Journal of Computer and Communications, **2013, 1, 32-39**
http://dx.doi.org/10.4236/jcc.2013.15006 Published Online October 2013 (http://www.scirp.org/journal/jcc)

Scientific Research

Smart Development Process Enactment Based on Context Sensitive Sequence Prediction

Andreas Rausch, Michael Deynet

[1]Chair of Software Systems Engineering, Department of Computer Science, TU Clausthal, Clausthal-Zellerfeld, Germany; [2]Embedded Systems Development, Fraunhofer IESE, Kaiserslautern, Germany.
Email: andreas.rausch@tu-clausthal.de, Michael.Deynet@iese.fraunhofer.de

Received July 2013

ABSTRACT

Actual software development processes define the different steps developers have to perform during a development project. Usually these development steps are not described independently from each other—a more or less formal flow of development step is an essential part of the development process definition. In practice, we observe that often the process definitions are hardly used and very seldom "lived". One reason is that the predefined general process flow does not reflect the specific constraints of the individual project. For that reasons we claim to get rid of the process flow definition as part of the development process. Instead we describe in this paper an approach to smartly assist developers in software process execution. The approach observes the developer's actions and predicts his next development step based on the project process history. Therefore we apply machine learning resp. sequence learning approaches based on a general rule based process model and its semantics. Finally we show two evaluations of the presented approach: The data of the first is derived from a synthetic scenario. The second evaluation is based on real project data of an industrial enterprise.

Keywords: Software Engineering; Software Process Description Languages; Software Processes; Process Enactment; Process Improvement; Machine Learning; Sequence Prediction

1. Introduction

Customers are placing growing demands on the software industry. They are looking for more complex products and at the same time that are easier to use, have higher quality, and are faster produced and shipped. The continuous increase in size and functionality of those software systems [1] has now made them among the most complex man-made systems ever devised [2].

Following a statement of Boehm [3], the ability of any industry enterprise to survive the rough market conditions will depend more and more on software and hence the capabilities to deliver in time, budget and quality required software. For that reasons over the years, a variety of software process models have been designed to structure, describe and prescribe the software systems construction process.

These models range from generic ones, like the waterfall model or the spiral model, to detailed models defining not only major activities and their order of execution but also proposing specific notations and techniques of application, like for instance RUP or V-Modell XT—also called as rich process models. In the last decade agile process models, like Scrum or Kanban, have become more and more popular.

Independently what kind of process model you apply, it can only strengthen your software development capabilities as long as you apply it correctly. But, there is huge gap between the defined and the applied development process [4-7]. As shown in [4-7] one reason for this discrepancy is that a predefined, even organization specific adapted, development process model cannot reflect the specific constraints of the individual project. Tom DeMarco even mentioned about the nature of process models and methodologies in [8]. It doesn't reside in a fat book, but rather it is inside the heads of people carrying out the work.

Consequently following this idea, we describe in this paper an approach to smartly assist developers in software development process application. The basic idea is to observe the developer's actions and thereby collect the development process knowledge inside the heads of people carrying out the work. Hence the project process history provides the process knowledge for our approach. Based on this process knowledge we predict the next development step. Therefore we apply machine learning

resp. sequence learning approaches based on a general rule based process model and its semantics.

The rest of the paper is structured as follows: Section 2 contains a shore overview of the related work in the area of software process languages and sequence learning. In Section 3 we present an overview of our software process description language. Our approach for a smart development process enactment based on context sensitive sequence prediction is shown in Section 4. In the following Section 5 we show two evaluations of the presented approach: The data of the first is derived from a synthetic scenario. The second evaluation is based on real project data of an industrial enterprise. A short conclusion in Section 6 rounds the paper up.

2. Basic Terms, Related Work and Objectives

As already mentioned the goal of software processes is to help us to successfully develop and deliver (complex) software systems. A software process is a set of activities in a project which is executed to develop and produce the desired software system. A software process description is a textual representation of a software process. To elaborate software process description in a well-defined manner we use software process description language. Using software process description languages has many advantages: The clear defined syntax and semantics enable a tool-based interpretation. Thus, controlling, planning, and coordination of software projects can be (semi-)automated.

Various software process description languages have been developed. A good overview of software process languages provides surveys like [9-13]. We distinguish between three different language paradigms: rule based languages (pre-/postconditions), net based languages (petri nets, state machines), or imperative languages (based on programming languages). In addition there are languages using multiple paradigms.

Rule based languages have loosely coupled steps, which are flexibly combinable. The advantage is that developers can execute the step in their own manner. The disadvantage is that there exists no tool support during process. Thus, there is a lower benefit for the developer during process execution.

Imperative and net based languages implement the step order directly. The step order corresponds to a generally valid order. Hereby, a tool based process execution is possible. The problem is that the order of the steps defined in the process description does not reflect the developer's working method.

For that reason our claim is to define a process description language which prevents these disadvantages in a way that

A) the development process model does not prescribe the order of development steps the developer has to execute

B) the process enactment environment guides the developer during process execution by context sensitive recommending the next development steps.

Therefore we adapt sequence prediction approaches, which are a subgroup of machine learning approaches, to process enactment environments. Hartmann and Schreiber describe different sequence prediction algorithms in [14]. The sequence prediction technique which is adopted in this paper is based on Davison/Hirsh [15] and Jacobs/Blockeel [16].

3. Overview of the Process Description Language

In this section we present a short overview of the concepts of our process modeling language. This language contains only elements which are needed to assist the user during process enactment. The language does not contain other elements like phases or roles although these concepts could be easily added. We have designed a process language based on pre-/postcondition. One key element in our language is a step which is an activity during process enactment. For example there are existing steps like "Map requirement to Component" or "Specify Component".

Every step has pre- and postconditions, that describe if the user can start or stop the step execution. Furthermore, steps can contain a set of contexts. This can be an execution context which is a subset of the product model (e.g. all classes within component c are in one execution context) or a "parameter" which describes a certain situation. This can be for example the implementation language, the used case tool, framework, the project, and so on. The number of step types is fix at project execution time; the number of context instances can vary (e.g. if a new product is added to the product model, one or more execution context instances are created corresponding to the process description). A more detailed description of our language can be found here [17-19].

4. Smart Development Process Enactment Based on Context Sensitive Sequence Prediction

For user (developer) assistance we have developed an approach to predict the next step the user wants to start. Our approach observes the last couple of started steps (and corresponding contexts) and tries to build a database where identified sequences are stored.

As already mentioned our work is based on the work of Davison/Hirsh [15] and Jacobs/Blockeel [16] which will be presented in more detail in the next subsection.

4.1. Overview of the Underlying Work

IPAM is the work of Davison/Hirsh and addresses the prediction of UNIX commands. The approach implements a first order markov model, so the prediction which is made is based on the last observed element. IPAM stores unconditional and conditional probabilities in a database. After each observation the entries of the database are updated as follows:
$P'(x|y) = x \cdot (1|\ ) + (\ -\ )$ for x = current observation and y = last observation. The entries with x ≠ current observation are updated with: $P'(x|y) = y \cdot (\ |\ )$. α is a parameter between 0 and 1. Davison/Hirsh recommend a value of $\alpha \approx 0.8$.

The work of Jacobs/Blockeel is based on IPAM but implements a higher order markov model. If $\cdots y_1 y_0 x$ was observed and the prediction for x was correct (e.g. $P(x|y_2 y_1 y_0)$ has highest probability) new entries are stored to the database. Let C be the set of suffixes of $\ldots y_1 y_0$ with $P(a|c) > 0$, for all $c \in C$. Let l be longest Suffix of $\ldots y_1 y_0 x$ with $P(a|l) > 0$. The following new entries are stored to the database:
$P(z \mid c \circ x) = P(z \mid l)$, for all $c \in C$ and $z \in$ observed elements.

4.2. Our Approach

The central part of our approach is the LookupDB which stores the experience data **Figure 1** (middle box). Each of the elements of the LookupDB stores, among other things, a condition (cond), a step prediction, and a probability (P). This describes the probability (P) that the next step (prediction) will be started, given the occurrence of the step sequence cond. For example: An element of LookupDB can store the condition 1→2→3. When the steps 1→2→3 are observed, the probability that step 1 will be started is 0,9. Further, for each element in the condition sequence a set of relevant context probabilities are stored. Those are these contexts which were observed in the past.

For prediction (see **Figure 1**, right) of the next step all relevant elements of LookupDB are taken and for each of these elements an "actual probability" is calculated. Here, our approach considers the learned context relevance in LookupDB and the actual observed contexts in the corresponding window.

After prediction and after observation our algorithm learns this observation by updating the conditional probabilities and context relevance of all matching elements in the LookupDB (see **Figure 1**, left). In the following out approach is specified in a more detailed way.
First, let us define some relevant elements of our meta model (see [17-19]):

(1) Let *S* be index set which represents the set of Steps
(2) Let *C* be index set which represents the set of

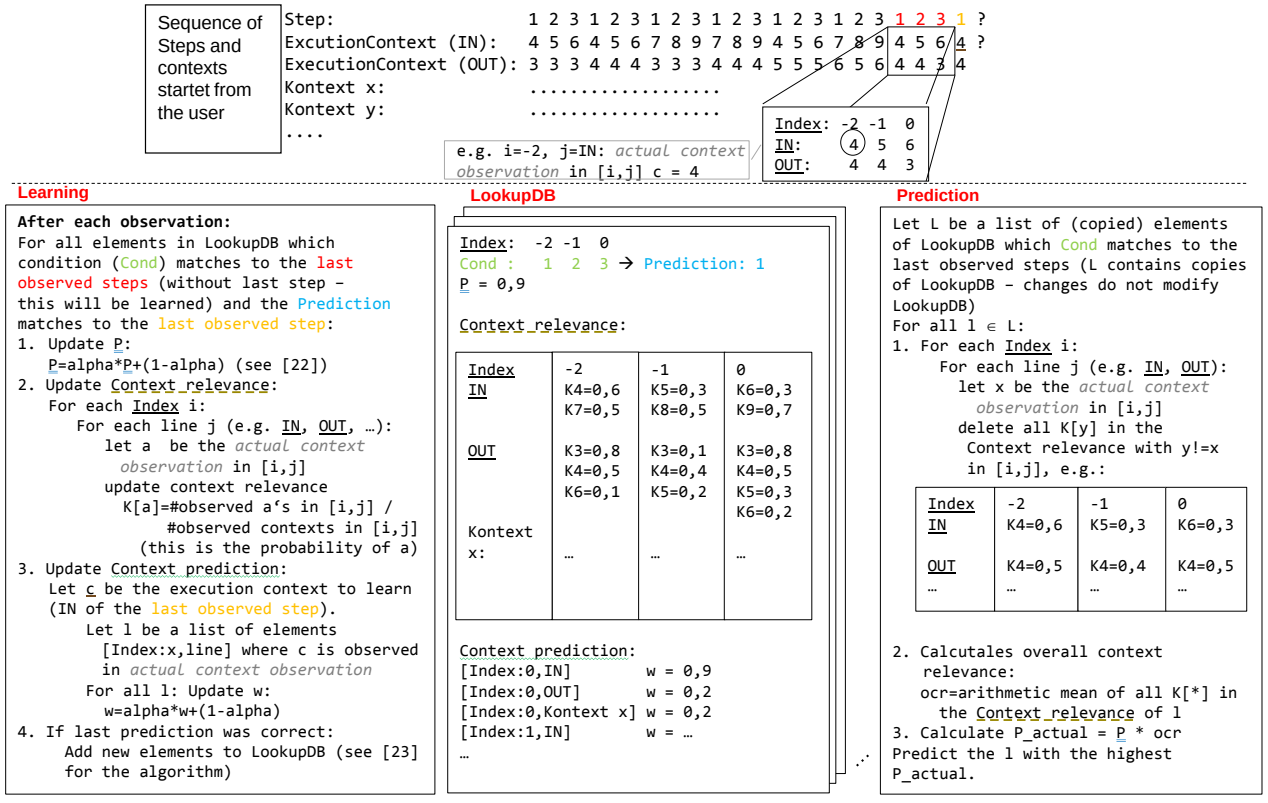


Figure 1. Overview of the approach.

Contexts

(3) Let *CC* be index set which represents the set of Context Classifications.

To address the past observed steps we need an index set *I*:

(4) Let *I* be index set $[-n,\ldots,0]$.

The function observation returns the observed step at the index *i*, where *i* = 0 means the last observation, *i* = -1 the last but one observation, et cetera (see **Figure 1** top right):

(5) $observation_s :=_{DEF} I \to S$

Furthermore, we need a function which returns the observed context at index *i* and at the Context Classification cc:

(6) $observation2 :=_{DEF} (I \times CC) \to C$

In the example in **Figure 1**, *observation* 2(0,*IN*) returns 6.

4.3. LookupDB

We define our LookupDB:

(7) Let $LOOKUPDB :=_{DEF} \{0,1,2,\ldots\}$ be Index Set

Each element of *LOOKUPDB* represents an element of the LookupDB. Every element of our LookupDB has a condition cond. We define

(8) $COND :=_{DEF} \{LOOKUPDB \times I \times S\}$

$COND$ is the set of all step sequences of elements of the LookupDB. For example: The following elements exist for the LookupDB element shown in **Figure 1**: (69,−2,1), (69,−1,0), and (69,0,3) where 69 is the ID of the shown LookupDB element. Let $ldb \in LOOKUPDB$; $cond_{ldb,i}$ describes the step of *ldb*at position *i*.

(9) $LENCOND :=_{DEF} \{LOOKUPDB \times \mathbb{N}\}$

Is the set that describes the length of the condition defined in (8). The following element exists for the LookupDB element described above: (69,3). The condition length is 3. Let $ldb \in LOOKUPDB$; $lenCond_{ldb}$ describes the condition length of *ldb*.

The set $PREDICTION$ describes the predicted step of the element of the LookupDB:

(10) $PREDICTION :=_{DEF} \{LOOKUPDB \times S\}$.

(11) $P :=_{DEF} \{LOOKUPDB \to [0,1]\}$ defines the probability of the element of the LookupDB that the predicted step occurs. For the element in **Figure 1** $(69,1) \in PREDICTION$ and $(69,0.9) \in P$. For easy handling p_{ldb} describes the probability of the LookupDB element ldb. Theset CONTEXTWEIGHT describes a weight for each context classification (**Figure 1**: the lines of the table, e.g. *IN*, *OUT*), *index* (**Figure 1**: rows of the table) and *context* (number in the fields of the table) for each element of the LookupDB:

(12) $CONTEXTWEIGHT \in \{LOOKUPDB \times CC \times I \times C \times \mathbb{N} \times \mathbb{N}\}$

The following elements describe the example in **Figure 1** at index 1, CC=IN:

$(69, IN, -1, K5, 3, 10) \in CONTEXTWEIGHT$; (3/10) defines the probability that the context *K*5 at position *IN* and index-1 is relevant.

Now, we define a function that returns 1 if a specified entry of LookupDB corresponds to the last observed steps:

(1) $match :=_{DEF} (LOOKUPDB \times \mathbb{N}) \to \{0,1\}$

$$match(ldb, offset) := \begin{cases} 1 \text{ if } cond_{ldb,q} = observation_s(q - offset) \\ \quad \text{for all } q \in [-lenCond_{ldb}, 0] \\ 0 \qquad\qquad\qquad\qquad \text{otherwise} \end{cases}$$

Anymore, we define the following function that finds a specific element from the set *CONTEXTWEIGHT*:

(2) $getCW :=_{DEF} LOOKUPDB \times CC \times I \times C \to CONTEXTWEIGHT$

$getCW(ldb, cc, i, c) := cw$ with $cw \in CONTEXTWEIGHT$ and $id_{cw} = ldb, cc_{cw} = cc, i_{cw} = i \text{ and } c_{cw} = c$

4.4. Prediction

For prediction we define the following functions:

(3) $getAMC :=_{DEF} \{(I \times CC \times C \times [0,1])\} \to [0,1]$

$getAMC(\{(i, cc, c, w)\}) := amc$ with $amc :=$ arithmetic mean of all $w > \theta$(θ is threshold).

(4) $f :=_{DEF} LOOKUPDB \to \{(I \times CC \times C \times [0,1])\}$

$f(ldb) := \{(i, cc, c, w)\}$ with $i \in [-lenCond_{ldb}, 0]$ and $c := observation2(i, cc)$ for all *i* and *cc* and *w*

$:= x / y \; with (x, y) := getCW(ldb, cc, i, c)$

The function *f* takes an entry of *LOOKUPDB* and returns a set of elements (*index*,*cc*,*c*,*w*). *w* corresponds to the context weight of *ldb* for each index *i* and *cc* in *ldb* and for the corresponding *c* found in the observation.

The function *getAMC* takes a set of elements (*index*,*cc*,*c*,*w*) and returns the arithmetic mean of all *w*. *getAMC*(*f*(*ldb*))returns a "parameter" that the element of the LookupDB (and the learned context weights inside) corresponds to the observed contexts.

The function *getActualP* takes an entry of *LOOKUPDB* and calculates an actual P weight dependent on *ldb* and the actual observation (this weight is used to select the best entry of LOOKUPDB for prediction):

(5) $getActualP :=_{DEF} LOOKUPDB \to [0,1]$

$getActualP(ldb) := getAMC(f(ldb)) * p_{ldb}\}$

Now, we can describe the function to predict the next

step:

(6) $makeprediction :=_{DEF} Pow(LOOKUPDB) \to S$
$makeprediction(ldb) := s) with$
$s := prediction(ldb)$ for maximal $getActualP(ldb)$

To predict the next step we call the function *make prediction* with a set of all elements *ldb* of *LookupDB* with $match(ldb,0) = 1$.

4.5. Learning

To update the LookupDB, the following steps are done (note: $observation_s(0)$ is the step a prediction we made and we want to learn):

If there exists no $ldb \in LOOKUPDB$ with $(ldb,1) \in LENCOND$ and $((ldb,0), observation(-1)) \in COND$ and $(ldb, observation_s(0)) \in PREDICTION$ a new entry is added to the LookupDB (the entry with the shortest cond):

(19)
$addEntry :=_{DEF} \{LOOKUPDB\} \times \{COND\} \times \{LENCOND\} \times \{PREDICTION\} \times \{P\} \to \{LOOKUPDB\} \times \{COND\} \times \{LENCOND\} \times \{PREDICTION\} \times \{P\}$
$addEntry(ldb_0, cond_0 lc_0, pre_0, p_0) :=$
$(ldb_1, cond_1, lc_1, pre_1, p_1$

with:

- $ldb_1 \setminus ldb_0 = \varnothing, cond_1 \setminus cond_0 = \varnothing, lc_1 \setminus lc_0 = \varnothing, pre_1 \setminus pre_0 = \varnothing, p_1 \setminus p_0 = \varnothing)$
- $|ldb_0|+1 = |ldb_1|, |cond_0|+1 = cond_1, |lc_0|+1 = |lc_1|, |pre_0|+1 = |pre_1|, |p_0|+1 = p_1$
- $n \notin ldb_0$ and $n \in ldb_1$
- $(n, 0, observation(-1)) \notin cond_0$ and $((n,0), observation(-1)) \in cond_1$
- $(n,1) \notin lc_0$ and $(n,1) \in lc_1$
- $(n, observation_s(0)) \notin pre_0$ and $(n, observation_s(0)) \in pre_1$
- $(n, 1-\alpha) \notin p_0$ and $(n, 1-\alpha) \in p_1$

Let $LDB \subseteq LOOKUPDB$:
$LDB = \{ldb \in LOOKUPDB \mid match(ldb,1) = 1 and (l, observation_s(0)) \in PREDICTION\}$.

For each element $ldb \in LDB$ the following steps are done:

(20) $updateP :=_{DEF} \{P\} \times LOOKUPDB \to \{P\}$ with:
$updateP(p_0, ldb) := p_1 with (ldb, pp) \in p_0, (ldb, pp) \notin p_1 and (ldb, \alpha * pp + (1-\alpha)) \in p_1$ and $p_0 \setminus \{(ldb, pp)\} = p_1 \{(elb, \alpha * pp + (1-\alpha))\}$ (see [15])

(21)
$updateContextweight :=_{DEF} \{CONTEXTWEIGHT\} \times LOOKUPDB \to \{CONTEXTWEIGHT\}$
$updateContextweight(cw_0, el) := cw_1$ with

- For all $(ldb, cc, i, c, x, y) \in cw_0$ with $ldb \neq el \Rightarrow (ldb, cc, i, c, x, y) \in cw_1$
- For all $(ldb, cc, i, c, x, y) \in cw_0$ with $ldb = el$ and $c = observation2(i, cc) \Rightarrow (ldb, cc, i, c, x, y) \in cw_1$ and $(ldb, cc, i, c, x+1, y+1) \in cw_1$
- For all $(ldb, cc, i, c, x, y) \in cw_0$ with $ldb = el$ and $c \neq observation_s(i) \Rightarrow (ldb, cc, i, c, x, y) \notin cw_1$ and $(ldb, cc, i, c, x, y+1) \in cw_1$

This function updates the probabilities of the context information in *ldb*.

If the last prediction was correct new entries are added to the LookupDB according to the work of Jacobs *et al.* [16]. Let $Q \subseteq LOOKUPDB$ be a subset of *LOOKUPDB* with:

(22) $Q :=_{DEF} \{ldb \in LOOKUPDB | cond_{q,i} = observation_s(i-1)$ for all $i \in [lenCond_q, 0]\}$

Let $L \subseteq LOOKUPDB$ be subset of *LOOKUPDB* with:

(23) $L :=_{DEF} \{ldb \in LOOKUPDB \mid cond_{l,i} = observation_s(i)$ for all $i \in [lenCond_{ldb}, 0]\}$

Let *ll* be the element of *L* with the longest *lenCond* and $P(ll)>0$. The function *update LOOKUPDB* is defined as:

(7)
$updateLOOKUPDB :=_{DEF} (\{LOOKUPDB\} \times Q \times \{ll\} \times \{COND\} \times \{LENCOND\} \times \{PREDICTION\} \times \{P\} \times \{CONTEXTWEIGHT\}) \to (\{LOOKUPDB\} \times \{COND\} \times \{LENCOND\} \times \{PREDICTION\} \times \{P\} \times \{CONTEXTWEIGHT\})$
$updateLOOKUPDB(LDB_0, q, l, c_0, lc_0, pre_0, p_0, cw_0) := (LDB_1, c_1, lc_1, pre_1, p_1, cw_1)$ with (see [16]):

- $\{0,1,\ldots,n\} \in LDB_0$ and $\{0,1,\ldots,n,\ldots m\} \in LDB_1 with \mid LDB_0 \mid + \mid q \mid = \mid LDB_1 \mid$

Let *ldiff* be $LDB_1 \setminus LDB_0$. For all $qq \in q$ there is an *ldiffel* in *lediff* with:

- $((q,i),s) \in c_0$ and $((ldiffel, i-1), s) \in c_1$ for all *i* and *s*; and: $((ldiffel, 0), oberservation(0)) \in c_1$
- $(q,l) \in lc_0$ and $(ldiffel, l+1) \in lc_1$
- $(q,s) \in pre_0$ and $(ldiffel, s) \in pre_1$ for all *s*
- $(q,p) \in p_0$ and $(ll, p2) \in p_0$ and $(ldiffel, p2) \in p_1$

This function adds new entries to LookupDB by taking the entries which have predicted the observation correctly

and “extending” the corresponding conditions by adding the observation (see [16] for detail).

5. Evaluation

5.1. Synthetical Process

For the evaluation of the approach described in Section 4 we derived a synthetic project scenario. In this scenario the requirements of the system are existent. The goal is to develop an architecture (components, classes) and a corresponding implementation. The process description consists of 4 steps: 1) Identify component, 2) Map requirement to Component, 3) Specify component (refine requirement), 4) Implement component.

In the system two types of components exist: a) Complex/hardware related components. Here, the engineer has a prototypical method to develop the component (steps 2 - 4 are executed sequentially). b) Components which classes have a high coupling. Here, the engineer has a broad design method (step1; all steps 2; all steps 3; all steps 4).

For evaluation three sequences of steps (with corresponding contexts) were build: i) Development of components only of the type a), ii) Development of components only of the type b), and iii) random mix of a) and b).

Our approach is compared with the algorithm from Jacobs/Blockeel [16] which is the underlying sequence prediction technique of our approach. The results are shown in **Figure 2**. This figure contains two graphs for each scenario: The first (top) describes the total number of correct predictions and the second describes the percentage distribution of correct prediction after each step (horizontal axis). The Jacobs Blockeel(JB) approach is shown in red color and our approach (MD) is shown in blue color.

In all scenarios our approach predicts equal to or better than the Jacobs Blockeel algorithm. Remarkable is that in scenario 3 (the “real world” scenario) our approach predicts the steps substantially better than the JB approach. After 2/3 of all steps our algorithm predicts always correctly. On the other hand the algorithm of JB “drifts” to 53% correct predictions. 81% of correct predictions in scenario 3 might be an indication that our approach is more applicable.

5.2. Real Project Process

In addition to the synthetical evaluation (described in section V.A) we evaluated our approach with real project data from anindustrial enterprise. Are there differences between these two evaluations? In the synthetical evaluation we described a process description using the description language presented in Section 3. This is a rule based language which is flexibly executable for the engineer. On this basis, we derived three test data sets and evaluated our approach with these data. The evaluation of the industrial case study is based on real project data with 5.600 individual development steps. The underlying process description language of the industrial case study was a net based language. Therefore, the process description (and the enacted process) is more stringent than our process used for evaluation in section V.A.

Like in the synthetically evaluation our approach is compared with the approach of Jacobs/Blockeel. The results are shown in **Figure 3**. Again, there are two cate-

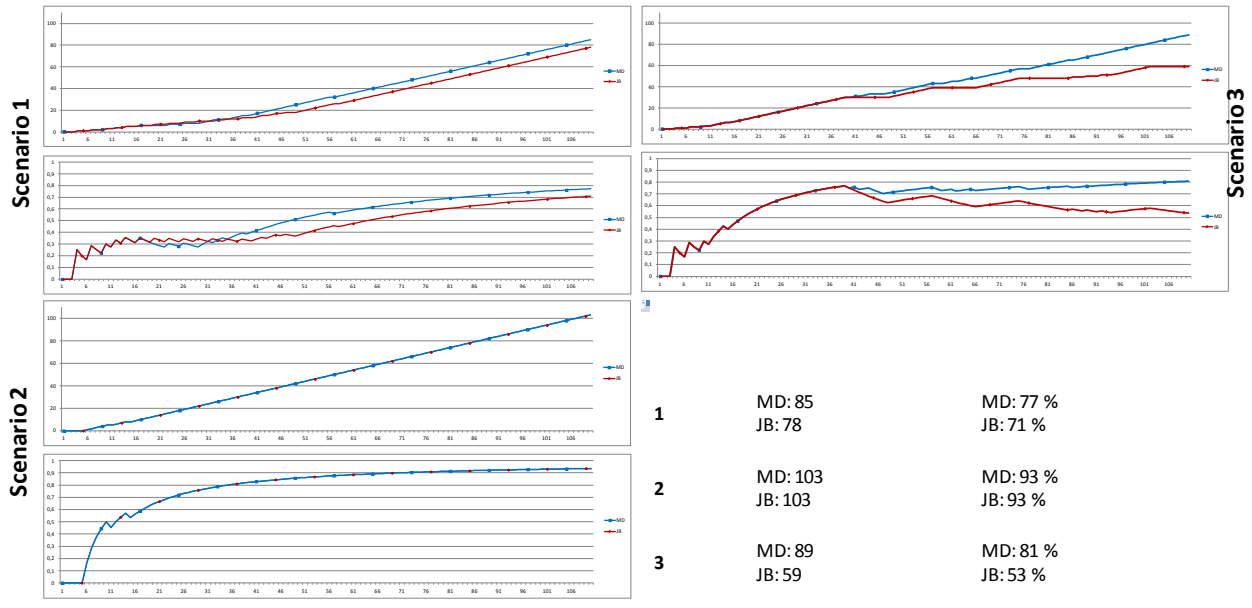
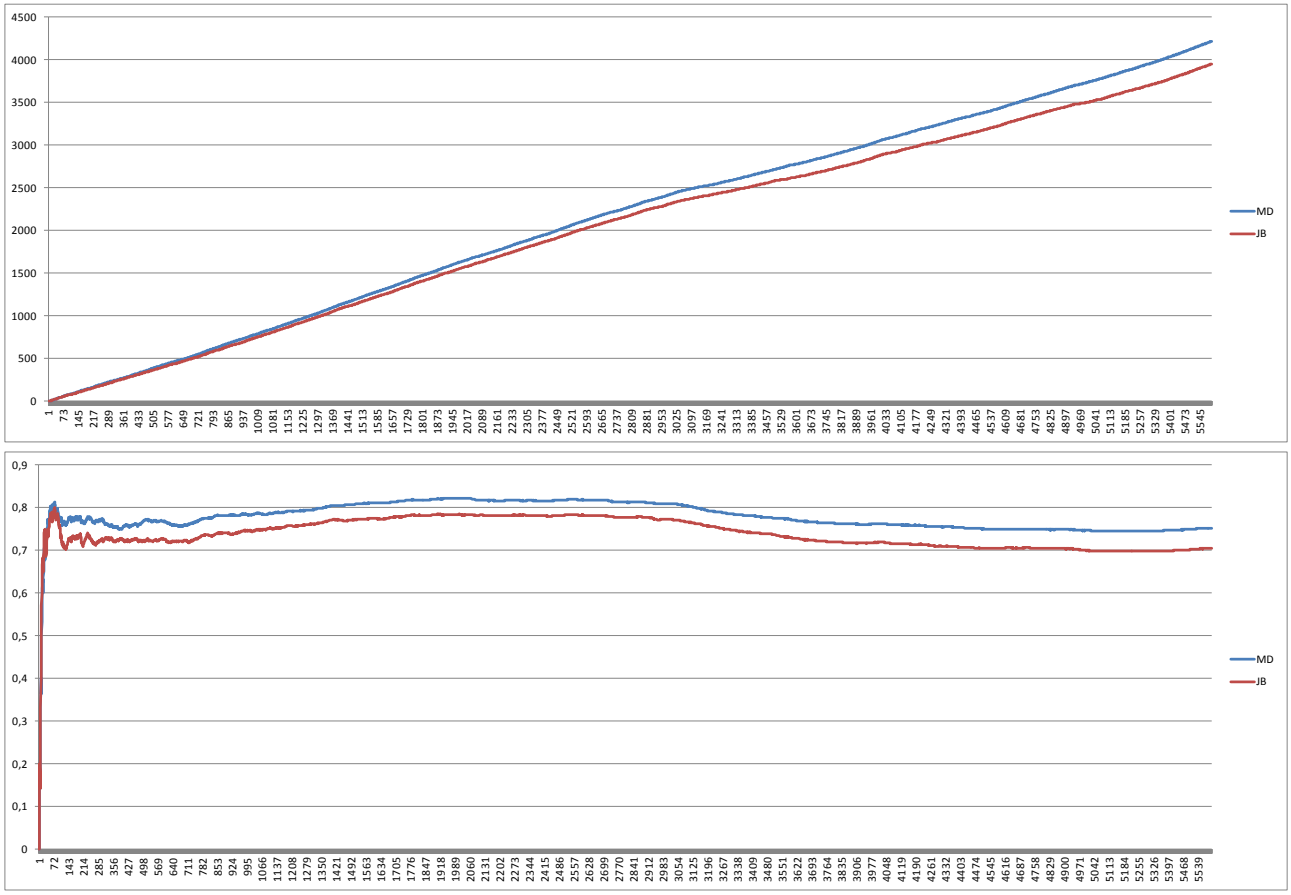


Figure 2. Results synthetic scenario.



Total number correct predictions	% correct predictions
MD: 4215 JB: 3949	MD: 75 % JB: 70 %

Figure 3. Results of the real project evaluation.

gories shown: the first shows the total number of correct predictions and the second shows the percentage distribution of correct predictions after each step. Similar to the synthetic approach our approach generates more accurate predictions than the Jacobs/Blockeel approach.

6. Conclusions, Further Work

Customers are placing growing demands on the software industry. The ability of any industry to survive the rough market conditions will depend more and more on software and hence the capabilities to deliver in time, budget and quality required software. For that reasons over the years, a variety of software process models have been designed. Software process models can only strengthen your software development capabilities as long as you apply it correctly.

But, there is huge gap between the defined and the applied development process. One reason is that the predefined general process flow does not reflect the specific constraints of the individual project. For that reasons we claim to get rid of the process flow definition as part of the development process. Instead we describe in this paper an approach to smartly assist developers in software process execution.

The basic idea is to observe the developer's actions and thereby collect the development process knowledge inside the heads of people carrying out the work. Hence the project process history provides the process knowledge for our approach. Based on this process knowledge we predict the next development step. We apply a sequence learning approach to predict the next development step with respect to project process knowledge.

We have evaluated our approach on a synthetic and a real world project setting. In both settings our approach was able to provide more accurate predictions than classical non context sensitive approaches. However the resulting accuracy of 75% to 90% is not high enough for a wide acceptance by developers. However if we present to

the developer not only a single development step prediction but instead presenting the best three predictions—similar to Google which presents a result list after a web query, the prediction accuracy rate would increase to 99.9%. Note, this resulting accuracy has been proved in the two presented evaluation scenarios.

Based on such a prediction soundness a process enactment framework could be widely accepted and applied by developers. The next step is to set up a broader empirical experiment to validate the applicability of our process enactment framework.

Moreover additional research and application directions of the presented approach are to support novice developers (e.g. developers works in a new project/new company) by providing the experience data of other developers. Using the experience data (of the developers of one or more projects) for organization-wide process improvement (e.g. derive a standard process description of the available knowledge) could be another interesting direction to follow.

REFERENCES

[1] L. Northrop, P. Feiler, R. P. Gabriel, J. Goodenough, R. Linger, T. Longstaff, R. Kazman, M. Klein, D. Schmidt, K. Sullivan and K. Wallnau, "Ultra-Large-Scale Systems —The Software Challenge of the Future," Software Engineering Institute, Carnegie Mellon, Tech. Rep., 2006. http://www.sei.cmu.edu/uls/downloads.html

[2] A. W. Brown and J. A. McDermid, "The Art and Science of Software Architecture," In: F. Oquendo, Ed., *ECSA, Ser. Lecture Notes in Computer Science*, Vol. 4758, Springer, 2007, pp. 237-256.

[3] B. Boehm, "Some Future Trends and Implications of System and Software Engineering Processes," Wiley InterScience, 2006.

[4] D. Rombach, "Integrated Software Process and Product Lines, Unifying the Software Process Spectrum," 2006.

[5] M. Kabbaj, R. Lbath and B. Coulette, "A Deviation Management System for Handling Software Process Enactment Evolution," *Making Globally Distributed Software Development a Success Story*, 2008, pp. 186-197.

[6] K. Mohammed, L. Redouane and C. Bernard, "A Deviation-Tolerant Approach to Software Process Evolution," *Ninth International Workshop on Principles of Software Evolution in Conjunction with the* 6*th ESEC/FSE Joint Meeting*, 2007, pp. 75-78.

[7] D. Jean-Claude, K. Badara and W. David, "The Human Dimension of the Software Process. In: Software Process: Principles, Methodology, and Technology, Lecture Notes in Computer Science," Bd. 1500, Springer US, 1999, pp. 165-199.

[8] T. DeMarco and T. Lister, "Peopleware, Productive Projects and Teams, Second Edition Featuring Eight All-New Chapters," Dorset House Publishing Corporation, 1999.

[9] K. Z. Zamli, "Process Modeling Languages: A Literature Review," Dez-2001. http://myais.fsktm.um.edu.my/278/

[10] S. T. Acuna, "Software Process Modelling."

[11] P. Armenise, S. Bandinelli, C. Ghezzi und A. Morzenti, "A Survey and Assessment of Software Process Representation Formalisms," *International Journal of Software Engineering and Knowledge Engineering*, Bd. 3, 1993, pp. 401-401. http://dx.doi.org/10.1142/S0218194093000197

[12] M. I. Kellner, "Representation Formalisms for Software Process Modeling," *Proceedings of the* 4*th International Software Process Workshop on Representing and Enacting the Software Process*, Devon, United Kingdom, 1988, pp. 93-96. http://dx.doi.org/10.1145/75110.75125

[13] P. H. Feiler und W. S. Humphrey, "Software Process Development and Enactment: Concepts and Definitions," *Software Process*, 1993, Second International Conference on the *Continuous Software Process Improvement*, 1993, pp. 28-40.

[14] M. Hartmann und D. Schreiber, "Prediction Algorithms for User Actions," *Proceedings of Lernen Wissen Adaption*, ABIS, 2007, pp. 349-354.

[15] B. D. Davison und H. Hirsh, "Predicting Sequences of User Actions," Notes of the AAAI/ICML 1998 Workshop on Predicting the Future: AI Approaches to Time-Series Analysis, 1998.

[16] N. Jacobs und H. Blockeel, "Sequence Prediction with Mixed Order Markov Chains," *Proceedings of the Belgian/ Dutch Conference on Artificial Intelligence*, 2002.

[17] M. Deynet, "User-Centric Process Descriptions," *Proceedings of the 3rd International Conference on Software Technology and Engineering*, Kuala Lumpur, Malaysia, 2011, pp. 209-214.

[18] M. Deynet, "Kontextsensitiv Lernende Sequenzvorhersage zur Erfahrungsbasierten Unterstützung bei der Softwareprozessausführung," Dr. Hut, 2013.

[19] M. Deynet, "Predicting User Actions in Software Processes," *4th Workshop on Intelligent Techniques in Software Engineering at the European Conference on Machine Learning and Principles and Practices of Knowledge Discovery in Databases* (ECML-PKDD), Athen, Ägypten, 2011.