Zarządzanie konfiguracją oprogramowania w odniesieniu do TEX-a

Tomasz E. Serafiński
Katedra Informatyki Stosowanej PŁ
Aleja Politechniki 11, 90-924 Łódź
tel./fax (+48) 42 631-27-50
tomaszek@cyberdude.com

Pracę zgłosił: Tomasz Przechlewski

Streszczenie

Praca porusza problematykę zarządzania konfiguracją w odniesieniu do systemu składu tekstu TEX. Począwszy od kontroli zmian i przepływu dokumentów oraz kodu źródłowego, aż do bardziej złożonych zagadnień związanych z cyklem życiowym TEX-a, potraktowanego jako ewoluujący projekt informatyczny.

Wprowadzenie

Zarządzanie konfiguracją jest dziedziną inżynierii oprogramowania, której znaczenie gwałtownie rośnie wraz ze stopniem złożoności systemów informatycznych. Swoim zakresem obejmuje w całości cykl życia projektu, a szczególnego znaczenia nabiera w trakcie jego dalszego rozwoju. Planowanie, kontrola zmian, koordynacja prac zespołu, sterowanie i wykrywanie potencjalnych zagrożeń, to tylko niektóre z funkcji jakie pełni. Jej znaczenie jest więc poważne, nie tylko dla zespołów technicznych, ale również dla szczebla zarządczego.

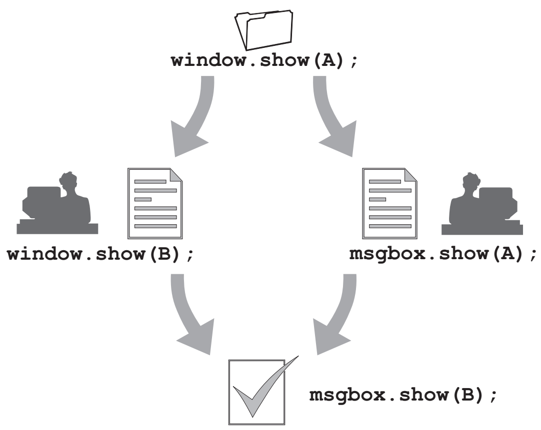
Szerokie pole zastosowań technologii określanych jako SCM[1] sprawia, że dotyka ona różnych dziedzin nie związanych bezpośrednio z informatyką takich, jak ekonomia czy zarządzanie. Dlatego w niniejszym artykule skupiono się wyłącznie na aspektach SCM związanych bezpośrednio z pracą z artefaktami projektu, ze szczególnym uwzględnieniem zagadnień, które mogą się okazać przydatne podczas użytkowania TEX-a.

Podstawowe problemy

Zarządzanie konfiguracją oprogramowania ma zapewnić spójność i kompletność wytwarzanych artefaktów projektu informatycznego (dokumentacji zarządczej, dokumentacji użytkowej, kodu programu i wielu innych). Celem jest ograniczenie ilości błędów, zwiększenie wydajności pracy zespołu zaangażowanego w projekt, a także podniesienie jakości wytwarzanego oprogramowania. Aby osiągnąć założenia należy rozwiązać podstawowe problemy, jakie pojawiają się podczas pracy złożonych zespołów ludzkich, nad projektami informatycznymi, w szczególności, gdy projekt taki ma wiele wersji, często jednoczesnych, różniących się nierzadko funkcjonalnością. Dodatkowo należy zdefiniować powiązania między poszczególnymi artefaktami tak, aby zapewnić kompletność wykonywanych operacji. Przykładowo dodanie pewnych funkcji do określonego modułu powinno pociągać za sobą odpowiednie zmiany w jego dokumentacji.

[1] *ang.* Software Configuration Management – Zarządzanie konfiguracją oprogramowania

Pierwszym problemem jest jednoczesny dostęp, modyfikacja, jednego i tego samego artefaktu, przez kilku członków zespołu w tym samym momencie. Po pierwsze, zmiany dokonane przez jednego mogą zostać zniszczone lub okazać się niepotrzebne w wyniku działania drugiego członka zespołu. Po wtóre, pojawia się problem połączenia dwu zmodyfikowanych artefaktów w jeden wynikowy, przy zachowaniu zmian wprowadzonych w obydwu wersjach.



Rysunek 1: Jednoczesna modyfikacja tego samego artefaktu przez dwóch członków zespołu.

Najczęściej stosowanym rozwiązaniem są repozytoria, z których artefakty są pobierane (*check-out*), a zmiany w nich dokonane, aktualizowane przy okazji zwrotu do repozytorium (*check-in*). Dzięki temu można zarówno kontrolować dostęp do poszczególnych artefaktów, poprzez przydzielanie praw dostępu do nich, jak i łatwo prześledzić historię zmian, każdego z nich. Dodatkowo w razie potrzeby (np. stwierdzenia błędów) można łatwo cofnąć zmiany i przywrócić poprzednią wersję. Często stosuje się również narzędzia zwane *merge-rami*, które pozwalają na częściową automatyzacje procesu łączenia wielu wersji artefaktu w jedną, wynikową. Potwierdza to obserwacja, że repozytoria o różnym stopniu złożoności, wspomagane narzędziami do łączenia wielu wersji artefaktów w jedną, są częścią składową większości dostępnych narzędzi wspomagających zarządzanie konfiguracją oprogramowania.

Kolejnym problemem jest rozwój projektu przeznaczonego do pracy w kilku odmiennych środowiskach. Za przykład posłużyć może program typu PIM, którego wersje są dostępne na platformy: Windows, Unix i PalmOS. W takim wypadku zadanie aktualizacji zmian, w tym usuwanie usterek zgłoszonych przez użytkowników, zostaje poszerzone o konieczność zachowania spójności określonej wersji programu z jego środowiskiem, tak programistycznym jak i uruchomieniowym. Często struktura tej samej wersji projektu dla różnych środowisk różni się znacząco. W takich wypadkach rozwiązanie SCM jest podstawą, integrującą produkt z narzędziami oraz środowiskiem wytwórczym i roboczym, na której budowany jest określony projekt informatyczny, co implikuje wspomniane na wstępie powiązania ze środowiskiem biznesowym.

Można rozróżnić cztery modele realizacji SCM. W prostszych przypadkach stosowany jest model **check-in/check-out** oprate na repozytorium przechowującym wersje artefaktów. Podobnie realizowany jest model **Change Oriented** gdzie zatwierdzone zestawy zmian przypisywane są do baz (*baselines*), które są podstawą określania kompletności danej wersji. W bardziej złożonych przypadkach stosuje się **Composition Model** gdzie założeniem jest określanie konfiguracji na podstawie struktury systemu. Mówimy wtedy już o SCM na poziomie systemowym. Najbardziej złożonym rozwiązaniem SCM jest **Long Transaction Model** w którym, poszczególne zespoły (członkowie zespołu) są odseparowani od zmian dokonywanych przez innych.

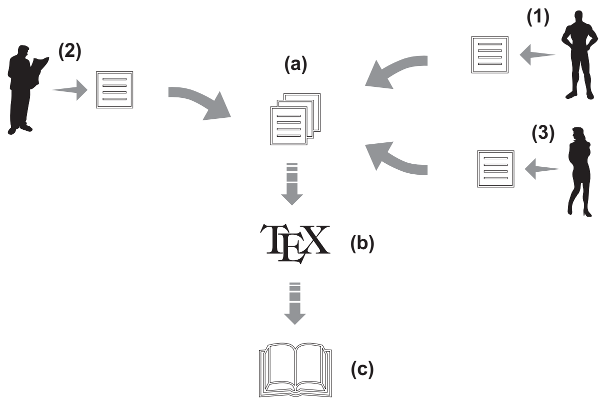
Widać z tego, że zarządzanie konfiguracją oprogramowania ma charakter długofalowej strategii, a rozmaite dostępne narzędzia, mają za zadanie jedynie wspierać i ułatwiać jej realizowanie. Poniżej wskazano kilka typowych problemów zarządzania konfiguracją, które mogą wystąpić podczas pracy z systemem składu tekstu TEX, a także ewentualne sposoby ich rozwiązania.

Zagadnienia SCM i TEX

Podobnie jak podczas realizacji i późniejszego rozwoju niewielkich, autorskich, programów komputerowych, tak i w przypadku składu prostych dokumentów w TEX-u, zagadnienia SCM nie mają aż tak dramatycznego znaczenia. Jednak w miarę wzrostu stopnia złożoności realizowanego projektu, ilość problemów z opanowaniem „całości” zaczyna być dostatecznie duża, aby uniemożliwić dalsze prace. Jest to znak, że zaistniała pilna potrzeba zarządzania konfiguracją. Przyjrzyjmy się trzem typowym problemom z zakresu SCM i potencjalnym sposobom radzenia sobie z nimi.

Książka posiadająca wielu autorów to najbardziej typowy przykład, w którym podczas składu zarządzanie konfiguracją wpływa w znacznym stopniu na ułatwienie, przyspieszenie oraz podniesienie jakości wykonywanej pracy.

Analogie do projektu informatycznego w rozumieniu rozwoju oprogramowania są daleko idące. Źródła dokumentu są odpowiednikiem kodu źródłowego programu, poszczególni autorzy *zastępują* programistów, TEX jest środowiskiem pracy, a złożona książka odpowiada wydaniu programu (*release*).



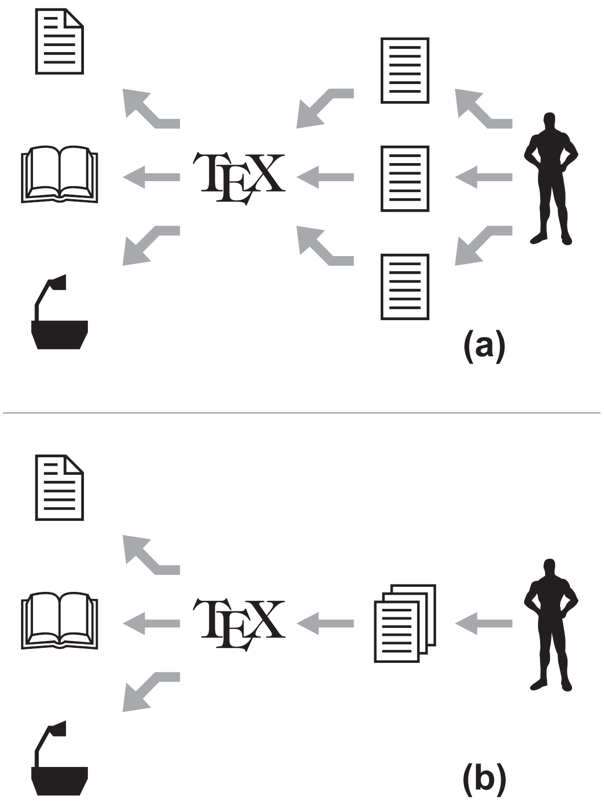
Rysunek 2: Autorzy (1), (2) i (3) tworzą rozdziały składające się na źródła książki (a), które po kompilacji (b) dają efekt końcowy w postaci złożonej książki (c).

Zadania przed którymi stajemy w takim wypadku to: (i) zapewnienie kompletności źródeł, tak aby w każdej chwili możliwa była ich kompilacja, (ii) zapewnienie dostępności ostatniej wersji źródeł, aby poszczególni autorzy mogli obejrzeć wpływ pracy innych na całość projektu, (iii) zdefiniowanie takiej struktury źródeł, aby zmiany wprowadzane przez jednego z autorów nie wpływały na pracę pozostałych. W odróżnieniu od klasycznego SCM możemy w tym przypadku zaniedbać dokumentację zmian dokonywanych przez poszczególnych autorów zwłaszcza, gdy spełniony jest postulat (iii).

W opisanej sytuacji zastosowanie dostępnych narzędzi SCM typu check-in/check-out lub Change Oriented jest oczywiste i dość proste. Kompletność zapewnia dostęp autorów do repozytorium, które przechowuje całość źródeł, a dostępność ostatniej wersji może być realizowana poprzez zdefiniowanie bazy, reprezentującej ostatnią zatwierdzoną wersję.

Można również pokusić się o próbę poradzenia sobie z zaistniałym problemem przy wykorzystaniu samego TeX-a. W rozwiązaniu takim dostęp do pliku głównego (*main file*) musi być ograniczony, tak aby modyfikacje w nim mógł wprowadzać jedynie upoważniony nadzorca projektu. Poszczególne rozdziały powinny mieć zadeklarowane w swoim ciele flagi jednoznacznie określające ich wersję. Plik główny natomiast, zawierałby instrukcje warunkowe, sprawdzające dostępność określonych flag w poszczególnych rozdziałach. Podczas kompilacji (składu) TeX napotykając flagi w kolejnych rozdziałach reagowałby odpowiednio na napotkane wersje, zgłaszając ostrzeżenia lub błędy. Rozwiązanie takie w większości przypadków nie jest jednak zadawalające. Przede wszystkim ograniczamy się jedynie do informacji o potencjalnych nieprawidłowościach, bez mechanizmów wymuszania założonej dyscypliny. W przypadku, gdy każemy TeX-owi reagować błędem na określone zdarzenie (np. wersja rozdziału starsza niż oczekiwana) ryzykujemy całkowite unieruchomienie środowiska wytwórczego projektu, co jest zdecydowanie bardziej niekorzystne, niż dopuszczenie do niezgodności wersji wewnątrz samego projektu. Jasne jest więc, że użycie mechanizmów TeX-a do rozwiązania przedstawionego problemu jest niewystarczające, a przynajmniej wymaga stałego nadzoru nad projektem, co powoduje, że może pełnić jedynie funkcje pomocnicze czy informacyjne.

Reusability czyli ponowne wykorzystanie wytworzonych komponentów, to kolejne typowe zadanie, którego wykonanie wspierają rozwiązania z zakresu zarządzania konfiguracją oprogramowania. Dla autora przygotowującego dużo rozmaitych materiałów przy użyciu TeX-a, podniesienie stopnia wykorzystania raz wytworzonego materiału, może okazać się znacznym ułatwieniem, dzięki któremu zaoszczędzi wiele czasu i pracy. Wyobraźmy sobie proces tworzenia monografii, na którą w dużej mierze składają się materiały publikowane przy różnych okazjach. Dodatkowo poszczególne tematy podejmowane w tej hipotetycznej pracy mogły zostać przedstawione w postaci referatów, pojawia się konieczność wykonania prezentacji w postaci slajdów. Oczywiste jest, że znaczna część materiału zawartego we wszystkich wymienionych pracach będzie się powtarzała. Nie znaczy to jednak, że autor nie musi zużyć dodatkowych nakładów czasu na ich wykonanie.



Rysunek 3: Brak ponownego wykorzystania raz wytworzonego materiału (a), prowadzi do niepotrzebnego wzrostu czaso i pracochłonności.

W tym przypadku zastosowanie powszechnie dostępnych narzędzi SCM nie jest już tak oczywiste jak poprzednio. Repozytorium nadal znakomicie pomaga uporządkować i kontrolować wytwarzany materiał, jednak spełnienie postulatu *reusability* wymaga dodatkowych czynności. Konieczne jest takie zaplanowanie struktury dokumentu (podział na pliki), aby poszczególne z nich można było włączać do projektu, w zależności od jego przeznaczenia, bez dokonywania w nich jakichkolwiek zmian. Przykładowo, dowody poszczególnych twierdzeń można umieścić w osobnych plikach i odwoływać się do nich w zależności od potrzeb.

Rozwiązanie takie ma jednak pewne mankamenty. Po pierwsze, wymaga starannego przemyślenia i zaprojektowania struktury powstających dokumentów, co często jest zadaniem niebanalnym. Po drugie, autor nadal musi napisać osobne pliki główne do kompilacji poszczególnych wersji. Po trzecie, efektem jest często zbędne rozczłonkowanie dokumentu, co utrudnia jego pielęgnacje (archiwizowanie, przenoszenie, wprowadzanie zmian).

Okazuje się, że znacznie lepszym rozwiązaniem dla zapewnienia ponownego użycia, jest zastosowanie mechanizmów zawartych w samym TEX-u. Używając instrukcji odnośników i etykiet, w połączeniu z warunkową kompilacją, możemy stworzyć dokument, w którym przed kompilacją zmieniamy jedynie etykietę, oznaczającą przeznaczenie danej wersji. TEX każdorazowo z jednego i tego samego dokumentu złoży odpowiedni rodzaj tekstu, w zależności od znalezionej w źródłach etykiety. Przykładowo: poniższy kod w zależności od zdefiniowanej flagi określającej przeznaczenie dokumentu spowoduje złożenie slajdów w formacie PDF lub artykułu w formacie Postscript.

```
\ifx\screenoutput\undefined
 \documentclass[11pt,a4paper,oneside]
  {article}
 \documentoutput{ps}
\else
 \documentclass[11pt,pdftex,a4paper,
        oneside]{slides}
 \usepackage[pdftex,plainpages=false,
        pdfborder={}]{hyperref}
 \documentoutput{pdf}
\fi
```

W podobny sposób można włączać w kompilacje określonych wersji dokumentu, rozdziały, akapity, rysunki, tabele, fragmenty teksty itp. W bardziej skomplikowanych przypadkach można wyobrazić sobie zastosowanie obydwu sposobów, tzn. pliki ze źródłami zawierają stosowne instrukcje kompilacji warunkowej, a całość materiału podzielona jest dodatkowo na osobne pliki, przechowywane w repozytorium narzędzia SCM.

Konserwacja systemu jakim jest TEX to ostatni omówiony i jednocześnie najbardziej złożony problem z zakresu zarządzania konfiguracją oprogramowania. W tym wypadku zarządzaniu konfiguracją podlegają nie tylko pliki źródłowe publikacji, ale całe środowisko wytwórcze. Oznacza to, że mamy do czynienia z SCM na poziomie systemowym. Należy w tym miejscu zaznaczyć, że nie przesądzamy czy zarządzaniu konfiguracją podlegają również źródła poszczególnych użytkowników systemu. Oczywiści staje się to niezbędne, jeśli użytkownicy ci wytwarzają kod źródłowy, który może nieć wpływ na funkcjonowanie samego TEX-a (np. rozwijają kolejne wersje pakietów makrodefinicji).

Dla osiągnięcia sukcesu, kluczowe jest spełnienie następujących wymogów:

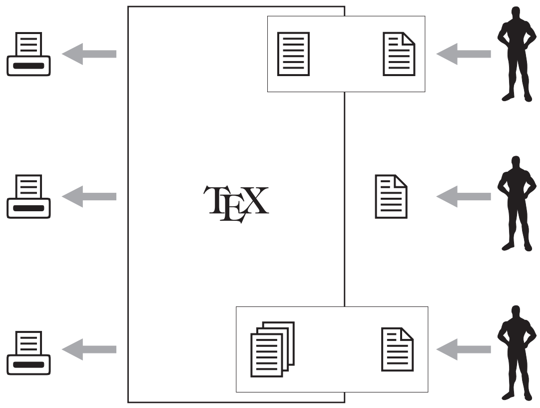
- Zapewnienie ciągłości funkcjonowania systemu
- Łatwość powrotu do poprzedniej konfiguracji
- Dokumentacja dokonanych zmian dostępna dla użytkowników

Zarządzanie konfiguracją i wersjami całego środowiska jest już zadaniem dość złożonym i wymaga stosowania zaawansowanych narzędzi SCM. W wypadku zmiany wersji całego systemu (np. TEXLive 6 do TEXLive 7) umożliwienie powrotu do poprzedniej wersji może się okazać kłopotliwe i zasobochłonne. Operacje zmiany całego środowiska wytwórczego na nowe (nową wersję) wykonuje się sporadycznie, a zmiany takie mają zazwyczaj charakter permanentny.

Inaczej wygląda sprawa uaktualniania działającego systemu o nowe komponenty lub zmiana wersji niektórych pakietów składowych. W takiej sytuacji doskonale sprawdzają się repozytoria, w których pliki źródłowe określonego dokumentu, są powiązane np. z pakietami makr potrzebnymi do ich kompilacji. Uzupełnienie takiego rozwiązania o przydzielenie odpowiednich praw odczytu i uaktualniania repozytorium, może doprowadzić do sytuacji, w której każdy z autorów używa odrębnych (wersji) pakietów do składu swoich dokumentów, nie wpływając przy tym na integralność systemu (Rysunek 4).

Mankamentem jest natomiast konieczność stosowania dość zaawansowanych narzędzi SCM wykorzystujących Composition Model lub Long Transaction Model, w celu dobrej integracji źródeł dokumentu i pakietów z nim związanych, ze środowiskiem działającego TEX-a. Warto w tym miejscu zauważyć, że na użytkowników zostaje przerzucona część zadań administratora systemu, takich jak zapewnienie poprawnego działania ze wszystkimi zainstalowanymi pakietami.

Niestety rozwiązania wykorzystujące wewnętrzne mechanizmy samego TEX-a nie są w tym wypadku wystarczające i mogą być użyte jedynie w bardzo ograniczonym zakresie. Odnosi się to szczególnie do twórców pakietów makrodefinicji, którzy testują ich działanie (różnych wersji) z określoną dystrybucją TEX-a. W takim wypadku można z powodzeniem



Rysunek 4: Poszczególni użytkownicy mogą kompilować swoje dokumenty, używając specyficznych pakietów lub różnych wersji tego samego pakietu.

stosować mechanizm kompilacji warunkowej opisany na stronie 31. Jednak przy większej liczbie użytkowników, rozwiązanie takie jest niewystarczające.

Podsumowanie

Środowisko systemu składu tekstu TEX jest złożonym systemem, w ramach którego użytkownicy wykonują czynności analogiczne do zespołów pracujących nad rozwojem projektów informatycznych. Dlatego zarządzanie konfiguracją oprogramowania ma tu dużą rolę do odegrania, przyczyniając się do zwiększenia efektywności pracy i podniesienia jakości produktu końcowego.

Dobra architektura i idąca z nią w parze elastyczność TEX-a sprawiają, że w niektórych przypadkach problemy zarządzania konfiguracją oprogramowania można rozwiązywać bez konieczności uciekania się do dodatkowych, zewnętrznych rozwiązań. Jednak przy zagadnieniach bardziej złożonych, gdzie mamy do czynienia z zarządzaniem konfiguracją oprogramowania na poziomie systemowym, musimy uciec się do specjalizowanych narzędzi. Jednak w tym przypadku korzyści z tego płynące są duże, szczególnie, gdy w grę wchodzi zarządzanie (administrowanie) systemem obsługującym dużą liczbę, niezależnych użytkowników.

Literatura

[1] Bruce Angstadt *SCM: More Than Support and Control* The Journal of Defense Software Engineering Software Technology Support Center

[2] James E. Tomayoko *Software Configuration Management* Carnegie Mellon University, Software Engineering Institute Sponsored by the U.S. Department of Defense

[3] Chris Kliewer *SENG 621: ESSAY, Software Configuration Management*

[4] Karol Frühauf and Andreas Zeller *Software Configuration Management: State of the Art, State of the Practice* INFOGEM AG - Informatiker Gemeinschaft für Unternehmensberatung, Universität Passau

[5] Laura Wingerd and Christopher Seiwald *High-level Best Practices in Software Configuration Management* Eighth International Workshop on Software Configuration Management

[6] Tresa Butler, Faith Turner, Verla Standley, Elaine Sullivan *Software Configuration Management: A Discipline with Added Value* Software Technology Support Center The Journal of Defense Software Engineering

[7] Alan Radding *Scalable SCM: Avoiding the Trauma, Disruption and Expense of Changing Software Configuration Management Tools* Rational

[8] Ming-Fang Chen, Jim Fluke, Herb Grote *A Proven Software Configuration Management for Workstation Development* NOAA Forecast Systems Laboratory

[9] Donald E. Knuth. *The TEXbook*. Addison-Weseley Publishing Company, 1984. ISBN 0-201-13448-9.