MESCONF 2017
KONGRESSLETTER

Eine Einführung
Das Manifest
Vorträge

Ausstellerbeiträge
Sieben auf einen Streich
200 Testfälle in 20 Minuten?
HIL and SIL Tests for Everybody

Visionen
das Buch zur MESCONF

MESCONF
Modeling of Embedded Systems Conference

MESCONF
Eine Einführung

Die MESCONF legt den Fokus auf den Nutzen der Modellierung in der Entwicklung eingebetteter Systeme. Unsere Welt wird immer komplexer und dynamischer. Der stetige Wandel ist einer der Herausforderungen für Unternehmen. Erfolgreich sind die Unternehmen, die flexibel und schnell mit qualitativ hochwertigen Produkten auf die wechselnden Anforderungen des Marktes reagieren können. Wir sind überzeugt, dass Modellierung einen entscheidenden Beitrag leistet, diese Fähigkeit zu erreichen.

Die MESCONF findet auf dem Campeon der Infineon Technologies AG statt. Infineon stellt ihre Räumlichkeiten und das Catering zur Verfügung, um die Konferenz zu ermöglichen. Ein modernes und inspirierendes Umfeld – ideal für einen erfahrungsreichen Austausch auf der MESCONF.

Manifest (lateinisch manifestus “ handgreiflich gemacht “) öffentliche Erklärung von Zielen und Absichten

Seit Januar 2015 diskutieren Spezialisten für eingebettete Systeme, wie durch modellgetriebene Vorgehensweisen schneller, sicherer, den funktionalen und nichtfunktionalen Anforderungen besser entsprechend, entwickelt werden kann. Die gewonnenen Erkenntnisse wurden auf der ersten MESCONF am 06.10.2015 dem Fachpublikum als Manifest der modellgetriebenen Entwicklung eingebetteter Systeme zur Diskussion vorgestellt.

Die klassische Softwareentwicklung erlebte innerhalb kurzer Zeit in den Jahren 1990 bis 1995 radikale Paradigmenwechsel. Diese Umbruchphase war gekennzeichnet durch die Verfügbarkeit von Systemen mit 1 Megabyte Programmspeicher und mehr, dem Wechsel von 8/16 Bit zu 32 Bit Architekturen, dem Übergang zu grafischen Benutzeroberflächen sowie einsetzender Vernetzung.

Parallel mit dieser Entwicklung haben sich die Paradigmen, Methoden, Techniken, Werkzeuge und Programmiersprachen der Softwareentwickler radikal verändert.

Eingebettete Systeme unterliegen derzeit einer enormen Entwicklungsdynamik. Es etablieren sich neue 32 Bit Standardarchitekturen. Die Systemgröße und die Leistungsfähigkeit nehmen rasant zu. Softwareentwickler für eingebettete Systeme sehen sich heute, wie die Anwendungsentwickler in den 90ern, mit entsprechend zunehmender Komplexität konfrontiert.

Inzwischen weiß man wie Systeme dieser Größe sicher und mit akzeptablem Aufwand entwickelt werden. Einen wertvollen Beitrag leistet die modellgetriebene Entwicklung.

http://mesconf.de

Manifest Modeling of Embedded Systems

For the first time published and discussed at the MESCONF 2015.

Theses:

INVOLVE STAKEHOLDERS
through domain-appropriate abstractions, notations and views

STRIVE FOR LONG-LIVED MODELS
through the right abstractions, separation of concerns and appropriate style guides

MAKE MODELS VALIDATABLE, TRANSFORMABLE AND EXECUTABLE
through semantically well-defined languages for functional and non-functional concerns

FRONT-LOADING
use models to validate or fail early in the development cycle

AVOID DUPLICATION AND REPETITIVE WORK
through automation and integration of models for different concerns

MAKE MODELING ACCESSIBLE
by providing a scalable infrastructure and user-friendly, learnable tools

ESTABLISH A MODELING CULTURE
through education, training and alignment with the development process

Wir lernen Sprachen, indem wir diese sprechen. Mit Sprachen die wir beherrschen können wir uns besonders gut ausdrücken. Das macht sogar Spaß. Sprachen, die man gern spricht benutzt man auch oft. Häufig genutzte Sprachen werden in der Anwendung Schritt für Schritt verfeinert und auf zunehmend höherem Niveau gesprochen. Das ist ein typischer, sich selbst verstärkender Erfolgszyklus. Diesen streben wir in Allem an was wir tun. Eine weitere Binsenweisheit ist, dass der Anfang, das Beginnen, die ersten unsicheren Schritte, oft die größte Hürde darstellen.

„Und jedem Anfang wohnt ein Zauber inne,
Der uns beschützt und der uns hilft, zu leben.
Wir sollen heiter Raum um Raum durchschreiten,
An keinem wie an einer Heimat hängen,
Der Weltgeist will nicht fesseln uns und engen,
Er will uns Stuf' um Stufe heben, weiten."
Hermann Hesse, Stufen

Wie gelingt es, Modellierung nicht als etwas Besonderes, als Ausnahme vom Täglichen, im Projektverlauf singuläre Aktivität, sondern als täglich wie selbstverständlich angewendete Sprache zu etablieren. Auch wenn es durchaus eine gute Idee ist, muss nicht unbedingt sofort ein Modellierungswerkzeug angeschafft werden. Denn dieses besitzt neben der Modellierungssprache zusätzlich werkzeugeigene Anfangshürden, die in der Summe zu einem Akzeptanzproblem führen können. Es ist auch möglich zum Beispiel am allgegenwärtigen Whiteboard oder Flipchart zu beginnen. Oft reichen wenige Modellierungsartefakte aus, um diese als Allgemeingut schnell in

der Kommunikation der Teammitglieder zu etablieren. Warum sollte bei der Erklärung der Systemarchitektur an dem einen Tag ein Softwarebaustein als Rechteck und an einem anderen Tag auf die Schnelle als Ellipse grafisch visualisiert werden? Warum benutzen wir beim Auflisten von möglichen Zuständen mal eine mit Komma getrennte Aufzählung und beim nächsten mal Bindestriche? Diesen „Luxus“ der freien, spontanen Wahl der Ausdrucksmittel leisten wir uns doch bei der Beschreibung der Hardware auch nicht. Zunächst findet sich in fast jedem Team jemand, der eine gewisse Affinität zur Modellierung besitzt. Da haben wir schon mal unseren Modellierungsexperten. Er kann für alle anderen als Ansprechpartner dienen und er könnte auch dafür verantwortlich sein, die Anwendung und Einhaltung einmal eingeführter Modellierungsstandards zu überwachen. Es ist notwendig, sich auf eine geeignete Modellierungssprache zu einigen. Daran muss das Team beteiligt werden. Es ist unendlich schwer eine Modellierungssprache gegen den Widerstand der Anwender zu etablieren. Die frühzeitige Einbeziehung des Teams kann irrationale Widerstände verhindern. Die Modellierungssprache muss übrigens nicht zwingend eine grafische Sprache wie die UML sein. Eine domainspezifische Sprache (DSL) ist für viele Anwendungsfälle ein mächtiges Werkzeug. Die Verfügbarkeit und Leistungsfähigkeit der im weiteren Verlauf einzuführenden Modellierungsumgebung sollte jedoch bei der Wahl der Sprache berücksichtigt werden. Einführungsseminare in die Modellierungssprache sind mit Sicherheit zweckmäßig. Die schrittweise Einführung ausgewählter Elemente der Modellierungssprache in die tägliche Arbeit, zum Beispiel in Besprechungen am Witheboard und in Dokumentationen, ist jedoch um Vieles wichtiger. Schritt für Schritt wird die Sprache im Team gesprochen und akzeptiert. Spätestens jetzt muss die Modellierungsinfrastruktur eingeführt werden. Werkzeuge, die ein effizientes Bearbeiten der Modelle und leistungsfähige Generatoren für die automatisierte Produktion bieten, sind unentbehrlich für eine erfolgreiche Modellierung. Im weiteren Verlauf der Etablierung einer Modellierungskultur ist zu überlegen, welche Modellsichten für weitere Projektbeteiligte wie Stakeholder, Subauftragnehmer und Auditoren sinnvoll sind.

Dieser kurze Abriss zur Einführung modellgetriebener Entwicklung zeigt, dass es keinen Modellierungs-Instant-Pudding gibt. Es gibt aber brauchbare Rezepte, die behutsam aber konsequent angewendet, jedes Projektteam auf eine neue Stufe in der Qualität seiner Softwareprozesse heben können. Alle Voraussetzungen sind gegeben. Leistungsfähige Modellierungssprachen und mächtige Werkzeuge. Es ist Zeit, fangen wir an.

Initial Signatories: Andreas Foltinek, Baris Güldali, Alexander Huwaldt, Thomas Schütz, Markus Völter, Tim Weilkiens, Andreas Willert

http://mdse-manifest.org

Die Vorträge auf der MESCONF 2017

Bruce P. Douglass: Effective Embedded Model-Based Development

Moving from a code-based to model-based development (MDD) can be a bit daunting. There are five key elements of using MDD effectively:

1. Modeling Language Skill,
2. An efficient process that incorporates modeling as a key approach,
3. Deployment of model-based verification,
4. Power modeling tooling, and
5. A strong yet flexible technology adoption strategy.

This talk reveals the best practices and approaches for each of these five key areas.

Robert Hellebrand, Tim Weilkiens: Produktlinienentwicklung auf dem Weg – Standards und Offene Herausforderungen der Variantenmodellierung

Produktlinienentwicklung, die systematische Entwicklung variantenreicher Produktfamilien hat in den letzten Jahren deutlich an Bedeutung gewonnen, um die Herausforderungen im magischen Dreieck von Qualität, Entwicklungszeit und verfügbarem Entwicklungswissen zu bewältigen.

Marc-Florian Wendland: UML Testing Profile 2

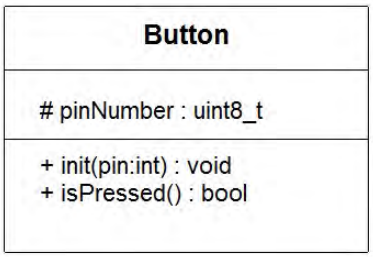
The UML Testing Profile 2 is on the verge of publication. As far back as 2001, a dedicated working group at Object Management Group (OMG) started collecting industry accepted testing concepts and practices in order to make them available via the Unified Modeling Language (UML). These efforts resulted in the adoption of the UML Testing Profile (UTP) by OMG in 2005. UTP is a standardized language for designing, visualizing, specifying, analyzing, constructing, and documenting testing artifacts commonly used for testing software-intensive systems. Ever since the interest in and application of the UTP was growing bigger and bigger. In 2014 the UTP working group at OMG has decided to rethink and renovate the UTP in order to align it with most recent requirements a modern modelling language for test design activities has to cope with.

Sieben auf einen Streich

Alexander Huwaldt, Laser & Co. Solutions GmbH, Aussteller mit dem UML-Tool SiSy

Sie sind kein erfahrener C++ Programmierer? Objektorientierung kennen Sie mehr oder weniger vom Hörensagen? Sie glauben UML das ist komplizierte Theorie, die dicke Bücher füllt? Um das alles zu lernen brauchen sie Jahre? Stimmt. Aber! Um mit der UML loszulegen reicht das grobe Verständnis von sieben Modellelementen völlig aus. Das ist sozusagen das Grundschulniveau, um erste einfache Modelle zu erstellen. Sie müssen nicht alles komplett verstanden haben, um mit der UML sinnvoll zu arbeiten. Wer also schnell anfangen möchte muss sich lediglich mit den folgenden Elementen des UML-Klassendiagramms auseinandersetzen: *Klasse, Instanz, Attribut, Operation, Sichtbarkeit, Generalisierung und Aggregation.* Was muss ein UML-Grundschüler über Klassen wissen? Zunächst sollte man der Fachsprache verständlichere Begriffe als Synonyme beistellen. Eine Klasse repräsentiert vereinfacht einen Baustein des Systems. Genauer gesagt einen Bausteintyp. Solche Bausteine haben einen Namen, eine fachspezifische Bezeichnung. Dieser Name ist der Klassenname. Das war es für das Erste auch schon. Wenn es in unserem System also Taster gibt, sollte es im Klassenmodell eine gleichnamige Klasse geben. Wenn sie diese in Ihrem UML-Tool anlegen wird die Klasse als einfaches Rechteck dargestellt. Hinweis: Verwenden Sie bei Klassennamen immer die Einzahl und es ist in der Regel eleganter den englischen Begriff zu nutzen.

Instanzen werden in der Sprache der Objektorientierung auch Objekte genannt (Nomen est Omen). Um besser zu verstehen was Instanzen sind, assoziieren wir diesen Begriff mit der hinlänglich bekannten Variable. Die gibt es faktisch in allen Programmiersprachen. Jeder Programmierer weiß, dass er einen Typ nur benutzen kann, wenn er davon eine Variable, also eine Instanz anlegt. Genauso verhält es sich mit den Klassen unseres UML-Modells. Um eine Klasse zu nutzen, benötigt man eine Instanz von dieser und die erzeugen Sie vorerst ganz genauso, wie Sie es von den Variablen gewöhnt sind. Attribute und Operationen, das sind die "Innereien" der Klassen. Attribute sind wieder mit Variablen vergleichbar nur eben, dass diese der Klasse zugeordnet sind. Operationen (oft auch als Methoden bezeichnet) sind das, was Sie bisher als Funktionen kennen, aber wiederum einer Klasse zugeordnet. Welche Attribute und Operationen könnte man dem Taster zuordnen? Hier ein Vorschlag: Der Taster soll sich merken, an welcher Pinnummer er angeschlossen ist und er muss initialisiert werden. Des Weitern wäre es nett, wenn er eine Funktion hat, mit der man abfragen kann, ob der Taster gedrückt ist. Und so sieht das dann in einem UML-Klassenmodell aus. Die Zeichen vor dem Attribut und den Operationen sind deren Sichtbarkeit. Der Codegenerator macht daraus später die C++ Schlüsselworte *protected* und *public*. Das sind Mechanismen für Zugriffsrechte auf diese Elemente und sorgen für funktionale Sicherheit. Generalisierung und Aggregation sind Beziehungen zwischen Klassen. Wenn Sie eine Klassenbeziehung mit "ist ein" übersetzen können, handelt es sich um eine Generalisierung. Das Synonym "hat" können Sie als Aggregation modellieren. Es kann tatsächlich so einfach sein. Mehr dazu unter **www.embeddeduml.de - www.sisy.de - www.myXMC.de - www.mySTM32.de** .



200 Testfälle in 20 Minuten?

Dr. Martin Beißer, Abteilungsleiter bei sepp.med, Aussteller mit dem Testfallgenerator MBTsuite und Dienstleistung im Umfeld MBT

Der Test ist integraler Bestandteil des Softwareentwicklungsprozesses. Das ist nicht nur so, wenn nach dem V-Modell entwickelt wird, sondern auch im agilen Prozess. Dabei ist die Testdurchführung nur ein Teil der Aufgabe. Entscheidend für die Qualität des Tests ist das Testdesign. Nur wenn die Testabdeckung geeignet definiert wurde und alle dafür nötigen Testfälle auch gefunden und implementiert wurden, kann eine sichere Aussage über die Zuverlässigkeit der Funktionalität gemacht werden.
Der modellbasierte Test (MBT) hat zwar schon eine lange Tradition was die Diskussion darüber betrifft, an seiner Umsetzung zeigen sich aber erst jetzt mehr und mehr Anwender interessiert.
Warum ist das so? Ein wesentlicher Grund ist die Komplexität der zu testenden Systeme. Das heißt, es werden zunehmend mehr Testfälle benötigt. Projekte, die mehrere tausend Testfälle benötigen, sind nicht selten. Die Erstellung, Pflege und Wartung dieser Testfälle ist entsprechend aufwändig bzw. stößt immer öfter an die Grenzen des Machbaren.
Der modellbasierte Ansatz schafft hier Abhilfe. Die Idee hinter MBT ist ja gerade, das Testdesign in einem graphischen Modell zu erstellen, aus dem die Testfälle automatisch in ausführbarer Form abgeleitet werden. Damit ist der lineare Zusammenhang zwischen der Anzahl der Testfälle und dem Aufwand aufgebrochen. Nicht mehr die Anzahl der Testfälle spielt die entscheidende Rolle, sondern die Testabdeckung. Es werden exakt die Testfälle generiert, die zur Erreichung der Testabdeckung nötig sind, nicht mehr aber auch nicht weniger. Da die Wartung und Pflege der Testfälle im Modell stattfindet, ist es z.B. kein Problem 200 Testfälle in 20 Minuten zu überarbeiten und neu zu generieren – ein riesiger Vorteil, wenn es wieder einmal Änderungen an den Anforderungen gegeben hat.

Weitere Vorzüge eines modellbasierten Testdesign sind:

- eine eindeutige, nachvollziehbare Testabdeckungsdefinition
- eine eindeutige Darstellung der Anforderungen
- eine Visualisierung des Testraumes
- hohe Wiederverwendung
- automatisches Requirement-Tracing
- gleichbleibende Testfallqualität

Modellbasiertes Testdesign ist im Vergleich zum manuellen Testdesign wie E-Biking zum klassischen Radfahren. Es ist zwar nicht so sportlich aber deutlich effektiver und schneller.

Wenn wir Ihr Interesse geweckt haben, finden Sie hier weitere Informationen:
http://www.mbtsuite.com

sepp.med
Qualität sichert Erfolg

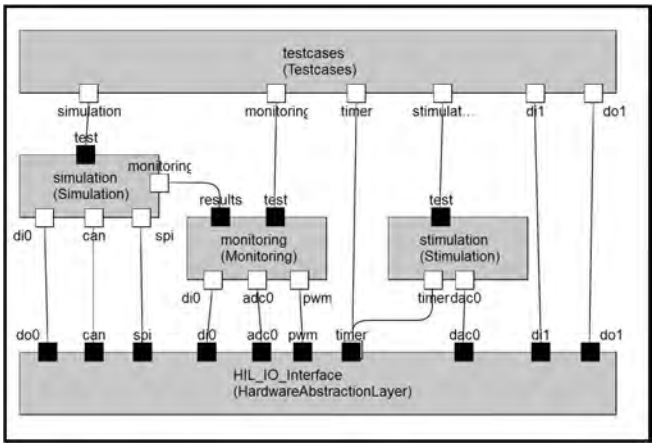
HIL and SIL Tests for Everybody

Thomas Schütz, PROTOS Software GmbH, Toolchains für Embedded Software

Die meisten Embedded Systeme werden zu spät oder nur unzureichend getestet. Fakt ist: Je später Fehler entdeckt werden, desto teurer ist die Behebung - die Kosten steigen exponentiell mit Fortschreiten des Projekts an. Im Extremfall einer Rückrufaktion für bereits gelieferte Produkte kann das zu Kosten in Millionenhöhe führen.
Es muss also Ziel sein, deutlich früher zu testen – im Idealfall bereits während der Implementierung.

Warum ist es schwierig Embedded Systeme zu testen?
Es gibt vielfältige Gründe warum Embedded Systeme so schwer zu testen sind: Im Gegensatz zu „normaler“ Desktop Software haben Embedded Systeme Schnittstellen zu physikalischen Systemen und nicht nur zu anderen Softwaresystemen. Die Software, welche diese Systeme kontrolliert, ist hochgradig nebenläufig und zustandsbehaftet. Mit den üblichen Testverfahren wie z.B. sequenziellen Unit Tests kann man solche Systeme meist nur schlecht testen.
Simulationsbasierte Software in the Loop- (SIL) oder Hardware in the Loop-Tests (HIL) sind grundsätzlich gut geeignet für solche Tests. Die gängigen Hardwareplattformen und Softwaretools sind aber häufig zu teuer, um sie bereits während der Entwicklung an jedem Arbeitsplatz einzusetzen. Außerdem ist für die Entwicklung der Tests anderes Methoden- und Toolwissen nötig als für die Entwicklung der Applikation. In den meisten Fällen werden diese Methoden daher erst sehr viel später in der Testabteilung angewendet. Der Entwickler kann so kaum strukturiert testen sodass die entstehenden Embedded Systeme während der Entwicklung häufig nicht getestet sondern eher „ausprobiert“ werden.

Wie kann man dennoch entwicklungsbegleitend testen?
Um dennoch entwicklungsbegleitend testen zu können sollten folgende Voraussetzungen erfüllt sein: Der Entwickler sollte die Methoden und Tool bereits kennen, oder sich schnell einarbeiten können. Hardwareplattformen und Softwaretools müssen günstig sein wenn sie an jedem Arbeitsplatz eingesetzt werden sollen. Methoden und Tools sollten die Entwicklung von Tests für Embedded Systeme auf unterschiedlichen Ebenen unterstützen (zumindest Komponenten- und Integrationstests). Das Open Source-Modellierungswerkzeug Eclipse eTrice ermöglicht den Aufbau einer kostengünstigen und dennoch mächtigen Testplattform: Im Modell können Komponenten für Stimulation, Monitoring und Simulation entwickelt und generiert werden. Mit dem eTrice Add-on T-CaGe können Komponenten mit Testcases für das System beschrieben und generiert werden. Die kombinatorische Testcasegenerierung ermöglicht eine sehr schnelle Entwicklung von Testcases mit hoher Abdeckung.



Alle Komponenten zusammen bilden einen portablen, echtzeitfähigen „Test Harness“ für die Applikation. Diesen kann man als SIL-Test (z.B. auf dem Entwicklungsrechner) oder als HIL-Test auf kostengünstiger Hardware (z.B. Standard Evaluation Boards von Infineon oder ST-Microelectronics) ausführen.
Die vollautomatische Durchführung der Tests wird durch einen Jenkins Continuous Integration Server (ebenfalls Open Source) übernommen.

Test First für Embedded Systeme!
Durch die gewählte Kombination aus Standard-Hardware und größtenteils frei verfügbaren Open Source Tools kann man eine modellgetriebene SIL- oder HIL-Testlösung aufbauen. Diese ermöglicht es bereits während der Applikationsentwicklung Embedded Systeme mit hoher Abdeckung automatisiert zu testen.

http://www.eclipse.org/etrice/ - http://www.protos.de

Visionen für die MESCONF

Die MESCONF soll sich im nächsten Jahr weiterentwickeln. Bei der diesjährigen Konferenz testen die Firmen Willert Software Tools und Laser & Co. Solutions erstmals eine Erweiterung des Kongressformates um einen zweiten Veranstaltungstag. Der erste Veranstaltungstag soll wie bisher werkzeug- und anbieterunabhängig Themen rund um die modellgetriebene Entwicklung eingebetteter Systeme anbieten. Der zweite Tag soll in Zukunft Raum und Zeit für die praktische Anwendung konkreter Werkzeuge oder spezifischer Lösungen verschiedener Anbieter schaffen. In den kommenden Veranstaltungen sollen alle Aussteller die Möglichkeit erhalten zum Beispiel Hands-On-Workshops mit ihren Werkzeugen zu organisieren. Für die Teilnehmer ergänzt sich die Theorie des ersten Tages mit der Praxis des zweiten. Interessenten können sich bei den Firmen Willert Software Tools und Laser & Co. Solutions über eine mögliche Teilnahme an den Praxis-Workshops informieren.

Das Buch zur MESCONF

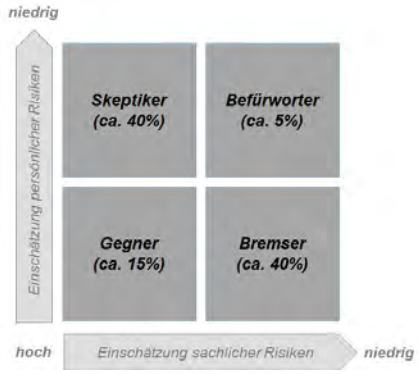
Tim Weilkins, oose Innovative Informatik eG

Bis zur nächsten MESCONF wird das praxisorientierte Buch "*Modellbasierte Entwicklung verstehen und anwenden, Der erfolgreiche Weg zur professionellen Modellierung Eingebetteter Systeme"* für alle Interessenten verfügbar sein. Das Buch wird vom Verlag dpunkt verlegt. An dieser Stelle ein kleiner Ausschnitt aus dem Inhalt.

Einführung eines modellbasierten Ansatzes in einer Organisation
Die Einführung eines neuen Vorgehens in eine Organisation ist auch immer ein Veränderungsprozess. Vielmehr als die fachlichen Themen der Einführung, sind die menschlichen Faktoren entscheidend für ihren Erfolg. Die dazu sinnvollen Vorgehensweisen in einem Veränderungsprojekt können ein eigenes Buch füllen. Wir beleuchten diese Themen trotzdem kurz und fokussieren auf die fachlichen Aspekte der Einführung.

Ausgangslage
Wir gehen davon aus, dass in der Organisation, in der das modellbasierte Vorgehen eingeführt werden soll, bisher gar nicht oder nur punktuell und ungeplant modelliert wurde. Die Einführung steht also noch ganz am Anfang. Innerhalb der Organisation gibt es drei wesentliche Interessensgruppen:

1. Die Initiatoren und Treiber,
2. die Organisationseinheiten, die modellbasiert vorgehen sollen und
3. die für Zeit, Budget, usw. verantwortliche Organisationseinheiten.



Alle drei Interessensgruppen müssen an einem Strang ziehen, damit die Einführung erfolgreich ist. Das bedeutet insbesondere, dass die Personen in der umzusetzenden Organisationseinheit das Vorhaben wollen und unterstützen. Und das die verantwortliche Organisationseinheit bereit ist, das notwendige Geld und die Zeit zur Verfügung zu stellen. Die Abbildung zeigt die typische prozentuale Verteilung des Unterstützungsgrads bei den von einer Veränderung betroffenen Personen. Sie sollten sich vor allem auf die große Gruppe der Unentschlossenen (Skeptiker und Bremser) konzentrieren und nicht auf die kleine Gruppe der Gegner, die aber in der Regel am lautesten ist und, wenn man nicht aufpasst, wertvolle Ressourcen bindet.

Wichtig für den Erfolg der Einführung ist auch die Begleitung des Vorhabens durch einen erfahrenen, ggf. externen Experten, um

- das neue Wissen über das Vorgehen schnell und direkt in die Organisation zu tragen.
- typische Fallstricke, die der Experte aus seiner Praxis kennt, frühzeitig zu erkennen und somit Zeit zu sparen und Scheitern-Ängste in der Organisation nicht zu schüren.
- ein neutrales Bild von außen ohne Betriebsblindheit zu erhalten.

Achten Sie darauf, dass der Experte die Einführung nur begleitend unterstützt, aber nicht übernimmt. Die Verantwortung und Aufgaben müssen innerhalb der Organisation bleiben, um die neue Vorgehensweise nachhaltig zu verankern.

Bildquelle: Mohr, N., Woehe, J. M., Diebold (1998). Widerstand erfolgreich managen. Professionelle Kommunikation in Veränderungsprojekten. Frankfurt/Main: Campus Verlag.

Organisatoren

Die MESCONF wird organisiert von Andreas Willert (Willert Software Tools), Alexander Huwaldt (Laser & Co. Solutions), Alexander Schneider (Mentor Graphics) und Tim Weilkiens (oose) sowie dem Verlag Elektronik Praxis.

Sponsoren:

Aussteller