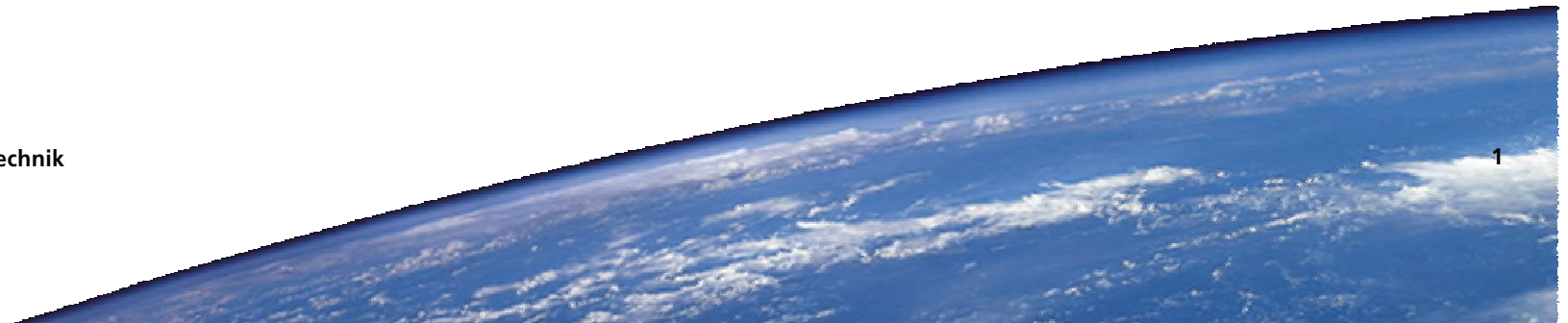




Einsatz von Software Engineering Methoden zur Senkung des Ressourcenverbrauches in Software Projekten

Anita Herrmann
Simulations- und Softwaretechnik

Köln, 6.11.01



Erfolgsquote von neuen, innovativen Ideen und Erfolgsfaktoren

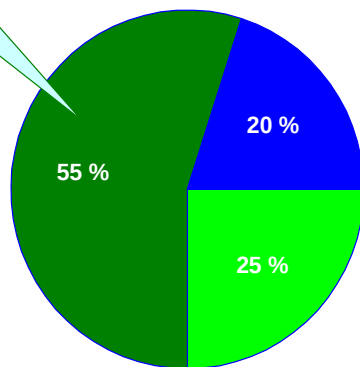


Neue innovative Ideen werden oftmals erst durch ihre Umsetzung in Software als Produkt oder Dienstleistung auf dem Markt wirksam.

Nur eine von sieben innovativen Ideen führt zu einem auf dem Markt erfolgreichen Produkt

Bei **55 %** wird die Entwicklungs- und Testphase nicht beendet oder ist nicht erfolgreich

Gescheiterte Projekte



Erfolgsfaktoren

- Sicherheit bei der Bewertung des Markterfolgspotenzials
- Zielgerichtete Entwicklung durch klare Definition der Funktionalitäten und realistische Planung
- Dialog mit den künftigen Nutzern des Systems während der gesamten Entwicklung
- Effektive Organisationsstruktur und ein kompetentes motiviertes Projektteam
- Effektiver Einsatz von Methoden und Werkzeugen bei der Entwicklung

Wie sollten die Ressourcen für die Entwicklung geplant und verteilt werden, um den Erfolg zu sichern?

Softwareprojekt InnoGuide

Ziele



- **Entwicklung einer PC-gestützten Stufen-Bewertungssoftware für Innovationsprojekte.**
- Analyse der Entwicklung von InnoGuide als **Untersuchungsobjekt** bzgl. Anwendung **effektiver und einfachster Software Engineering Verfahren** für kleine Software-Projekte
- **Praxisorientierter Ansatz** für die Überarbeitung der DLR Software-Minimalstandards (SQS)

Status:

- **seit März 2000 entwickelt** (durch SISTEC und die Firma webdynamix im Auftrage des Innovationsmanagements)
- **seit Jan 2001 auf dem Markt mit einer Vorversion = 60 Kunden**
- **Derzeitiger Umfang: 16 000 LoC = 52 Zeilen/Arbeitstag**



Software Engineering Umgebung

- **Visual Studio mit VB 6.0 Enterprise**, Win API Funktionen und VBA-Anbindung an MS Office (Access, Word, PowerPoint); lauffähig unter Win 95-2000, NT 4.0; VBFramePlus als Zusatzbibliothek
 - Datenbankmodellierung/ERM: manuell mit Access
 - Konfigurationsmanagement: **Visual SourceSafe**
 - Code-Verpackung mit **InstallShield**

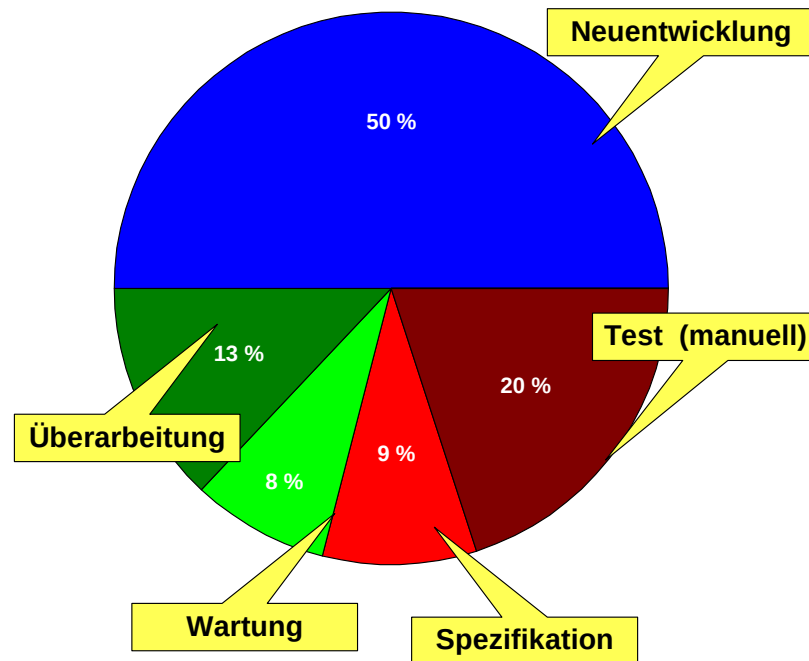
Bisher noch nicht wirksam (verspätete Bereitstellung durch Rational):

- CASE-Modellierung: UML - **Rational Rose**
- automatisierter Test mit **Rational Test Robot**



Modellierung und Test bisher weitgehend manuell

Planung des Entwicklungsaufwandes Ist - Anteile an der Gesamtentwicklungszeit

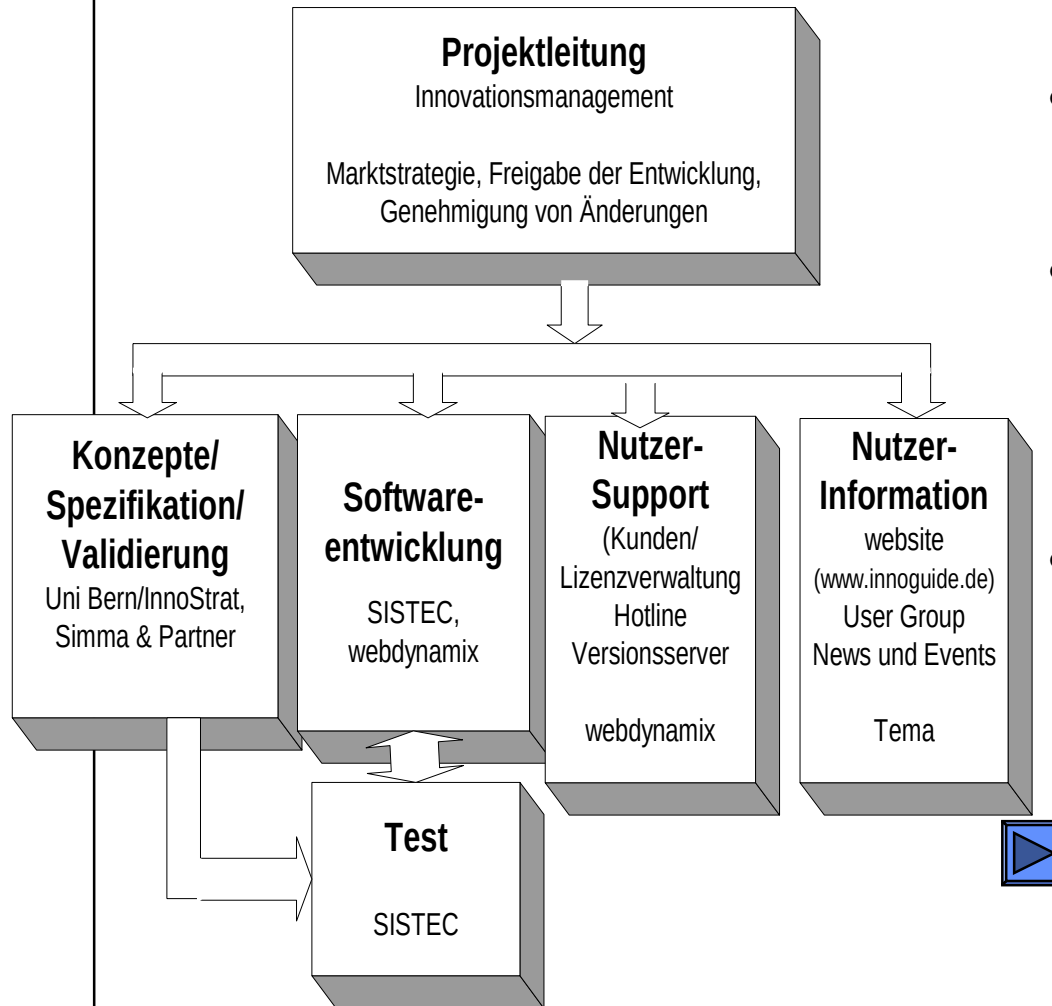


- Schätzung des Entwicklungsaufwandes für die spezifizierten Einzel-Funktionalitäten
- **Weitere Kostenfaktoren sind zu berücksichtigen:**
 - Spezifikation (9 %)
 - Test (20 %)
 - nachträgliche Anpassungen an internationale Trends, Überarbeitungen (13 %)
 - Klärung von schwierigen Problemen (z.B. Passfähigkeit von MS .dll's) und
 - Wartungsaufwand für schon auf dem Markt befindliche Versionen (8 %).



Ca. 100 -120 % des funktional kalkulierten Aufwandes waren zusätzlich erforderlich.

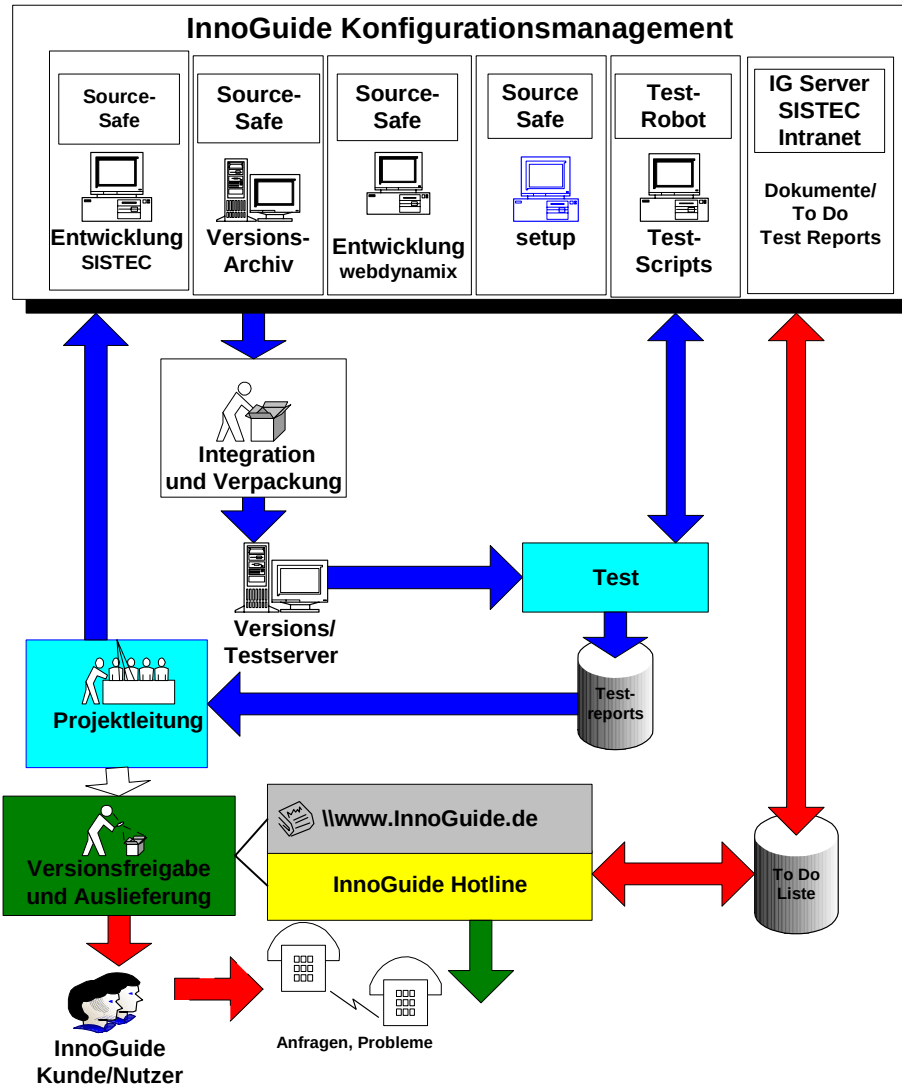
Prinzipien der Projektorganisation



- Entwicklung der Software verteilt in einem Netzwerk von Partnern
- Spezifikation und Validierung der Software durch international anerkannte Experten des Innovationsmanagements
- Der Nutzerdialog ist Bestandteil der Projektorganisation

Die Entscheidungs- und Bearbeitungsprozesse sind definiert und es werden einheitliche Entwicklungsstandards angewendet.

Konfigurationsmanagement



Aufgaben:

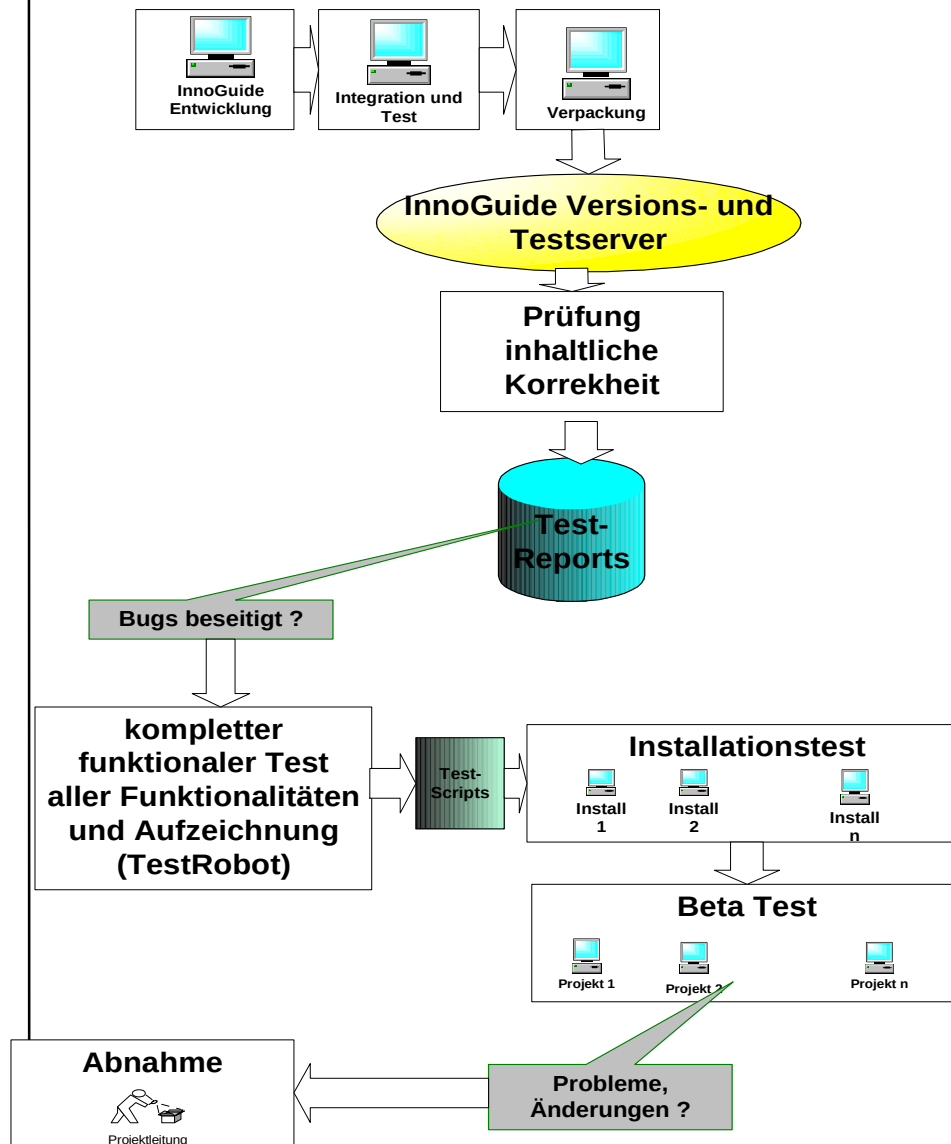
- automatische, konsistente Bereitstellung aller bisher erarbeiteten Versionen
- automatisierter Codevergleich und Statusanalyse der Versionen
- systematische Verwaltung aller Problemmeldungen und Änderungsanforderungen (nicht durch SourceSafe abgedeckt)



Wird kein Konfigurationsmanagement angewendet, dann kann ein **Mehraufwand bis zu ca. 25 %** der gesamten Entwicklungsaufwandes notwendig sein!



Testphasen

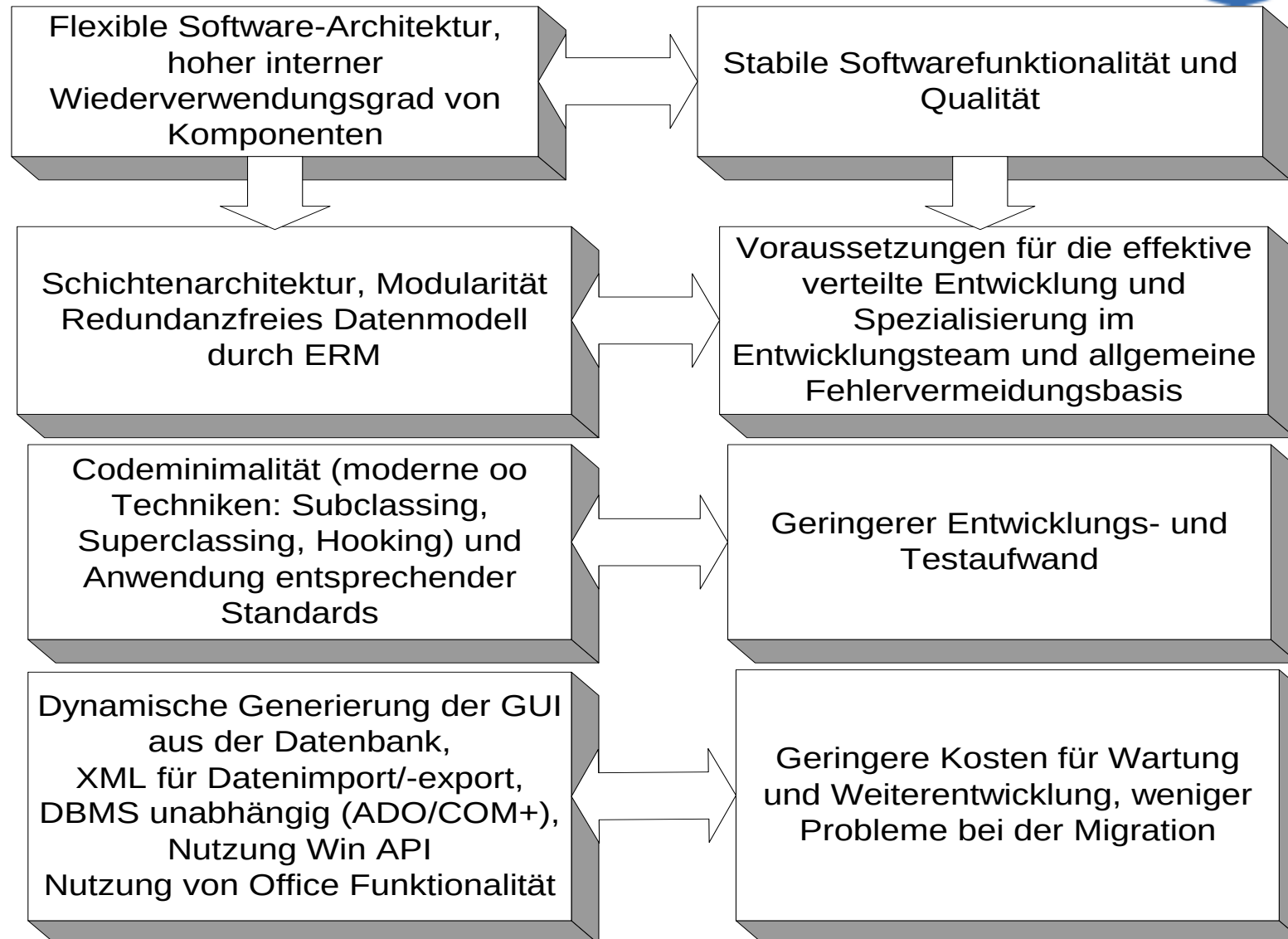


- **Jede Funktionalität** der Software muss **in den** (wie im Handbuch angegebenen) **Umgebungen** korrekt lauffähig sein.

1. **Funktionsvalidierung** - inhaltliche Korrektheit
2. **Kompletter Funktionalitätstest** - softwaretechnisch
3. **Installationstest** (Kompatibilität der Umgebung/Bibliotheken)
4. **Beta-Test** - Konformität zu den Nutzererwartungen/Bedienerfreundlichkeit

▶ **Ressourcenreduzierung** durch Automatisierung von wiederholten Tests: **TestRobot**

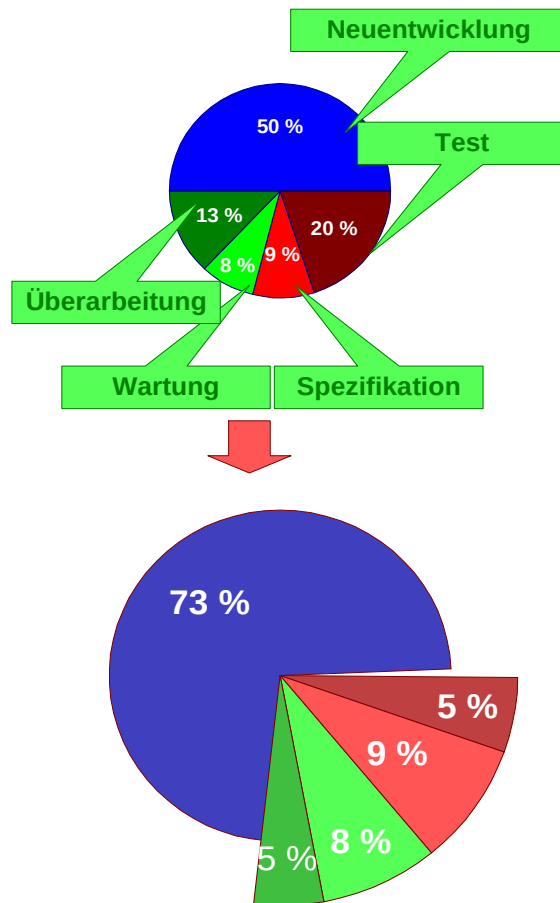
Einfluss der Designstrategie/-flexibilität auf die Kostenfaktoren bei InnoGuide



Kosten/Nutzen Software Engineering InnoGuide



Ressourcengewinn durch effektivere
Hilfsprozesse, Softwareflexibilität und
Wiederverwendung



Software Engineering Kosten(InnoGuide)

- Entwicklungsumgebung ca. **20 TDM**
- Aufwand für Installation/Anpassung von Tools **1,5 MM**

Möglicher Nutzen (grob geschätzt bezogen auf die notwendige Gesamtentwicklungszeit für InnoGuide)

Reduzierung bis zu:

- 10 %** durch Konfigurationsmanagement
- 15 %** durch Testautomatisierung bei größerer Verlässlichkeit der Versionen
- 5-8 %** durch Vermeidung von Überarbeitungen (Designflexibilität)

Wartungs- und Weiterentwicklungsphasen:

- 50-60 %** Reduzierung durch Designstrategie, Standards und Anwendung moderner Entwicklungstechniken



Fazit im Hinblick auf die Aktualisierung der DLR Software-Minimalstandards (SQS)

Vorschläge für neue Inhalte:

- **Frühe Identifikation von Schwachstellen der Projektdefinition durch Innovationsbewertung der Software-Entwicklung,**
- **Minimalanforderungen an den Investitionsschutz, Projektmanagement, Empfehlungen zum Software Life Cycle**
- **Aktualisierte Empfehlungen zu „software development best practices“** (speziell zu verteilten Systemen und neueren Sprachen):
 - Konfigurationsmanagement
 - Kommunikation und effektive Software Dokumentation
 - Software Spezifikations- und Designmethoden
 - Sammlung von Regeln, Prinzipien, Patterns zur Code- und Designflexibilität und Software-Wiederverwendung
 - Effektive Testverfahren
 - Empfehlungen zur Software Engineering Toolanwendung