


AGEDIS CONSORTIUM

# AGEDIS



## AGILE SOFTWARE DEVELOPMENT AND AGEDIS

---

by Joachim Hofer (imbus AG)

[joachim.hofer@imbus.de](mailto:joachim.hofer@imbus.de)

# Agile Software Development and AGEDIS

---

## OVERVIEW

---

Introducing AGEDIS in a software development process will change and improve the process. In this article the possible influence of AGEDIS on Agile Software Development is discussed.

---

## FEATURES OF AGILE DEVELOPMENT

---

The agile approach to software development breaks with the traditional view of software development as a process of well-defined and sequential stages. It is probably possible to view it as a special kind of iterative process with extremely short development cycles. However, this would be a bit beside the point of Agile Development.

Agile Development is based on doing away with the process overhead of defining and documenting everything in as much detail as possible. Instead, it is based on the four values *simplicity*, *communication*, *feedback* and *courage*. The *Manifesto for Agile Software Development* (at <http://www.agilemanifesto.org/>) summarizes the key aspects of Agile Development very well. There are some variations of agile “process definitions”, the most popular of which probably is *Extreme Programming (XP)*.

The main features of XP to achieve the goals of Agile Development are (according to [2]):

- the “Planning Game”,
- testing (before programming and continuously),
- Pair Programming,
- refactoring,
- keeping the design simple,
- collective code ownership,
- continuous integration,
- an on-site customer,
- small releases and
- a 40 hour week,
- coding standards and
- a metaphor for the system.

Through the use of these values and features, Agile Development tries to keep the *Cost of Change Curve* flat, whereas other processes just assume that this curve has to be steep and are built around this assumption.

---

## ADVANTAGES AND RISKS

---

The advantages and risks of Agile Development are currently discussed very hotly. Within the scope of this document, only aspects related to testing will be considered.

One huge advantage of Agile Development over all other processes is that it emphasizes the importancy of testing as early as possible and as thorough as possible. For example, the acceptance tests (system tests) in XP are done by the customer and thus define what the customer requires the

software to do. There is no other explicit requirements documentation, which makes these tests crucial to the success of the project.

The other kind of testing done in XP is the unit testing. The developers specify the tests first (before implementing anything), then implement the tests (still before implementing the unit under test), then execute the tests (resulting in a first reference test run of “all fails”), and only after all that start to implement the unit under test and continue until the tests run. This has the advantage that everything ever programmed in a project will have automated unit tests checking it and that everything programmed is specified clearly (mostly in the form of tests and their specifications).

There are some risks associated with this approach, of course: In general, there is nearly no systematical empirical data yet about how well Agile Development works in practice. Then, telling the customer to do the system tests may be asking too much of her, leading to tests that are not very useful. Letting the developers write their own unit tests may also lead to sloppy tests when they focus too much on the implementation already while writing the tests. Also, the missing explicit documentation of the system to develop makes it hard to understand the project “from the outside”.

---

## **INFLUENCE OF AGEDIS**

---

Integrating AGEDIS with agile development is a challenge, as there will not normally be any well-defined and fixed model of the behaviour of the software under test you can just take, adapt and throw at AGEDIS. However, it also offers a great opportunity, as AGEDIS will naturally accompany the development process from the beginning until the end.

### **USING AGEDIS FOR THE SYSTEM TEST**

The AML language of AGEDIS is designed to be easily understandable for non-technical people. Thus, it is very well suited as a tool for the customer to formulate her tests with. By creating a model, she will be able to focus on the required *behaviour* of the system even more than with conventional test approaches. Also, by creating an AML model for the behaviour of the system, the customer will normally be forced to document her requirements in a better way than if she used the conventional test specification approach.

The model-based approach will also guarantee that the system test will be automated, so that repeating it after each of the usually many releases can be fully tested at nearly no additional cost (assuming that the model of the behaviour will remain relatively constant compared to the software design).

The model probably should also be developed in an agile way (especially releasing it early and often, keeping it simple), as the cost of changing the AML model in AGEDIS is very low, considering that everything afterwards runs fully automated.

### **USING AGEDIS FOR THE UNIT TESTS**

It is not immediately obvious how the developers can profit from using AGEDIS as their tool for writing the unit tests. There may very well be many areas (many types of units) where this does not make any sense.

However, units in the object-oriented world often have internal states and are in a large part defined by these states. In this case, an AGEDIS model is probably well suited to describe the behaviour of the unit in question, as it internally translates into a state machine.

The aim of using AML models in unit testing should probably not be to describe the behaviour of the units completely and thus to substitute conventional unit testing, but it could prove to be a useful complement, especially focussing on the internal states of the units under test.

## SUMMARY

Although there are starting points for using AGEDIS in the developers' unit testing, its main application is the system test. Model-based testing by AGEDIS is especially useful in the system testing of Agile Development, as

- it is started very early (the first thing to be done after the Planning Game),
- the AGEDIS modelling language provides an easy means to the crucial end of the customer to specify her requirements as tests,
- developing the model in parallel to the software integrates AGEDIS seamlessly into the agile development process,

Agile Development requires fully automated tests, and AGEDIS provides them.

---

## REFERENCES

---

- [1] AGEDIS Consortium, Automated Generation and Execution of Test Suites for DIstributed Component-based Software.
- [2] Auer, Miller, Extreme Programming Applied, Addison-Wesley, 2002