здійснення запису в репозиторій. Репозиторій PharoByExample є доступний для читання всім, але не для запису, тож якщо ви спробуєте здійснити запис в цей репозиторій – то ви побачите повідомлення про помилку.

Однак ви можете створити ваш власний репозиторій і зберегти код туди. Тепер популярними репозиторіями для коду є http://www.smalltalkhub.org та http://ss3.gemstone.com . Це є дуже корисний механізм, щоб ділитися вашим кодом з іншими і бути впевненим, що ви можете використовувати системи автоматизації компілювання Jenkins або Travis.

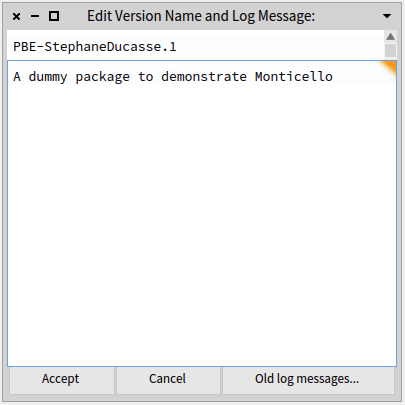


Рис. 8.6 Лог повідомлення для нової версії пакету.

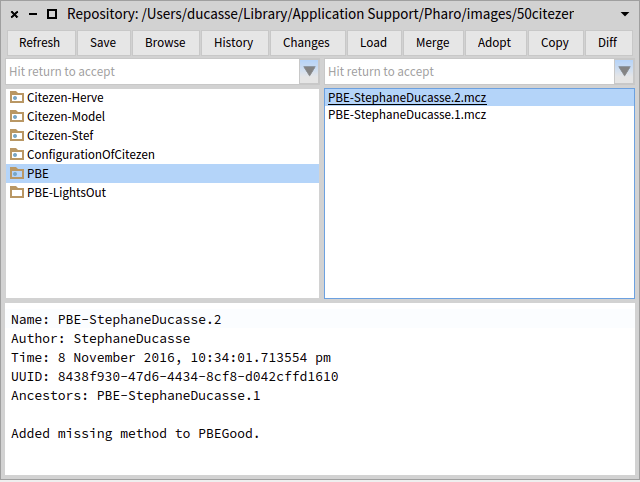


Рис. 8.7 Дві версії нашого пакету в кеші пакетів.



Рис. 8.8 Браузер версій, який показує дві версії ObjTest>>testIVOffset методу.

Якщо ви спробуєте і збережете до репозиторію, де ви не маєте права на запис, версія все ж таки буде збережена в package-cache. То ви можете відновитися, просто змінивши інформацію про репозиторій (правий клік миші на браузері Monticello) або ж обравши інший репозиторій, і тоді використовуючи Copy (копіювання) з package-cache браузера.

**8.2 Контроль за версіями**

Версії методу

Коли ви зберігаєте нову версію методу, стара не зникає. Фаро зберігає всі версії (припускаємо, що ви використовуєте той самий образ системи і зберегли його)., і дозволяє порівнювати різні версії і здійснювати відкати назад, до старіших версій.

Browse > versions (v) меню дає доступ до наступних модифікацій, які були зроблені у вибраному методі. На рисунку 8.8 ви можете бачити дві версії методу.

Верхня панель зображує кожною лінією іншу версію методу, вказуючи ім’я програміста який її написав, час та дату коли було здійснено збереження, і протокол, в якому його визначено. Поточна (активна) версія методу є на горі списку.

Завдяки існуванні браузера версій вам не потрібно хвилюватися за збереження коду, про який ви думаєте, що він не потрібний. Якщо ви вирішите, що він вам таки потрібен, то ви можете відновити стару версію або ж скопіювати цю ділянку коду в потрібне вам місце. Візьміть за звичку використовувати версії. Коментування коду вже є не потрібним і вважається поганою звичкою, адже погіршує читабельність коду. Фаровці цінують читабельність коду надзвичайно високо.

Підсказка. Що якщо ви видалили код повністю, а потім захотіли його вернути? Ви можете знайти видалену частину в change set, де ви можете бачити версії за допомогою правого клацання миші. Change set браузер описаний в секції 8.2.

Change set i change sorter

Коли ви працюєте в Фаро, будь які зміни, які ви робите в методі і в класі, записуються в change set . Це включає і створення нових класів, перейменування класів, зміна категорій, додавання методів до існуючих класів – просто все важливе. Однак, звичайні команди “Do it” не записуються.

В будь який момент, багато множин змін (change set) існує, але лише поточна (change set current) збирає зміни, які були зроблені в образі. Ви можете переглянути яка множина змін є поточна і переглянути зміни використовуючи сортувальник змін (change sorter), доступний в World > Tools…> Change Sorter.

На рисунку 8.9 зображено подвійну зміну в браузері сортувальника змін. Заголовок вказує, яка множина змін є поточна. І ця множина змін є вибрана, коли відкривається сортувальник змін.

Інші множини змін можуть бути вибраними у верхній лівій панелі; меню «правого кліку» дозволяє вибрати обрати різні множини змін як поточну чи створити нову множину змін. Наступна панель подає список всіх класів, яких торкнулися зміни в даній множині змін (з їхніми категоріями).При обиранні одного з класів, відображається імена його методів, які також є в множині змін (не всіх методів класу) в лівій центральній панелі, і при обиранні певного методу, відображається визначення методу в панелі знизу.

Сортувальник змін також дозволяє видалити класи чи методи з множини змін використовуючи меню правого кліка миші на відповідному елементі.

Сортувальник дозволяє одночасно проглядати дві множини змін, одну на лівій стороні, а іншу на правій. Така особливість забезпечує основну властивість сортувальника змін, як можливість переміщувати чи копіювати зміни з однієї множини змін до іншої, як це відображено на рис. 8.9 Також є можливим копіювати індивідуально методи з одного боку на інший.

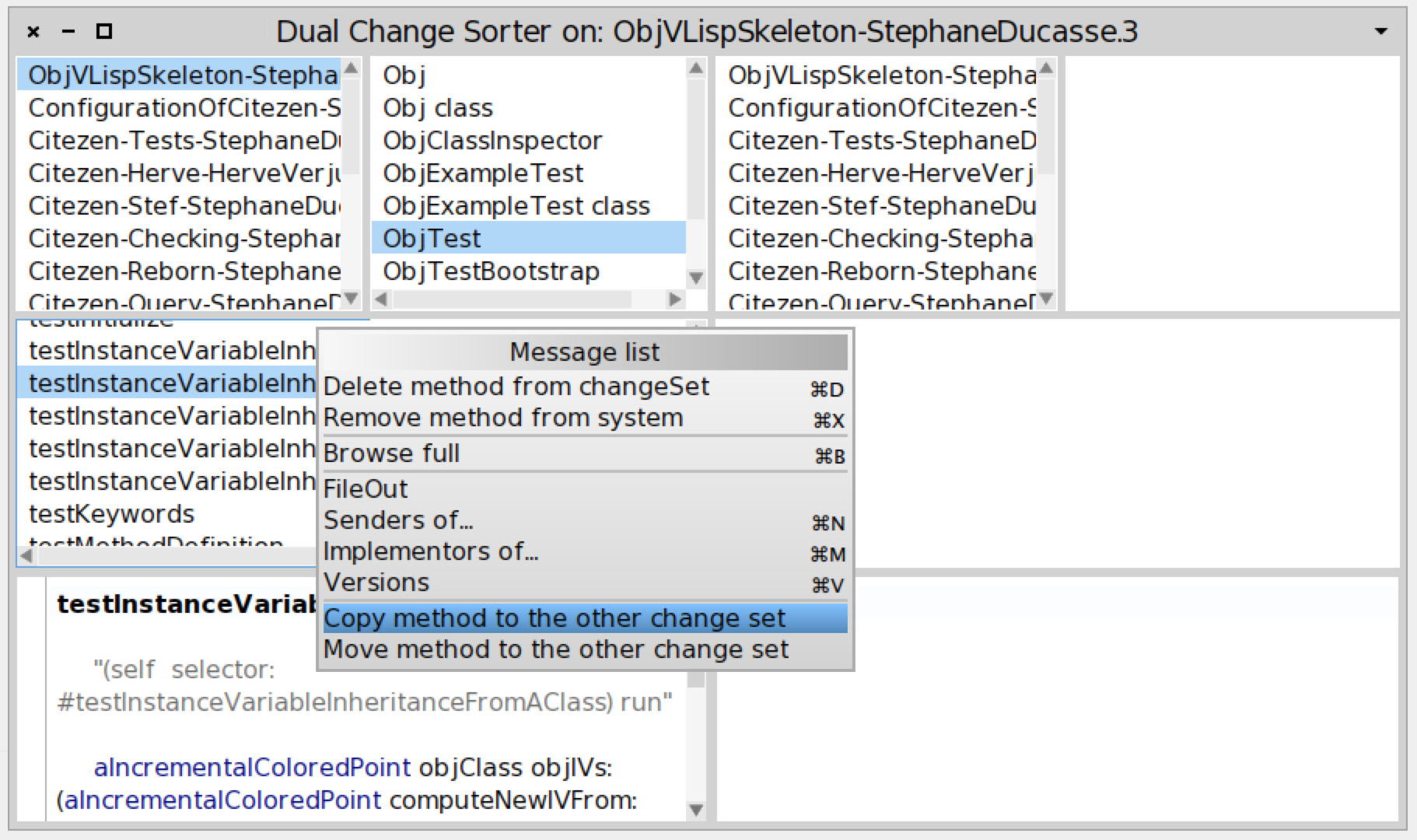


Рис. 8.9 Сортувальник змін показує всі зміни однієї множини змін і пропонує перемістити зміни до інших множин змін.

Вас може дивувати, чому вас повинно хвилювати композиція множини змін. Відповідь є такою, що множина змін забезпечує простий механізм експорту коду з Фаро у файлову систему, звідки він може бути перенесений на інший образ Фаро. Експорт множини змін відомий як Filling-out, і може бути здійсненим завдяки правому клацанні мишею на будь якій множині змін, класі чи методі в браузері. Повторювані file outs створюють нову версію файлу, але множина змін не є засобом для контролю над версіями як Montecello: вони не слідкують за залежностями.

Перед появою Montecello, множина змін була найважливішими засобом для обміну кодом у фарорівців. Її перевага була в простоті ( file out - це просто текстовий файл, однак ми не рекомендуємо його рекомендувати текстовими редакторами), і ступеню переносу.

Основний недолік множин змін, порівняно до пакетів Monticello, є те, що вони на слідкують за залежностями. Не зважаючи на ці недоліки, множини змін все ще використовуються. Зокрема, ви можете знайти множини змін в інтернеті, на які ви б хотіли подивитися і, можливо, скористатися. Отже, здійснивши filed out множини змін, використовуючи сортувальник змін, ми можемо сказати як здійснити обернену операцію (file in). Дана операція потребує іншого засобу, браузеру списку файлів (file list browser). Зауважте, що при перетягуванні множини змін у Фаро, остання спитається чи хочете здійснити імпорт змін з даної множини за допомогою випливаючого вікна.

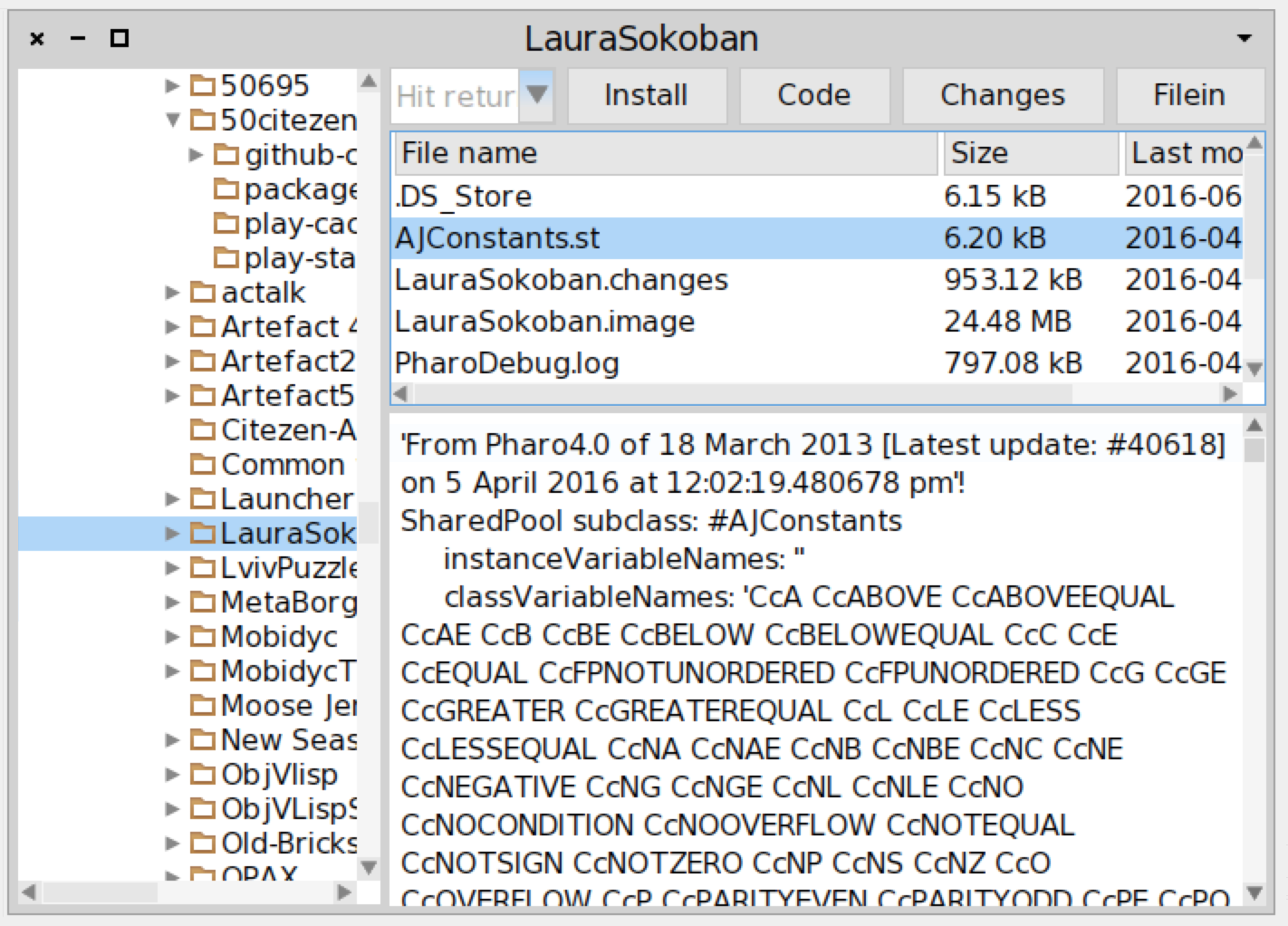


Рис. 8.10 Браузер списку файлів (A file list browser).

**8.3 Браузер списку файлів (The File List Browser)**

Браузер списку файлів є програмою загального призначення для перегляду файлів у файловій системі (так і на FTP серверах) з Фаро. Ви можете його відкрити з World > Tools… > File Browser menu. Що ви побачите залежить від вашої системи, але типове зображення ви можете побачити на рисунку 8.10.

Коли ви вперше відкриєте браузер файлів він вам покаже вашу поточну директорію, тобто ту , з якої ви стартували Фаро. Заголовок відображає назву директорії. Більша панель зліва може бути використана для звичайної навігації файловою системою. Коли директорія є обрана, файли, які вона містить (але не папки) будуть відображені справа. Цей список файлів може бути відфільтрований за допомогою ввожу шаблонів в Unix стилі в маленькому боксі зверху і зліва вікна. Спочатку це шаблон – це \*, з яким співпадають усі імена файлів, але ви можете ввести інший рядок туди і застосувати його, змінивши шаблон фільтру.(Заувага, \* буде неявно додано спереду і позаду до вашого шаблону). Порядок сортування може бути змінено, використовуючи кнопки «Ім’я» (Name), «Дата» (Date), «Розмір» (Size). Інші кнопки залежать від файла, який ви виберете в браузері. На рисунку 8.10, ім’я файла має суфікс .st, отже браузер припустить, що це код Фаро, і забезпечить відповідними можливостями через кнопки.

Тому що вибір кнопки для відображення залежить від імені файла, який ви виберете, а не від його наповнення, часом кнопка, яку ви хочете не появиться на екрані. Однак, повний набір опцій можливий через контекстне меню > more… опції, тож ви легко можете обійти цей недолік.

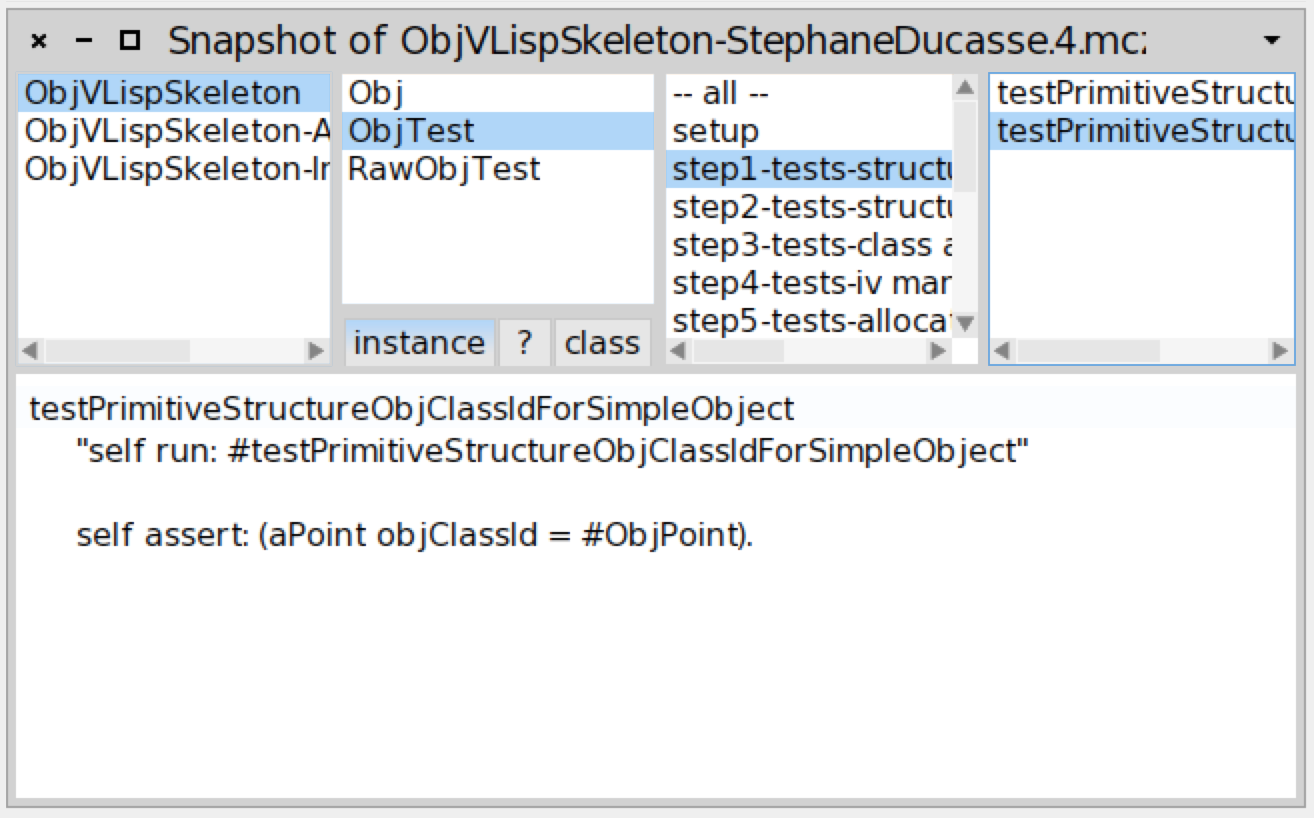


Рис.8.11 Браузер списку файлів, відкритий на пакеті.

Кнопка code є, можливо, найбільш корисною для роботи з множинами змін. Вона відкриває браузер на вмісті множини змін чи MC пакету. Приклад зображено на рисунку 8.11. Браузер вмісту файлів є подібним до браузеру, крім того що код не завантажується в пам’ять.

**8.4 У Фаро ви не можете втрати код.**

Цілком можлива є аварія Фаро. Як експериментальна система Фаро дозволяє міняти все, включаючи речі, які є необхідні для функціонування Фаро!

Хороші новини в тому, що ви ніколи не загубите вашу роботу, навіть якщо станеться аварія Фаро і вона відкотиться до попередньої робочої версії, яка може створена аж годину тому назад. Це все відбувається тому, що весь код, який ви виконуєте зберігається в .change файлі. Він не містить коду Playground, який зберігається в середині (*/pharo-local/playchace*) і може бути отриманий за допомогою Spotter (результати відображатимуться під секцією Playground-cached).

Там є інструкції, як отримати ваш код назад. Немає потреби читати це, аж поки це не стане потрібним. Однак, вона вас там очікує, як тільки ви станете її потребувати.

В гіршому випадку, в можете використати редактор тексту для відкриття .change файлу, але, оскільки розміри файлу можуть сягати мегабайти, це не рекомендується. Фаро пропонує кращі шляхи.

Як отримати код назад.

Рестартніть Фаро з останнього знімка, і оберіть World > Tools … > Recover lost changes.

Smalltalk recover: 10000.

ChangeList browseRecentLog.

ChangeList browseRecent: 2000.

Це дасть вам можливість вирішувати наскільки далеко в історію в хочете заглиблюватися. Зазвичай, достатньо останнього знімка. ( Ви можете отримати схожий ефект, редагуючи ChangeList browserRecent: 2000 так, що число 2000 стане іншим, шляхом спроб і помилок).

Як тільки ви отримаєте браузер останніх змін, який показує, скажімо, зміни аж до останнього знімку, ви будете мати список всього, що було зроблено у Фаро протягом того часу. Ви можете видалити речі зі списку використовуючи контекстне меню. Коли будете задоволені, ви можете file-in, те що лишилося, таким чином вмонтовуючи зміни у ваш новий образ. Хорошою ідеєю є розпочати нову множину змін. Ви можете потім file out цю множину змін.

Ще одною корисною річчю, щоб зробити, є видалення doIts. Зазвичай ви не захочете file in (таким чином виконавши його) doIts. Однак, є вийняток. Це створення класу, який з’являється як doIt.

Перед тим як виконати file in для методів класу, клас має існувати. Отже, якщо ви створили новий клас, спочатку file in створення класу через doIts, потім видаліть doIts і файли в методі.

Коли я завершив з відновленням, я зазвичай роблю file out моєї нової множини змін, виходжу з Фаро без збереження змін, рестартую, і переконуюся , що нова множина змін робить files back in чисто.

**8.5 Підсумок глави**

* Monticello - це засіб для експорту, імпорту, контролю версій та діленням пакетів з класами і методами.
* Пакети Monticello складаються з класів, пов’язаних методів в інших пакетах.
* Множини змін – це автоматично згенерований лог всіх змін програмного коду вашого образу. Вони значною мірою були витіснені Monticello, як засіб для зберігання і обміну версіями програмного коду, але досі корисні, особливо при відновлені після катастрофічних помилок.
* Браузер списку файлів (File list browser) – це засіб для перегляду файлової системи. Він також дозволяє виконувати File in програмного коду з файлової системи.
* У випадку аварії образу перед тим, як ви його зберегли чи зробили резервну копію з допомогою Monticello, ви можете відновити ваші останні зміни , використовуючи браузер змін (change list browser). Ви можете вибрати зміни, які ви хочете заново внести і завантажити їх у вашу найбільш пізню копію образу системи.