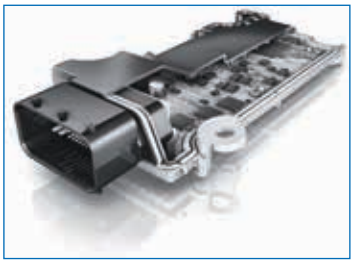
ZVEI:
Die Elektroindustrie

Best Practice Guideline
Software Release






Platform Automotive –
Electronics, Infrastructure & Software

Impressum
Best Practice Guideline
Software Release

Published by:
ZVEI
German Electrical and Electronic
Manufacturers' Association
Platform Automotive –
Electronics, Infrastructure & Software
Lyoner Strasse 9
60528 Frankfurt am Main, Germany

Phone: +49 69 6302-276
Fax: +49 69 6302-407
E-mail: zvei-be@zvei.org
www.zvei.org

Responsible: Dr. Stefan Gutschling, ZVEI

Authors:
Gunther Bauer, ZF Friedrichshafen
Dr. Stefan Bunzel, Continental
Thorsten Geiselhart, Marquardt
Dr. Günther Heling, Vector Informatik
Markus Langhirt, Brose Fahrzeugteile
Henning Möller, NXP Semiconductors Germany

May 2015
1st Revision April 2016

Table of Contents

1. Objectives of This Guideline

The automotive industry is becoming more and more focused on the necessity of a robust and efficient software development process. This is due to the increasing significance of software based functions in vehicles, the increasing interconnection of control units and the rapidly growing complexity. More and more requirements have to be implemented in ever shorter time spans. The growing complexity can only be managed when the development process in the network of vehicle manufacturers, suppliers and service providers is coordinated (incl. clear definition of tasks and responsibilities)

In previous years, the emphasis was on the improvement of company internal processes (such as in the implementation of automotive SPICE or CMMI). The interfaces between vehicle manufacturers and the suppliers were not considered in detail. The lack of common standards right at the "software release" interface leads to a considerable coordination effort and possible misunderstandings. The optimisation of the software release process is of common interest for all participants – it can make an important contribution ensuring the maturity of the software development process.

This guideline summarises experiences and best practices for the essential aspects. This creates awareness of where early bilateral cooperation is helpful, even if clear cross-company recommendations are not possible everywhere. This guideline deals with both parties, contractor and purchaser (e. g. OEM and supplier). A common conception is very important to handle both viewpoints, not just for new business relationships but particularly in this case.

The objective is to present suggestions for an optimisation of the interface (communication and documentation) between the vehicle manufacturer and the supplier. This is an essential pre-requisite for meeting new challenges in the field of software development (driver assistance systems, service focused communication, Car2X, etc.) more efficiently. The definition of the artefacts belonging to a software release is at the centre of this. The primary focus is an equal understanding of the contents with less emphasis being placed on standardised data formats.

2. Release Process

2.1. Software release

Software release means that the software is cleared to be passed on to the user or customer. With the release of software, the supplier gives a statement about the implemented functions and properties and hands them over to the customer within the defined framework for use. Release of the software typically results in the fulfilment of contractual elements of the business relationship between the customer and the supplier. On the other hand software is released as an item, which is delivered at the end of the development process.

Software development for embedded control units can also be considered in a further context. To build up the complete control system, software components, which may be delivered from various suppliers, are integrated on individual control units in a first step. In a second step, all control units are integrated to a complete network in the vehicle. From the vehicle manufacturer's point of view, this applies right up to distributed functions which involve the complete vehicle. Automotive software development is part of a system which involves many participants. It has to consider disciplines such as electrics, electronics, mechanics and interconnection as boundary conditions. For this purpose, a process definition with many synchronisation points has become established in the automotive industry for which a software release timeline is required.

In linguistic interaction – and also subsequently in these guidelines – the term release has different meanings. First of all it refers to the result of the release process, meaning the artefact to be released and also the associated documents and metrics. We will call it "release item" in the following. The process which leads to the release and the release item is the release process. This guideline casts light on the release process from various perspectives in the sections which follow. These perspectives are from the point of view of the supplier and that of the vehicle manufacturer. The release item is to be regarded generically as a component in this release process. This may involve a software component – from the point of view of the supplier. It may, however, also involve a group of software components which are to be released jointly, such as the software for a complete control unit. From the point of view of the OEM, a "component" often also includes hardware and then refers to a control unit to be released for example.

2.2. Process from the point of view of the supplier

The process from the point of view of the supplier is broken down into various development cycles over various sub-systems (mechanics, hardware, software). They can progress at different speeds and they are synchronised with milestones for the complete system (see figure 1). The development methods of the individual subsystems can be independent of one another (e. g. V-model for the hardware and Agile methods for the software development).
The software release process of the supplier contains the following steps:

- Functional extensions, modifications and bug fixes are integrated into the software components according to the release plan.
- The verified components are integrated into the complete software.
- The integrated software is verified as planned (e. g. based on the results of impact analyses for modifications)
- The integrated software may possibly be released together with a calibration dataset (see figure 2).
- The complete software may be delivered together with other system components such as hardware if necessary. The scope of the software release must be defined and validated prior to delivery.
- The complete software and calibration dataset together with the supporting documentation represent the software release item from the point of view of the supplier.

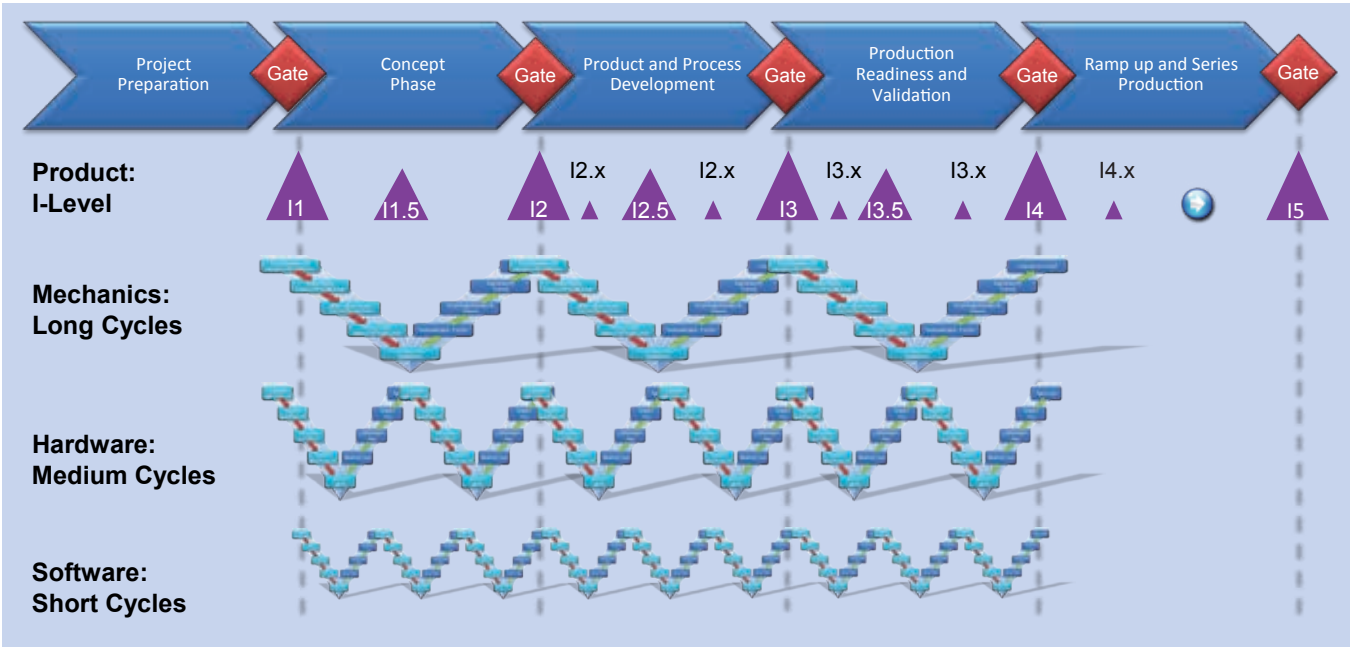


Figure 1: Process: Synchronization of the sub system development (picture source: ESG Elektroniksystem- und Logistik)

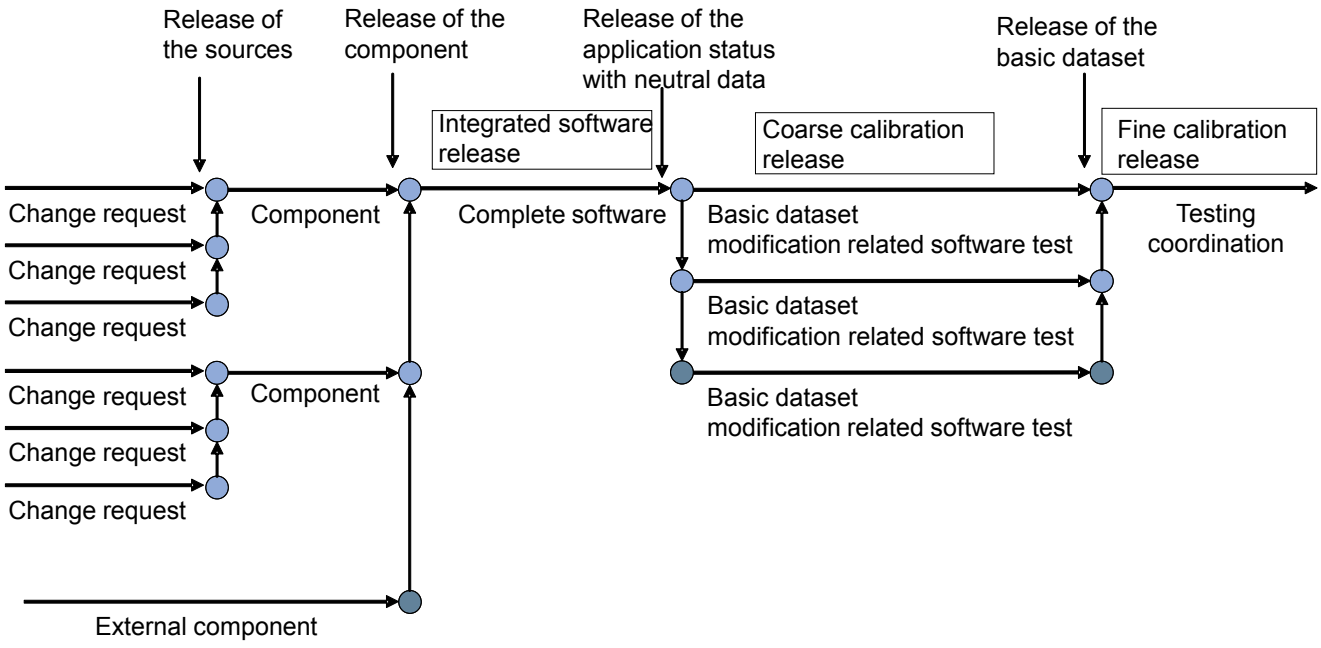


Figure 2: Sequence change request/bug fix (picture source: ZF Friedrichshafen)

2.3. Process from the point of view of the OEM

The OEM expects verified software for the required maturity level from the supplier. From the point of view of the OEM, the software is part of the control unit and thus part of a component. The OEM tests this component in-house in various steps. Component tests, subsystem tests and system tests are carried out. The components (control units) are brought closer and closer to the complete vehicle and tested (see figure 3) in the various validation stages.

In component tests, the component is tested intrinsically, for functional capability and flash capability for example. Subsystem tests validate the interaction between the component and the direct communication partners. The freedom from side effects and the functionality in the vehicle are tested in the complete system. If errors occur during a test, these are fed back to the component developer (supplier) for bug fixing in subsequent releases (see figure 4).

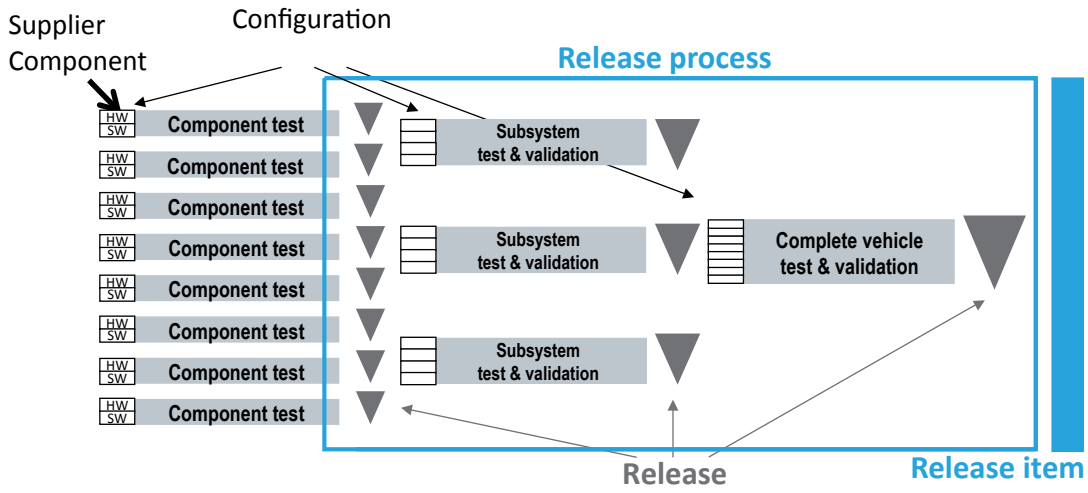


Figure 3: Release process from the point of view of the OEM (picture source: ESG Elektroniksystem- und Logistik)

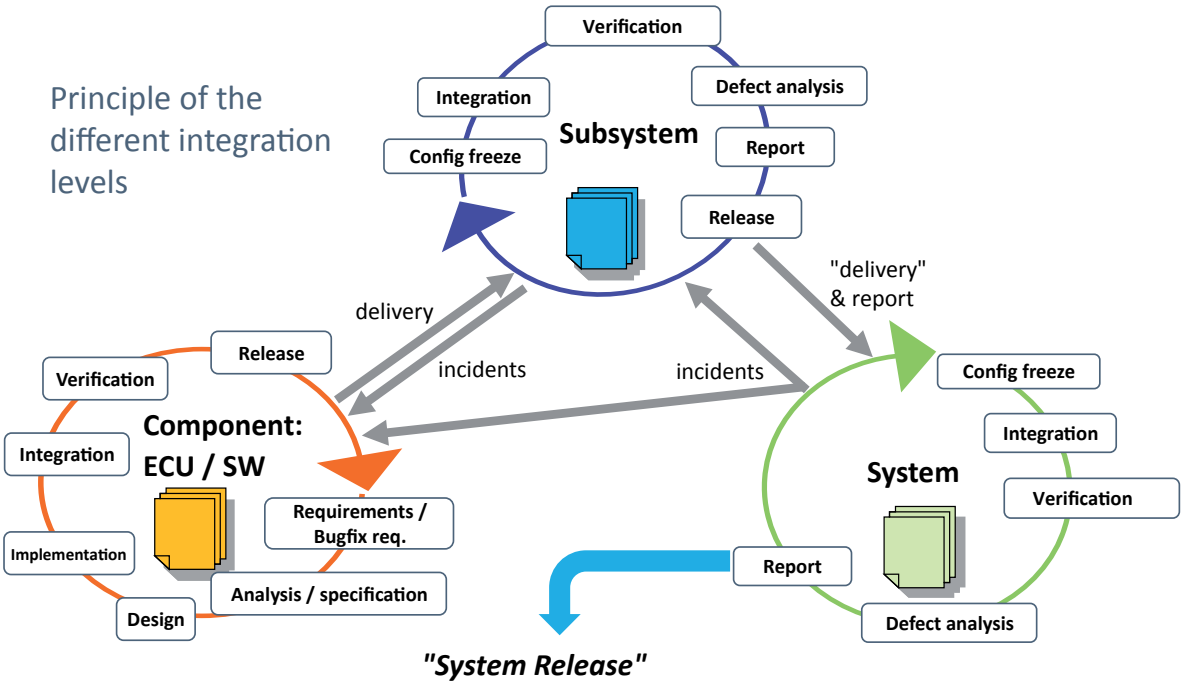


Figure 4: Principle of the different integration levels (picture source: ESG Elektroniksystem- und Logistik)

Release process from the point of view of the OEM – time sequence

The integration of software into the system consists of three different integration stages. The first stage is the integration on the component level, and then the integrations at the subsystem and system level are performed. Testing can be started consecutively but run in parallel. Once the quick checks of an integration stage have been completed successfully, the next test stage can be started. Additional deliveries of software are only permitted for show stoppers. The system freeze is done at a previously defined point in time. From this point in time onwards, no additional deliveries are permitted and the concluding tests are carried out (see figure 5).

Release process from the point of view of the OEM – Theory

Integration stages with defined functional extensions are planned by the OEM. At the start of the development, the intervals are longer, between two and four months. Shortly before SOP, the functional extensions are no longer as complex and come at intervals of 2 to 6 weeks (see figure 6).

Release process from the point of view of the OEM – Reality

Bug fix loops are pushed between the integration stages through unplanned bug fix measures. This can be traced back to the fact that bug fixes are delivered additionally, regardless of the planned timeline. Bug fixes and functional extensions are often not separated in practice.

Capacities for further development and validation may be factored in for these unplanned bug fix loops. Unplanned loops may delay the final software release. At the same time there is a probability that the comprehensibility and transparency of the actions carried out, will suffer due to the loops which are slotted in (see figure 7).

Release process from the point of view of the OEM – Best practice

In reality errors are to be expected at every integration stage. Therefore bug fix loops should be planned right from the start. This ensures a higher quality and transparency in the software development process. The same amount of time should be planned for the additional bug fix loops for each iteration (functional extension) (see figure 8).

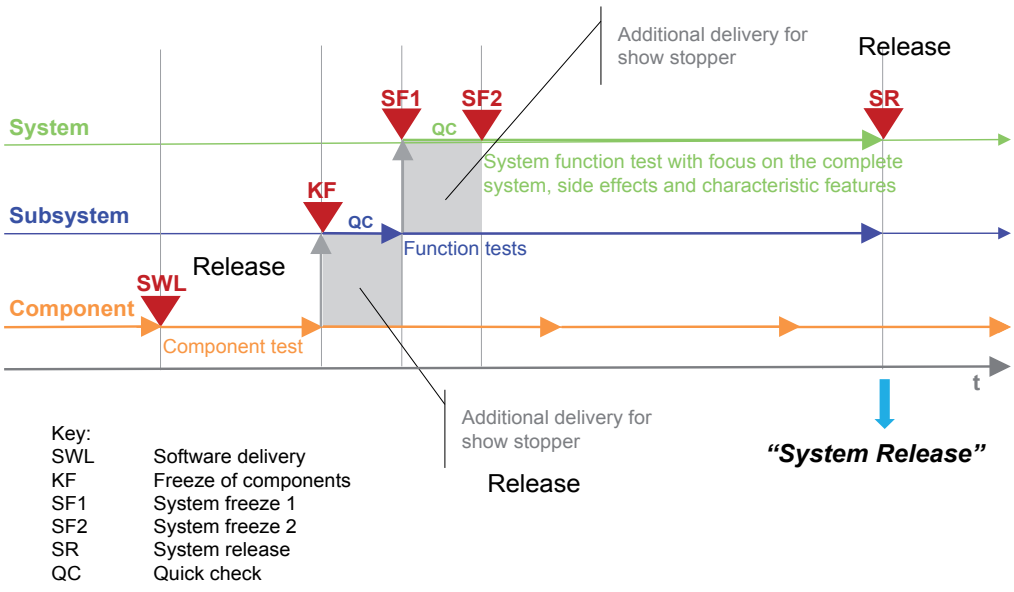


Figure 5: Release process: temporal sequence (picture source: ESG Elektroniksystem- und Logistik)

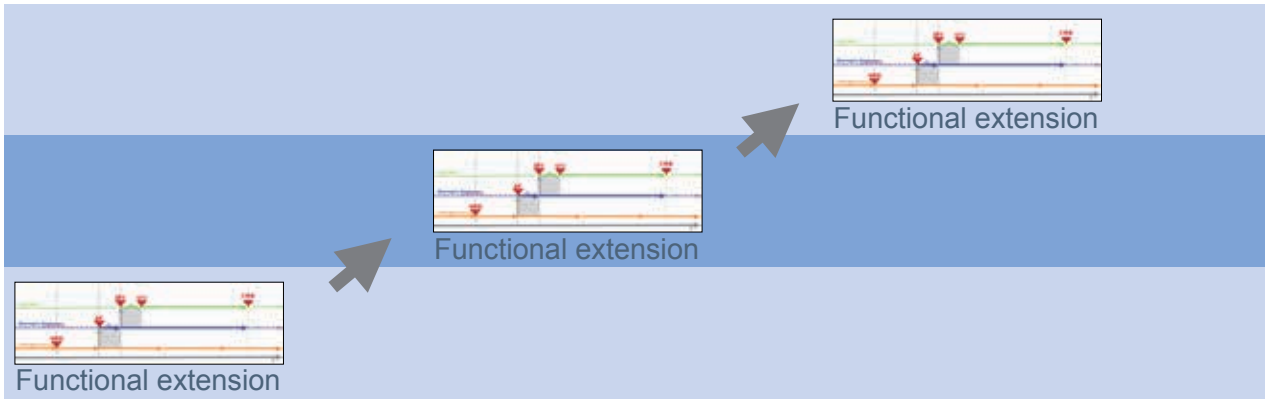


Figure 6: Release process from the point of view of the OEM: Theory (picture source: ESG Elektroniksystem- und Logistik)

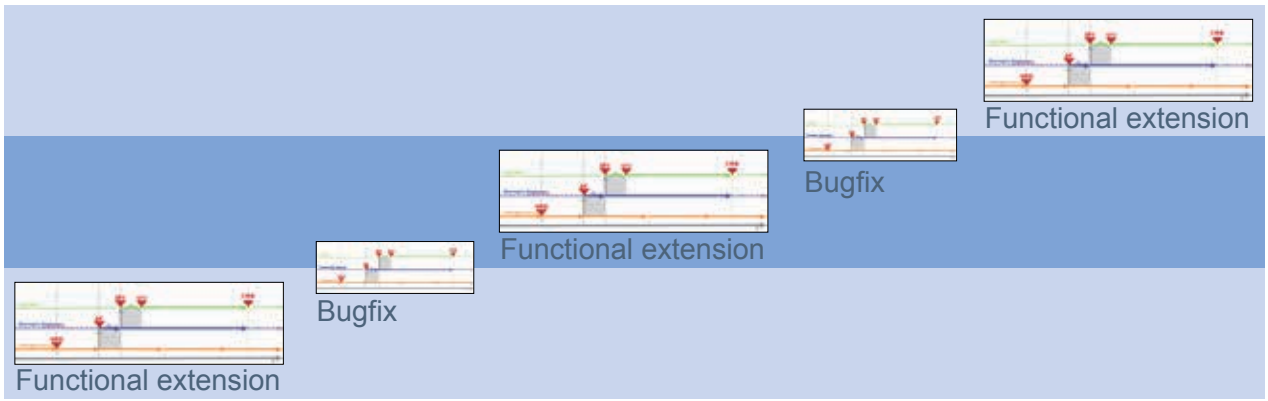


Figure 7: Release process from the point of view of the OEM: Reality (picture source: ESG Elektroniksystem- und Logistik)

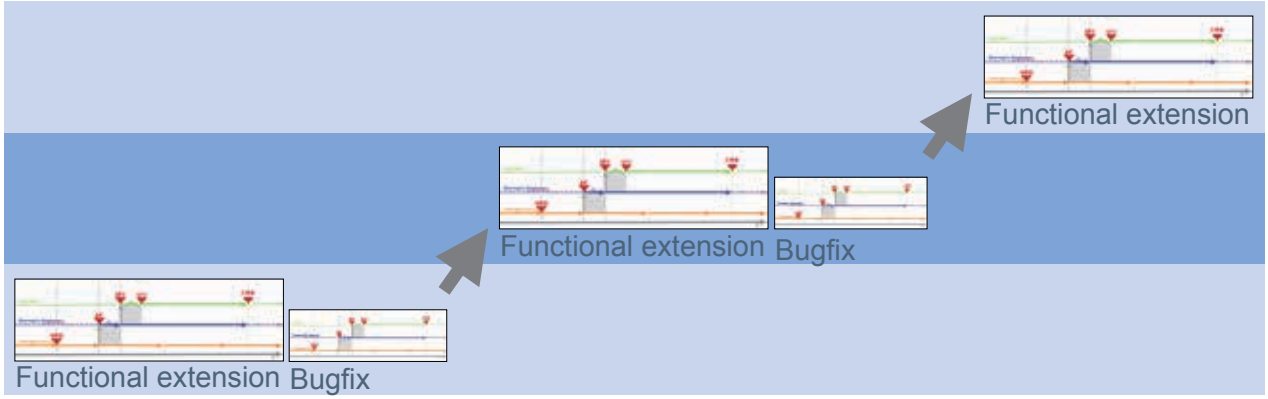


Figure 8: Release process from the point of view of the OEM: Best practice (picture source: ESG Elektroniksystem- und Logistik)

2.4. Impact of externally developed software

Automotive software releases increasingly contain software from multiple suppliers, even from a cascade of suppliers. This extends the classic relationship between the OEM as customer and Tier-1 as supplier in several aspects. On the one hand, a supplier has to assume the perspective of an OEM, when he integrates software from a Tier-2 supplier into his own component. On the other hand, an OEM also has to assume the supplier perspective, when he provides software that a supplier integrates into a software package or into hardware. For the software release, there is an important difference whether the integration of external software is ordered by the customer or if it is a free decision of the supplier. The responsibility for releasing such external software parts should be clarified between customer and supplier before the first delivery.

In a cascading sequence of software suppliers, ideally the software requirements related to quality, maturity, development processes, timing of delivery should be forwarded to each supplier on each level. In practice, this forwarding is limited. For example:

- **Commercial-off-the-shelf (COTS) software components, e. g. AUTOSAR basic software**
  For the most part, COTS software has to be integrated as is. Quality assessments at the supplier may not be possible. Software changes can be difficult with regard to content, timing, or even in general. Desired release documentation and artefacts (cf. chapter 3) may not be provided by the supplier in a format and with the content that can easily be integrated into the overall documentation of the Tier-1 supplier. In order to address these uncertainties and to mitigate corresponding risks the integrator of such software has to transform the information of the COTS provider or even has to add appropriate quality assurance measures.
- **Open source software**
  Although for open source software the source code is fully transparent to an integrator, the implications are quite similar and even more likely than with COTS software. Open source software is often maintained by a community, so that the availability of any needed updates is not assured. Even a reliable issue reporting is often not guaranteed. Additionally, an assessment of the development process of an open source component usually is not feasible. Such implications should be clarified with the customer, even if the customer requested the usage of this open source component and even if it is the OEM. A very important issue is the license type of open source software. If it is a strong copyleft license (GPL) for example, all code has to be shared. Due to this reason, open source license information is an important part of the release notes.
- **Proprietary software (e. g. functional software from OEM)**
  Such software often contains innovative functions and thus is linked to specific intellectual property. To protect this, an integrator does not often have transparent insight into the source code, e. g. if he is requested to integrate pure object code. In that case, the capability of the integrator and consequently his responsibility is limited to the integration purpose. Depending on the scope of the integrated software and its functionalities, the functional responsibility stays with the supplier of the component. The distribution of these responsibilities should be clearly defined in the contractual relationship between customer and supplier. In practice, this is particularly important for an integrator if an OEM assumes the additional role of a software supplier, beneath the role of the top-level awarding authority (customer).

2.5. Impact of configurable software

Configurable software is a key principle to realise ever growing software content with reasonable effort and quality or to enable reuse. But configuration of software brings along some drawbacks that have to be addressed in the context of releasing and providing software.

One example is AUTOSAR basic software with thousands of parameters including configuration parameters that significantly influence the behaviour of the software. A similar case is a platform software of a supplier, which is developed to be used in many projects for different OEMs.

Differentiate:

- **Configuration by supplier**
  Part of the configuration is done by the supplier before delivering the software. This restricts the configuration freedom for the integrator/customer – we could call it "pre-configuration".
- **Configuration by integrator**
  Part of the configuration is done by the integrator/customer (OEM who integrates an ECU into a vehicle variant or Tier-1 who integrates software components into an ECU).

Configuration leads to functional variants and the main challenge is to adequately test the variants, because testing of all possible variants may not be possible with a reasonable effort. Different strategies can be used to meet this challenge that are not discussed here. Regarding software releases at the interface of different organizations, maximal transparency is the major goal. This will be dealt with in chapter 3.

3. Documentation and Artefacts

This chapter provides a description of supporting documents accompanying the delivery of a release item.

3.1. Overview document

Practice shows that an overview document provides the best access to release documentation for the customer. The following aspects are summarised in this document:

- A short description of the release item, purpose of use (construction phase, test run, intermediate release ...). Using a clearly structured nomenclature in the release designation enables the distinction between main releases and bug fix releases. In addition it is useful to state in the nomenclature, whether the software has interface compatibility with its predecessor or not.
- Overview of the documents delivered, overview of the documentation.
- Project schedule with reference to the current phase in the project (A, B, C sample)
- Short description of the agreed standard timeline or the process model for a release (see figure 9).

3.2. Detailed release documentation

3.2.1. Delivery scope

This document supplies a comprehensive overview of the software components or the control units. This includes an overview of variants e. g. in the form of a matrix, which contains all the necessary information for the integration of the component. The 3rd party software components contained in this delivery also have to be listed, independent of whether they are OEM standard software, open source software, or any kind of proprietary software.

The information concerning software version, configuration files, flash bootloader, memory data (RAM/ROM), interface description, HW etc. denoting a software release item must be stated clearly and in detail here. If required, parameter sets and diagnosis data inputs are also described.

Furthermore it is reasonable to describe the exact data of the "build environment" used, for instance compiler versions. In Table 1, the matrix description of the B1 sample stage of a control unit is given as an example. Similar presentations can also be used for pure software deliveries. In particular with 3rd party software, it is important to also mention the license (see Table 1).

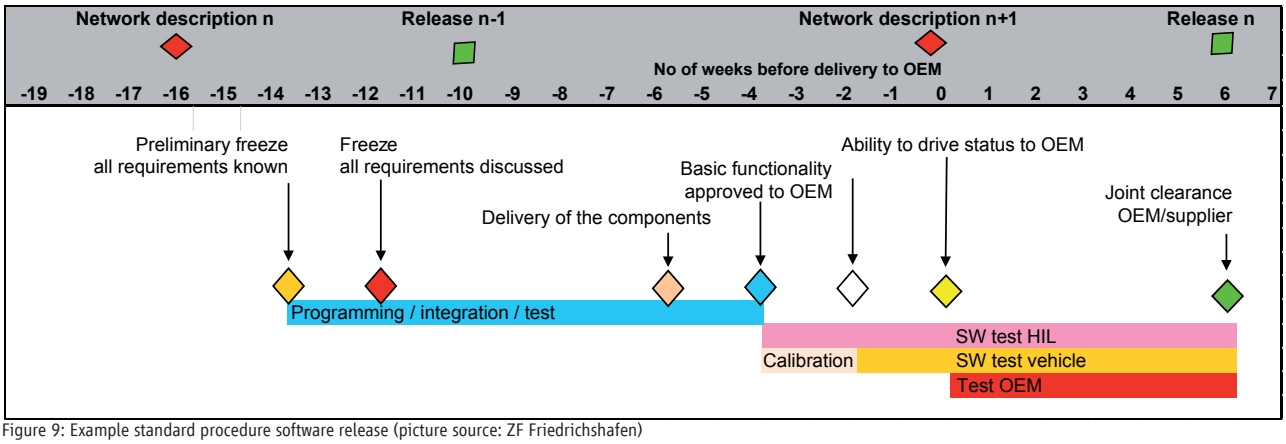


Figure 9: Example standard procedure software release (picture source: ZF Friedrichshafen)

Sample status	Variant	Tier-1 Part no.	OEM LU no. (delivery scope)	OEM ZB no. (assembly)
B1	1. Flexray single	**XY**		
	2. Flexray x4	**XY**		
	3. Flexray x8	**XY**		
	Identification			
	SW	**unique identification for delivered software version (e. g. software part number)**		
	HW	**unique identification for delivered E/E hardware version (e. g. E/E hardware part number)**		
	MECH	**unique identification for delivered mechanic hardware version (e. g. number and index of drawing)**		
	DBC file OEM	**...**		
	ECU file name	**...**		
	Standard SW package OEM	**...**		
	Flash bootloader – status	**...**		
	SW article code	**Extended SW article code**		
	...			

Table 1 : Example of release information for a control unit:

3.2.2. Subsystem description

In addition to the unique identification of the release item in the delivery scope, a clear reference to the subsystem or control unit level is added for a software release item. This is especially helpful for the customer if he wishes to check functionalities on a release item and wishes to have the corresponding framework conditions available quickly.

The following information is part of the subsystem description for example:

- Circuit diagram of the control unit
- Interface description of the control unit or subsystem
- Block wiring diagram
- Connector description

This important block of release documentation can frequently be transferred from one release to the next release. Information on the compatibility of the software release item to various hardware states must also be documented for changes in the system or control units HW.

3.2.3. Change log

The task of the change log is to give the customer an overview of the modifications of the release item. A central element is a listing of the software changes and bug fixes with reference to the previous release item. The changes will ideally be exported from the workflow management system in order to avoid consistency problems of software and documentation. It is sensible to reference the document “Delivery scope” in order to document the compatibility of hardware, software and tools clearly.

It is useful to make a distinction between new or modified requirements and bug fixes which are implemented. Tables with the following columns as categories are commonly used:

- Consecutive number
- Unique supplier change ID (reference to supplier's change management system)
- Headline of the change
- Change type (requirement or bug fix)
- Unique customer change ID (reference to requirements or problem management system of the customer)

3.2.4. Function list

The functions to be implemented for a release are specified during release planning and stated in the function list. They are referenced to the relevant requirements documents. Further important information is whether the planned function was implemented completely. If it was only implemented in part, a statement of the resulting limitations is added. If not all the variants that can be activated by configuration parameters are implemented in a specific release, this is also documented.

The trend towards ever finer, more granular reporting, is a challenge which can extend to the level of individual requirements under certain circumstances. An agreement with the customer about a suitable depth or an average granularity minimises time and effort. A reference to superordinate functions or function groups may thus be appropriate. The term "Feature" is used for this at many points.

A requirement specification of good quality and stability, supports the creation of an informative function list. In case of a poor or volatile requirement specification quality, the importance of the function list for obtaining an overview of the functionality increases.

Bug fixes are already shown in the change log. The function list additionally shows faults which are known but which have not yet been fixed ("known issues"). This increases transparency and reinforces confidence.

It also makes sense to create metrics for the functional extensions between the releases. How does the number of requirements develop over the project? Are the agreed functional extensions achieved? How great are the deviations?

Figure 10 shows the degree of implementation and fulfilment assumed for a notional project progression. Here a distinction is made between requirements, which were planned and implemented for the release, and requirements which were planned but could not be implemented.

3.2.5. Configuration parameters

Configuration parameters can either be parameter set at build time in a makefile configuration or a post build time calibration parameter set. A list of configuration parameters that are intended to be used by the integrator is documented. A detailed description of the effect on the functionality is not part of the release description, but rather of the detailed technical specification. A list of the changes in comparison to former releases increases readability.

3.2.6. Verification results

The purpose of this document is to make the verification results of the supplier available to the customer in a suitable way. The customer and supplier agree which test methods and test end criteria are to be used in the project right at the start of the project.

It is necessary to agree to a suitable abstraction level for the verification outcomes. This minimises time and expense for the customer and the supplier and safeguards know-how. Normally it is sufficient to report the test coverage with reference to the customer requirements (see also section 3.2.4). The detailed test results are only then made available, if this is agreed on contractually. A good compromise often involves making it possible to view detailed results without these being passed on.

In case of configurable software, the supplier states which sets of configuration parameters have been verified in the current release item. If the test level is different over the range of variants this is documented (e. g. full test for the standard variant and only test of standard behaviour – "Geradeauslauf" – for other variants). It could be useful to add information about those sets of configuration parameters

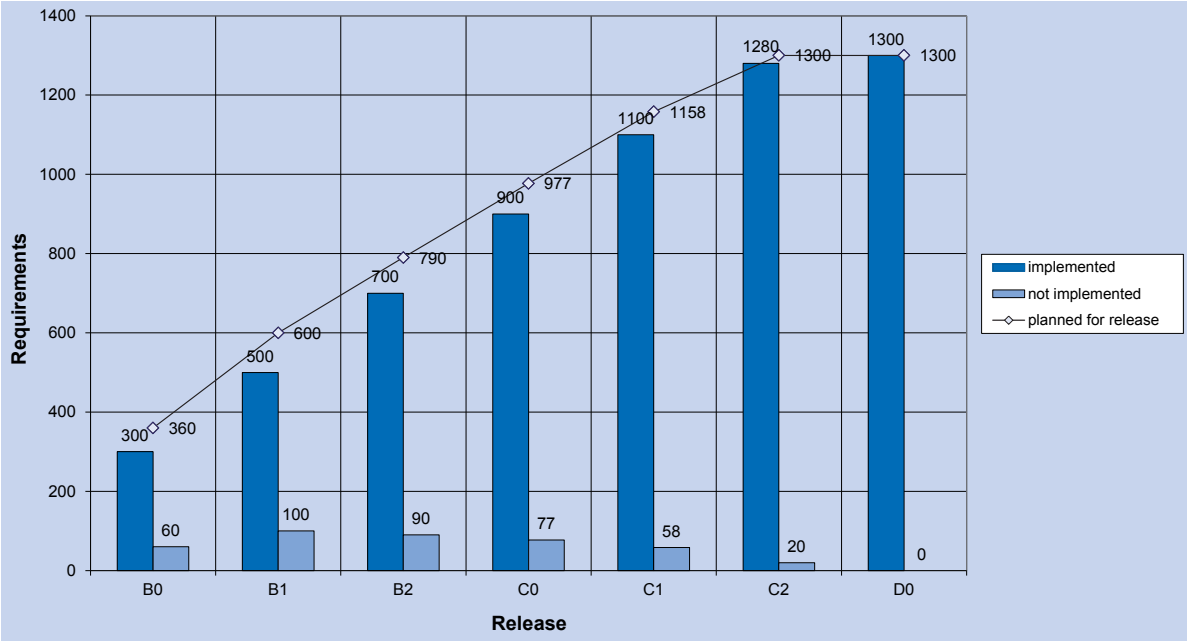


Figure 10: Implementation status requirements specifications (picture source: Leopold Kostal)

that have been tested in former release items. For the software release, the expectations on the verification test of configurable software depend on the configuration time point. If the software is configured by the supplier, complete test coverage for this configuration is expected by the OEM. If the software is configured during runtime by calibration values, the verification has to be performed by the integrator with this special dataset.

Notes:

- The defined document scope can deviate, depending on the project phase. In early phases the document scope may possibly be incomplete or adapted.
- In the event of several deliveries and recursion loops for a release item, it may be beneficial to carry out a delta analysis.

3.2.7. Metrics

Metrics assist in the all-round evaluation of a software release item. This makes them an element of quality management. In many cases it makes sense to document the same metrics over the project progression so that trend statements can be derived from them. Therefore, the careful definition of the metrics at the start of the project is important. Any later modification may require recalculations and in any case, make statements on the long-term trend more difficult.

The following metrics are commonly used:

- Number of the function modifications implemented ("functional extensions")
- Number of bug fixes
- Number of faults which have not been fixed or open points ("known issues")
- Test coverage with reference to the requirements
- Test coverage with reference to the code created (e. g. "function coverage" or "code coverage")
- Test coverage with reference to configuration parameters (e. g. coverage of functional variants)
- Metrics for evaluation of the product quality (e. g. MISRA or HIS)
- Maturity Index
  The maturity index is a very useful tool for determining the product quality or product maturity during product development. It takes problems and modification wishes into consideration depending on their degree of difficulty and processing status. Details about this can be found in the glossary.
- Resource usage (with regard to RAM, ROM and runtime, see figure 11)

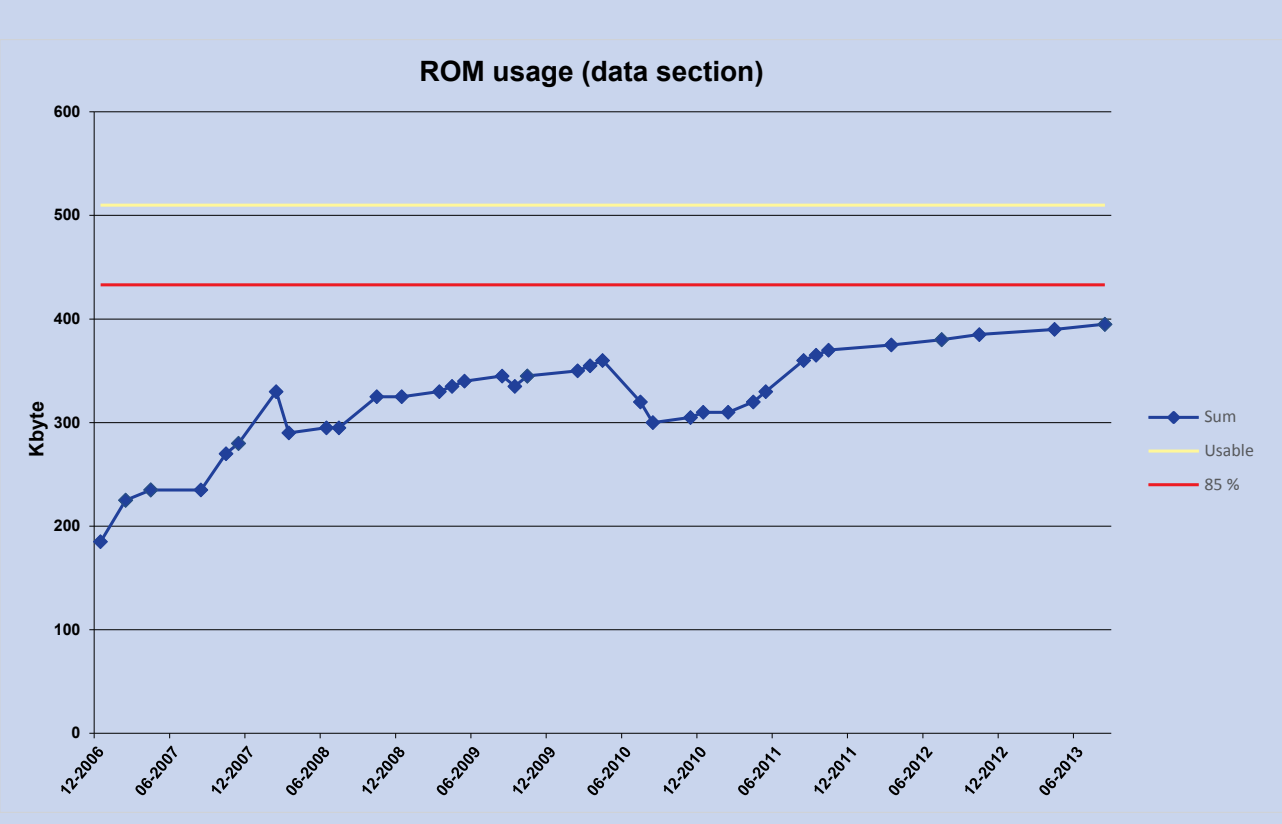


Figure 11: Example of resources consumption actual/target (picture source: ZF Friedrichshafen/ZVEI)

3.2.8. Releasing Software

With this document, the release process of the involved development departments is summarised in a multi-stage process (see figure 12).

Releasing a software release item for a defined use is declared by the authorized persons. This is based on the technical release recommendations of the functions involved (e. g. software development, test, quality assurance, safety management) and also has to reflect the 3rd party software contained.

Different release levels can be issued depending on the project phases. Examples of this include:

- Releasing software for testing in the vehicle in a closed testing area for specially permitted drivers in prototype testing
- Releasing software for testing in the vehicle on public roads for a restricted group of people who may drive the vehicle
- Releasing software for unrestricted use of the vehicle on public roads

In the case that the release level varies for different variants (due to different configurations of the software), this difference is documented. It could be useful to apply a characterisation like "Released for public roads with restrictions" supplemented by information denoting the restrictions in terms of variants that do not fulfil the release level "public road" (e. g. a specific functional variant is supported by the software but not thoroughly tested – therefore the customer must not use this variant on public roads).

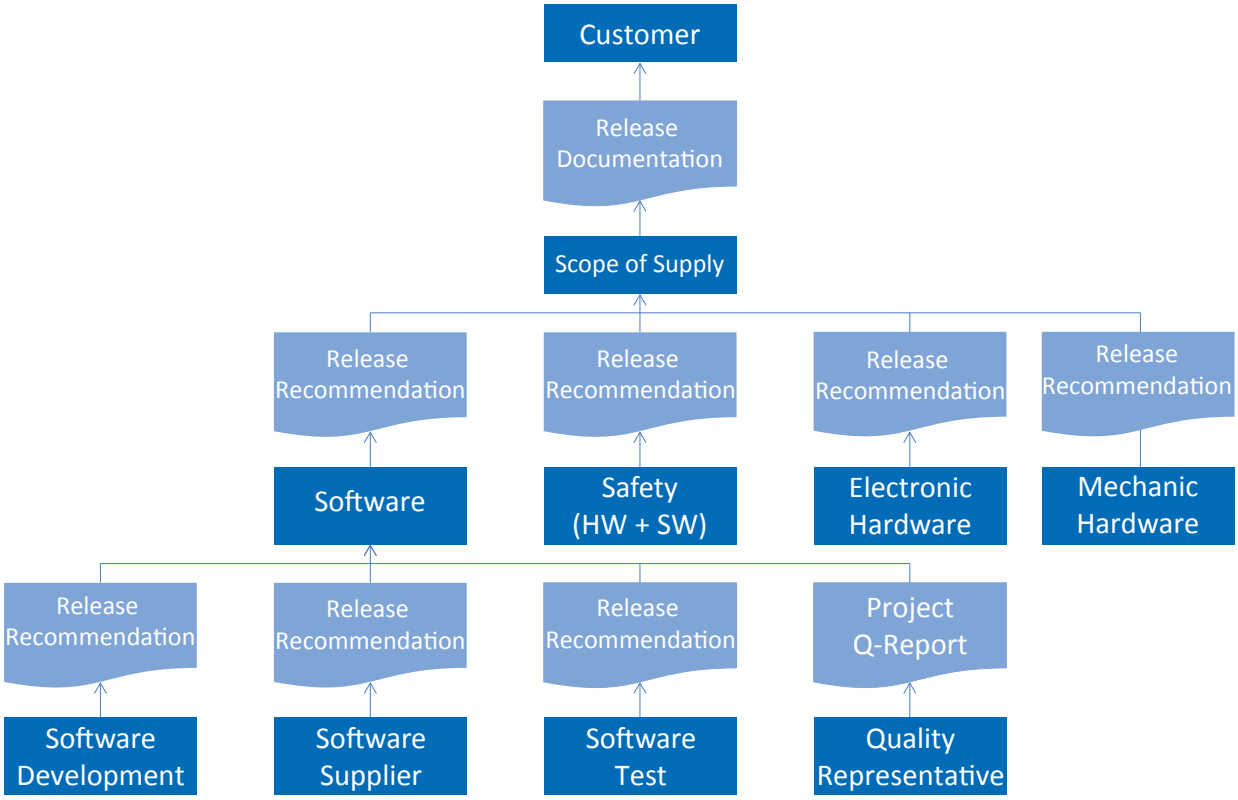


Figure 12: Example of a multi-stage release process (picture source: ZF Friedrichshafen)

3.3. Proposal for basic release documentation

In most cases, the release documentation from supplier to customer looks different from company to company, and also differs sometimes for different projects in the same company. But most release documents cover the same basic information, just in different ways and with some additional project-specific information added to this basis.

This chapter proposes which basic information may always be present in the release documentation to the customer. The content can differ according to the kind of customer. For an OEM, other information is important than for a Tier-1 (see Appendix A).

Examples for the basic content of a release documentation are shown in Appendix B and C. Aside from the proposal for the basic content, the release document can be supplemented by the additional information mentioned in the chapters 3.1 and 3.2 or additional information requested by the specific customer.

4. Principles for Use in Practice

Two central challenges need to be addressed for an optimal implementation of the software release process. These two challenges are maintaining the communication between the partners during the release process and creating informational transparency.
A close cooperation between the OEMs and their partners on the supplier and service provider side can help to ensure that:

- Responsibilities in the release process are discussed and documented at the beginning of the project.
- A high degree of transparency regarding the status of the software forms the basis for seamless cooperation and trust between all partners.
- Stability is maintained in the processes, which is essential during critical project phases in order to avoid frictional losses.

Consistency and continuity of the information are of importance to the level of transparency aimed for. They form the basis of efficient communication. Today the exchange formats for the documentation of software release items between customer and supplier are different for each customer. There is a need for standardisation of the interfaces for change management and problem management. Also, a simple alignment of toolchains supports the consistency and continuity of information and avoids misunderstandings, adaptation efforts, and integration issues caused by incompatibility (e. g. dbc-files, arxml-files, …).

Transparency ensures that customer and supplier have a common understanding about requirements and expectations for each software release item. This can be ensured through project and release kick offs in which expectations and scope are clarified jointly. Clarification also occurs through a transparent presentation of relevant information in release planning and documentation. Here, it is important that a suitable abstraction level is defined jointly which is not too fine granular. An excess of detailed information does not result in higher transparency. The degree of fulfilment of requirements is reported in the release documentation. A reporting not based on a single source database can cause inconsistency. A good overview of what this software version is and is not capable of, is a decisive factor.

Initial process quality evaluations (e. g. SPICE assessment) can be updated during the project if required. They can contribute to building up a basis of trust between the customer and the supplier. This can be more helpful than reporting in the project at a very detailed level. Requirements on the maturity of development processes are exchanged between customer und suppliers on each level. In practice this forwarding may be limited when handling COTS software, open source software or proprietary software with restricted insight. Such limitations are clarified between all involved parties – from the OEM to any Tier-2 supplier.

Punctual planning and coordination of the release contents is the basis for a high-quality delivery which is on time. Unplanned modifications can easily lead to a delay in the project. The number of software release items to be delivered is defined in the project planning and adhered to as far as possible. Action for action's sake through "daily" releases, only affects the stability of the process. Daily software deliveries within the framework of agile development in particular at the component level can be extremely advisable. Nightly builds by means of continuous integration are an example of this. They are not software releases in the sense of this guideline because fewer formal requirements have to be met.

Late modifications on the basis of customer decisions must be evaluated jointly. The balance between adherence to schedules, quality and modification requirements of the customer can only be optimised successfully in regular and open communication. Agreed metrics and a joint evaluation are an important basis for this. In the planning of the release it must be ensured that test results and thus corrections can flow into subsequent releases.

The definition of different release levels and thus test scopes per release and even per functional variant (implemented by software configuration) can be documented in the release planning equally. The distinction between bug

fix on a side branch and further development on the main branch is helpful. Bug fixes which cannot be integrated in the main branch are usually restricted to the most necessary.

The passing on of a software release to the customer may be accompanied by a release review. The development status achieved for the requirements and expectations of the customer are reflected here. A joint understanding is created, regarding the purposes for which the release item can be used. An evaluation of the development phase since the corresponding release kick off ("lessons learned") is logically an additional component of a release review.

5. Definitions and Terms

Bug fix
Rectification of a fault.

Calibration parameters
Parameters that configure the software post build time.

Coarse calibration release
Calibration parameter set providing a basic functionality.

Configuration parameters
Parameters that configure the software at build time.

Feature
Superordinate functionality or set of functionalities. See also functional extension.

Fine calibration release
Calibration parameter set providing a specified functionality completely.

Functional extension
Planned implementation of additional functionality.

Known issue
Not rectified known fault.

Maturity index (Product/Release maturity)
The aim of the maturity index is to summarise the product maturity in a classification number. Here the "issues" are weighted according to their severity and their processing status (see table 2). All release relevant faults, wishes for modifications and feature implementations are treated equally as "issues". The maturity index is ultimately made up of the sum of the weighted issues and is logically plotted as a graph over time (see figure 13).

Release
Statement about the implemented functions, properties and intended use for a release item.

Release item
Unambiguously identifiable element with stated functions, properties and purpose.

Severity status	Show stopper	Major	Minor
New or analyzing	40	20	10
Open or implementing	20	10	5
Testing	10	5	1
Closed or rejected	0	0	0

Table 2: Weighting factors for establishing the maturity index

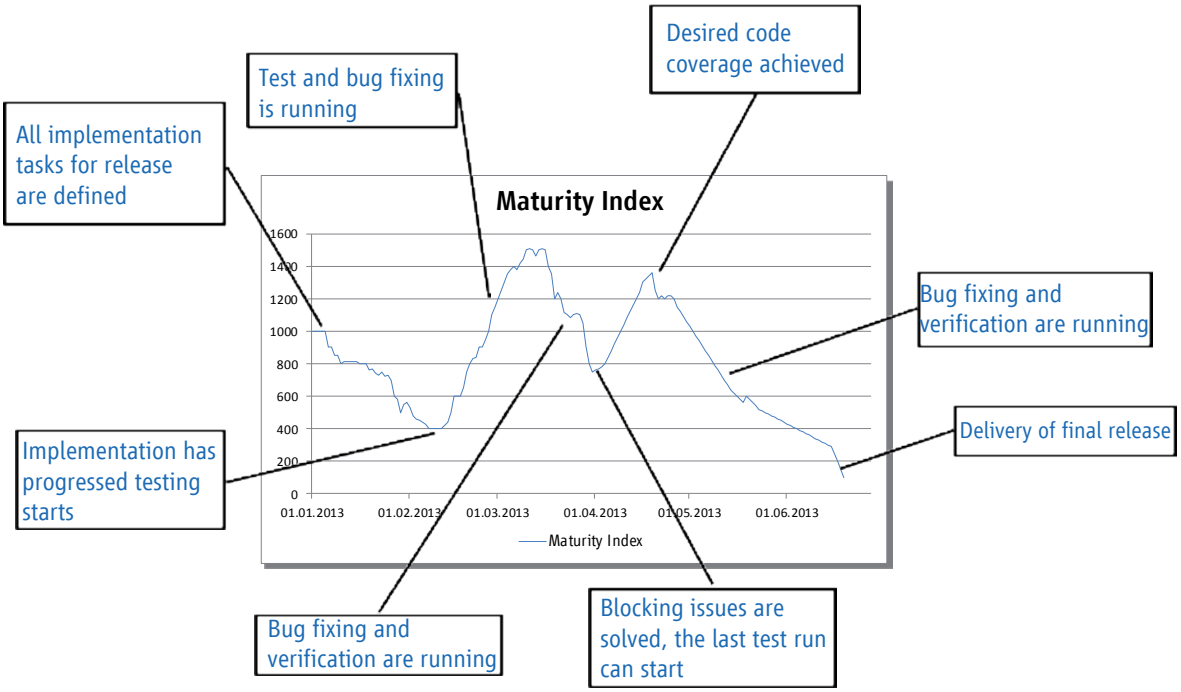


Figure 13: Typical process of the maturity index (picture source: TTTech Computertechnik)

Release kick off

A release kick off is a meeting attended by the customer and supplier. Participants are decision makers, project managers and technical experts.

In the meeting, the customer presents:

- release objectives
- acceptance criteria

and the supplier presents:

- Planned validation measures and a detailed time schedule where necessary.

The aim of the release kick off is balancing the expectations of both sides and the definition of rules regarding communication, processes, escalation etc.

Release plan

Description of the content and timing of the releases and the release items.

Release review

Activity to determine the quality of a software release item with respect to the intended functions, properties and purpose.

Standard timeline

Schedule overview of the planned and coordinated project phases and milestones with the OEM. The expected outcomes are in time relation to the delivery for this integrated software release.

6. Participating Companies

Participating companies in the "Software Release" working group:

Automotive Lighting Reutlingen
Brose Fahrzeugteile
Continental
Lear Corporation
Leopold Kostal
Marquardt
NXP Semiconductors Germany
OptE GP Consulting
Robert Bosch
Schaeffler Technologies
Vector Informatik
Webasto
ZF Friedrichshafen

Appendix A

Proposal for software release note content			
Content	**Proposal to Tier-1**	**Proposal to OEM**	**Chapter**
Document Number (Unique ID)	r	r	3.1
Release Date	r	r	3.1
Contact Person	r	r	3.1
Software Release	r	r	3.1
Software Version	r	r	3.1
Customer Version	r	r	3.1
Release Result (Released / with Restriction)	r	r	3.1
Cause of Restriction	r	r	3.1
Known Issues	r	r	3.1
Revision History	o	o	3.1
Purpose of Release	r	r	3.1
Referenced Documents	r	r	3.1
Remarks	r	o	3.1
Delivery Content / Material List	r	r	3.2.1
Used External Module Names	o	o	3.2.1
Used External Module Version	o	o	3.2.1
Known Issues from Used External Modules	r	r	3.2.1
Interface Compatibility	r	o	3.2.1
Used Compiler with Version	r	o	3.2.1
Information about Build Environment	r	o	3.2.1
Open Source Software Information	r	r	3.2.1
Implemented Functions / Change List for this Release	r	r	3.2.3
Implementation Status (See figure 10)	o	o	3.2.4
List of Open Change Requests	r	r	3.2.4
RAM Usage in Percent	r	r	3.2.7
ROM Usage in Percent	r	r	3.2.7
CPU Load in Percent (Worst Case)	r	r	3.2.7
Resource Consumption Metric (See figure 11)	o	o	3.2.7

Caption	r = recommended		
	o = optional		

Appendix B

Example for Chapter 3.1 Overview Document

Document Number	123456 R2.0	**Release Date**	Date
Software Release	X-Sample	**Contact Person**	John Doe
Software Version	vX.X	**Customer Version**	vX.Y
Release Result	Released / Released with restriction		
Cause of Restriction			
...			
Known Issues			
...			

Revision History

Revision	Date	Description
R1.0	Date	Information about revision
R1.1	Date	Information about revision
Rx.x	Date	Information about revision

Document Purpose
Explanation of this document and the intention of this software release.
Release Purpose
Purpose of the release (for construction phase, test run, ...)

Reference Documents

Number	Name /Short Description	Version	Date
1	Document name	vX.X	Date
2			
3			
...	...	...	...

Appendix C

Example for chapter 3.2.3 Change Log and chapter 3.2.4 Function List

Implemented Functions / List of Changes		
Number	**Task / Change ID**	**Short Description of Task**
1	Unique ID	...
2		...
3		...
...		
Software Release Plan		
Reference to the release plan in the referred documents		
List of Open Change Requests		
Number	**Change ID**	**Short Description of Change Request**
1	Unique ID	...
2		...
3		...
...		...
Status of the Requirements and Implementation		
See figure 10 “Implementation status requirements specification“		

Note

ZVEI
German Electrical and Electronic
Manufacturers' Association
Lyoner Strasse 9
60528 Frankfurt am Main, Germany

Phone: +49 69 6302-0
Fax: +49 69 6302-317
E-mail: zvei@zvei.org
www.zvei.org

Sources: Continental Teves, ESG Elektroniksystem- und Logistik, NXP Semiconductors Germany, TTTech Computertechnik, Vector Informatik, ZF Friedrichshafen