Mit Software-Archäologie auf dem Weg zu A-SPiCE® Level 3

Umgang mit gewachsener Software

Kosmas Kopmeier, Christian Steinmann; Synspace Group

Die Prozesse der meisten Unternehmen beschreiben den Entwicklungsablauf eines Produkts oder Software auf der grünen Wiese beginnend. Oft gibt es aber auch Projekte, bei denen der Prototyp, oder fast schon voll funktionsfähiger Code, schon lange, teilweise seit Jahren, vorhanden ist. Dieser ist aber nicht nach den geordneten und dokumentierten Abläufen entstanden, die für ein Serienprojekt gefordert sind. Diesen Fall nennen wir hier Software-Archäologie. Der reguläre Ablauf, dass erst Anforderungen niedergeschrieben werden und dann ein Architektur-Entwurf erstellt wird, bevor dieser dann in Code runtergeschrieben wird, funktioniert nicht, da der Code z.B. nicht dem schönen, neuen Layer-Model entspricht, das der neue Software-Architekt entwickelt hat.

In der Automotive Branche wird üblicherweise ein Automotive SPiCE® Level 2 oder Level 3 gefordert. Das heißt unter anderem, dass umfängliche Entwicklungsnebenprodukte erstellt werden müssen, die zueinander auch konsistent sein müssen. Wenn der Prototyp nicht verworfen werden, sondern als Basis verwendet werden soll, passen die normalen Entwicklungsabläufe nicht, zumindest nicht so ganz.

In dieser Präsentation wird erörtert, wie die Prozessabläufe so angepasst werden können, dass das Qualitätsziel Automotive SPiCE® Level 3 erreicht werden kann.

Grundlagen Automotive SPiCE® Level 1-3

Automotive SPiCE® ist ein Modell zur Überprüfung der Prozessreife eines Projekts. Es definiert also nicht selbst den Prozess, sondern ist vereinfacht gesagt eine Sammlung von idealisierten Handlungsmustern, sogenannten „*Base Practises*“ und „*Generic Practises*“, die als Fragenkatalog verwendet werden können, um zu prüfen, ob ein Entwicklungsprozess bestimmten Qualitätsanforderungen entspricht.

Bei einem Assessment beschränkt man sich üblicherweise auf einen bestimmten Teil der Prozesse, der vom Verband der Automobilhersteller als VDA-Scope definiert wurde.

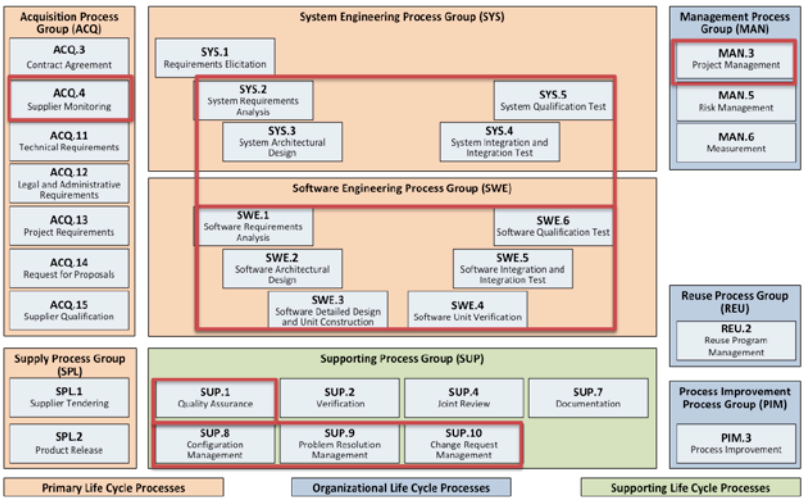

Abb. 1: Prozessgebiete nach Automotive SPiCE® mit VDA-Scope

Automotive SPiCE® definiert dabei die Level 0-5 der Prozessreife und dazugehörige Prozessattribute, die jeweils eine Bewertung erhalten.

Liste der Level und zugehörigen Prozessattribute bis Level 3:

Level 0: Incomplete Process
Level 1: Performed Process
PA1.1 Process Performance Process Attribute
Level 2: Managed Process
PA2.1 Performance Management Process Attribute
PA2.2 Work Product Management Process Attribute
Level 3: Established Process
PA3.1 Process Definition Process Attribute
PA3.2 Process Deployment Process Attribute

Ein Level wird erreicht, wenn die zugehörigen PA mit F oder L und alle PA der Level darunter mit F bewertet sind.

Prozess	PA1.1	PA2.1	PA2.2	PA3.1	PA3.2		Level
MAN.3	F	L	F	L	P		2
ACQ.4	L	L	P	L	P		1
SWE.1	F	F	F	L	L		3
SWE.2	L	F	F	F	F		1
SWE.3	F	L	F	P	P		2
SWE.4	N	N	P	L	N		0
SWE.5	P	P	L	F	P		0
SWE.6	L	L	P	L	P		1
SUP.1	L	L	L	L	L		1
SUP.8	F	F	L	F	F		2
SUP.9	L	F	L	F	L		1
SUP.10	F	F	F	F	L		3

Legende:

F	fully achieved
L	largely achieved
P	partially achieved
N	not achieved

Level:

3	Established Process
2	Managed Process
1	Performed Process
0	Incomplete Process

Abb. 2: Typisches Beispiel einer Prozessbewertung

Entwicklungsablauf auf der grünen Wiese

Wenn ein neues Produkt, eine neue Software entwickelt wird – so stellt sich der Top-Manager oder Kunde das zumindest immer vor - wird zuerst überlegt, was die Software „können“ soll. Es werden Anforderungen aufgeschrieben. Die Anforderungen werden in Features eingeteilt, die anschließend den verschiedenen geplanten Releases zugeordnet werden können.

Danach erstellt man eine Architektur, z.B. mehrere Layer, und legt grundlegende Konzepte fest, z.B. preemptives oder nicht preemptives Betriebssystem, Zeitscheiben- oder eventgetriebene Steuerung etc.
Es wird entsprechend der Festlegungen der Architektur ein Design z.B. in UML oder Matlab erstellt und der Code daraus entwickelt.
Im Anschluss wird alles getestet, vom Unit-Test über einen Schnittstellentest, in dem das Design verifiziert wird, einem Integrationstest, in dem die Einhaltung der Konzepte der Softwarearchitektur überprüft werden, bis hin zum Test gegen die Anforderungen.

Gerahmt wird der gesamte Entwicklungsablauf zum einen von einem Projektmanagement, der, zumindest, wenn man einen A-SPiCE® Level 3 erreichen will, nicht nur die Planung aller Aufgaben beinhaltet, sondern auch die Aufwandsabschätzung, deren Überwachung und eine geeignete Reaktion auf Planabweichungen.
Zum anderen wird der Entwicklungsablauf von einer Qualitätssicherung gerahmt, in der die Qualitätsziele aller Arbeitsprodukte und die Prozessabläufe definiert sind, Reviews geplant und durchgeführt werden und Abweichungen bis zur Abstellmaßnahme verfolgt werden.

Selbstverständlich sind alle Arbeitsprodukte unter Versionskontrolle und für jedes Release werden die vorgesehenen Baselines in den verschiedenen Konfigurationsmanagement-Tools erstellt.

In der Entwicklung von Serienprodukten werden typischerweise mehrere Prototypen – in der Automobilwelt nennt man sie A-Muster, B-Muster, C-Muster – in iterativen Schleifen entwickelt. Dabei werden alle Schritte des Entwicklungsablaufs immer wieder neu durchlaufen und die Arbeitsprodukte und natürlich auch das Produkt immer weiter verbessert. Dieser iterative Ablauf wird durch einen Problemlösungs- und Änderungsmanagement-Prozess gesteuert.

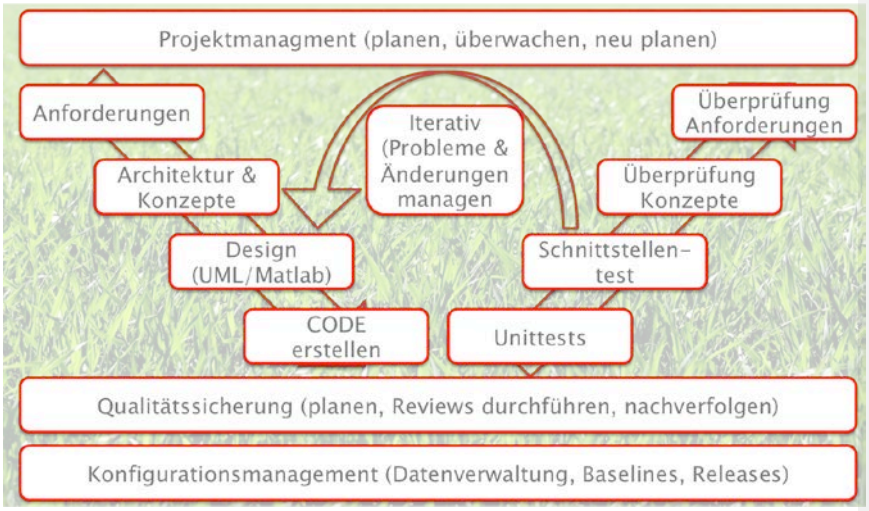

Abb. 3: V-Modell auf der grünen Wiese

Entwicklungsablauf bei Software-Archäologie
Es gibt aber auch Entwicklungsprojekte, die nicht bei Null anfangen, sondern solche, bei denen der Code schon in großen Teilen existiert, jedoch – mehr oder weniger – keine schriftlich dokumentierten Anforderungen, Architektur-Festlegungen, UML-Designs oder ähnliches existieren.

Der existierende Code wird dann z.B. mit Hilfe von Doxygen zunächst in ein für den Menschen besser lesbares Design verwandelt. Oft sieht man einige erste Optimierungen, die durch ein *refacturing* des Codes ohne inhaltliche Änderung gemacht werden können.
Erheblich schwieriger ist die Erstellung einer Software Architektur, wenn die Software nicht ursprünglich nach einem klaren schriftlichen Model mit verschiedenen Layern entwickelt wurde, dann entspricht sie nämlich meistens auch keiner noch so schönen nachträglich ausgedachten Struktur. Wichtig ist es trotzdem eine solche Soll-Struktur zu definieren, damit man sich in zukünftigen Iterationen dieser Struktur immer weiter nähern kann.

Etwas leichter, aber auch herausfordernd ist es die Anforderungen, denen die Software bereits genügt, nachträglich zu ermitteln und diese anschließend aufzuschreiben.

Im Folgenden wird – wie beim Entwicklungsablauf auf der grünen Wiese auch – alles getestet, vom Unit-Test über einen Schnittstellentest, in dem das Design verifiziert wird. Diese Tests sind als Regressionstest vor allem wichtig, damit bei dem *refactoring* des Codes keine versehentlichen Fehler passieren.

Weiter gibt es auch hier den Integrationstest, in dem die Einhaltung der Konzepte der Software-Architektur überprüft wird und den Test gegen die Anforderungen.
Selbstverständlich gibt es auch hier wieder ein Projektmanagement, auch wenn die Planung nicht nur die Umsetzung neuer Features sondern auch die Optimierung des vorhanden Codes beinhaltet, muss auch hier alles geplant werden. Die Aufwände müssen geplant und überwacht werden und bei Planabweichungen, die hier deutlich häufiger auftreten können muss entsprechend neu geplant und alle Pläne aktualisiert werden. Zumeist gibt es klar definierte Meilensteine, zu denen ein Release für den Kunden erstellt werden muss, so dass hier genau überlegt werden muss, wie viel Aufwand in die Optimierung des vorhanden Codes investiert wird und wie viel in die Entwicklung von neuen Features.

Auch hier wird der Entwicklungsprozess von einer Qualitätssicherung gerahmt. Wenn man einen kontinuierlichen Integrations- und Testprozess hat, kann man mit den richtigen Tools unterschiedliche Qualitätsvorgaben machen: eine für vorhanden Code und eine für neue Features. Dieses Vorgehen wird in den folgenden Kapiteln und Folien im Detail beleuchtet. In jedem Fall ist die Definition der Qualität der einzelnen Arbeitsprodukte, je Iteration, hier deutlich schwieriger, als bei einer Neuentwicklung. Nichtsdestotrotz müssen, um einen A-SPiCE® Level 3 zu erreichen, die Qualitätsziele der einzelnen Arbeitsprodukte geplant, überwacht und Abweichungen nachverfolgt werden.
Selbstverständlich sind auch bei diesem Ablauf alle Arbeitsprodukte unter Versionskontrolle und für jedes Release werden die vorgesehenen Baselines in den verschiedenen Konfigurationsmanagement-Tools erstellt.

Iterationen bei Neuentwicklungen
Bei der Neuentwicklung gilt es für die kommenden Releases neue Features und Bugfixes umzusetzen. Beides kommt in der Regel mit neuen Anforderung einher. Die Architekturkonzepte bleiben bestenfalls unverändert oder werden leicht erweitert und dann werden natürlich das Design und der Code erweitert. Die Tests werden gleichzeitig mit der Entwicklung erweitert.
Alles wächst gleich schnell an.

Iterationen bei Software-Archäologie
Bei der Software-Archäologie muss zunächst noch die gewünschte Testtiefe erreicht werden und die Coverage der Unit-Tests erhöht werden. Dann gilt es auch die Anzahl der Abweichungen zwischen dem Software-Architektur-Bild und dem Code zu verringern, in dem beide Seiten einander weiter angeglichen werden. Aufgrund des *refactoring* des Codes müssen auch die Schnittstellentests angepasst werden. Je besser das Produkt im Laufe der Projektlaufzeit verstanden wird, desto besser können dann auch die passenden Anforderungen aufgeschrieben werden, die dann im Test wiederum überprüft werden.
Die Anzahl der Unit-Tests, Anforderungen und Software-Test wächst an, während die Anzahl der Architekturelemente und *Lines of Code* konstant bleiben.

Regeln und Ziele bei Neuentwicklung

- Die Architektur basiert auf den nicht funktionalen Anforderungen an die Software.
  - ➔ Ziel ist die Einhaltung dieser Anforderungen sicher zu stellen.
- Design und Code entsprechen der Architektur.
  - ➔ Ziel ist eine gute Wartbarkeit.
- Die Anzahl der MISRA Warnungen (Verstöße gegen Codier-Richtlinie) werden reduziert.
  - ➔ Ziel ist eine Risikominimierung.
- Es gibt Unit-Tests für jede Funktion oder sogar jede Codezeile. Dies kann zeitlich parallel zum Entwicklungsaufwand gut realisiert werden.
- Es gibt Testfälle für jede Anforderung.
  - ➔ Ziel ist die Einhaltung der Anforderungen sicher zu stellen.

Regeln und Ziele bei Software-Archäologie

- Die Architektur soll den nicht funktionalen Anforderungen entsprechen. Dies dauert mehrere Iterationen.
- Eine Veränderung der externen Schnittstellen führt zu Inkompatibilität.
- Eine Veränderung des Verhaltens führt möglicherweise zu Inkompatibilität.
- Eine Anpassung des Codes an die Architektur führt zu einer besseren Wartbarkeit, ist aber auch eine Quelle für neue Fehler.
- Eine Anpassung der Architektur an den existierenden Code führt möglicherweise zu einer besseren Wartbarkeit aber auch zu einer unlesbaren Architektur.
- Die Behebung von MISRA Warnungen führt zwar zur Reduzierung des Risikos von ungewolltem Verhalten ist aber auch eine Quelle für neue Fehler.
- Die Erstellung von Unit-Tests für jede Funktion (100% *function coverage*) dauert nach Releaseplanung mitunter mehrere Iterationen.
- Testfälle für alle Anforderung zu haben, geht nur für bereits identifizierte Anforderungen.

Kontinuierliche Integration und kontinuierliches Testen

Um eine nachträglich erfundene UML-Architektur mit dem realen Code vergleichen zu können benötigt man eine Tool-Unterstützung. Eine gute Möglichkeit bietet die Bauhaus Suite von Axivion, die einen kontinuierlichen Integrations- und kontinuierlichen Testansatz verfolgt. Hier kann kontinuierlich überprüft werden, wie viele Verstöße zwischen Architektur und realem Code existieren, ob die Verstöße neu hinzugekommen sind oder schon in früheren Versionen da waren. Ähnliches gilt für Fehler, hier kann auf einfachste visuelle Weise unterschieden werden, ob eine MISRA Warnung neu hinzugekommen ist oder bereits in einer per Schieberegler einstellbaren früheren Version enthalten war.

Der Vorteil dieser Unterscheidung von alten und neuen Fehlern ist für die Qualitätssicherung sehr entscheidend. Hier können unterschiedliche Qualitätsvorgaben gemacht werden:

- Für Code der unberührt bleibt
- Für Code der verändert wird
- Für Code der neu geschrieben wird

Hierdurch wird Qualität deutlich planbarer, was sich positiv auf die Automotive SPiCE® Prozessattribute PA2.1 und PA2.2 auswirkt und damit auf SPiCE® Level 2 und 3.

Praxiserfahrung mit einem Software Archäologie Projekt auf dem Weg zu Automotive SPiCE® Level 3

Die Erfahrungen mit Software Archäologie basieren unter anderem auf einem konkreten Projekt im Automobilumfeld. Hier hat ein Lieferant den Auftrag erhalten, eine 8 Jahre lang gewachsene, auf einem Open Source Projekt basierende Software, die bereits im Serieneinsatz ist „SPiCE® Level 3 konform“ zu machen. Hierzu mussten Entwicklungsprozesse angepasst werden, neue Entwicklungstools eingeführt werden und vor allem der Planungsprozess sehr engmaschig gelebt werden. Das Entwicklungsteam wurde ein Jahr intensiv von uns, prozessseitig begleitet, bis ein SPiCE® Level 2 in allen Prozessen nachgewiesen werden konnte und ein weiteres Jahr später voraussichtlich auch der SPiCE® Level 3 in wirklich allen Prozessen nachgewiesen werden soll.

Weblinks

http://www.automotivespice.com/fileadmin/software-download/Automotive_SPICE_PAM_30.pdf
https://www.axivion.com/de/produkte-58
https://www.synspace.com

Autoren

Herr **Kosmas Kopmeier** ist Dipl.-Inform. (FH) und intacs™ zertifizierter Automotive SPiCE® Assessor.
Er arbeitet seit 2017 als Director Engineering Consulting für die SynSpace Group GmbH in Freiburg und ist als Berater für Prozessqualität und funktionale Sicherheit und als Automotive SPiCE® Assessor bei vielen Automobilzulieferern und Automobilherstellern, aber auch bei Herstellern von Industrieprodukten im Einsatz. Er hat über 20 Jahre eigene Erfahrung in der Entwicklung von Embedded Systemen und geht deshalb pragmatisch an alle Prozessthemen heran.

Kontakt

Internet: www.synspace.com
Email: kosmas.kopmeier@synspace.com
Tel.: +49 179 530 620 8

Herr **Christian Steinmann** ist Senior Consultant für Prozessverbesserung und seit 1995 Geschäftsführer des SynSpace-Sprösslings 'HM&S IT-Consulting GmbH' in Graz. Er ist Diplomingenieur der Technischen Mathematik, intacs™ zertifiziert, Scrum-Master und Inhaber eines technischen Ingenieurbüros. Er leitet die (Weiter-)Entwicklung des SPiCE 1-2-1 Assessment Tools. Im Banken- und im Automotive-Segment hat er mehr als 100 Assessments nach SPiCE bzw. CMMI und auch Trainings zu diesen Themen durchgeführt. Freitags spielt er die Rolle des Lehrbeauftragten für Informatik und SW-Entwicklung an der FH Joanneum in Graz für den Studiengang Fahrzeugtechnik und bringt den Studenten das Programmieren bei.

Kontakt
Internet: www.synspace.com
Email: christian.steinmann@synspace.com
Tel.: +43 699 10 696 100

Feldfunktion geändert