AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI

KATEDRA INFORMATYKI

Praca dyplomowa magisterska

Debuggowanie aplikacji kominukujących się asynchronicznie oparte o historię komunikatów.

History-based approach for debugging applications using asynchronous communication.

Autor: *Krzysztof Romanowski*
Kierunek studiów: *Informatyka*
Opiekun pracy: *dr hab. Arkadiusz Janik, dr inż*

Kraków, 2015

Oświadczam, świadomy(-a) odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Serdecznie dziękuję . . . tu ciąg dalszych podziękowań np. dla promotora, żony, sąsiada itp.

Spis treści

1. Wprowadzenie

1.1. Motywacja

Podczas tworzenia aplikacji asychronicznych twórcy wielokrotnie napotykają ograniczenia narzędzi którę są nie przystosowane do pracy z tą klasą aplikacji. Podstawowe techniki takie jak analiza wyjątków czy klasyczne debuggery przeważnie nie daje nam wystarczających informacji o naturze problemów. Iulian Dragos w swojej prezentacji [**?**] przedstawił koncepje asychronicznego debuggera przeznaczonego do debugowania aplikacji stworzych przy wykorzystaniu technologi Akka oraz mechanizmu Feature'ów z języka Scala. Po jej wysłuchaniu uznałem że przedstawiona koncepja jest bardzo dobra, jednakże sposoby persystencji komunikatów będą miały zby duży wpływ na działania debuggowanej aplikacji.

1.2. Cele pracy

Celem poniższej pracy jest zbadanie możliwości oraz efektywaności debuggowania aplikacji komunikujących się asychronicznie w oparciu o historię komunikatów. Głównym obszarem zainteresowań pracy będzie narzut sposobu persystowania i analizy komunikatów na czas wykonywania poszczególnych części aplikacji. Zamierzam zaimplementować, przetestować różne podejścia oraz zestawić wyniki wraz z ograniczeniami danej metody.

1.3. Środowisko

Jako że przedmiotem tej pracy nie jest stworzenie asychronicznego debuggera zamierzam wykorzystać pracę Iuliana [**?**]. Zaimpelemtowany debugger jest częścią ScalaIDE - IDE dedykowanego Scali. Zamierzam testować wydajność wykorzystując aplikacje napisane w frameworku Akka - najpopularniejszej technologi do pisania aplikacji asychronicznych opartych o wymiane komunikatów w ekosystemie Scali.

2. Pierwszy dokument

W rozdziale tym przedstawiono podstawowe informacje dotyczące struktury prostych plików LaTeXa. Omówiono również metody kompilacji plików z zastosowaniem programów *latex* oraz *pdflatex*.

2.1. Struktura dokumentu

Plik LaTeXowy jest plikiem tekstowym, który oprócz tekstu zawiera polecenia formatujące ten tekst (analogicznie do języka HTML). Plik składa się z dwóch części:

1. Preambuły – określającej klasę dokumentu oraz zawierającej m.in. polecenia dołączającej dodatkowe pakiety;
2. Części głównej – zawierającej zasadniczą treść dokumentu.

```
\documentclass[a4paper,12pt]{article}        % preambuła
\usepackage[polish]{babel}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{times}

\begin{document}                             % część główna

\section{Sztuczne życie}

% treść
% ąśężźćńłóĘŚĄŻŹĆŃÓŁ

\end{document}
```

Nie ma żadnych przeciwskazań do tworzenia dokumentów w LaTeXu w języku polskim. Plik źródłowy jest zwykłym plikiem tekstowym i do jego przygotowania można użyć dowolnego edytora tekstów, a polskie znaki wprowadzać używając prawego klawisza `Alt`. Jeżeli po kompilacji dokumentu

polskie znaki nie są wyświetlane poprawnie, to na 95% źle określono sposób kodowania znaków (należy zmienić opcje wykorzystywanych pakietów).

2.2. Kompilacja

Załóżmy, że przygotowany przez nas dokument zapisany jest w pliku `test.tex`. Kolejno wykonane poniższe polecenia (pod warunkiem, że w pierwszym przypadku nie wykryto błędów i kompilacja zakończyła się sukcesem) pozwalają uzyskać nasz dokument w formacie pdf:

```
latex test.tex
dvips test.dvi -o test.ps
ps2pdf test.ps
```

lub za pomocą PDFLaTeX:

```
pdflatex test.tex
```

Przy pierwszej kompilacji po zmiane tekstu, dodaniu nowych etykiet itp., LaTeX tworzy sobie spis rozdziałów, obrazków, tabel itp., a dopiero przy następnej kompilacji korzysta z tych informacji.

W pierwszym przypadku rysunki powinny być przygotowane w formacie eps, a w drugim w formacie pdf. Ponadto, jeżeli używamy polecenia `pdflatex test.tex` można wstawiać grafikę bitową (np. w formacie jpg).

2.3. Narzędzia

Do przygotowania pliku źródłowego może zostać wykorzystany dowolny edytor tekstowy. Niektóre edytory, np. Emacs, mają wbudowane moduły ułatwiające składanie tekstów w LaTeXu (kolorowanie składni, skrypty kompilacji, itp.).

Jednym z bardziej znanych środowisk do składania dokumentów LaTeXa jest *Kile*. Aplikacja dostępna jest dla środowiska KDE począwszy od wersji 2. Zawiera edytor z podświetlaną składnią, zestawy poleceń LaTeXa, zestawy symboli matematycznych, kreatory tabel, macierzy, skrypty kompilujące i konwertujące podpięte są do poleceń w menu aplikacji (i pasków narzędziowych), dostępne jest sprawdzanie pisowni, edytor obsługuje projekty (tzn. dokumenty składające się z wielu plików), umożliwia przygotowanie i zarządzanie bibliografią, itp.

Na stronie `http://kile.sourceforge.net/screenshots.php` zamieszczono kilkanaście zrzutów ekranu środowiska *Kile*, które warto przejrzeć, by wstępnie zapoznać się z możliwościami programu.

Bardzo dobrym środowiskiem jest również edytor gEdit z wtyczką obsługującą LaTeXa. Jest to standardowy edytor środowiska Gnome. Po instalacji wtyczki obsługującej LaTeXa, edytor nie ustępuje funkcjonalnościom środowisku Kile, a jest zdecydowanie szybszy w działaniu. Lista dostępnych wtyczek dla

tego edytora znajduje się pod adresem http://live.gnome.org/Gedit/Plugins. Inne polecane wtyczki to:

– Edit shortcuts – definiowanie własnych klawiszy skrótu;

– Line Tools – dodatkowe operacje na liniach tekstu;

– Multi-edit – możliwość jednoczesnej edycji w wielu miejscach tekstu;

– Zoom – zmiana wielkości czcionki edytora z użyciem rolki myszy;

– Split View – możliwość podziału okna edytora na 2 części.

2.4. Przygotowanie dokumentu

Plik źródłowy LaTeXa jest zwykłym plikiem tekstowym. Przygotowując plik źródłowy warto wiedzieć o kilku szczegółach:

– Poszczególne słowa oddzielamy spacjami, przy czym ilość spacji nie ma znaczenia. Po kompilacji wielokrotne spacje i tak będą wyglądały jak pojedyncza spacja. Aby uzyskać *twardą spację*, zamiast znaku spacji należy użyć znaku *tyldy*.

– Znakiem końca akapitu jest pusta linia (ilość pusty linii nie ma znaczenia), a nie znaki przejścia do nowej linii.

– LaTeX sam formatuje tekst. **Nie starajmy się go poprawiać**, chyba, że naprawdę wiemy co robimy.