Effort estimation for Agile Software Development Projects

Andreas Schmietendorf, Martin Kunz, Reiner Dumke

Abstract
Agile methods of the software development are increasingly used for industrial projects. The application of effort estimation methods in such kind of projects is very difficult, but an important task. Classical estimation methods need well defined requirements. Agile methodologies don’t support this behaviour. Rather, they see changed requests as important challenge. A method for the effort estimation must consider this requirement. This paper provides an investigation about estimation possibilities, especially for the eXtreme Programming (so called XP-projects) paradigm. After a short introduction of key characteristics of agile projects the paper deals with available estimation approaches from the literature. Furthermore the paper shows selected results of an empirical analysis.

1. Introduction
Agile Software Development Methods are nowadays wide spread and accepted. The procedure to carry out such kind of projects takes especially constructive aspects and short realisation times into account. From the Software Measurement point-of-view not all metrics and methods from conventional lifecycle models can be used without adaptation. Within this paper we want to investigate especially the aspect of effort estimation activities for agile software development projects. For this we want to show the possibilities and borders of effort estimation tasks and the new challenges. The following questions are examined:
1. Which conditions do projects that are executed with help of agile process models assume?
2. Which basic approaches (state of the art) of the effort estimation can be identified for agile executed projects at present?
3. Are the classic methods of the estimation procedures, like Function Points or COCOMO applicable?
4. Which experiences can be found in the industrial and academic surroundings with the use of agile procedures?

Beside the clarification of these questions, a short overview is given to the key characteristics of agile software development methods. For this we want to consider especially the industrial accepted XP approach.

2. Characteristics of agile methods
Many technological ambitious products were designed with new complex functionality. The demand for functions establishes a need for new software requirements to deliver new functionality. Due to the fast alteration and the high cost of change in the late life cycle phases the agile software development method becomes more important in this field of application.
Agile software development methods like eXtreme Programming try to decrease the cost of change and therewith reduce the overall development costs. Agile methods try to avoid the deficits of classic software development procedures.
Mostly the following methodologies are considered to reach this aim: short release cycles, simple design, continuous testing, refactoring, collective ownership, coding standard and continuous integration.

Currently, there are different approaches for agile software development methodologies. Typical approaches can be found with:

- eXtreme Programming (Kent Beck - 1999).
- Feature Driven Development (Jeff De Luca - 1999).
- Scrum (Ken Schwaber and Mike Beedle - 2001).
- Agile Implementation of RUP (Craig Larman - 2002).
- Crystal – as family of agile methodologies (Alistair Cockburn - 2004).

Extreme Programming (XP) is a usual and wide spread agile software development method. Therefore, we have used XP as a reference model for agile software development. In the centre of software development projects should be the human and not documents, processes and tools. XP provides a set of practices, values and principles [1]:

- Values (e.g. communication, simplicity, feedback, courage)
- Principles (e.g. incremental changes, honest measuring).
- Practices (e.g. pair programming, short version cycles)

These set is derived from "best practices" and should help to carry out successful software development projects. To realise these totally different project characteristics, the agile software development has established a different type of product life cycle in comparison with traditional life cycle models like waterfall-model or V-Model.

Figure 1 shows the necessary process steps of a XP-project. The major element of the XP life cycle is the “Iteration”. The iteration is a recurring event in which an actual version is edited for example by:

- Adding additional functionality.
- Correcting errors.
- Removing unnecessary functionality.

Each software version will be validated through an acceptance test. In XP the duration of each iteration is very short (1 until 4 weeks) in comparison with traditional software development.

![Figure 1: Process Steps of a XP-project (under consideration of [12])](image_url)
The planning phase is also very short and mainly realised by the definition of tasks. The task of the effort estimation can be found within this step. In the following section this activity will be investigated in detail.

3. Effort estimation for XP-projects

The consideration of the effort estimation activity within agile software development methodologies was also investigated by [3]. He describes shortly the estimation approaches used in various agile software development methods. In the following representations, we want to concentrate on the eXtreme Programming (in short XP) approach. For this, available approaches for the effort estimation will be shown. In addition, further influence criterions will be analysed.

3.1. Consideration of feedback and change

As mentioned earlier within this paper, feedback- and change-activities are central principles of the XP-methodology. Feedback and changes are important for effort estimation activities too, as Gilb [8] noticed. He considers the budget-related requirements of the software development with the following statement:

“Accurate estimation is impossible for complex technical projects, but keeping to agreed budgets is still possible using feedback and change”.[8]

The activities of feedback and change are key activities to fulfil budget restrictions of a software development project. An exact estimation of the corresponding effort at beginning of a complex development project is not reachable as Gilb [8] mentioned. He sees the reasons for this in:

- Inaccurately defined requirements.
- Missing experience background with the effort of completed projects.
- New projects differ often from realised projects.

In order to reach a successful realisation of technical requirements, he proposes to combine the effort estimation with the following “principles of resource control”[8]:

- Risk principles (under consideration of: DRIVERS, EXPERIENCE, ARCHITECTURE, STAFF, SENSITIVITY).
- Control principles (under consideration of: LEARN SMALL, LEARN ROOT, PRIORITISE CRITICAL, RISK FAST, APPLY NOW).

While the risk principles show the potential causes of inadequate effort estimation, the control principles accept possible solution alternatives. Similar to the agile methodologies in the context of the software development [8] puts the person-referential aspects (acceptance of the learning and profits of experiences) as well as the guarantee of feedback (short iterations and small increments) in the foreground of a successfully executed software development.

For the effort estimation of agile executed software development projects [11] gives the following recommendation:

“Don't estimate too far into the future, if the future is unclear!”

If effort estimations are difficult, projects should work more with feedback and change. Future methods of the effort estimation must take this challenge into account. That means effort estimations must be executed frequently, but within a very short time. The established methods of the effort estimation mostly do not possess this quality.
3.2. Identification of the functional size

The effort estimation within XP-projects is a part of the so called planning game. The aim of this activity is the planning of a new software version as well as the planning of a single iteration. The figure 2 shows the different activities during the planning game. The recommended duration of the planning game can be assumed with 4 hours. Within this time, customers and developers work together.

The User Stories are the basis for the effort estimation. These stories should be written by the customers. They should be a description of expected business functionalities by the later software system. To reach a common understanding of the described User Stories, the discussion of these requirements between the customer and developer side takes place in the next step. As a result of this discussion, the developers are able to design a coarse grained system architecture. The architecture should show a first idea about the needed subsystems. Each subsystem is related to several features and each feature is related to several User Stories.

With the help of the coarse architecture the developers try to identify the corresponding functional size. Mostly, the following “relative” numbers are used for this task [13, p. 200]:

- **STEP** – Story Effort Points.
- **FEEP** – Feature Effort Points.
- **SSEP** – Subsystem Effort Points.

With the use of these numbers the developers estimate the effort in person days for each User Story. The won experiences in completed iterations provide the basis for this effort estimation. Experiences can be won also with the implementation of small prototypes (sometimes called “spike”).

A so named Load-Factor is used for the objectification of these experiences [12]:

$$ XP_{LoadFactor} = \frac{Days_{required}}{Days_{estimated}} $$

- **Days_{required}** – considers the required time to complete a task.
- **Days_{estimated}** – considers the estimation to do the activity.
If the estimated effort is too big, the corresponding User Story will be split into several User Stories. After the identification of corresponding efforts, each User Story must be prioritised to complete the project planning.

The procedure implies a mathematical problem, because the relationship between these ordinal scaled point types can be only characterised as following:

\[
SSEP \leq FEEP \leq SSEP
\]

Sometimes, a direct conversion of these point types can be observed in the literature. However, the direct representation of mathematical relations between these numbers is questionable. In order to justify this procedure, the following assumptions are postulated:

- Linearity of the used effort points.
- Constance of the relationship values between the point types.
- Constance of the productivity.

From the authors’ point of view, these assumptions are debatable. It could be possible to guarantee these assumptions within small project organisation, but the necessary empirical background is still missing.

There are further approaches for the effort estimation of XP-Projects. For example, the following numbers can be found:

- ITEP – Iteration Effort Points, consider the effort of a whole iteration. They are based on won experiences and do not take the functional structure of the system into account.
- TAEP – Task Effort Points, consider specific execution tasks, during the development. In this case, a detailed view of the executing development tasks takes place.

Currently, no standardised approach for the effort estimation of XP-projects can be found. The interpretation of the used measurements can be only guaranteed within stable organised software development organisation. On this basis, it is difficult to establish an empirical experience background.

### 3.3. Application of classical effort estimation methods

The application of the Function-Point-method within XP-projects was investigated through Fuqua [7]. The following questions were examined:

- “Would Function-Points produce a more accurate schedule?”
- “Are Function-Points a good measure of velocity?”
- “Do Function-Points predict how long it will take to implement a story?”

Within a controlled experiment he investigated the XP-based implementation of an XML-editor. Under the use of 100 User Stories he tried to estimate the effort with function points (especially Mark II FP) and also with the number IED (Ideal Engineering Days). The number IED is another used measure inside XP-projects. The IED describes the working time without interruption. In the result of the executed examinations, he determined the following:

- The used Function-Point procedure was unable to estimate the required effort.
- Function-Points support the measurement of the load factor insufficient.
Basis of this statement was the low correlation of the estimated function points with the actual spent efforts in comparison to the results of the expert estimation. From view of the authors, this result must be assessed critically regarding to the following points:

- The Function-Point-method especially concentrates on information systems. The graphic-oriented development of a XML-editor is not considered.
- The used empirical values with the Function-Point-method, in the sense of identified effort, cannot be understood by the outsider.
- The investigation in the context of only one application cannot yet lead to a statistically secured statement. For this it needs further empirical analyses.

Despite these criticism points, important experiences can be won:

- Within agile executed software developments, the Function-Point-method offers a too fine granularity and therefore causes even too high effort.
- A key aspect of agile methods is the use of feedback and change. The Function-Point-method supports this procedure insufficient.
- The use of function points can help to validate expert estimations. This requires the use of the backfire method [4].
- Important User Stories can be identified with help of the Function-Point-method. Such kind of User Stories requires a stronger analysis.

4. Empirical analysis

4.1. Survey of the research project

Figure 3 provides a survey of the content of our research project. The project is processed in cooperation between the Berlin School of Economics and the University of Magdeburg, inside the Software measurement Laboratory (SMLab). Our aim is to implement controlled experiments, to investigate industrial XP-projects and to provide guidelines for measurement aspects within industrial XP-projects. Diverse measurement- and evaluation-aspects of XP-projects should be considered inside this research project [10].

![Figure 3: The research project at a glance](image-url)
In the context of the effort estimation of XP-projects (the main focus of this paper), we pursue the following goals within 2 steps:

**Carry out empirical analysis:**
- Identification of used effort estimation approaches.
- Importance of risk-driven approaches for the effort estimation.
- Analysis about the degree of the application of agile methods.

**Implementation of prototypes:**
- Compare the productivity between XP - and non XP-projects.
- Compare the productivity before and after introduction of XP-approaches.
- Analysis of the productivity of maintenance projects executed with help of XP.

The implementation of prototypes requires the use of comparable measurements. Therefore, the further explanations consider the empirical analysis, as precondition for controlled experiments and prototypical implementations.

### 4.2. Selected results of a first examination

For a first evaluation we established a questionnaire with 17 questions. The questions consider aspects of agile methodologies and the effort estimation, too. The complete questionnaire is available inside the appendix of this contribution. The questionnaire was sent to approximately 40 known experts in Germany.

The following section contains some selected results from a first analysis. However they show a first trend, but a statistical security cannot be guaranteed yet.

**Which agile methodologies do you know?**
5 participants of the survey didn't know any methods to the agile software development. After all, 12 participants of the survey knew agile methodologies. The knowledge of concrete methods can be taken from the figure 4.

![Figure 4: Knowledge of agile methods](image)
For how many years are projects executed under consideration of an agile procedure inside your company?
- Non experience → 29%
- Less than 2 years → 18%
- Less than 5 years → 18%
- More than 5 years → 35%

The participants with less than 2 years experience came all from academic facilities. All other participants came from industrial businesses. The average sizes of software development projects inside companies with more than 5 years experience was 5 to 30 developers.

How important is the effort estimation in agile operated software projects?
- Highly important → 43%
- Important → 36%
- Necessary → 21%

How important is it to carry out a risk evaluation during the planning game?
- Highly important → 43%
- Important → 29%
- Necessary → 7%
- Less necessary → 21%

Which temporal and personal effort should be caused by an effort estimation method in agile accomplished projects?
The question was answered by 11 experts. However, 6 experts could not make any statements about the necessary effort. The replies of this question brought the following results:
- Effort for the estimation: min. 1 person day, max. 2 person day.
- Effort estimation is a cyclic activity (weekly task).
- The effort should be fixed project-dependent.
- The effort should be proportionally towards the total expenditure.
- 5% to 10% of the whole development effort.
In your mind, is there a demand on adaptation or new development for an effort estimation method which is applicable in agile accomplished projects?

![Effort estimation methods in agile projects](image)

This question was difficult to answer for the most participants. The validations of the answers require the consideration of another question about the known effort estimation methods. For an elaborate analysis, this question must be formulated more precisely. The experienced developers of agile software projects consider the activities inside the planning game as effort estimation method too. The concept of “Yesterdays Weather” use the number of story points that are completed in the last iteration. Under consideration of this experience, this number will be used as predictor for the next iteration.

4.3. Further findings

The introduced survey is not yet finished at the time of the preparation of this contribution. Therefore a concluding evaluation is impossible at this time, but the derivation of first trends is possible.

* In large IT-related enterprises, agile methods are only reflected to the software development conditionally. Mostly, only selected elements of this procedure are used. A common procedure to the effort estimation cannot be identified currently.
* The absence of a common method for the effort estimation implies problems inside large project organisations. In such organisations stable project teams are not a must. Therefore the estimation approaches from section 3.2 are difficult to use.
* The Pair Programming or the complete assignment of the co-workers exclusively to one project could not be observed. The costs related to this procedure are mostly not accepted.
* The iterative procedure with short release cycles is well established inside many development companies. So, a fast Feedback of the customer side can be guaranteed. Therefore, effort estimation methods are frequently used.
* Caused by the use of integration architectures (e.g. Service Oriented Architectures), the importance of the software development activities decrease in the comparison to integration tasks. Future estimation methods must take this into account.
The benefit of agile methods is difficult to assess at the moment. Especially the costs of maintenance projects must be taken into account for it. Solutions which were developed with the help of agile procedures exist only for few years.

5. Conclusions

This paper presents an analysis of the effort estimation possibilities within agile software development methodologies. An industrialisation of the software development implies the demand inevitably at dependable statements about the corresponding efforts and costs of a new IT-solution. This statement is valid for the agile methodologies, too. Boehm [2] provides a classification of possible estimation methods. Selected elements of these procedures can also be observed within XP-projects.

- Parkinson's estimation - under consideration of available resources. Corresponding projects can be settled, however only after expenditure.
- Top Down estimation – with the starting point of the general functionality. The Iteration Effort Points (ITEP) tries to use this approach.
- Bottom Up estimation – consider the estimation of the required system components. This procedure is currently used mostly, as mentioned in section 3.2.
- Analogy, through comparison with similar projects. For the identification of the effort, this base stands behind all used procedures.
- Expertise, in the context of the beginning of XP-Projects. Intuition and experiences from other projects build the basis of the effort estimation.

Parametric as well as algorithmic models form an exception. The application of such procedures like Function Points or also the COCOMO-method could not be observed within industrial projects. However, some of the interviewee could imagine the application of these methods within agile software development projects. In one case, the application of the CO-COMO-method was tested at the time of the survey.

6. References

**Effort estimate in times of agile accomplished development projects**

In the framework of a research project at the software measurement laboratory (shortened SMLab) of the University of Magdeburg „Otto-von-Guericke“ we deal with the empirical estimation of agile accomplished software development projects. One main research refers to interlocking efforts which are involved in such projects. By means of this questionnaire we want to detect the current consideration of cost estimate methods in case of agile realised software development projects.

**Q1:** In which field do you work?
- Academic field
- Industry (employee)
- Industry (free assistant)

**Q2:** Which average size have software development projects inside your business?
- Up to 5 developers
- 5 to 30 developers
- 30 to 100 developers
- More than 100 developers

**Q3:** What is your prior part or the one of your organisation within the framework of software development projects?
- Development company for all phases of software design
- Customer for external accomplished development projects
- Agent for specialised software component processing
- Realisation of integration projects using standard software

**Q4:** Are you aware of agile software development approaches?
- Yes?
- No? (go on with question F11)

**Q5:** For how many years are projects executed under consideration of an agile procedure inside your company?
- Less than 2 years
- Less than 5 years
- More than 5 years

**Q6:** Which agile implementations do you know?
- Extreme Programming XP
- Feature Driven Development
- Crystal
- Agile Implementation of RUP
- Others _______________
Q7: How important is the effort estimation in agile operated software projects?
- 1 – highly important
- 2 – important
- 3 – necessary
- 4 – less necessary
- 5 – not necessary

Q8: Are parametric and algorithmic models applicable for the effort estimation of agile software projects?
- Function Points (yes/no – unknown)
- COSMICFFP (yes/no – unknown)
- Object Points (yes/no – unknown)
- CoCoMo (yes/no – unknown)
- Others …

Q9: In your mind, is there a demand on adaptation or new development for an effort estimation method which is applicable in agile accomplished projects?
- Adaptation of conventional methods
- New development of an method for effort estimation
- Use of existing methods without adaptation

Q10: Which temporal and personal effort should be caused by an effort estimation method in agile accomplished projects?
- Personal effort:  _______ person days
- Temporal effort:  _______ hours

Q11: How important is it to carry out a risk assessment of the development project as well within the scope of the cost estimate?
- 1 – highly important
- 2 – important
- 3 – necessary
- 4 – less necessary
- 5 – not necessary

Q12: Which part does the contemporary feedback of already developed software components have regarding to the cost estimate?
- Important mechanism for effort control
- Minimise the risk of an inaccurate cost estimate
- Supports the prioritisation on implementing functional features
- Considers available learning curve if necessary
- Further considerations  ________________
Q13: Do you know selective elements of agile proceeding?
- Short release cycles
- Simple design
- Pair programming
- Continuous design improvement (refactoring)
- Continuous testing
- Others ________________

Q14: Do you know and use the knowledge databases of already realised software development projects, for example ISBSG?
- Known, but not used
- Known and used
- Use of company internal knowledge database
- Not known

Q15: Which coherences do you see between agile proceeding and service orientated developed software solutions?
- 1 – complementary approaches
- 2 – existing interactions
- 3 – concurrent aims
- 4 – up to date inadequately tested
- 5 – no coherences

Q16: In which percentage are tasks of integration of already available components and new development of components in your field?
- Integration %
- New development %

Q18: Further notes and suggestions concerning this topic

Q17: Are you interested in the results of our research work?
- Yes
- No

Please write your answers and notations directly into the questionnaire. In the case of given answers please mark the appropriate one in **bold**. As a matter of course multiple choices are possible. We thank you for the assistance of our research project und would be pleased a-bout constructive feedback regarding to this topic at any time.