# Generic tools and individual research needs in the Digital Humanities – Can agile development help?

Gerhard Heyer[1], Christian Kahmann[2], Cathleen Kantner[3]

**Abstract:** Many Digital Humanities research projects from many different target disciplines regularly encounter the same recurring key problems and key procedures such as preprocessing, standard text analytics, and visualization, which would be very time consuming if conducted without DH tools. This calls for the use of generic platforms. However, there is a trade-off, since different researchers from many different disciplines look at their objects from different theoretical perspectives. This raises the general question how we can deal with this very typical conflict, and whether *agile development* might be a suitable development method to cope with it. By addressing this question, we report on an experience during a DH summer school as a condensed experiment in dealing with this trade-off using the *iLCM* as a generic platform. In summary, although we have not arrived at a procedural solutions for balancing individual user needs and generic problems which call for generic tools, our summer academy experience well illustrates the high potential of a software eco-system supporting the approach of agile development in Digital Humanities, and may help to better understand the role of generic software tools and their role in DH.

**Keywords:** interactive modelling; agile development; iLCM

## 1 Introduction

If we take the term digital in the Digital Humanities not only to refer to digital infrastructures like the ESFRI initiatives CLARIN[4] and DARIAH[5], but also to refer to a *digital* – in contrast to an analog – representation of research questions, then a key issue in the Digital Humanities is to transform a Humanities or Social Science research question into a format where *digital data* and *computational analyses* become key components of the research. Assuming thus that theory-driven empirical research in the Humanities or Social Sciences defines a need for computational methods in a new digital research paradigm in their own fields, this implies that Digital Humanities is more than just using software tools for e.g. text analytics or image analysis in the domain of Humanities and Social Science, but that it is an effort of

[1] University Leipzig, Department for Natural Language Processing, Augustusplatz 10, 04109 Leipzig, Germany heyer@informatik.uni-leipzig.de

[2] University Leipzig, Department for Natural Language Processing, Augustusplatz 10, 04109 Leipzig, Germany kahmann@informatik.uni-leipzig.de

[3] University Stuttgart, Department of International Relations and European Integration, Breitscheidstr. 2 , 70174 Stuttgart, Germany cathleen.kantner@sowi.uni-stuttgart.de

[4] https://www.clarin.eu/

[5] https://de.dariah.eu/

doi:10.18420/inf2019_ws19

- a transdisciplinary cooperation between researchers from different scientific fields with dissimilar research traditions
- in order to find answers to challenging research questions in their respective fields
- by employing digital data and methods.

Yet, how can this transformation be technically supported? Traditional software engineering mainly focused on domains like business or production workflows: Software development roughly follows a well defined path of first defining the application requirements, then implementing the specifications, and finally testing and fine-tuning the resulting software [So12]. This model, generally known as the *waterfall*, or *V-model*[Ba01], also seems to be the main paradigm in the Digital Humanities: In most projects it is expected that scholars (be it from the Humanities, the Social Sciences, or Computer Science) are able to define the technical requirements for the tools to be employed from scratch. Computer Scientists then implement it, and finally – often only at the very end of DH projects – the resulting data are tested and evaluated against the original hypotheses by the Humanities and Social Science scholars. The traditional software engineering approach is rather rigid, time consuming, and inflexible when it comes to finding an optimal match between the requirements and the available algorithmic solutions [Ma03]. In particular, iterative adaptations between requirements on the one hand, and the implemented functionalities on the other, are not encouraged. This suffices in domains of applications where user needs can be well defined in advance and implemented based on a common knowledge shared between users and software engineers of what is technologically possible. However, it does not work optimally in a context of non-standard tasks, lack of routine, unknown dynamics of data, theory-building, technology, highly interdisciplinary, and possible surprising findings.

## 2 Agile development

With the rapid development of new digital applications and technological changes, however, even in some business applications, a more flexible paradigm, the so-called agile development has been introduced. *Agile development* is based on the idea that a development group's "team ability” is crucial to the success of a project. Rather than optimizing on specifications and the individual developers efficiency, the goal is to change the process of software development into a team effort to develop quality software in a short time. *Agile Software Development* is perceived as being a bit like playing a game [Co02]. The Manifesto for *Agile Software Development* reflects this idea: While traditional software engineering focuses on comprehensive documentation of requirements, code, and usability, the paradigm of *Agile Software Development* values „individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan“ [Be01]. The twelve guiding „principles of Agile Software“ spell out the need and value of working software in short interactive development cycles open for revisions and fine-tuning. Development methodologies like Scrum[6] help to implement the guiding principles of agile development.

We would like to propose the *Agile Software Development* paradigm for research in the Digital Humanities, too: Rather than deciding upon software requirements too early in the collaboration, we would like to recommend an iterative procedure of mutual rapprochement between Humanities, or Social Scientists, on the one hand, and Computer Scientists on the other as a means to arrive at the best results with respect to both parties research goals involved. *Agile development* in Digital Humanities projects would allow for an early interaction between defining software requirements and reflecting on how they can possibly be implemented in dialogue with the further specification and adaptation of the research questions of theory-driven empirical research in the Humanities or Social Sciences. In practice, *agile development* in mixed teams of researchers can effectively be supported by using a software ecosystem that allows to rapidly select suitable data and interactively develop research hypotheses based on a truly *digital* representation of the research question.

Using *agile development* as a guiding paradigm for DH projects has also been followed in projects like Derrida's Margins [Ko18]. In fact, many Digital Humanities research projects from many different target disciplines regularly encounter the same recurring key problems and key procedures such as preprocessing, standard text analytics, and visualization, which would be very time consuming if conducted without DH tools. This calls for generic platforms such as *weblicht*[7] or generic tools such as the interactive Leipzig Corpus Miner (*iLCM*[8]). By "generic" we mean that the tool is not tied to one specific area of application, but can easily be ported to other domains and applications. Generic tools support the development of software in DH by automating general tasks that are not specific to a particular application (such as preprocessing and defining task-specific data collections) , and thus help to structure the development process and to improve development productivity. However, there is a trade-off, since different researchers from many different disciplines look at their objects from different theoretical perspectives:

- They use different corpora with different characteristics for which a generic platform is more or less well prepared,
- they discover different ways of using the generic platform, some of which are surprising, and
- they call for different additional tools (e.g. POV / emotion classification tool) that are not part of the original platform, etc.

Using a generic platform, this raises the general question how we can deal with this very typical conflict, and whether *agile development* might be a suitable development method

[6] https://www.scrumguides.org/
[7] https://www.clarin-d.de/de/sprachressourcen-und-dienste/weblicht
[8] http://ilcm.informatik.uni-leipzig.de/

to cope with it. By addressing this question, we want to report on an experience during a DH summer school as a condensed experiment in dealing with this trade-off. We were surprised, how in a very short time four parallel student projects on very different subjects from different scientific backgrounds could be developed quite far in adapting the generic *iLCM* to very different user needs by engaging in a process that has a lot in common with agile software development.

## 3 The eco-system iLCM

The *iLCM* project jointly carried out by the Computer Science Department of Leipzig University and GESIS, pursues the development of an integrated research environment for the analysis of structured and unstructured data in a "Software as a Service" architecture (SaaS). The research environment addresses requirements for the quantitative evaluation of large amounts of qualitative data with text mining methods as well as requirements for the reproducibility of data-driven research designs in the Social Sciences [Ni18]. *iLCM* is not a stand-alone program, but rather a server infrastructure comprising a number of components including a document database (MariaDB), an NLP pipeline for preprocessing text data (spaCy), a full-text index (Solr), a collection of text mining processes (for this, we rely on a selection of mature external packages for the R statistical programming language and additional own implementations), and finally a web application GUI (R Shiny). To make the infrastructure available as a decentralized installation for end-users, it is embedded in a virtual machine ensemble (Docker), which can be easily set up with predefined configuration scripts. Docker ensures the executability of the *iLCM* on every system with a running docker environment. All required libraries, software dependencies, and R packages are provided with the docker containers. To set up the *iLCM* container ensemble, docker compose, which automates the creation of all involved services in separate containers, is used. In summary, *iLCM* integrates components for document management and retrieval, R scripting capabilities for text data, and a GUI for analysis process management and result visualization to enable researchers to conduct text mining on large collections in a systematic manner.

## 4 Agile development using iLCM at summer school

We tested the *iLCM* in a real life scenario using data from German and British newspapers during a 2 week summer school of Studienstiftung des deutschen Volkes[9]. During the first week, the students developed small Social Science research projects stating specific hypotheses. The comparison between German and British newspaper coverage of Brexit was one of those questions. The students decided to deploy a form of sentiment analysis on relevant documents which was implemented within the *iLCM* infrastructure. Throughout the second week, several principles of the agile development manifesto had been adopted.

[9] https://www.studienstiftung.de/

Using the *iLCM* simplified data acquisition, data management, data preprocessing, filtering the data, extracting and selecting relevant documents in a processable format. Moreover, the *iLCM* provided an integrated development environment with a set of necessary libraries and functions already installed (R-Studio Server with pre installed libraries as running Docker container having shared volumes with the R-ShinyApp container). All this converged towards the production of a first prototype in a very short period of time (1st principle). Apparently, *agile software development* principles help to get users on their way in working with a generic DH platform such as the *iLCM*, and encourage them to continue with their research on their own. We do not claim to be the first and only to apply agile software development principles and we do not propose them in a 'fundamentalistic' or all to naive attitude. However, in the course of the realization of the project, it became apparent that the students from various fields of science, viz. politics and computer science, played the role of stakeholders as-well as developers simultaneously. Therefore, the principles demanding close communication and feedback between the partners, were intensely realized. Generic tools are useful, but have to be adapted to different individual user needs. The first version of the implementation used two pre defined sentiment lists with words and a binary assignment towards positive or negative sentiment expression. After applying the initial algorithm to the selected documents, the students identified limits of binary sentiment weights leading to incomprehensible results (3rd, 4th, 6th and 7th principle). These outcomes lead to a new feature request (2nd principle): A list of sentiments with continuous weights to obtain more reasonable results. The application of the principles of agile development (fast first prototype) led to new and previously unseen demands on the software. A more classical implementation strategy would not have offered the possibility to integrate further requirements and consequently new implementation methods so spontaneously. The already described approach in addition to a visualization of the results of the sentiment analysis of the newspaper collection on Brexit over time using R-plotly, was successfully implemented and incrementally improved (see 1) by the students. Due to the lack of time and the exemplary character of the student's projects, we also consider the principle of simplicity (10th principle) as one of the basic features of the cooperation. Apparently, platform operators learn in that process more about their needs as users, they develop a kind of Beta-Version of possible new features and learn how they can improve generic platforms such as the iLCM. This is a second very important positive effect.

## 5 Conclusion

Although we have not arrived at a procedural solution for balancing individual user needs and generic problems which call for generic tools, our summer school experience well illustrates the high potential of a software eco-system supporting the approach of agile development in Digital Humanities, and may help to better understand the role of generic software tools and their role in DH.

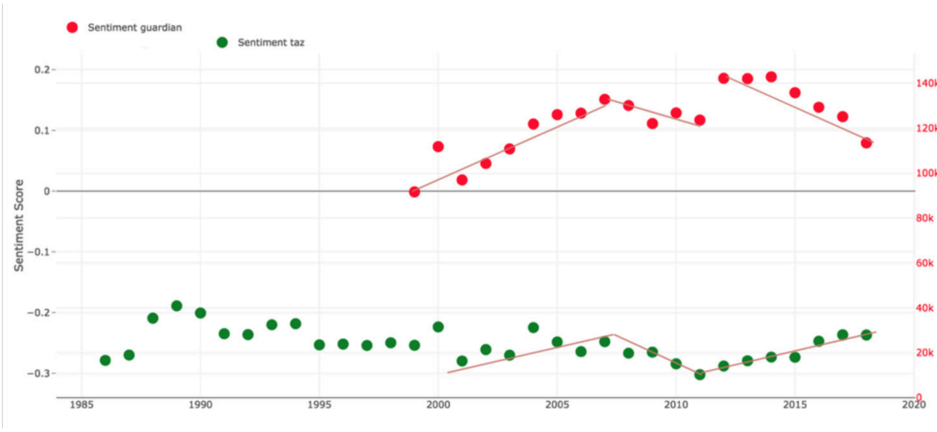

Fig. 1: Visualization for implemented Sentiment Analysis created by the students using the iLCM infrastructure. In the graphic the measured sentiments for documents (from the German newspaper *taz* and the British newspaper *the Guardian*) dealing with *Brexit* are shown in a diachronic matter. The different result levels are caused by the used sentiments sets, which differ for the two languages. Therefore the relative progressions is of main interest showing a worsening tendency since 2012 for the Guardian and in contrast to that an improving sentiment in the German texts since 2011.

## Bibliography

[Ba01] Balzert, Helmut: Lehrbuch der Software-Technik. Bd. 1. Software-Entwicklung. Spektrum Akademischer Verlag, 2001.

[Be01] Beck, Kent; Beedle, Mike; van Bennekum, Arie; Cockburn, Alistair; Cunningham, Ward; Fowler, Martin; Grenning, James; Highsmith, Jim; Hunt, Andrew; Jeffries, Ron; Kern, Jon; Marick, Brian; Martin, Robert C.; Mellor, Steve; Schwaber, Ken; Sutherland, Jeff; Thomas, Dave: Manifesto for Agile Software Development, 2001.

[Co02] Cockburn, Alistair: Agile Software Development. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[Ko18] Koeser, Rebecca Sutton: Lessons learned from building "Derrida's Margins", 2018.

[Ma03] Martin, Robert Cecil: Agile Software Development: Principles, Patterns, and Practices. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2003.

[Ni18] Niekler, Andreas; Bleier, Arnim; Kahmann, Christian; Posch, Lisa; Wiedemann, Gregor; Erdogan, Kenan; Heyer, Gerhard; Strohmaier, Markus: ILCM - A Virtual Research Infrastructure for Large-Scale Qualitative Data. In (chair), Nicoletta Calzolari (Conference; Choukri, Khalid; Cieri, Christopher; Declerck, Thierry; Goggi, Sara; Hasida, Koiti; Isahara, Hitoshi; Maegaard, Bente; Mariani, Joseph; Mazo, Hélène; Moreno, Asuncion; Odijk, Jan; Piperidis, Stelios; Tokunaga, Takenobu, eds): Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018). European Language Resources Association (ELRA), Miyazaki, Japan, May 7-12, 2018 2018.

[So12] Sommerville, Ian: Software Engineering. Pearson Studium, München, 9 edition, 2012.