Metody inżynierii wiedzy

XTT+ Apps

Cel i opis projektu

Wybranie przykładowych projektów informatycznych, najlepiej opisanych technikami UML i MVC. Stworzenie implementacji tychże przykładów w języku XTT/ARD za pomocą narzędzia HQED.

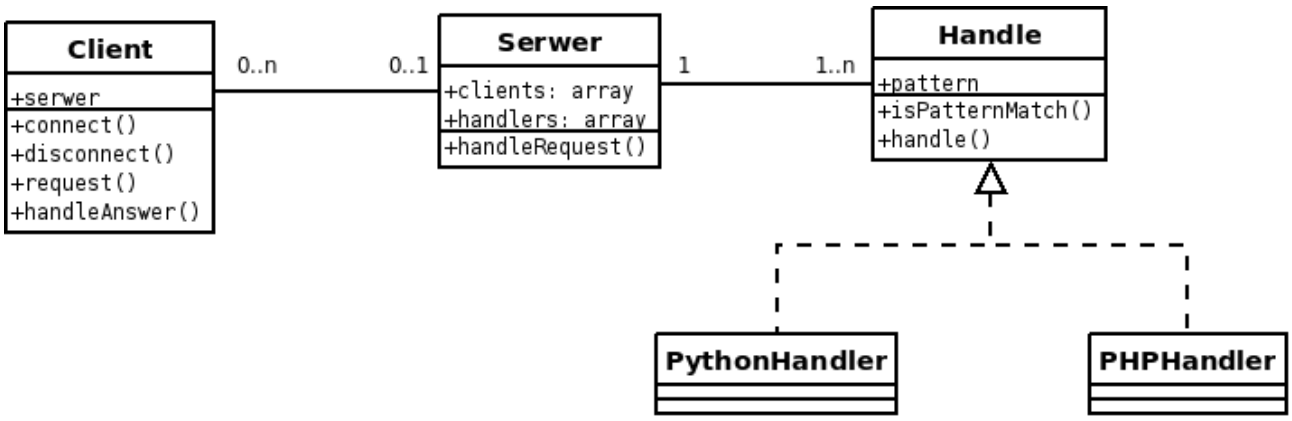
Projekt ma za zadanie ukazać możliwości modelowania w XTT+, opisać zalety i wady tego podejścia. Przykłady które wybrałem:

1. Webserwer – komunikacja client-serwer
2. Subversion – system kontroli wersji
3. Inne przykłady dobrze znanych problemów programistycznych.

Opis przykładów

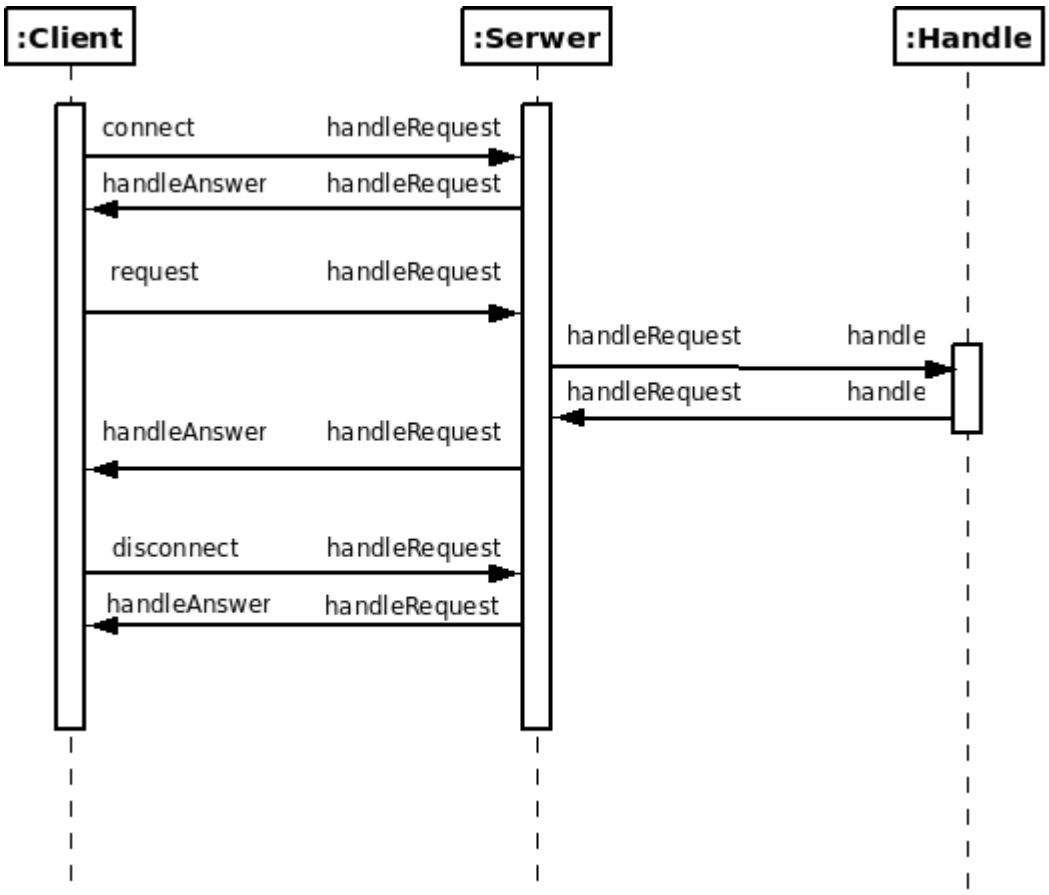
Webserwer

Jednym z najbardziej popularnych i używanych na szeroką skalę serwerów WWW jest Apache – w chwili obecnej wersja 2.2.9, wersja 1.3 jest opisana tutaj [1]. Jest to serwer z którym praktycznie każdy kto miał kontakt z Internetem, się spotkał – możliwe, że nawet o tym nie wiedząc. Ma on budowę modułową dzięki czemu łatwo można go rozbudować. Ale dość wstępu – wybrałem taki przykład gdyż pokazuje on przejrzyście grupę obiektów w którym każdy ma określone z góry obowiązki. Uproszczoną budowę przedstawię na diagramie UML:



Rysunek 1: Diagram klas serwera Apache

Na powyższym diagramie, przedstawione są zależności między instancjami obiektów. Widzimy że serwer posiada informację o kliencie i modułach obsługujących (Handle), klient zaś tylko informację o serwerze HTTP. Opis jak przebiega komunikacja między tymi obiektami zostanie przedstawiony poniżej.



Rysunek 2: Diagram sekwencji komunikacji z serwerem Apache

Klient wysyła żądanie „connect” (rozpoczęcie połączenia) jest to żądanie które serwer potrafi „zrozumieć” więc nie odwołuje się do żadnych modułów. Podobnie żądanie „disconnect” kończące połączenie z serwerem. Inne żądania „request” są obsługiwane poprzez moduły, serwer musi zdecydować który z tych modułów jest odpowiedni do żądania klienta i przekazać je do tego modułu. Odpowiedź jaką uzyska od modułu przekazuje dalej do klienta. Jeśli serwer nie znajdzie odpowiedniej obsługi dla danego zapytania zwraca błąd.

Subversion

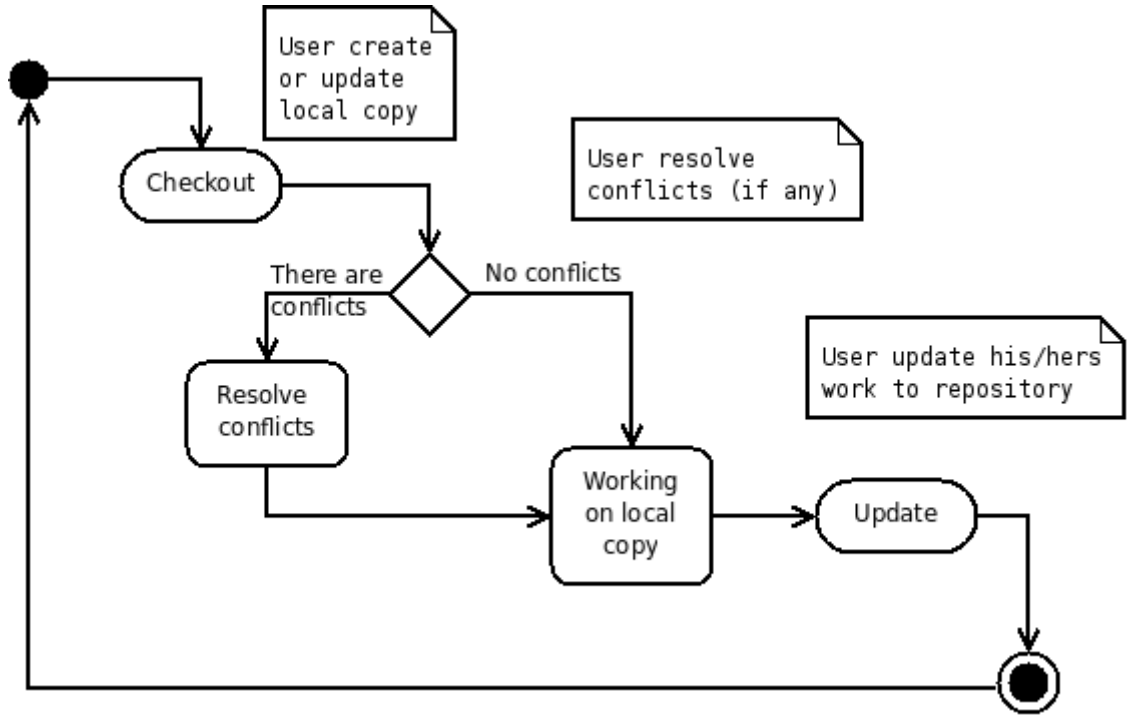
Najpopularniejszy obecnie system kontroli wersji – nieodzowne narzędzie w przypadku pracy w grupie. Założenia tego systemu są proste:

1. Wiele osób może pracować nad jednym fragmentem tekstu (kodu) – jednocześnie,
2. Po wprowadzeniu poprawek zawartość pliku jest uaktualniana w repozytorium,
3. Inna osoba może w tym czasie wprowadzić swoje poprawki będące w konflikcie z naszymi – należy wtedy rozwiązać konflikt wersji,
4. Kolejne wersje plików nie nadpisują poprzednich, lecz rozszerzają je poprzez zapisanie zmian jakie nastąpiły w pliku i kolejnego numeru wersji – umożliwia to powrót do wcześniejszych wersji plików.

Świetne wprowadzenie możemy przeczytać w książce [2], jak również ogólne reguły działania systemu kontroli wersji, ponadto porównanie możliwości z wciąż popularnym CVS.

Pierwszym przykładem był serwer Apache – system Subversion także możemy zakwalifikować do aplikacji klient - serwer, jest to jednak tak wyspecjalizowane narzędzie, że na pierwszy rzut oka trudno zauważyć podobieństwa. Ciekawostką jest, że istnieje moduł do Apache umożliwiający dostęp do repozytorium poprzez właśnie ten serwer. W przedstawionym opracowaniu skupię się

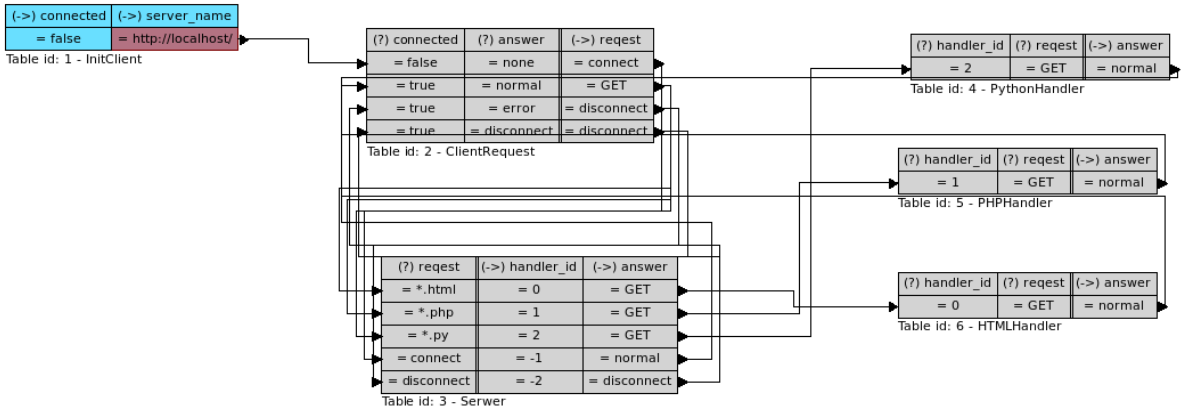
jedynie na cyklu korzystania z Subversion tj. założenia kopii roboczej, aktualizowaniu jej, zapisywaniu zmian w repozytorium i rozwiązywaniu konfliktów. Schemat tych działań obrazuje diagram:



Rysunek 3: Diagram stanów

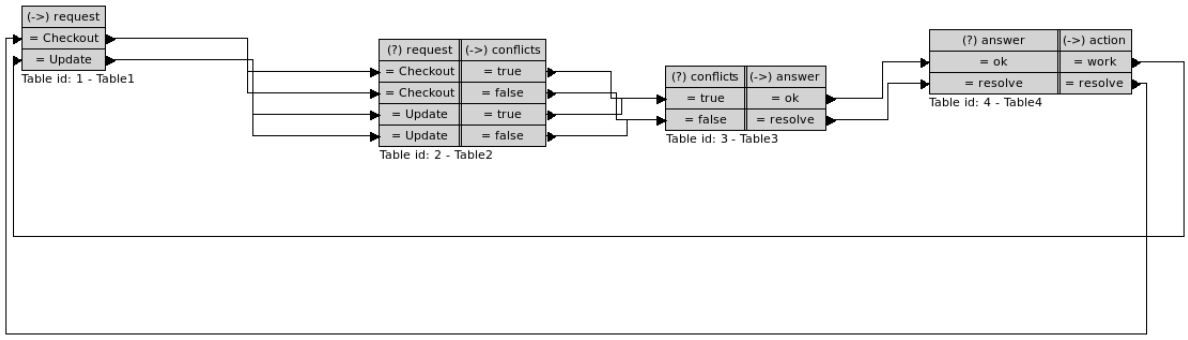
Próba zamodelowania przykładów w XTT+

Webserwer



Rysunek 4: Schemat XTT+

Subversion



Rysunek 5: Schemat XTT+

Problemy i wnioski

Okazuje się, że problemy Knowlege Engineering i Software Engineering pojawiają się już w trakcie próby ujednolicenia zapisu modelu. Podczas gdy dla SE potrafimy wskazać szereg technik, metod i narzędzi pomocnych przy tworzeniu modelu, w przypadku KE jesteśmy bardzo skąpo wyposażeni w te narzędzia. Ponadto okazało się wyjątkowo trudne przejście z modelowania w UML – niejako naturalnego języka SE, do postaci zapisu reguł wykorzystywanych przez KE. Trudności te zostały opisane m.in. w publikacjach [3],[4]. Podczas gdy model zapisany w XTT+ odpowiada w całości zapisowi w Prologu (w chwili obecnej jest to chyba najlepszy język KE) i na odwrót, z zapisanego programu w Prologu możemy odtworzyć model XTT+ - UML nie daje nam takiego komfortu. Model UML zawiera wiele niedomówień, nic nam nie mówi o przebiegu procesu modelowania, potrzeba wiele różnorodnych diagramów aby w pełni omówić daną funkcjonalność. Z drugiej strony w UML opieramy się o naszą intuicję tworząc tak naturalne dla człowieka obiekty i powiązania między nimi – co daje dużą swobodę projektantowi. XTT+ brakuje pojęcia obiektu jako takiego – obiektem jest pewien zbiór atrybutów i opierających się na nich reguł. Nie pozwala to na swobodne przechodzenie z projektowania w sposób obiektowy, na sposób zapisu reguł – nie pomaga w tym również skąpy zasób informacji zawarty w diagramach UML.

Podsumowując, tak jak pojęcia SE i KE oznaczają całkowicie różne podejścia do rozwiązywania problemów przez komputery, tak i nie ma prostego sposobu na przejście z jednej metody do drugiej. Miejmy nadzieję, że podobnie jak w dziedzinie SE nastąpił ogromny rozwój i postęp, tak KE będzie niedługo obfitowało w dogodne narzędzia, metody i techniki.

Bibliografia

1: Engelschall, Ralf S., Apache Desktop Reference, 2001
2: Collins-Sussman, Ben; Fitzpatrick, Brian W.; Pilato, C. Michael, Version Control with Subversion: For Subversion 1.4, 2002
3: Nalepa, Grzegorz J.; Wojnicki, Igor, Using UML for Knowledge Engineering - A Critical Overview, 2007
4: Nalepa, Grzegorz J.; Kluza, Krzysztof, UML Representation Proposal for XTT Rule Design Method,