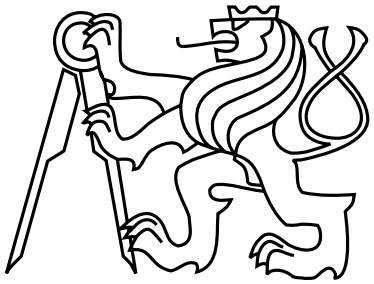
Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science and Engineering



Master's Thesis

Semantic Data Analysis and Visualization of User Interactions

Bc. Michal Švácha

Supervisor: Ing. Ivo Malý Ph.D.

Study Programme: Open Informatics

Field of Study: Software Engineering

April 27, 2016

Aknowledgements

I would like to thank dearly to my cat, dog and goldfish I never had. You were a true inspiration to me. With you, I would have never made this document happen. Also, my PlayStation 3 proved to be an amazing tool to keep me sane while writing this thesis. Thank you SONY.

Declaration

I hereby declare that I have completed this thesis independently and that I have listed all the literature and publications used.
I have no objection to usage of this work in compliance with the act §60 Zákon č. 121/2000Sb. (copyright law), and with the rights connected with the copyright act including the changes in the act.

In Prague on May 27, 2016 .....

Abstract

Current status quo in data collection is to collect everything that is available. This approach has given birth to a trend of the last couple years - "Big Data". Data that is inconveniently large for processing, interpretation and inference. When a company decides to leverage big data, it usually ends up with having too much information and no real business value. Each department uses different domain vocabularies and therefore there is no synchronization and understanding. This master thesis is trying to take the opportunity of this disorder and connect the endpoints together while leveraging big data to provide an end-to-end solution.

The focal part of this thesis is the focus on user interactions - collection of data from mobile applications. Before such collection can even happen, interactions must be defined - the "what", "when" and "why". Starting with management, over to architecture and engineering to interpreting results as the destination - uniting all steps to form a bigger picture.

Abstrakt

Momentální status quo ve sběru dat je sbírání a ukládání všeho, co je k dispozici. Tento přístup dal vzniknout trendu posledních let - "Velká data". Tedy data, která jsou nepohodlně objemná pro zpracování, výklad, a odvozování závěrů. Když se společnost rozhodne, že chce využít velká data, dopadne to většinou tak, že má příliš mnoho informací bez žádné reálné hodnoty. Každé oddělení používá vlastní doménové názvosloví a tudíž chybí synchronizace a porozumění. Tato diplomová práce se snaží využít této příležitosti neuspořádanosti pro spojení všech konců dohromady a vytvořit tak komplexní řešení.

Těžištěm práce je zaměření se na uživatelské interakce - sběr dat z mobilních aplikací. Než nějaký sběr vůbec nastane, je třeba mít interakce definované - tedy "co", "kdy" a "proč". Počínaje projektovým managementem, přes architekturu a softwarové inženýrství až k interpretaci výsledků jako konečným bodem - spojení všech kroků k vytvoření uceleného náhledu.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the lean/agile product development, it is necessary to have a formalized user feedback loops in place, to measure the product performance against various (quantitative) metrics. Such feedback loops are getting information and statistics regarding user engagement and interaction with the product. Are the users using it the way the creator imagined it or did they find any other means of utilizing it? What stops the users from doing the task they intended? Do they get everything they need and at the same time does the creator get what was expected? In other words - are the dominant means of usage incentive compatible for the users?

Not only that the current solutions for gathering such user data for both quantitative and qualitative metrics work out-of-the-box with low-level semantics only (everything is a general activity on a general resource), but also tend to run on somebody else's servers. What if the product developer is in a highly regulated market, such as pharmaceuticals, and has to own all their users' data? How can the current solutions' space be utilized and tweaked in order to fit in such schema?

The most problematic issue in large corporations is the disconnectivity of the data. It may physically be all there, however, nobody knows what and how should it be connected in order for it to make sense and drive value. Do I have good data or do I just have petabytes of useless log trace? Am I gathering what the application was inteded to do or am I only filling the database of irrelevant information. Most importantly, though - Am I gathering the information I need in a consistent fashion that coresponds to the domain of the shareholders?

Let's draw an analogy here.

Example

The management comes with a need to create an e-commerce mobile application to drive sales of their products. The main KPI (= Key Performance Indicator) is the customer turnover (= how many incoming customers buy a product) and retention (= how many customers that previously bought a product come back). This idea is passed/assigned to middle management for processing. At this point one can assume that the domain vocabulary for the project is almost the same, as the direct source (management) was the one to mediate the idea. From then on, project is passed on to various people in various departments (design, back-end,

front-end). Every department has a different technique of work and may also have different jargon. That has the effect that the first idea may be fulfilled but not properly measured. The management was clearly interested in *customer turnover* and *customer retention*. Those two activities were translated into "buying customer" and "repeated shopping" or any other equivalents. While it does bring the message in the end, it does slow down the whole process that can be automated. Starting from the development and ending with the interpretation for the management.

Scope of work

First, I will present the context of software development in a regulated market. Then I will take a look at the current tracking solutions being used in mobile development (focus is on iOS development, but most of the tools are multiplatform). I will examine and analyze their strengths and weaknesses and further propose an architecture to fit the needs of a company operating in a regulated market. And last I will propose a working PoC (Proof of Concept) tested in a real environment.

Chapter 2

Analysis

2.1 Regulated environment constraints

Regulated markets, especially pharmaceuticals have multiple rules that need to be carefully followed in order to be allowed to use new IT products. Not only is important how much value does the final product bring to the end user, but also how data storage is handled and how prone it is to exploitations and attacks.

From development process point of view it is equally important to follow specific guidelines and processes during the development phase. Each step has to be carefully documented and approved by specific audit department. For that reason, most of the big pharmaceutical companies use the waterfall model - SDLC (= Software Development Life Cycle), which enables companies to follow specific steps in order to get their software product certified. This all hassle is not for the sake of bureaucracy, it is to protect the customers and increases the level of tracability of a problem, should one ever occur. After all, it is their private data that goes on the server, so it is imperative that it is safe.

2.1.1 Personal interviews

To be able to draw any conclusions, I had to conduct interviews with stakeholders from various departments.

1. Associate Director, Applied Technology

   > "The real problem I see is the fact, that all the information I need is on somebody else's server. We can't store any sensitive, let alone confidential information *somewhere* with some random vendor. Unfortunately, sometimes sensitive data is exactly what we need to obtain from the applications to make an informed decision."

2. Associate Director, Mobile and Web

   > "Our needs for tracking KPIs are variable throughout the time and unfortunately the current tools we use are quite inflexible. Because we are in a regulated market, each change that requires new build of an application takes a longer period of time. And time *is* money."

3. Mobile Application Development Lead

   "I have noticed that the one of the biggest obstacles is how should we name what we measure. I have no vocabulary to help me during development. The only thing we have is a robust Google Analytics toolkit that only allows us to gather low-level actions."

4. UX Lead

   "Our team looks at the high level needs. We are trying to make activities in an application as smooth as possible for the user. When we design a low-fidelity prototype, we know what we want to measure. Having the opportunity to add a high-level KPI would help us a lot to gather feedback for our prototypes. We are not programmers, we don't know how to add it to the code, but I would love the idea of including the KPIs along with the prototype."

2.2 Opportunity Assessment

1. *What problem are we trying to solve?*

   The problem is that no platform allows by default storage of user data on custom servers. In a regulatory market it is vital to have that option. Also no higher semantic analysis other than defining in code customizations is provided. Enabling alignment of business language and user interaction reporting is a key part of success. Lack of interchangeability tends to end up with a vendor lock-in.

2. *For whom do we solve that problem?*

   For the entire scope of company, top to bottom. Setting a business goal, aligning it with the needs of managers, developers and users.

3. *How will we measure success?*

   Having gathered data that is securely placed on custom servers and analysed by NLP tools and vizualized by Kibana/D3.js or any other analytics frameworks.

2.3 Existing Tools

2.3.1 Google Analytics

Google Analytics is the by far most popular and widely used framework for monitoring user interactions in applications. Reports everything the developer wishes to. By default it doesn't report anything - the tool has to be activated on application start and then actions need to be wired to the framework. Action can be either wired up after certain custom occurrence has appeared (pressed button, refreshing data) or simply an identifier can be set to an actionable item (button) - then whatever happens with that item, gets reported.

The dashboard website is very detailed and responsive. All data is nicely visualized in graphs and corresponds well with the whole GA ecosystem. Higher order semantic interpretations are missing, though.

The source code is closed source and all of the statistics run ot Google servers. Data is accessible via REST API, but registration is required. Single user account is free, enterprise account is paid for. Enterprise account does NOT include an opt-out from Google servers.

2.3.2 Fabric (formerly Crashlytics)

Crashlytics was also a star framework. Acquired by Twitter, it is now a vital part all purpose platform - Fabric. Fabric is aside from a reporting tool a full featured developer platform used for variety of tasks and obstacles a developer may face - even beta version distribution that has always been a problem for iOS developers. The Crashlytics reporting has been taken a step further and is not only about reporting crashes. It also reports overall statistics, like Google. One nice feature Fabric has is by acquiring AppSee - a way of visualizing user movement in the app through the usage data. A video of steps users take in their app gives the developer a new perspective on how the app is used.

The source code is closed source and all of the statistics run ot Twitter servers. Data is NOT accessible. The whole platform is free to use. Registration is required along with installation of a custom program to install the framework parts in existing projects.

2.3.3 App Pulse (formerly Pronq)

App Pulse is a "new feature by acquisition" - acquired by HP in 2014, it is now part of the portfolio of the new Hewlett Packard Enterprise. It lets users try out their 30-day trial and then charges for everything (no free version).

App Pulse is very different from GA in a way that it reports everything at all times. The usage is fairly low - tens of kilobytes per week, but it is very thorough. Screen time, actions, movements - it is all there. No setup is required for the start. The SDK they supply simply has to be dragged and dropped in Xcode project and then it starts working out-of-the-box. The only issue is the need to have consistent naming of all views, labels, buttons etc. - as it does everything on its own, without hooking up the actionable items to the framework manually, it can be hard to determine which button was which.

The tools are closed source and all of the statistics run on HP servers. No API is provided.

2.3.4 Crittercism

Crittercism doesn't stand out from previously mentioned tools - has their own servers, SDK and works seemlessly. Has some more benchmarking than others, and seems very enterprise oriented - they enable 3rd party API integration into their system to see performance of other APIs used in the application to really find what can be the bottleneck of the app's performance.

The tools are closed source and all of the statistics run on their servers. API is provided.

2.3.5 New Relic

New Relic somewhat differs in a sense that as the only platform there is a mention on their website about "specific needs" - maybe custom server can be provided. Otherwise it is the same strategy - SDK installed in every app and statistics gets reported periodically. Nice alerting system is optionally provided - when crash occurs, web hook to ticketing system can be defined to streamline bug reports.

The SDKs are open source, the analytics runs on their servers (possibly 'not only'). No API is provided.

2.3.6 Apple

The last isn't considered framework, but it should be noticed. As companies fight for data from mobile applications - such as Twitter, who gives out their platform free for everybody, naturally the platform owners strive for keeping all that precious data for themselves. Apple announced at WWDC 2015 new iTunes Connect portal redesign and along with it also a new feature - App Analytics. It is fairly thorough in means of usage, downloads, screen time etc, but overall reporting is still very high level and really far from the code. It seems like it is not meant to be a developer tool at all, because there is no in depth code reporting. There are crashes reported, but not very detailed compared to Fabric.

These statistics are provided to every single developer of iOS apps for free on the iTunes Connect website. There is no framework and naturally Apple keeps all of the data for themselves.

Conclusion on tools

Why are they bad etc etc.

Chapter 3

Design

3.1 JIRA

JIRA is ...

3.1.1 Domain Specific Language

Domain Specific Language (in short DSL) is ...

3.2 System Architecture

PICTURE

Description

TBD depending on the picture

Components

3.2.1 Semantic Data Manager

This is the most crucial component of the whole solution - connecting JIRA and tracking capabilities. The main task is to obtain actionable items from JIRA and create configurations for application engineers to include in the source code. These configurations should be persisted for the sake of reproducibility. Persistence shouldn't be applied to any metadata (detailed measurement description) as it is subject to change in JIRA. Every issue/ticket has a finite unique ID and that should be the only persisted piece of data.

Connecting to JIRA will be handled via its JIRA Agile REST API. It should automatically retrieve a list of all projects and their subsequent issues/tickets that may or may not contain specific metadata regarding fine-grained measurement demands.

SDM will run as a stand-alone microservice and provide UI for easy measurement configuration, but also REST API, should the UI ever be replaced.

3.2.2 Tracking Engine

As a tracking engine, anything that provides API for data storage/retrieval is good enough. Google Analytics is a good feasible option, but for reasons listed in previous chapter (especially privacy), it is more convenient to use own tracking solution.

3.2.3 Tracked Device

This can be any kind of iOS mobile device - iPhone, iPad, Apple Watch or Apple TV. The key part is, that the data that is being sent to the tracking engine is in sync with the data that the semantic data server collected from JIRA.

3.2.4 Statistical Front-end

This component is the most visible one, because it interprets the collected data.

Analýza a návrh implementace (včetně diskuse různých alternativ a volby implementačního prostředí).

Chapter 4

Implementation

4.1 Deployment

UML diagram a popis komunikace

4.2 Semantic Data Manager

The central part of the project should be robust and reliable. For that reason I chose Java as the main technology. For convenience and standardization of the code-base I opted for Spring Boot framework.

The code is packaged by Maven and deployed as WAR file to AWS EB auto-scaled m3.medium instance, running Tomcat 8 server. Database runs on managed Amazon RDS (Relational Database Service), which is fast, secure and scalable deployment of database engine. The size of the instance is recommended for any application running JVM. The configuration is Intel Xeon E5-2670 v2 (Ivy Bridge), 4GB SSD storage and 3.75GB RAM. Any additional memory is handled via S3.

// JIRA - pagination problems, service account

4.3 Tracking Engine

Because the tracking engine was developed after the Semantic Data Manager, I opted to use Java again. I first tried to use play2 framework for educational purposes, but I encountered too many obstacles deploying play2 application so I had to drop it. Main reason was that play2 uses its own application server - Netty Server, which always runs on port 9000. When deploying to AWS EB (via tool Activator that prepares a package similar to WAR file - WAR files are not supported by play2), NGINX that handles HTTP requests searched for application deployed on port 5000. Because Netty does not support compile-time port configuration and NGINX does not support running Activator to set up the port during run-time I had to drop the idea of using play2. I tried to configure NGINX to not look for the application on port 5000, but AWS EB has a very limited configurability in this area. That was the end for play2 and so I used Spring Boot again.

// H2 DB, Flyweight + DB schema // REST API + JSON

The code is deployed in the same fashion as the Semantic Data Manager.

4.4 Statistical Front-end

4.4.1 Numerous

Numerous has two components and uses monolithic git repository to keep everything together. It is designed to be a standalone micro service and can be used outside of the scope of this solution.

4.4.1.1 Back-end

For computational convenience in future development (= data modeling, model training etc.), Python was selected to be the language this component will be written in.

The code is packaged in Docker image and is deployed in two EC2 t2.medium instances - one for the API and one for the database. The main difference between T2 and M3 is the processor computational power - T2 has Dual Core Intel Xeon 3.3GHz with Turbo. Also it is a Burstable Performance Instance, meaning that it is provided with a baseline level of CPU performance with the ability to burst above. T2 instances are for workloads that don't use the full CPU often or consistently, but occassionally need to burst (computing statistical models for example in my case).

4.4.1.2 Front-end

Requirements for mobile client were only for iOS devices. Therefore, as no code portability was required, I opted to go native. For native development, Apple's language Swift is used (as of March 2016, in version 2.2). Swift is a general-purpose, multi-paradigm (both object oriented and functional), compiled programming language. It was first released to support iOS and Mac OS X, now supporting also tvOS (Apple TV 4th generation and newer) and watchOS (Apple Watch). Many more cases of use are coming, because Swift compiler has been open-sourced. It is gaining popularity among non-Apple developers mainly due to its safety, robustness and ease of use.

Dependencies

// CocoaPods

User Interface

DASHBOARD screenshot + DETAIL screenshot

Color scheme of the application was provided by a graphical designer. Most of the graphics is set on code level. I am not in favor of bloated projects because of multitude of png files for every possible device. Code generated graphics may introduce some level of complexity, but the space saved on user's device is more important. Even with app-slicing

(method of distribution provided by Apple - only the resources needed for your device are downloaded), the amount of space saved is at least 3MB.

// ios-charts

Networking

// Alamofire + VPN

Data Model

// Realm.IO

Chapter 5

Testing

- Způsob, průběh a výsledky testování.
- Srovnání s existujícími řešeními, pokud jsou známy.

Chapter 6

Conclusion

6.1 Future development

The current solution can be deployed in testing environment as an MVP (= Minimum Viable Product). In order for it to run in production environment, legal steps have to be carried out in order to be fully compliant.

6.2 Technical debt

Currently unknown

6.3 Closing remarks

TBD

Bibliography

[1] M. Haindl, Ľ. Kment, and P. Slavík. Virtual information systems. In *WSCG'2000 — Short communication papers*, pages 22–27. University of West Bohemia, Pilsen, 2000.

[2] P. Slavík. Grammars and rewriting systems as models for graphical user interfaces. *Cognitive Systems*, 4(3/4):381–399, 1997.

[3] CSTUG — $\mathcal{CS}$TEX Users Group — hlavní stránka. `http://www.cstug.cz/`, stav z 2. 3. 2009.

[4] K336 Info — pokyny pro psaní bakalářských prací. `https://info336.felk.cvut.cz/clanek.php?id=504`, stav ze 4. 5. 2009.

[5] K336 Info — pokyny pro psaní diplomových prací. `https://info336.felk.cvut.cz/clanek.php?id=400`, stav ze 4. 5. 2009.

[6] Knihovna Grafické skupiny. `http://www.cgg.cvut.cz/Bib/library/`, stav z 30. 8. 2001.

[7] Grafický vektorový editor pro práce vhodný pro práci LATEXem. `http://tclab.kaist.ac.kr/ipe/`, stav z 4. 5. 2009.

[8] LATEX — online manuál. `http://www.cstug.cz/latex/lm/frames.html`, stav ze 4. 5. 2009.

[9] Wiki books LATEX. `http://en.wikibooks.org/wiki/LaTeX/`, stav z 3. 4. 2009.

[10] J. Žára, B. Beneš, and P. Felkel. *Moderní počítačová grafika*. Computer Press s.r.o, Brno, 1st edition, 1998. In Czech.

Appendix A

List of abbreviations

AWS Amazon Web Services

EB Elastic Beanstalk

EC2 Elastic Compute Cloud

KPI Key Performance Indicator

MVP Minimum Viable Product

PoC Proof of Concept

S3 Simple Storage Service

SDLC Software Development Life Cycle

UX User Experience

Appendix B

Instalační a uživatelská příručka

Tato příloha velmi žádoucí zejména u softwarových implementačních prací.

Appendix C

Obsah přiloženého CD

Tato příloha je povinná pro každou práci. Každá práce musí totiž obsahovat přiložené CD. Viz dále.

Může vypadat například takto. Váš seznam samozřejmě bude odpovídat typu vaší práce. (viz [5]):

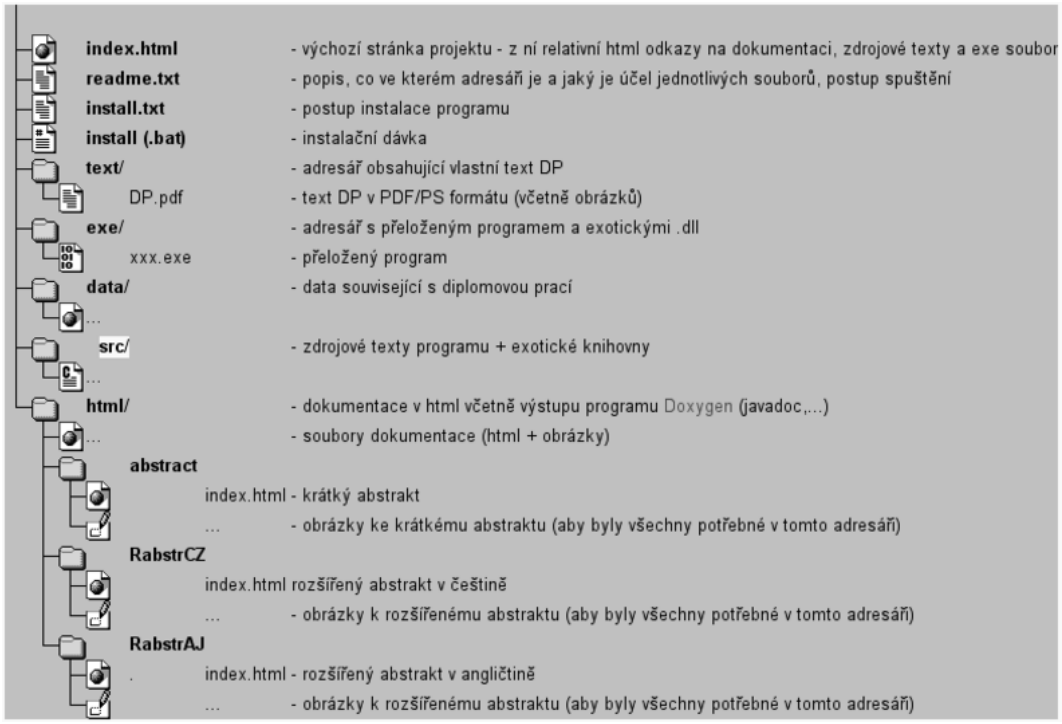


Figure C.1: Seznam přiloženého CD — příklad

Na GNU/Linuxu si strukturu přiloženého CD můžete snadno vyrobit příkazem:

```
$ tree . >tree.txt
```

Ve vzniklém souboru pak stačí pouze doplnit komentáře.

Z **README.TXT** (případne index.html apod.) musí být rovněž zřejmé, jak programy instalovat, spouštět a jaké požadavky mají tyto programy na hardware.

Adresář **text** musí obsahovat soubor s vlastním textem práce v PDF nebo PS formátu, který bude později použit pro prezentaci diplomové práce na WWW.

Appendix D

Pokyny a návody k formátování textu práce

Tato příloha samozřejmě nebude součástí vaší práce. Slouží pouze jako příklad formátování textu.

Používat se dají všechny příkazy systému LaTeX. Existuje velké množství volně přístupné dokumentace, tutoriálů, příruček a dalších materiálů v elektronické podobě. Výchozím bodem, kromě Googlu, může být stránka CSTUG (Czech Tech Users Group) [3]. Tam najdete odkazy na další materiály. Vetšinou dostačující a přehledně organizovanou elektronikou dokumentaci najdete například na [8] nebo [9].

Existují i různé nadstavby nad systémy TeX a LaTeX, které výrazně usnadní psaní textu zejména začátečníkům. Velmi rozšířený v Linuxovém prostředí je systém Kile.

D.1 Vkládání obrázků

Obrázky se umísťují do plovoucího prostředí `figure`. Každý obrázek by měl obsahovat **název** (`\caption`) a **návěští** (`\label`). Použití příkazu pro vložení obrázku `\includegraphics` je podmíněno aktivací (načtením) balíku graphicx příkazem `\usepackage{graphicx}`.

Budete-li zdrojový text zpracovávat pomocí programu `pdflatex`, očekávají se obrázky s příponou `*.pdf`[1], použijete-li k formátování `latex`, očekávají se obrázky s příponou `*.eps`.[2]

Příklad vložení obrázku:

```
\begin{figure}[h]
\begin{center}
\includegraphics[width=5cm]{figures/LogoCVUT}
\caption{Popiska obrazku}
\label{fig:logo}
```

[1] pdflatex umí také formáty PNG a JPG.

[2] Vzájemnou konverzi mezi snad všemi typy obrazku včetně změn vekostí a dalších vymožeností vám může zajistit balík ImageMagic (http://www.imagemagick.org/script/index.php). Je dostupný pod Linuxem, Mac OS i MS Windows. Důležité jsou zejména příkazy convert a identify.

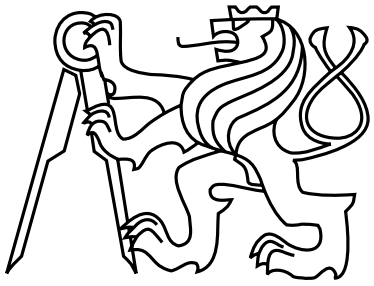


Figure D.1: Popiska obrázku

DTD	construction	elimination
|	`in1|A|B a:sum A B` `in1|A|B b:sum A B`	`case([_:A]a)([_:B]a)ab:A` `case([_:A]b)([_:B]b)ba:B`
+	`do_reg:A -> reg A`	`undo_reg:reg A -> A`
$*, ?$	the same like | and + with `emtpy_el:empty`	the same like | and + with `emtpy_el:empty`
R(a,b)	`make_R:A->B->R`	`a: R -> A` `b: R -> B`

Table D.1: Ukázka tabulky

```
\end{center}
\end{figure}
```

D.2 Kreslení obrázků

Zřejmě každý z vás má nějaký oblíbený nástroj pro tvorbu obrázků. Jde jen o to, abyste dokázali obrázek uložit v požadovaném formátu nebo jej do něj konvertovat (viz předchozí kapitola). Je zřejmě vhodné kreslit obrázky vektorově. Celkem oblíbený, na ovládání celkem jednoduchý a přitom dostatečně mocný je například program Inkscape.

Zde stojí za to upozornit na kreslící programe Ipe [7], který dokáže do obrázku vkládat komentáře přímo v latexovském formátu (vzroce, stejné fonty atd.). Podobné věci umí na Linuxové platformě nástroj Xfig.

Za pozornost ještě stojí schopnost editoru Ipe importovat obrázek (jpg nebo bitmap) a krelit do něj latexovské popisky a komentáře. Výsledek pak umí exportovat přímo do pdf.

D.3 Tabulky

Existuje více způsobů, jak sázet tabulky. Například je možno použít prostředí `table`, které je velmi podobné prostředí `figure`.

Zdrojový text tabulky D.1 vypadá takto:

```
\begin{table}
\begin{center}
\begin{tabular}{|c|l|l|}
\hline
\textbf{DTD} & \textbf{construction} & \textbf{elimination} \\
\hline
$\mid$ & \verb+in1|A|B a:sum A B+ & \verb+case([_:A]a)([_:B]a)ab:A+\\
&\verb+in1|A|B b:sum A B+ & \verb+case([_:A]b)([_:B]b)ba:B+\\
\hline
$+$&\verb+do_reg:A -> reg A+&\verb+undo_reg:reg A -> A+\\
\hline
$*,?$& the same like $\mid$ and $+$ & the same like $\mid$ and $+$\\
& with \verb+emtpy_el:empty+ & with \verb+emtpy_el:empty+\\
\hline
R(a,b) & \verb+make_R:A->B->R+ & \verb+a: R -> A+\\
 & & \verb+b: R -> B+\\
\hline
\end{tabular}
\end{center}
\caption{Ukázka tabulky}
\label{tab:tab1}
\end{table}
\begin{table}
```

D.4 Odkazy v textu

D.4.1 Odkazy na literaturu

Jsou realizovány příkazem `\cite{odkaz}`.

Seznam literatury je dobré zapsat do samostatného souboru a ten pak zpracovat programem bibtex (viz soubor `reference.bib`). Zdrojový soubor pro `bibtex` vypadá například takto:

```
@Article{Chen01,
  author  = "Yong-Sheng Chen and Yi-Ping Hung and Chiou-Shann Fuh",
  title   = "Fast Block Matching Algorithm Based on
             the Winner-Update Strategy",
  journal = "IEEE Transactions On Image Processing",
  pages   = "1212--1222",
  volume  =  10,
  number  =   8,
  year    = 2001,
}

@Misc{latexdocweb,
```

```
  author  = "",
  title   = "{\LaTeX} --- online manuál",
  note    = "\verb|http://www.cstug.cz/latex/lm/frames.html|",
  year    = "",
}
...
```

Pozor: Sazba názvů odkazů je dána BibTEX stylem (`\bibliographystyle{abbrv}`). BibTEX tedy obvykle vysází velké pouze počáteční písmeno z názvu zdroje, ostatní písmena zůstanou malá bez ohledu na to, jak je napíšete. Přesněji řečeno, styl může zvolit pro každý typ publikace jiné konverze. Pro časopisecké články třeba výše uvedené, jiné pro monografie (u nich často bývá naopak velikost písmen zachována).

Pokud chcete BibTEXu napovědět, která písmena nechat bez konverzí (viz `title = "{\LaTeX} --- online manuál"` v předchozím příkladu), je nutné příslušné písmeno (zde celé makro) uzavřít do složených závorek. Pro přehlednost je proto vhodné celé parametry uzavírat do uvozovek (`author = "..."`), nikoliv do složených závorek.

Odkazy na literaturu ve zdrojovém textu se pak zapisují:

```
Podívejte se na \cite{Chen01},
další detaily najdete na \cite{latexdocweb}
```

Vazbu mezi soubory `*.tex` a `*.bib` zajistíte příkazem `\bibliography{}` v souboru `*.tex`. V našem případě tedy zdrojový dokument `thesis.tex` obsahuje příkaz `\bibliography{reference}`.

Zpracování zdrojového textu s odkazy se provede postupným voláním programů `pdflatex <soubor>` (případně `latex <soubor>`), `bibtex <soubor>` a opět `pdflatex <soubor>`.[3]

Níže uvedený příklad je převzat z dříve existujících pokynů studentům, kteří dělají svou diplomovou nebo bakalářskou práci v Grafické skupině.[4] Zde se praví:

```
...
j) Seznam literatury a dalších použitých pramenů, odkazy na WWW stránky, ...
 Pozor na to, že na veškeré uvedené prameny se musíte v textu práce
 odkazovat -- [1].
Pramen, na který neodkazujete, vypadá, že jste ho vlastně nepotřebovali
a je uveden jen do počtu. Příklad citace knihy [1], článku v časopise [2],
stati ve sborníku [3] a html odkazu [4]:
[1] J. Žára, B. Beneš;, and P. Felkel.
     Moderní počítačová grafika. Computer Press s.r.o, Brno, 1 edition, 1998.
     (in Czech).
```

[3] První volání `pdflatex` vytvoří soubor s koncovkou `*.aux`, který je vstupem pro program `bibtex`, pak je potřeba znovu zavolat program `pdflatex` (`latex`), který tentokrát zpracuje soubory s příponami `.aux` a `.tex`. Informaci o případných nevyřešených odkazech (cross-reference) vidíte přímo při zpracovávání zdrojového souboru příkazem `pdflatex`. Program `pdflatex` (`latex`) lze volat vícekrát, pokud stále vidíte nevyřešené závislosti.

[4] Několikrát jsem byl upozorněn, že web s těmito pokyny byl zrušen, proto jej zde přímo necituji. Nicméně příklad sám o sobě dokumentuje obecně přijímaný konsensus ohledně citací v bakalářských a diplomových pracích na KP.

```
[2] P. Slavík. Grammars and Rewriting Systems as Models for Graphical User
    Interfaces. Cognitive Systems, 4(4--3):381--399, 1997.
[3] M. Haindl, Š. Kment, and P. Slavík. Virtual Information Systems.
    In WSCG'2000 -- Short communication papers, pages 22--27, Pilsen, 2000.
    University of West Bohemia.
[4] Knihovna grafické skupiny katedry počítačů:
    http://www.cgg.cvut.cz/Bib/library/
```

... abychom výše citované odkazy skutečně našli v (automaticky generovaném) seznamu literatury tohoto textu, musíme je nyní alespoň jednou citovat: Kniha [10], článek v časopisu [2], příspěvek na konferenci [1], www odkaz [6].

D.4.2 Odkazy na obrázky, tabulky a kapitoly

- Označení místa v textu, na které chcete později čtenáře práce odkázat, se provede příkazem `\label{navesti}`. Lze použít v prostředích `figure` a `table`, ale též za názvem kapitoly nebo podkapitoly.
- Na návěští se odkážeme příkazem `\ref{navesti}` nebo `\pageref{navesti}`.

D.5 Rovnice, centrovaná, číslovaná matematika

Jednoduchý matematický výraz zapsaný přímo do textu se vysází pomocí prostředí `math`, resp. zkrácený zápis pomocí uzavření textu rovnice mezi znaky `$`.

Kód `$ S = \pi * r^2 $` bude vysázen takto: $S = \pi * r^2$.

Pokud chcete nečíslované rovnice, ale umístěné centrovaně na samostatné řádky, pak lze použít prostředí `displaymath`, resp. zkrácený zápis pomocí uzavření textu rovnice mezi znaky `$$`. Zdrojový kód: `|$$ S = \pi * r^2 $$|` bude pak vysázen takto:

$$S = \pi * r^2$$

Chcete-li mít rovnice číslované, je třeba použít prostředí `eqation`. Kód:

```
\begin{equation}
  S = \pi * r^2
\end{equation}

\begin{equation}
  V = \pi * r^3
\end{equation}
```

je potom vysázen takto:

$$S = \pi * r^2 \tag{D.1}$$

$$V = \pi * r^3 \tag{D.2}$$

D.6 Kódy programu

Chceme-li vysázet například část zdrojového kódu programu (bez formátování), hodí se prostředí `verbatim`:

```
         (* nickname2 *)
Lego> Refine in1
             (do_reg (nickname1 h));
Refine by  in1 (do_reg (nickname1 h))
   ?4 : pcdata
   ?5 : pcdata
          (* surname2 *)
Lego> Refine surname1 h;
Refine by  surname1 h
   ?5 : pcdata
          (* email2 *)
Lego> Refine undo_reg (email1 h);
Refine by  undo_reg (email1 h)
*** QED ***
```

D.7 Další poznámky

D.7.1 České uvozovky

V souboru `k336_thesis_macros.tex` je příkaz `\uv{}` pro sázení českých uvozovek. „Text uzavřený do českých uvozovek.“