Software Project: Introduction

Alexis Palmer Magdalena Wolska

WS 2015/2016

Overview

- Time and Place: Di 14-16/16-18 SR 16 (INF 328)
- Website and additional Materials:
  `www.cl.uni-heidelberg.de/courses/ws15/softwareprojekt/`
- Requirements:
  Programmierprüfung, participation in Ressourcenvorkurs
- Expected work load and credit points:
  6 LP $\Rightarrow$ ca. 12h/Week + 4LP ÜK

The Software Project (1)

The aim of the software project is to accomplish *a computational linguistic (CL/NLP/ML) task*

- working in *autonomous teams*, you will
- *plan*
- *implement*
- *test / evaluate*
- *document* und *present* your system.

---

adapted from: Stefan Riezler & Laura Jehl; Yannick Versley

The Software Project (2)

At the end of the software project, you will have learned how to ...

- starting from a **theoretical approach/idea**, make a concrete plan for **realization of the approach** (**Projektplan**)
- **implement** the plan as a team
  and extend the original approach with additional functionalities
- **present** and **document** the realized project

Approach and Project Plan

. . . starting from a theoretical approach/idea, make a concrete plan for its realization:

- describe the **Task** in your own words
- define **Data Representations/Formats** and **System Components/Modules** that combine appropriately to realize the desired functionalities
- work out dependencies between modules, in order to make a concrete **Schedule** and a concrete **Distribution of Tasks**
- specify the **System architecture** and **Data format(s)**

Carrying out the Projects

. . . implement the plan as a team, and extend the original approach with additional functionalities.

- **Implement** the specified modules **according to plan**
- Test and document the modules such that **other team members** can also use them
- Ensure that you notice and promptly react to. . .
  - **Needs for coordination** with each other
  - **Questions and Problems**
  - Deviations from the planned schedule

  $\Rightarrow$ *Proactively* develop appropriate ways of doing this.

Documentation and Presentation

. . . present and document the implemented project:

- so that other teams understand **what you have done**
- and also the **Challenges** and **Difficulties** involved in your project
- identify problems with the progress of the project and/or weak points of the implementation
  - What was the problem?
  - What went wrong?
    What limitations does the system have as a consequence?
  - Why?
    (i.e. how *could* you have done it differently)
  - Is there anything that you *should* have done differently?

Working as a Team

- **Shared Time**
  (per Skype, at ICL, meetings with advisors)
  - for discussions
  - for working together
  - for integrating modules/components
- Shared **Logbooks** (Wiki, GoogleDocs, Trello, etc.)
  . . . e.g. calendar, logbooks, documents
- Shared **Process**
  - weekly status reports of all team members
  - common planning process
- Shared **Decisions**
  - consider appointing a project manager
  - clearly define areas of responsibility

What are expected time-consuming aspects of projects?

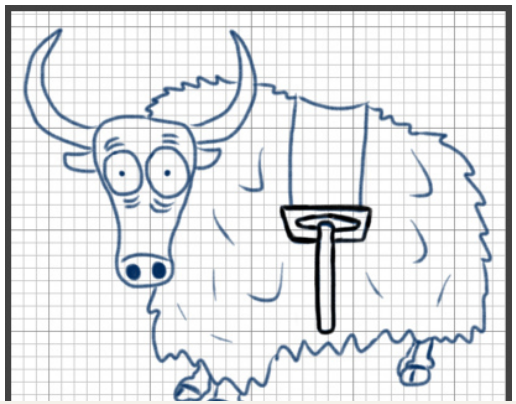
- Identify, read, and understand relevant articles/background material
- Find or collect **Data** and get it into suitable format
- Design and develop **Software**
- Write **Documentation** for Software (with details of Architecture and Implementation, Tutorials)
- Develop **Presentations** und other Deliverables

What are potential additional time-consuming aspects of projects?

- **Unexpected Behavior** of Software / Debugging
  (your own modules or external libraries and/or tools)
- **Compute/Processor Time**
  (more than 10 sec./ more than 10 min./ more than 8h/ more than 72h)
- Difficult/Odd **Special Cases**
  (Annotation, Rule-writing)
- Trying out **Alternatives**
- Cost of **Coordination** (external / in the group)

Where do we not want to spend time?

- **Missing Documentation**: If you have to spend a long time figure out how your own code works, something is wrong.
- **Yak Shaving**
  Time spent solving problems that are neither urgent nor important.
- **Feature Creep**
  Adding more and more features without actually making progress.
- **Rat holes**, **Bike shedding**
  Long discussions about unimportant things, when the important problems are difficult.



Broad Schedule

- *today* Introduction of the Course and the Projects
- 27.10. "Best practices" / Forming teams
- 3.11. Kickoff-Meetings (Project Groups)
- 1.12. Research plan due: Goal, Methods, Data, Tools, Schedule
- *Subsequently:* Group work, weekly meetings with Advisors
- 8.12. Presentations: Specification, Commit to Version 0.0
- 2.2. Presentation and Demo: finished systems!

Idea 1: Dictation-based Listening Comprehension for Language Learning

- Background: dictation exercises improve listening comprehension for language learners
- Goal: build system to allow learners to practice dictation
- Main project aspects: data collection, capture learner input, evaluate input
- Research question: explore various alignment algorithms for evaluating learner input
- Finally: deploy and test system with real users!

Idea 2: Visualization of Linguistic Features of Learner Essays

- Background: native language identification (NLI) aims to identify native language of learners writing essays in other languages (here, English)
  many systems have been built for this classification task
- Goal: build a web-based visualization tool for exploring linguistic features used for classification in order to form hypotheses about links between proficiency & L1, quality of features
- Main project aspects: process existing corpora of learner essays, develop visualization tools, focus on analysis of data
- Finally: present initial hypotheses for further investigation

Idea 3: Discourse features for Native Language Identification

- Background: one aspect of learning to write in a new language is how to convey the flow of a document. One way this is realized is by discourse connectives (e.g. *therefore, however, perhaps*) and discourse relations.
- Goal: evaluate effectiveness of various discourse-related features for identifying both native language and proficiency level of learners
- Main project aspects: feature extraction, application of discourse parsers, building and testing classifiers
- Finally: link to visualization from Project 2 - what do these discourse features actually capture?

Idea 4: Parallel Concordancing for Comparative Literary Analysis

- Background: concordancing is a core method in both quantitative and qualitative literary analysis, but current tools are generally limited to one corpus/text at a time.
- Goal: build system that allows simultaneous exploration of two corpora/texts based on existing mark-up (e.g. parsing)
- Main project aspects: data collection and preprocessing (parsing), conversion to TEI format, build flexible search and concordance system
- Finally: automatically confirm/refute existing manual analyses of particular texts

Idea 5: Classifying lexical aspect in German

- Background: verbs in any language have some tendencies as to whether they usually represent *events* or *states*.
  - *Johanna liebt Maria.* (stative)
  - *Johanna füttert die Katze.* (eventive)
  - *Ich habe schon gegessen.* (ambiguous)
- Two levels of classification: type-level and token-level (i.e. in context)

Idea 5: Classifying lexical aspect in German

- Goal: Develop a type-level system for classifying lexical aspect of verbs in German
- Main project aspects: collect and preprocess data, feature extraction, type-level features extracted from large corpus, build and test classifiers
- Finally: extend to token-level classification

Idea 6: Morphological Paradigm Induction for Uspanteko

- Background: Low-resource languages pose special challenges compared to high-resource languages. An important task for low-resource languages is morphological analysis. Uspanteko is a Mayan language with a complex but concatenative morphological system.
- Goal: develop a system to learn morphological paradigms from data, exploring both supervised and unsupervised approaches
- Main project aspects: some data preprocessing (annotated corpus already exists), supervised morphological analysis, unsupervised induction (discovery) of morphological paradigms
- Finally: perhaps extend system to closely-related Mayan language

Project Preferences

No later than 23.10: Please send a mail to `wolska@cl.uni-heidelberg.de,palmer@cl.uni-heidelberg.de`, with

- Subject: "SWP Project Preferences"
- 2-3 Project preferences (ranked list)
- Self-evaluation:
  - In which programming language(s) do you prefer to work? How do you estimate your competence in those? (1: Beginner, 5: Much experience; Mult. languages possible) Ex.: *Haskell(4), Go(2), OCaml(3)*
  - What practical programming experience do you bring to the project? Ex.: *Prog1/2, 1 Year internship at IBM*
- If you already have a preferred group in mind: the names of your group members

Forming teams: 27.10.

- (ideally) **3-4 Members** per team
- tentative project assignments
- preferences for teams will be followed as much as possible
- everyone gets a project

Kickoff Meetings in Teams: 03.11

- read and discuss articles
- identify questions or unclear things
- elaborate plan (tasks, timeline, responsibilities)

Project Plan: to be sent on 01.12

- Description of the overall task
- What is the **Goal**?
  (ideally: *specific, verifiable und realistic*)
- What is the plan for **Solving the Problem**?
  (what Assumptions / Methods need to be considered?)
- How will your System/Method be **evaluated**?
- Which **Tools** will you use?
- Which **Data** will you use (both for training and for evaluation)?
- Estimation of your work and time plan

$\Rightarrow$ *written, c. 2 Pages/Team*

Specification Presentation: 8.12.

Specification of Content

- Description of the problem, Approach to the solution, Evaluation ($\rightarrow$ *Research plan*)
- Necessary resources, methods/algorithms to be used

Modularization und **Distribution of Tasks**

- Definition of **Tasks**, **Modules** und **Dependencies** (also: *preliminary* vs. *final* version of the system)
- Association of **Names** with Tasks

Software specification

- Data formats
- (Program-)Interfaces
- Data structures
- Estimation of remaining work load und dependencies

Presentation, max. 25-30 Minutes

Requirements for Success

- Demo Day: **Presentation** of the finished System, Lessons Learned
  Final Presentation 2.2.
- **Completion of the Project** (careful and thorough Planning, Implementation, Documentation)
  Submission of Code+Documentation by ≈12.3.
- **Working together as a team**

Backup Idea 1: Semantic Textual Similarity (STS) using word2vec

- Background: Textual similarity is the problem of estimating how semantically similar two sentences or discourse are to each other.
- Goal: Explore effectiveness of word representations (from word2vec/word2phrase, e.g.) in addition to standard features for STS
- Main project aspects: obtain standard data and evaluation scripts, use existing word representation toolkits, build systems with and without word2vec features, compare
- Finally: consider other types of word/feature representations?