Quality Assurance of "Handed on Software"

Uwe Hehn

Presentation

F8

Friday 23rd November, 2001

International Conference On
Software Testing, Analysis & Review
November 19 - 23 Stockholm, Sweden

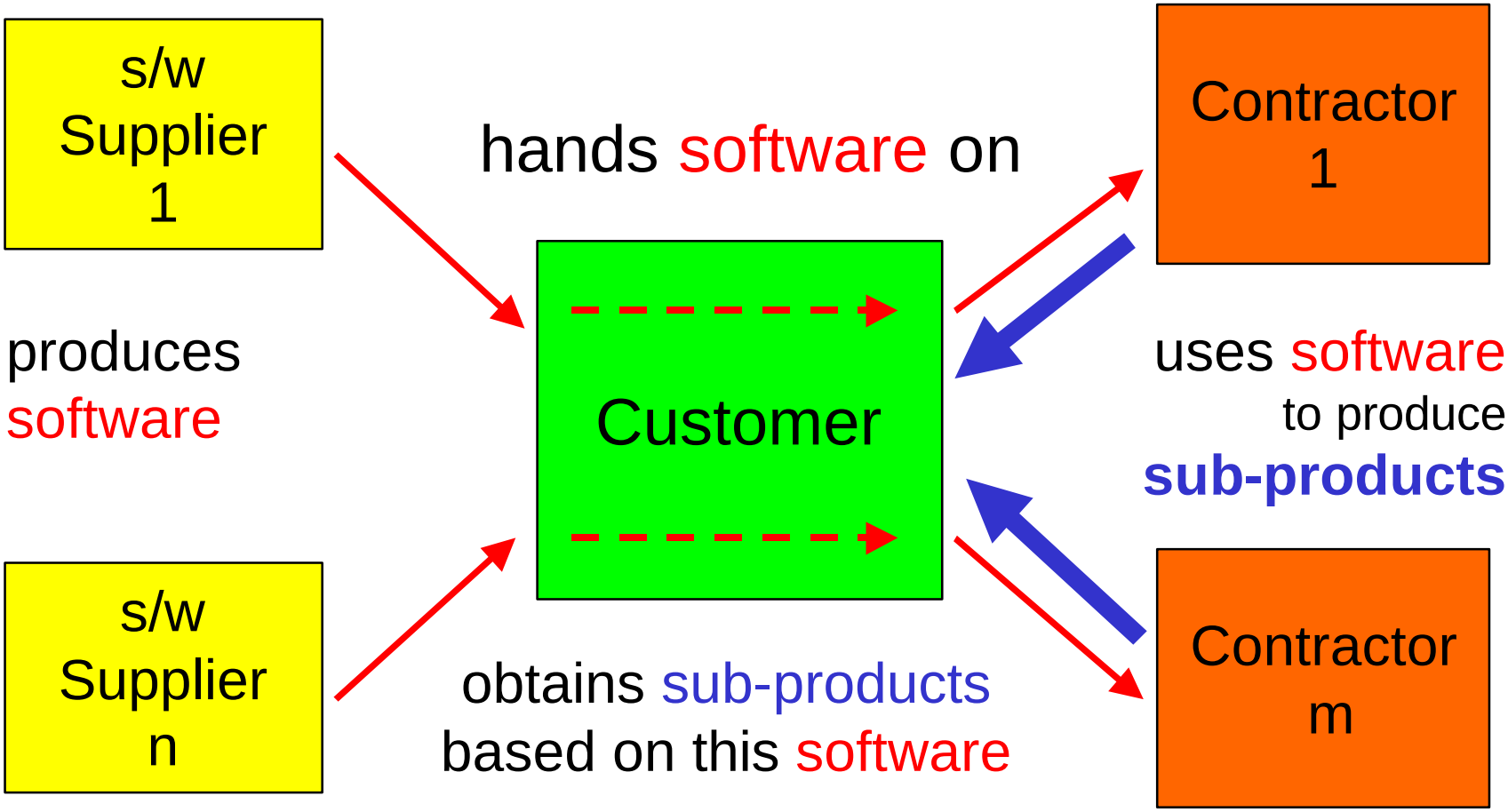
METHOD > PARK

Quality Assurance of ”Handed-On Software”

Uwe Hehn

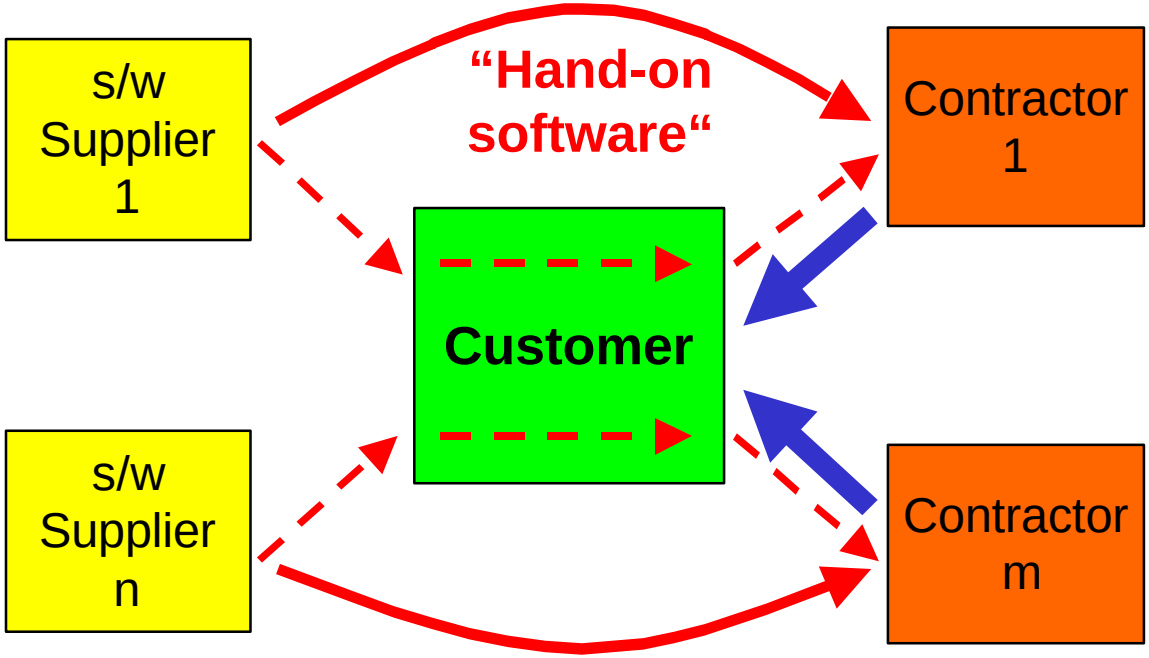
(Uwe.Hehn@methodpark.de)

Starting-Point



Starting-Point

- How can the **Customer make sure**,
  - that the **software delivered** has the **quality needed**
  - to be **used without problems by his contractors** to provide **sub-products** based on this software ?



Observation 1

The quality assurance
of the (hardware) sub-products
supplied by the Contractors
is covered by the process
“Supplier's Management“

This is an traditional technique, which is typically applied in hardware-oriented environments

Observation 2

From point of view of the Customer
the s/w suppliers are “Suppliers“ as well!

Similarly Supplier's Management should provide the framework for the quality assurance of the sub-product “handed-on software“

- But what if the customer does not know how to accomplish this task for software?
- Often the Customer neither has s/w knowhow nor does he want to setup such knowledge!

Thesis 1

Supplier's management is also necessary for software!

If there are no measures for assuring the quality requirements,
typically severe delays and high costs will result

- To do
  - Preparatory planning
  - Make suitable agreements with s/w Suppliers

Thesis 2

Supplier's management costs time and money!

Thorough planning as well as the realization of the agreements made with the s/w Suppliers means expense

The expense arises both at the s/w Supplier and at the Customer!

- To do
  - Consider the expense in project planning and calculation

Thesis 3

Supplier's management is rewarding!

The invest of time and money in quality assurance typically is lower than the delays and costs which would result without such provisions

Especially this is true, if there are many s/w Suppliers

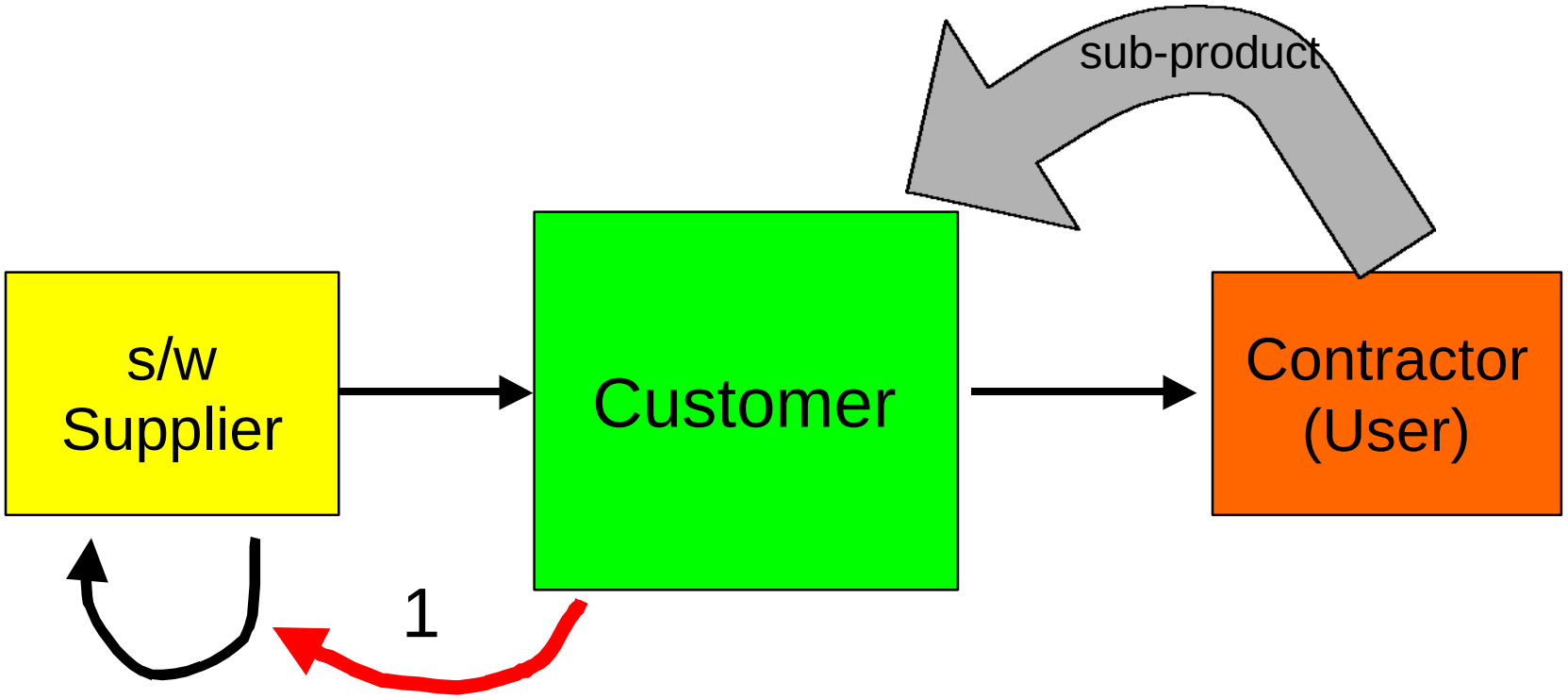
- Remark:

  In some areas - for instance medical software or software used in military applications - supplier's management is common practice for a long time.

How can the customer influence quality?

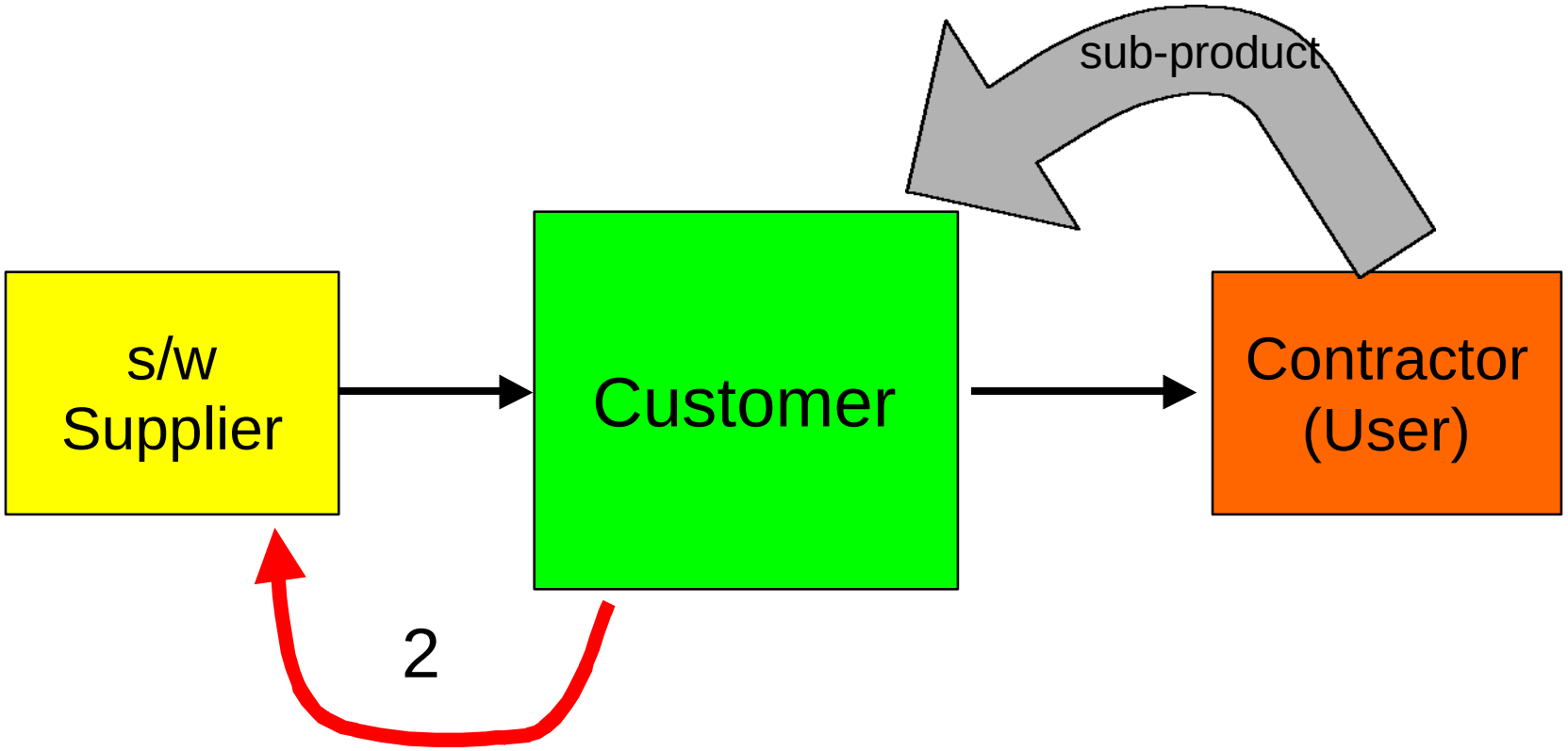
- In the following
  we will examine the practicable cases

How can the customer influence quality?



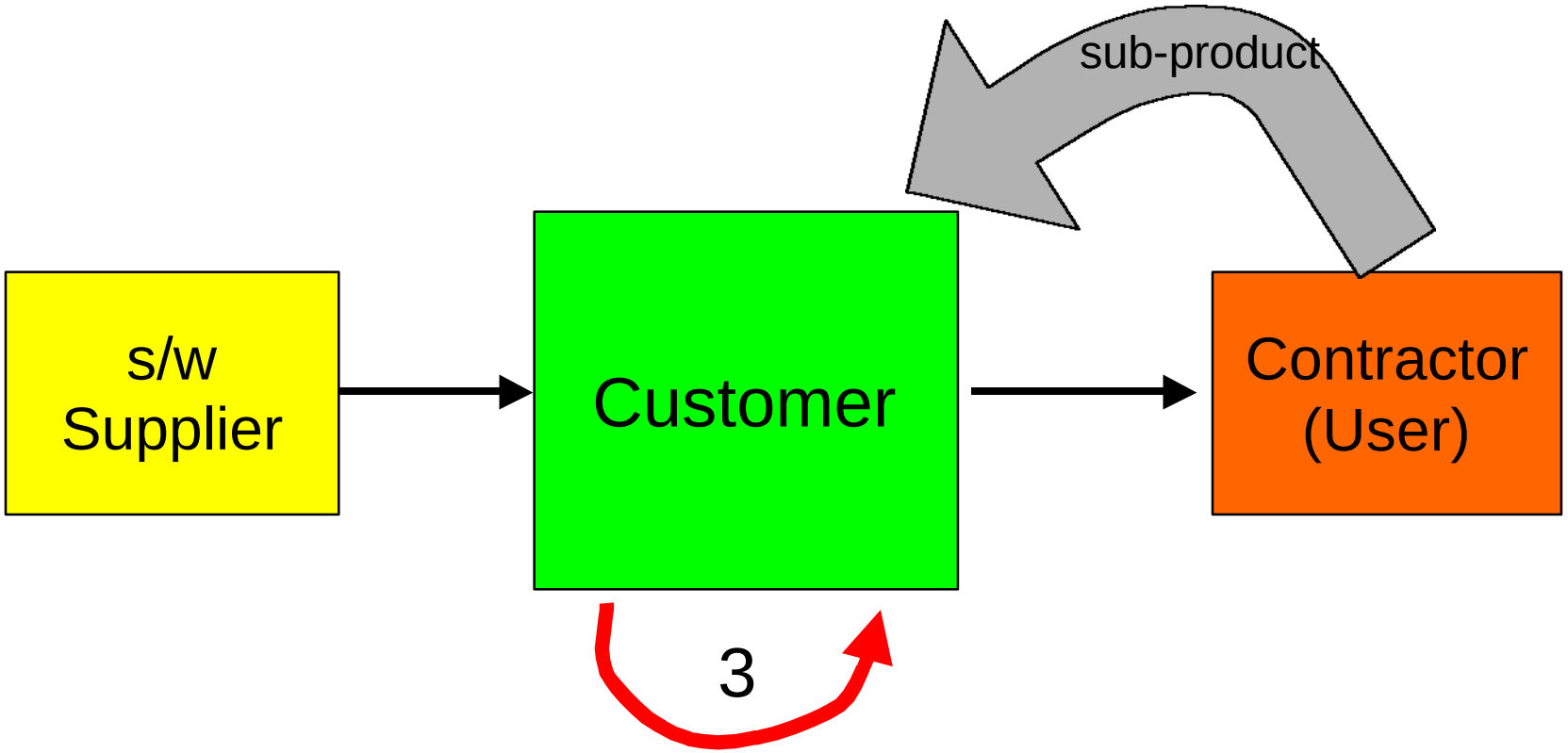
1: Direct access
to the quality control of the s/w Supplier

How can the customer influence quality?



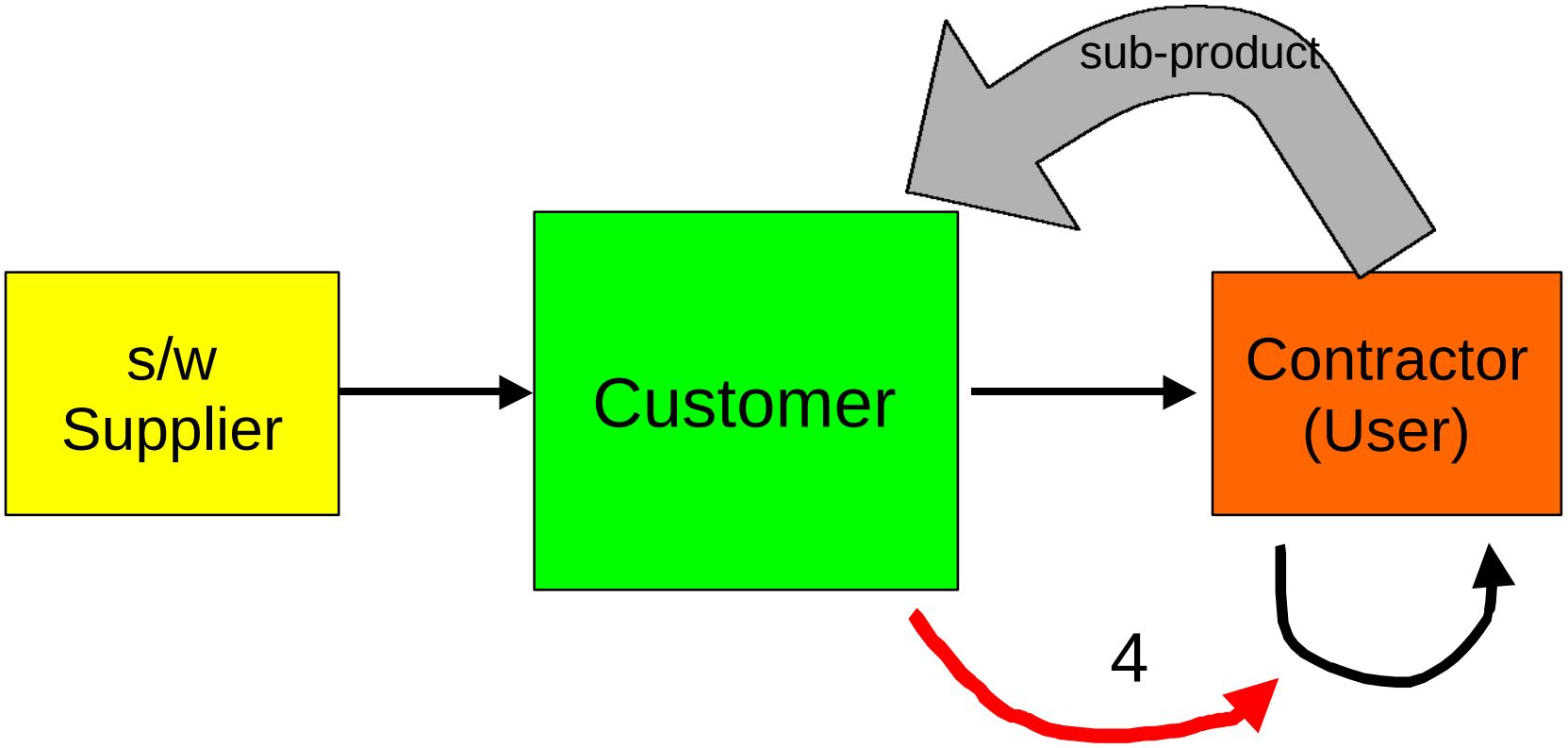
2: Define quality requirements regarding s/w Supplier

How can the customer influence quality?



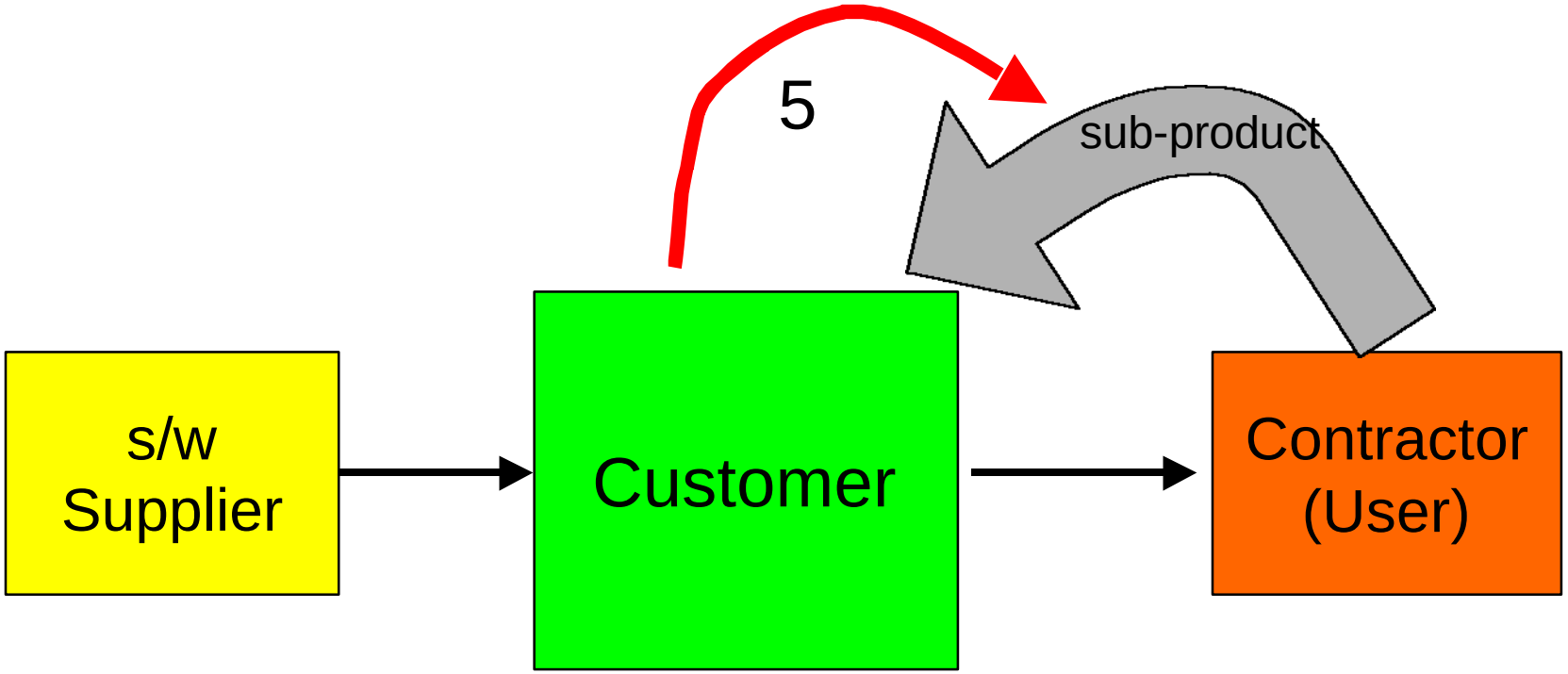
3: Check yourself
software obtained by the s/w Supplier

How can the customer influence quality?



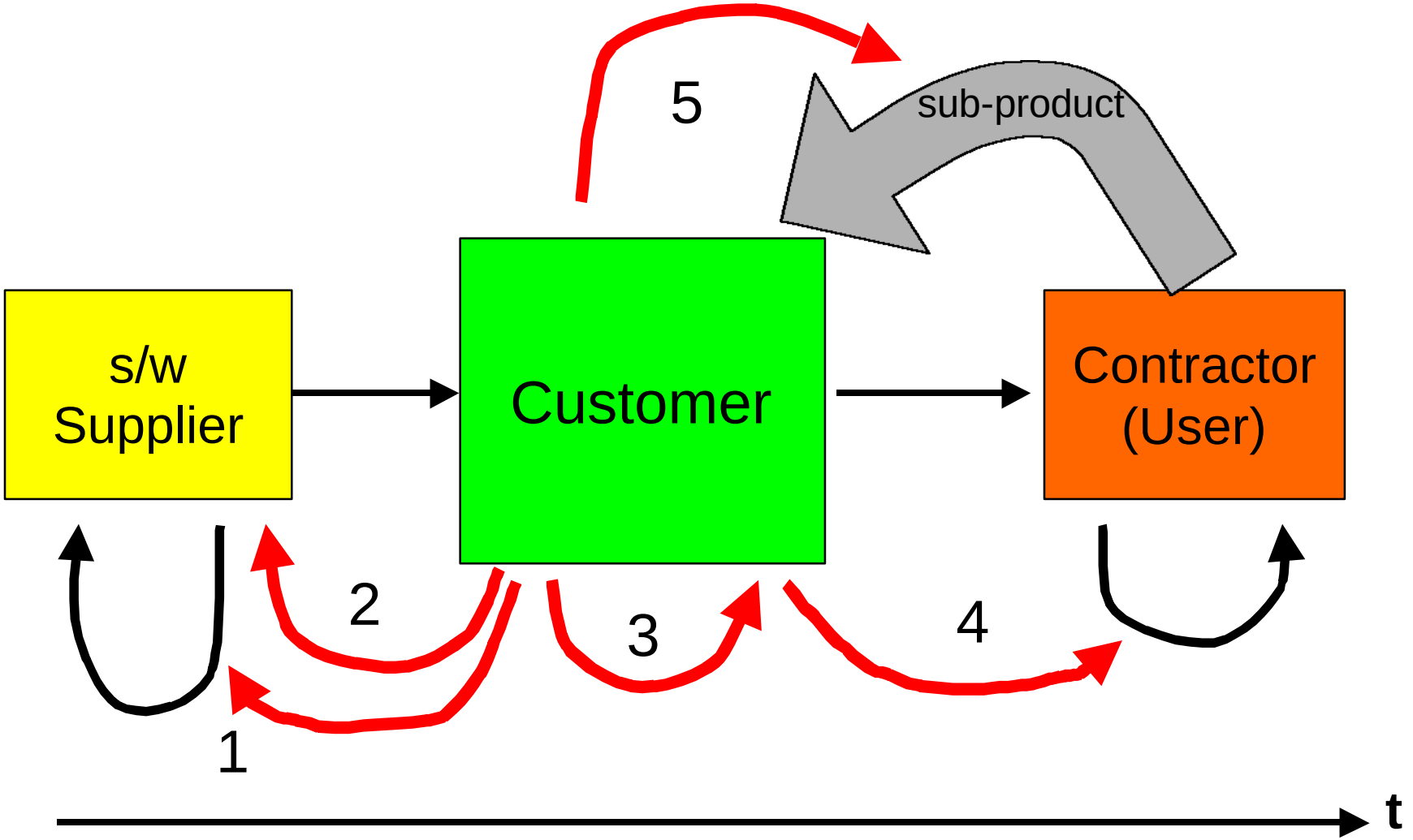
4: Direct access
to the quality measures of the Contractor

How can the customer influence quality?

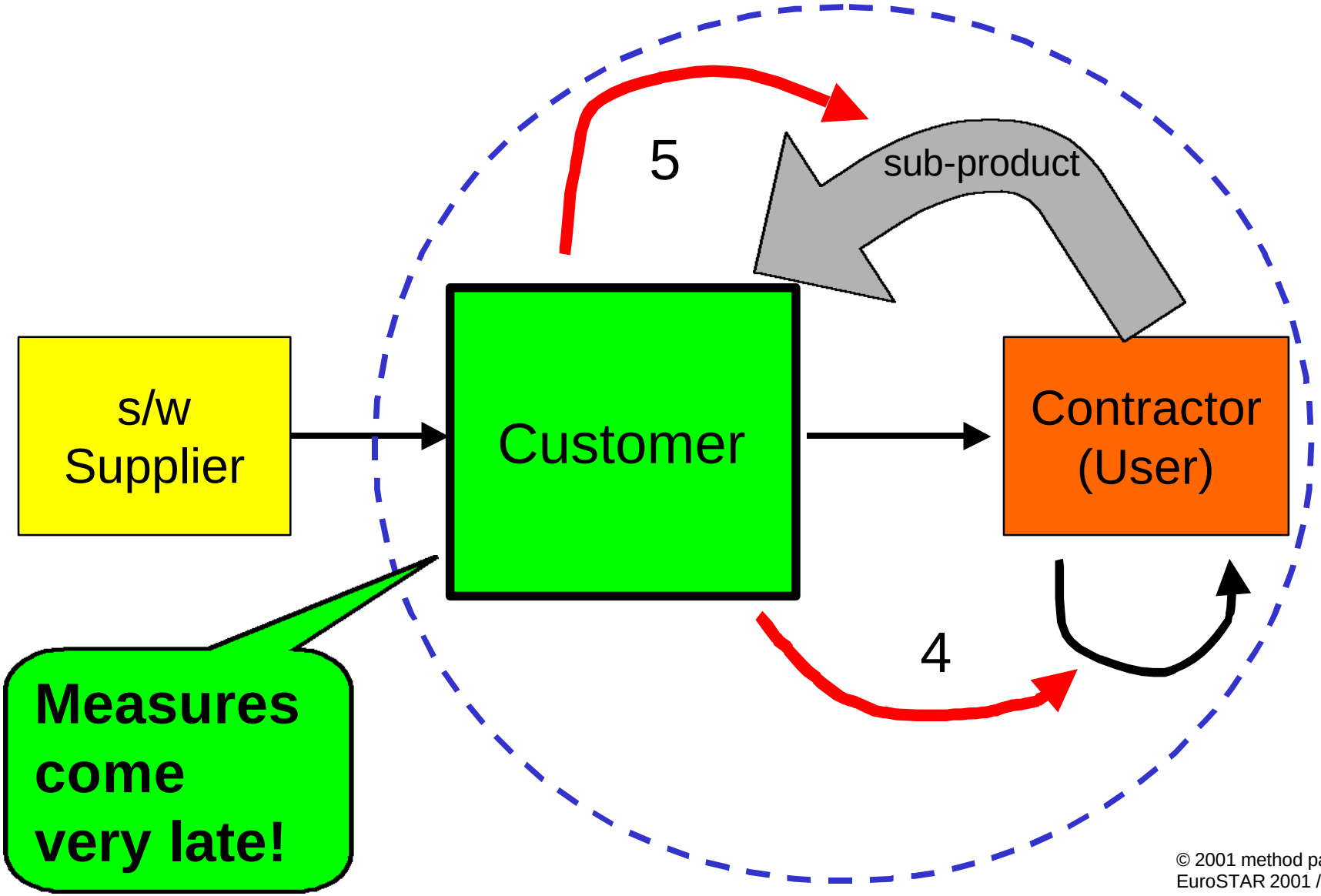


5: Check the sub-products obtained by the Contractor

Retrospect: How can the customer influence quality?



Late influence



Measures of Quality Control

Feedback by users of the software (4)

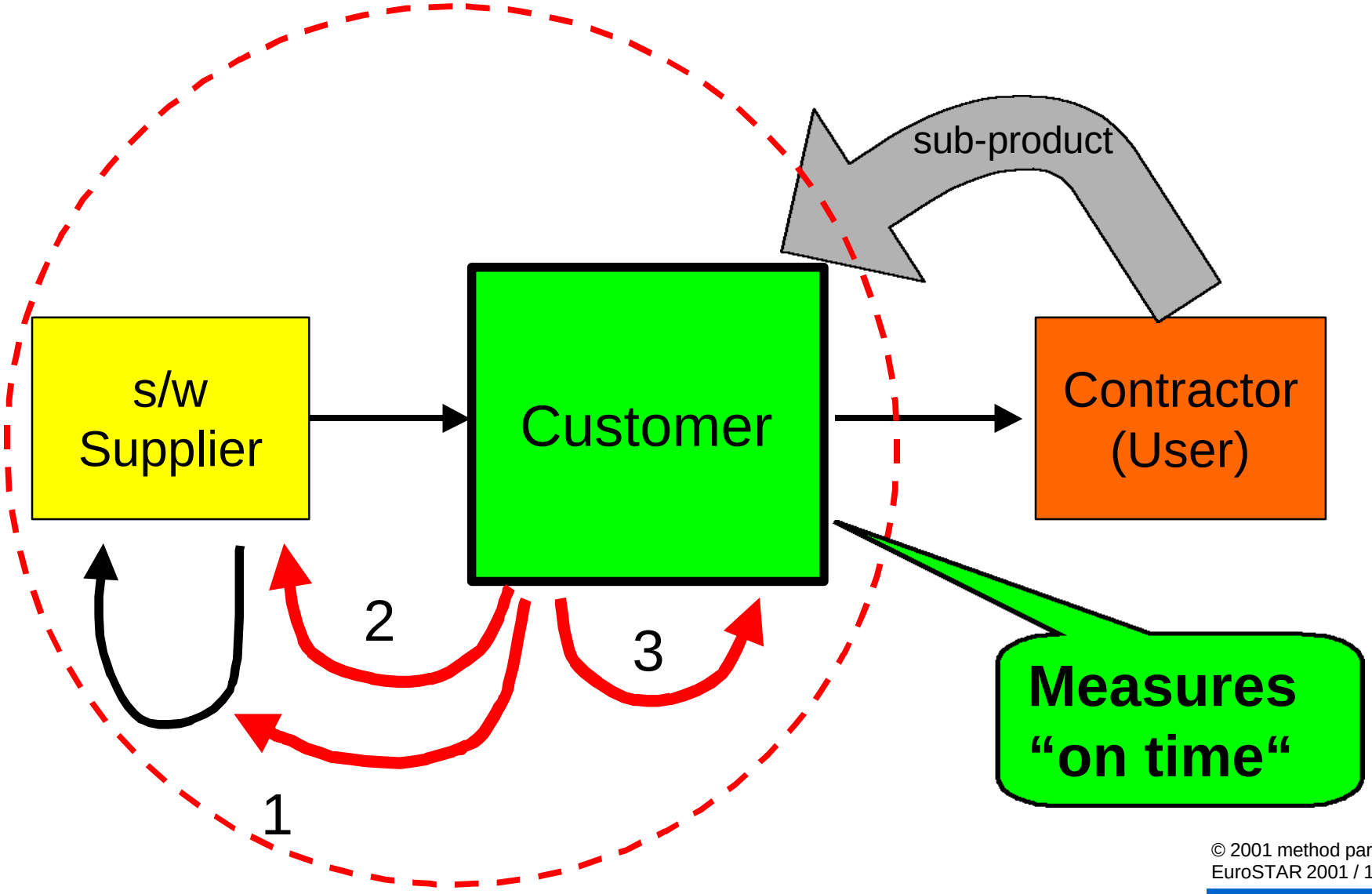
- Errors in software delivered are actually found
  - But: Cause of error really in delivered software?
  - Or: Wrong use of software?
  - Difficult to draw the border line!
  - Anyway: It‘s too late!
- If there remain errors even after using the handed-on software
  - fatal consequences have to be expected
- If problems arise when using the “handed-on“ software
  - cost explosion, severe delays
  - loss of good reputation

Measures of Quality Control

- Examination of the sub-products created by the Contractor (Software User) (5)
  - Result

    Essentially the same as "Feedback by users of the software" (4)

Early influence



Measures of Quality Control

Quality control of the s/w Supplier (1)

- Requirements specifications
- Development related documentation
  - Design specification
  - Interface specification
- Validation concept
  - Test and reviews
  - Corresponding test and review protocols
- Further possibilities
  - Development and use of a (automated) validation suite
  - Let authorities or other acknowledged institutions validate the software

Measures of Quality Control

Quality related requirements of the s/w Supplier (2)

- concerning its software engeneering process
  - for instance CMM, Spice; TPI
- concerning its software
  - see Quality control of the s/w Supplier (1)
- especially
  - making transparent methods and results of examination
  - validation suite
  - validation by authorities and other institutions
- Delivery of documentation to the Customer

Measures of Quality Control

Examinations of the software obtained by the s/w Supplier (3)

- Accept certificates of the s/w Supplier
- Perform examinations with support of the s/w Supplier
- Perform examinations independent from the s/w Supplier

- Methods of examination:
  - statical: statical code analysis, reviews
  - dynamical: tests using a validation suite (tailor-made by the Supplier or - better - a third party)

Proposal for Proceeding

- Quality assurance at the s/w Supplier
  - Deep and thorough quality assurance at the s/w Supplier (together with all required documents)
- Examinations at the Customer
  - Use of a “Button-Press“ validation test suite
    - acceptance test performed at/by the Customer
  - Use tools for evaluation of metrics and for statical analysis
  - Supplement by code and docu reviews performed by the Customer

Proposal for Proceeding

- The Customer should come early to an agreement with the s/w Supplier concerning
  - Extent and depth of the examinations to perform by the s/w Supplier (tests, reviews, ...)
  - Development directions, conventions, ranges of acceptance

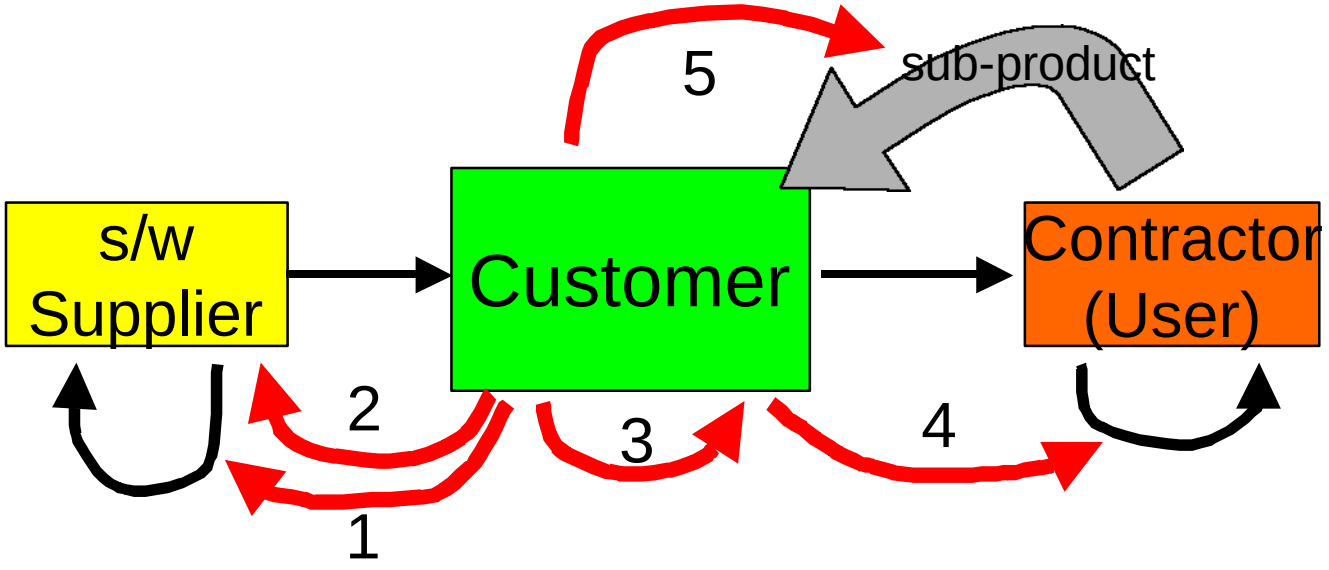
Proposal for Proceeding

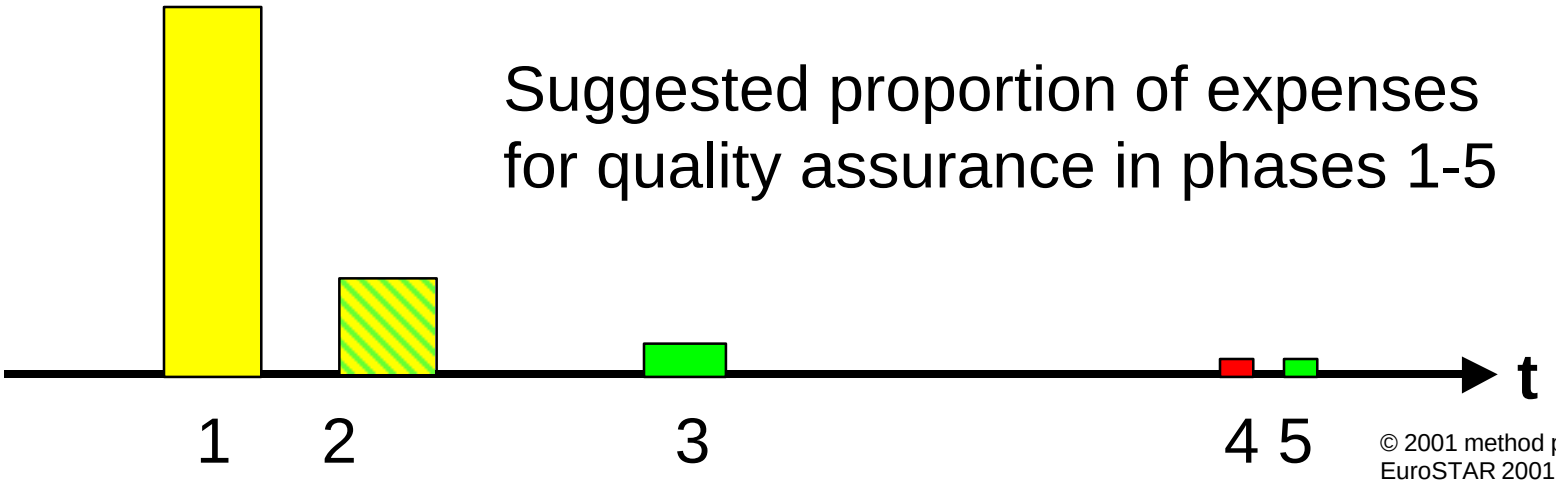
- Then check if the preconditions for effective tests or reviews are fulfilled
  - Records of the s/w Supplier okay?
  - Statical checks successful?
  - **Otherwise**: Send software back to the s/w Supplier!
- Only if software is acceptable
  - perform code reviews on samples (e.g. using metrics for finding critical parts of the software)
  - perform validation test suite

Lessons Learned

- The real quality assurance must be performed at the s/w Supplier
- The Customer‘s task is:
  define prerequisites and check them
- The Customer performs only a few inexpensive checks at pre-checks software, e.g.
  - Automatic performance of the validation test suite
  - Only a few samples of code and docu are reviewed
  - Certificates of the s/w Supplier are accepted, as long there is no cause to refuse them

Lessons learned





Suggested proportion of expenses for quality assurance in phases 1-5

Summary

- Supplier's management for “handed-on software“
  - is necessary (← thesis 1)
  - costs time and money (← thesis 2)
  - but is rewarding! (← thesis 3)
- In fact, there is no real alternative to Supplier's management for “handed-on software“

Quality Assurance of ’Handed-On Software’

Dr. Uwe Hehn
method park Software AG
Wetterkreuz 19a
91058 Erlangen / Germany

mailto:Uwe.Hehn@methodpark.de

1 Abstract

If a customer obliges ist contractors to use software delivered from his software suppliers, he has to make sure the necessary quality of this software – in the following called ”handed-on” software because it is handed-on to contractors by the customer without the customer typically using the software by himself. Otherwise problems as strongly growing costs and significant delays will result. So it must be the customer’s own interest to be able to assure a high quality of the ”handed-on” software. However this may be a difficult job, particularly if the customer does not utilize the software himself, and does not regard the implementation and verification of software as a domain of his core competence.

2 Introduction

The following scenario shall explain and motivate a situation often occurring in many areas such as, for example, automotive industry or industrial automation (Fig. 1):

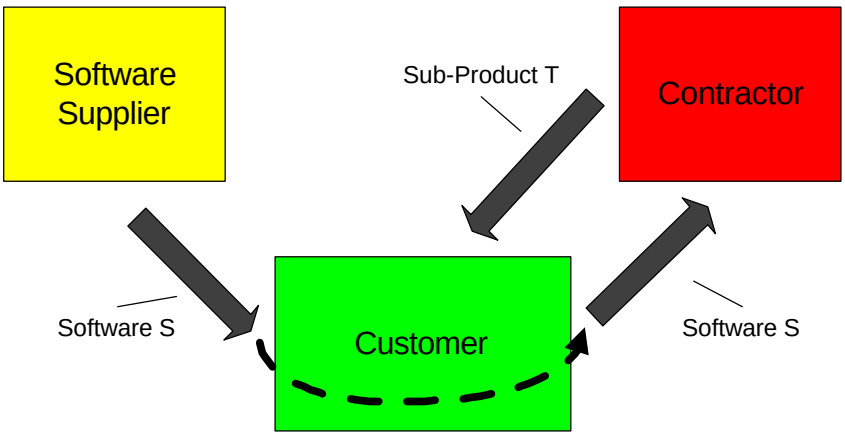


Figure 1: Customer hands on software from a supplier to a contractor

- A company - the *customer* - commissions its *contractor* to build and deliver a sub-product T, which the customer will then integrate into its product.
- The customer obliges the contractor to use the software product S – made by a third company, the *software supplier* – for the production of sub-product T. For this purpose, he provides the contractor with the software he received from the software supplier.
- Before handing on software S to the contractor, the customer wants to make sure that this software is as error-free as possible, complies with the given interfaces, etc. in order to avoid

  a) complications arising on the contractor's from the use of software S during the production of sub-product T and

  b) serious problems and delays during the integration of sub-product T into the product.

As the distinguishing property of the software regarded here is, that after delivery by the software supplier it is handed-on by the customer to the contractor, this kind of software shall be called "handed-on software" in the following.

Oftentimes, not only the software of one software supplier is used but also several software suppliers will provide software packages S1, S2, ... for the production of sub-products (see [4]). On the other hand, several sub-products T1, T2, ... will be built by several contractors using software packages, S1, S2, ... (Fig. 2).

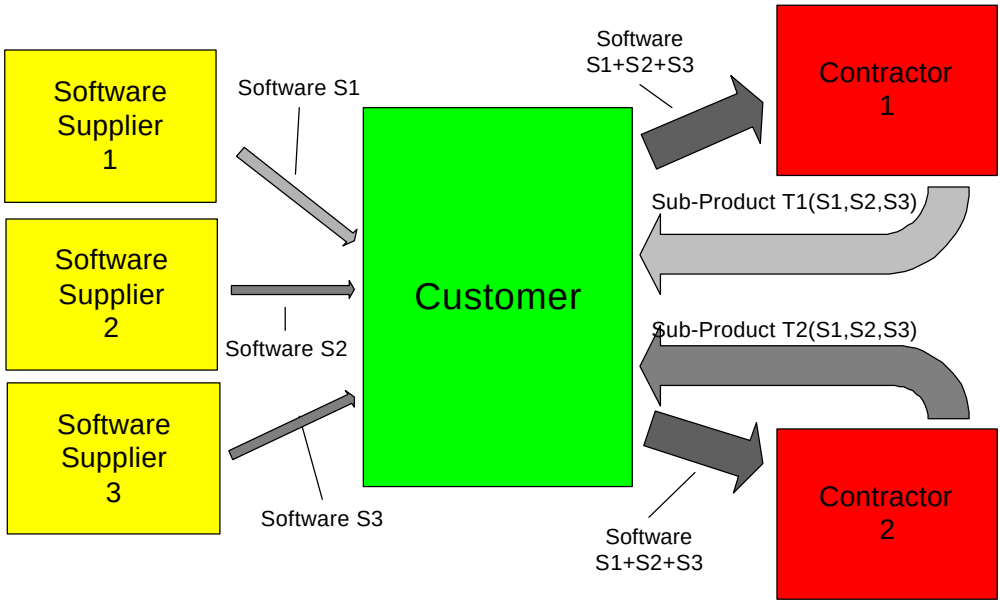


Figure 2: Contractors use software provided by several software suppliers

The customer has to solve this task successfully ”with adequate effort”. This means especially that

a) not all basically feasible measures can be applied,

b) redundant test measures must be avoided.

The solution will be for the customer to use a clever combination of possible measure to achieve good coverage with a given ”adequate effort”.

The first part of the paper outlines the basic possibilities of quality assurance a company may deploy in this scenario. Then a suitable combination of these possibilities is proposed, which allows an efficient quality assurance for ”handed-on” software with reasonable effort.

3 Basic Intervention Options

3.1 One Software Supplier

We first look at the simplified situation where the customer receives software S from only one supplier and hands it on to contractor Z for use during the production of sub-product T (Fig. 3).

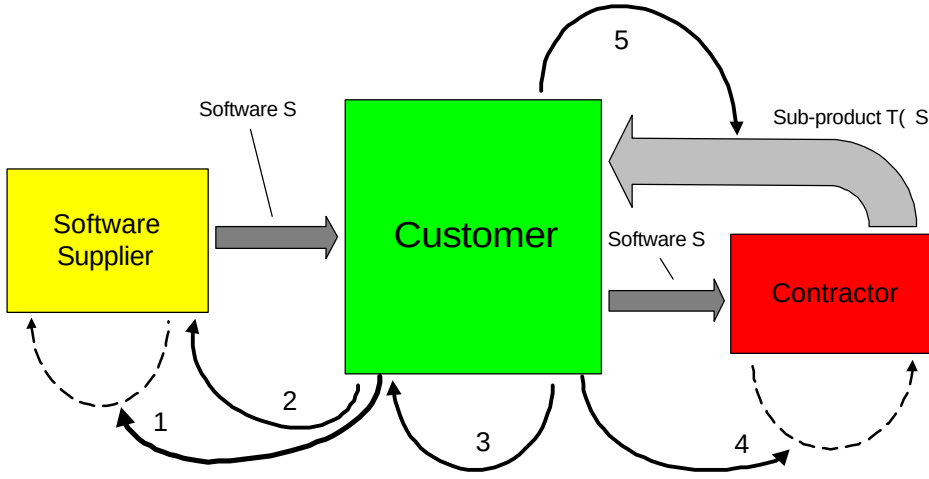


Figure 3: Basic intervention options of quality assurance

The customer can choose between the following intervention options:

a) The customer trusts the software supplier's own quality assurance.

b) The customer influences the software supplier's quality assurance.

c) The customer tests the software received from the software supplier before handing it on to the contractor.

d) The customer judges the quality of the software using feedback from the contractor.

e) The customer indirectly tests the quality of the handed-on software during the integration of the sub-products into the product.

The following examines the usefulness of these options.

3.2 Customer Uses the Software Supplier's Own Quality Assurance

The customer assumes that the software supplier has defined an adequate software development process. Part of this well-defined process are a precise requirements description as well as design and interface descriptions.

A testing concept based on the requirements, correctly filled-in testing protocols and review protocols of the document and code reviews conducted during the tests document the quality of the software supplier's software development process. Proof of this can be supplied with a mandatory confirmation given by the software supplier.

The software supplier can also have the tests conducted by a different company offering testing services.

Moreover, the availability of a validation suite for the judgement of the delivered software comes in very handy. A successful validation by an organisation such as the German TÜV ("society for supervision of technical systems") is another confidence-building measure.

If the delivered software is an established and proven standard product available in a same or similar form, and if the software has proven its functionality and reliability in wide-spread use, this kind of quality control is usually considered adequate. Typical examples are compilers or operating systems that have proven their quality for certain platforms.

3.3 Customer Makes Quality Demands on the Supplier

The customer has to define quality requirements for the custom-made software or the software specially configured for the current project. Furthermore, the customer will expect a good software quality from a certified software process.

An accepted certificate (e.g. CMM level [1,2], SPICE [2]), can be required as proof for the maturity of the software development process; an effective testing process can also be verified by a successful audit corresponding to a test-process-improvement model (e.g. TPI [3]).

The quality control measures undertaken by the software supplier – e.g. in the context described above – should be revealed to the customer; this is especially true for all testing methods and results. If desired, an available validation suite and all documentation pertaining to the software development process must be provided to the customer.

3.4 The Customer Tests the Software Provided by the Software Supplier

Theoretically, it should be possible for the customer to directly assure himself of the quality of the delivered software. Oftentimes, however, the customer does not have the facilities for conducting his own tests or other inspections, because, e.g. he does not have the necessary software development or testing know-how and he does not plan to engage in building up. Only if software development is one of the customer's main competences, he will really be interested in performing the test by himself.

The following approaches are theoretically feasible:

- Conducting tests with support from the software supplier
- Conducting tests independent from the software supplier
  a) in the customer's own test lab
  b) contracting the tests to an external test lab

3.5 Measures in order to evaluate software quality

In general the customer can apply the following measures:

- static inspections (static code analysis, reviews)
- dynamic tests (e.g. running an available validation suite, provided by the

supplier or a neutral organization)

In general, dynamic tests can only be conducted in special environments (e.g the software suppliers' development or testing environment). This is especially true when the software also accesses project-specific hardware. Dynamic testing outside of the software supplier's development/testing environment is very costly. Therefore, if the customer insists on conducting his own dynamic test, this testing should be done in cooperation with the software supplier, and within his environment, if possible.

Static testing does not require special technical efforts. In principle, static tests can be conducted without the assistance of computers, even though the use of static analysis tools can decisively increase the efficiency of static tests. Since computer-assisted static analysis is independent from the target platform, it can be conducted on a suitable available standard platform.

The customer may subcontract the execution of both dynamic and static tests to a software/test house that is independent from the software supplier.

3.6 Customer's Quality Control Using Feedback from the Contractor

If the contractor finds deviations in the software he is to use, these deviations must be analysed carefully: Is there really an error in the delivered software? Has the software been used incorrectly? Sometimes, this distinction is difficult to achieve and can require a lot of time. If an error is indeed found in the software, the delivery of an improved version will take some time, which can lead to a significant delay in the planned deadlines.

Of course, it is preferable for the contractor to find any remaining errors, rather than accepting those errors as a ticking "time-bomb" in the sub-product. Nevertheless, it should be attempted to find any possible errors before the software is handed on to the contractor – among other things to avoid additional claims made by the contractor. As is generally known, fixing errors gets more expensive the longer the time between its formation and its exposure and removal is. Costs as well as the required time especially, increase exponentially, which can have serious consequences.

Besides a costs explosion and drastic delays, the customer's good reputation might also suffer, which makes an effective quality assurance even more important.

3.7 Customer Indirectly Tests the Delivered Software during Integration of Sub-Products

This situation widely corresponds to the one described in 3.6. Since possible errors are found even later, however, the above-mentioned problems will get even worse (costs explosions, delays, loss of reputation).

3.8 More Than One Software Supplier

If there are more than one software suppliers, the situation is similar to that with one supplier with the additional problem that the software packages S1, S2, … delivered by the different suppliers generally must cooperate.

By adding a software integrator, the situation ”more than one software supplier” can be reduced to the situation ”one software supplier”.

4 Efficient Intervention Options for the Customer

After the general intervention options available to the customer have been clarified, the following describes a sensible concept for a quality assurance of the ”handed-on software” from the customer’s point of view. (The numbers in round brackets used in this section refer to figure 3 respective figure 4.)

4.1 Influence Too Small

The alternative ”software developer’s own quality control” (1) only applies to standard products where the customer can exert only little influence. If the customer needs more control, he should negotiate with the software supplier on that account. The software supplier, for example, can give the customer access to his quality assurance methods, or agree upon a joint validation of his standardized software. This, however, changes the customer/software supplier relationship towards the alternative ”customer makes quality demands on the software supplier” (2).

4.2 Influence Too Late

The influence options given by ”feedback from the contractor” (4) and the ”indirect test during integration of the sub-products” (5) do not help to prevent errors and problems. Errors are discovered at such a late point in time that significant consequences regarding costs and time schedule must be expected: the removal of errors in late phases generally is costly; the probability that the software contains even more undiscovered errors is high.

Therefore, feedback from the contractor and quality assurance during the integration of the product are not really suited to achieve a good quality standard for ”Handed-On-Software”.

4.3 Realistic Influence Options for the Customer

Based on the above conclusions, only the following influence options remain to the customer for an efficient quality assurance (Fig. 4):

- The customer actively influences the software supplier’s quality control (2).

- The customer subcontracts the integration of the individual software packages and the validation of the successful integration to a software integrator (2a)
- The customer himself tests the software received from the software supplier before handing it on to the contractor, or subcontracts this task to another company (test lab) (3).

These control options are combined in a suitable way to achieve a good quality assurance.

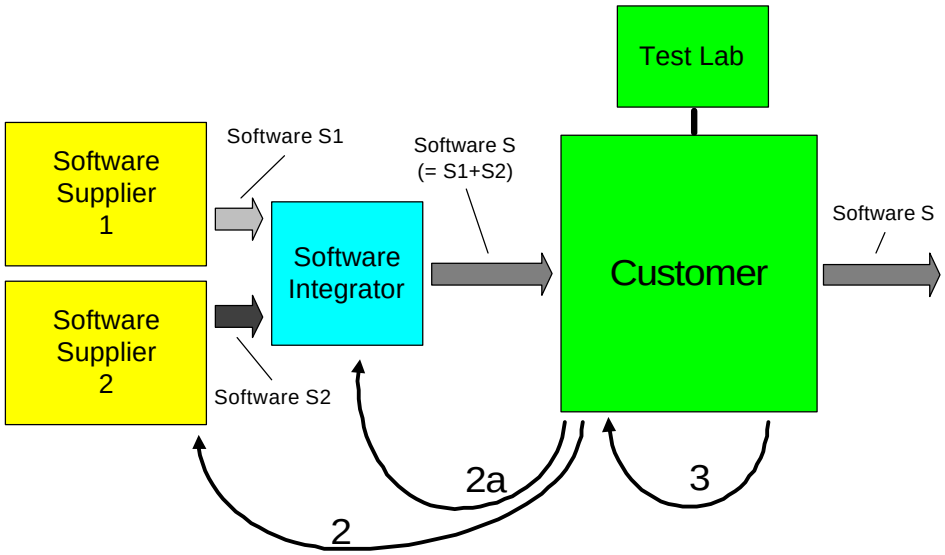


Figure 4: Workable possibilities to assure the quality of ”handed-on software”

5 Putting the Concept into Practice

The following assumes that even though the customer wants to achieve a high quality of the ”handed-on software”,

- the customer can afford only limited expenditures
- the customer does not want to or is not able to conduct his own tests, if they are not available in the form of a very-easy-to-use test suite
- and that the relative costs for implementing the suggested process decreases as such tests are repeated; i.e. a one-time expenditure for establishing the testing method is acceptable, the tests of regular updates or new versions, however, may require only little time and material costs.

In the following the subject ”dynamic tests” is kept only very short. The focus is put on the ”review” or ”static” approach.

5.1 Dynamic Tests

Dynamic tests are used to reveal discrepancies in the execution of specified

test cases. Since the tested software is to be used by different contractors for the development of diverse application software, i.e. the software will be used in various different ways, this must be considered during the software test. Drawing up several typical use scenarios that are made available as a test suite can do this. As soon as the test suite is available, the tests may be performed by an independent test lab or by the customer himself.

Requirements on such test suites are:

- ease to use by a user without special test know-how (”select test case, start test, get the result”)
- and maintainability and extendibility by experts (”new test cases, changed test cases”)

Because of the complexity and variety of the systems under test (for example in the automotive industry many different operating systems, many different processors and many different boards have to be considered), suitable test suites are not generally available but have to be developed tailored to the needs of the customer. Test suites of this kind are very powerful but typically very expensive, too.

5.2 Code Reviews

The following summarizes the different versions of code tests such as desk test, code inspection or code review (see [5]) under the label ”code reviews”.

Code reviews are a very valuable tool for testing and evaluating the source code of software, Together with dynamic tests, which can reveal selective deviations during the execution of specified test cases, code reviews can be used to check functional correlations and the technical quality of the developed software.

Depending on the participants in a code review, different emphases can be observed. We take a closer look at

- code reviews being conducted at the software supplier’s, and
- code reviews being conducted at the customer’s or in a test lab authorized by the customer.

5.3 Code Reviews Conducted by the Software Supplier

The main focus of code reviews conducted by the software supplier is determined by the following parameters:

- Code reviews are being conducted from the point of view of software development (the software developers participate.
- Code reviews can be conducted both during the development phase and at the completion of a software unit’s development.
- Development documents such as system design and module design are

available to the participants of the code review.

- The low-level aspects of the implementation can be tested competently.
- It is possible to run through use scenarios (testing of the functionality).

The customer should demand proof that the code reviews were conducted (review protocols). If necessary, some of the customer's employees can participate in the code review as observers.

5.4 Code Reviews Conducted by the Customer

In contrast to code reviews conducted by the software supplier, code reviews conducted by the customer are determined by different criteria:

- Normally, the software developers who worked on the software will not participate (this makes it impossible or difficult to ask them questions).
- The test will encompass the complete software released by the supplier.
- Testing the functionality is not possible or requires great effort since the required developer's know-how usually is not available to the customer.

The customer can subcontract the code reviews to a test lab but in general, the given criteria also hold for this situation.

5.5 Static Checks

Normally, the customer will accept the test protocols and affirmations provided by the software developer if they guarantee both the required functionality and the non-functional characteristics.

If necessary – e.g. when the suppliers increasingly have problems – the customer may, however, – based on respective agreements – inspect the software supplier's software development process as closely as desired. This makes it possible for the supplier to basically get by without his own complete test of the delivered software. Of course, the transfer of the concept described here can be supplemented with other analytic methods – such as automated regression tests –, which makes them even more effective.

Nevertheless, the customer should not dispense with testing the outer appearance of the delivered software – using static analysis – since this makes it possible to draw substantial conclusions about the care taken during the development of the software.

5.6 Static Checks with Tool Support

The following are examples for relevant checks that can be conducted statically with a relatively low effort:

- observance of development guidelines

- adherence to naming conventions
- testing whether documents and source code consistently contain requirement keys
- testing for adherence to the interface description.

This list can be continued as needed.

5.7 Evaluating Metrics

Metrics tools implement a special kind of static analysis. Software metrics numerically record characteristics of the inspected software. A multitude of commercial tools is available for calculating metrics. In order to be able to use metrics for the quality assurance of ”handed-on software”, sensible, acceptable value ranges must be defined for each one of these metrics.

5.8 Prerequisites for Effective Code Reviews

What can the customer do to effectively use the results of the static analysis or the determination of metrics (s. Fig. 5) ?

- It is necessary to determine, which guidelines and conventions are to be tested using the static analysis tools. The effort undertaken to adapt the tools should not become too large. If need be, one will want to dispense with automated tests for every individual requirements; the requirements that were not tested can then be spot-checked with manual code reviews.
- It is necessary to determine which metrics should be used.
- For every metric employed, it is necessary to define
  - which metrics values are unacceptable (exclusion criteria)
  - which metrics values can be accepted offhand.

  All other values that are neither unacceptable nor acceptable offhand signalise that the code should be checked manually by reviewing code.

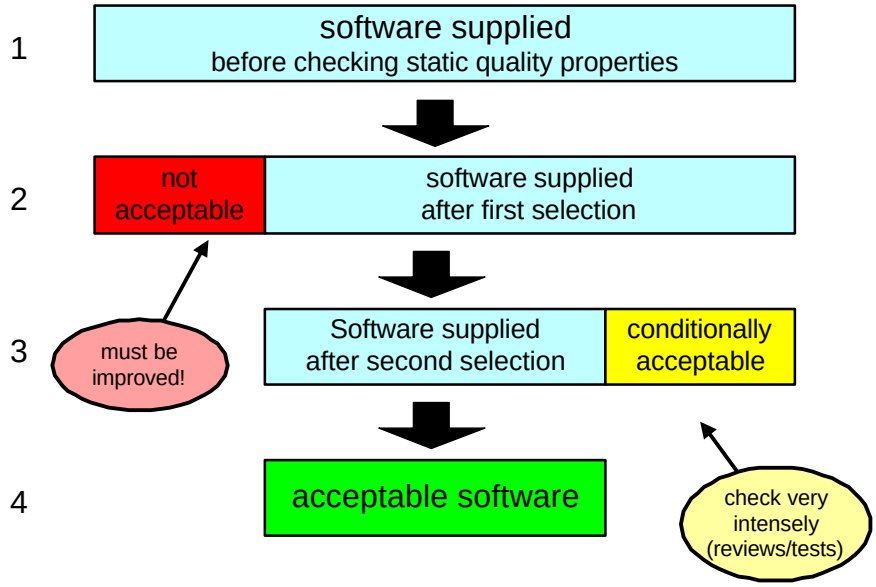


Figure 5: Filtering unacceptable or not necessarily acceptable software

After this preparation, the static tests can be conducted automatically. If all static tests are successful and all calculated metric values are acceptable, the software delivered by the supplier is accepted; otherwise, it is returned to the supplier for remedy.

It is recommended to agree with the supplier about how the acceptance criteria are determined early on, i.e. before starting the tests. Ideally, the acceptance criteria should be agreed upon at the beginning of the project; this way, the requirements on the software are clear from the beginning.

5.9 Extent of Static Tests

The automated tests should include all the delivered software sources, since this requires only little effort and costs. Since only tested and accepted software is handed on, the share of ”not necessarily acceptable” software will not be very high. This part of the software should be subjected to a more intensive code review ”from the customer’s point of view” (this can also be done by a test lab).

The formal aspects - adherence to the structure template, existence of a version history, etc.- of respective documents, release notes, etc. should also be checked with simple tools.

6 Outlook

The quality of ”handed-on software” can be assured by a combination of several methods. The influence of the customer on the software supplier’s quality assurance methods must be strong enough for these measures to show the expected results.

Automated tests using static analysis and metrics were suggested to supplement the software supplier's measures and to provide additional security.

Putting this approach into practice requires that the customer and the supplier agree upon which measures are to be undertaken and that these agreements are lived in everyday business. Considering the product quality and the importance for the end user, the necessity for suitable measurements is essential.

The problem was shown from the customer's point of view. Nevertheless, the presented approach does not only yield advantages for the customer:

- It is very useful for the contractor when the software handed on to him is in a good state. Very often, contractors work under strict time constraints; clearly, problems with the software they must use do nothing to improve this situation.
- The increased influence the customer has on the software supplier might at first be unusual and undesired for the latter; on the other hand, the increased software quality this brings about also strengthens the position of the software supplier, not to speak of the reduced effort and costs needed to process usually high-priority problem reports caused by discrepancies during the integration process at the supplier's or customer's.
- The very fact that someone will look into the software produced will help to make better software!

7 References

[1] Software Engineering Institute: Carnegie-Mellon-University, URL=www.sei.cmu.edu/cmm

[2] H. Balzert, Lehrbuch der Software-Technik: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung Spektrum Akademischer Verlag, Heidelberg 1998

[3] T. Koomen, M.Pol: "Testprocess Improvement", Addison-Wesley, 1999

[4] U.Hehn, B.Haworth: "Testing Distributed Embedded Systems; Factors and Solutions", Test Congress 2000, London, May 8-12

[5] K. Frühauf, J. Ludewig, H. Sandmayr: "Software-Prüfung - Eine Anleitung zum Test und zur Inspektion", vdf Zürich, 2000

Friday 23 November 2001
F8

Quality Assurance of “Handed on Software”

Uwe Hehn

Dr. Uwe Hehn, born in 1954, studied computer science and mathematics at the Technical University of Darmstadt (Germany). He acquired his doctorate at the University of Erlangen-Nürnberg (Germany). Dr. Hehn worked as developer und project leader in software development and test. In 1997 Dr. Hehn joined 3SOFT.

Since 2001 he is working with method park Software AG, which is a spin-off of 3SOFT. His work focuses on consulting, training and teaching in the subject of the definition and implementation of test processes and quality measures at customers development and test divisions. Dr. Hehn is guest lecturer at the University of Erlangen-Nürnberg. He published several papers on testing and quality management.