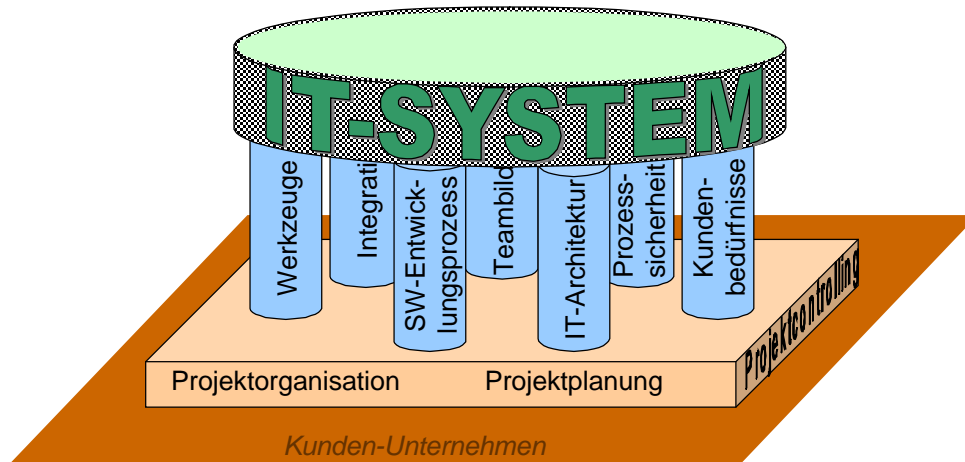


## Die 7 Säulen des IT-Projektmanagements

### Das Bild zum Thema



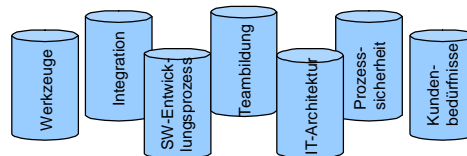
Die 7 Säulen des IT-Projektmanagement,  
Vortrag FG IT-Projektmanagement\_19032004 - Folie 1



© Steinbeis-Transferzentrum IT-Projektmanagement

## IT-Projektmanagement - Vortrags-Übersicht

- Einleitung und Übersicht
- Die Bedürfnisse des Kunden
  - ◆ Definieren das Ziel des Projekts, sind dessen wichtigster Maßstab
- Die IT-Architektur
  - ◆ Bildet das Rückgrat der zu schaffenden Lösung, muss nachhaltig sein
- Der SW-Entwicklungsprozess
  - ◆ Wasserfall-, V- und Spiralmodell, Iteratives Vorg. => Projektphasen, -planung
- Werkzeuge und Artefakte (Zwischenprodukte für das System)
  - ◆ Müssen angemessen sein, nur effektiver Umgang sichert Wirtschaftlichkeit
- Integration
  - ◆ Zielsystem muss ins Bestehende passen, EAI-Aspekte, organisator. Aspekte
- Teambildung
  - ◆ Auch mit dem Kunden ! - nur das Team gewinnt - Rollen und Verantwortung
- Prozess-Sicherheit, Performance und QS - ggfs. incl. Projektcontrolling
- Zusammenfassung



Die 7 Säulen des IT-Projektmanagement,  
Vortrag FG IT-Projektmanagement\_19032004 - Folie 2



© Steinbeis-Transferzentrum IT-Projektmanagement

## Was zeichnet IT-Projekte aus ?

### ➤ Welche Rolle spielt die IT (Informationstechnologie) ?

- + Sie soll in der Regel die Arbeitsprozesse unterstützen, die Abläufe automatisieren und - wo möglich - vereinfachen, standardisieren
  - + Sie soll bessere Dienstleistungen ermöglichen, reichere Informationen bieten, dem Kunden / Anwender dienen
  - ± Sie hat technische Spezifika, die sich immer wieder ändern / erneuern und die nicht alle kennen - diese können die Produktivität erhöhen oder auch die Beteiligten verwirren
  - Sie erfordert möglicherweise Umorganisationen, Änderungen, die nicht jede/r Beteiligte mag
  - Sie kann zur Rationalisierung bzw. Jobabbau eingesetzt werden
  - Mit ihr kann ggfs. die Arbeitskontrolle erhöht ("verbessert") werden
- ....

➔ Die IT ist ein mächtiges Werkzeug, das kennengelernt und sinnvoll genutzt werden muss, das Verbesserungen, aber auch Bedrohungen schaffen kann.

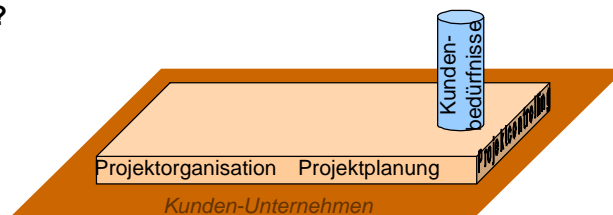
## Die erste Säule: Die Bedürfnisse des Kunden

### ➤ Aus den Bedürfnissen des Kunden resultiert das Projekt !

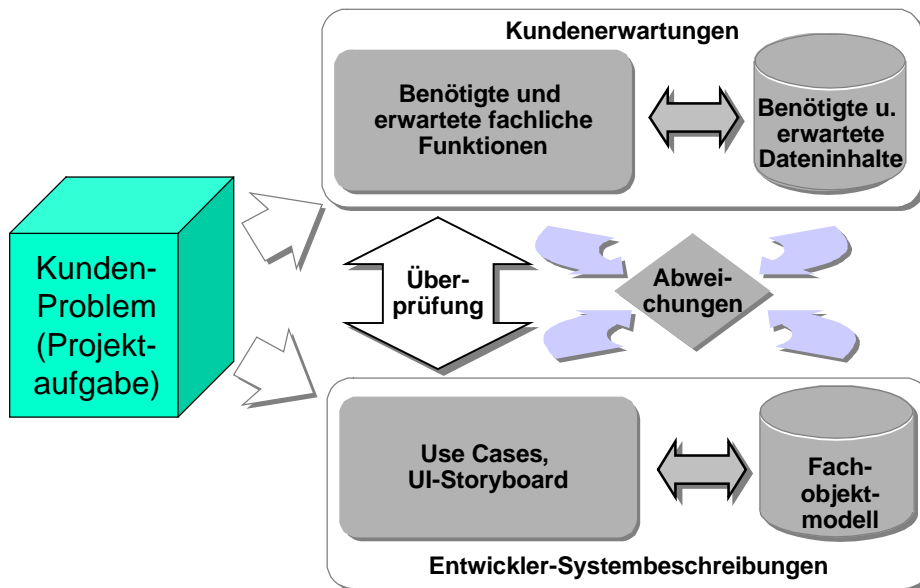
- ◆ Diese Bedürfnisse müssen also wesentlich das Projekt bestimmen

### ➤ Fragestellungen daraus:

- ◆ Wodurch werden die Kundenbedürfnisse deutlich gemacht?
- ◆ Wie werden sie "operationalisiert" (umgesetzt)?
- ◆ Sind die "genannten" Bedürfnisse die "tatsächlichen" Bedürfnisse?
- ◆ Welche Rolle spielt die Beratung durch den Auftragnehmer?
- ◆ Welche Rolle spielen unterschiedliche Abteilungen, Organisationseinheiten, Einflussgruppen beim Kunden?
- ◆ Wie versteht "das Projekt" den Kunden, und wie der Kunde, was das Projektteam vor hat?



## Beschreibungen von Anforderungen und Modellen



## Kundenbedürfnisse: Bekannte Probleme

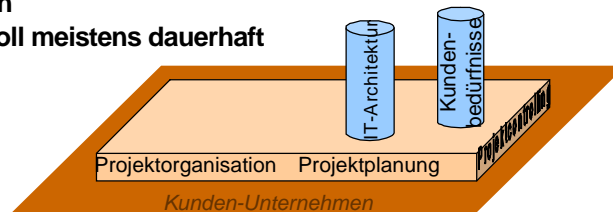
- **Auftraggeber hat nicht gleiche Ziele wie die Fachabteilung**
  - ◆ Teilweise haben Geldgeber und "System-Empfänger" deutlich unterschiedliche Interessen
- **Verbindlichkeit lässt sich auf Kundenseite nur schwer herstellen**
  - ➔ Teilabnahmen, (Er-)Klärungs-Workshops, Präsentationen, Iteratives Vorgehen
- **Irrationale Terminvorstellungen**
  - ➔ Gemeinsame transparente Planung mit Kunden-Seite, am besten iterative Schritte
- **Kundenseite versteht zu wenig von der IT des neuen Systems**
  - ◆ Dadurch lassen sich bestimmte Vorteile nicht nutzen
  - ◆ Risiken und Probleme beim späteren Systemeinsatz werden ggfs. falsch eingeschätzt

## Kundenbedürfnisse: Best Practices

- **Auftragsklärung als erste "Gemeinschaftsaufgabe" vertiefen**
  - ◆ Gemeinsame Interessenlage verdeutlichen
- **Gemeinsam ein fachliches Glossar entwickeln**
  - ◆ Begleitet ein Projekt bis zum Projektende, ist "Gold wert" für alle späteren Projektmitarbeiter (auch später bei neuen Versionen)
- **Eine frühe erste Version erstellen**
  - ◆ Hilft, eine Menge Irrtümer - auf beiden Seiten - zu entdecken
  - ◆ Dadurch wächst das gegenseitige Verständnis (fachlich ↔ technisch)
- **Um einfache Darstellungen ringen**
  - ◆ Gilt nicht nur für die Kundenseite - wieviele komplizierten Diagramme werden nicht hinterfragt, weil sie nicht durchschaut werden ?
- **Projektmitarbeiter des Kunden erklärt Prototyp seinen Kollegen**
  - ◆ Unter Beisein des Auftragnehmers hilft dies, die Resonanz beim Kunden sachlicher einzuschätzen

## Die zweite Säule: Die IT-Architektur

- **Die IT-Architektur prägt und strukturiert wesentlich das zu schaffende IT-System**
  - ◆ Die "richtige Architektur-Entscheidung" muss in der Regel sehr zeitig getroffen werden
- **Stichworte dazu**
  - ◆ Existiert ein ähnliches "Schwester-System", so wird die Architektur oft von diesem (ggfs. mit Modifikationen) übernommen
  - ◆ Basiert das neue System auf einer Standard-SW, so schränkt diese häufig die verbleibenden Architektur-Alternativen ein
  - ◆ Mit der IT-Architektur sind oft die Möglichkeiten einer "Zerlegung" des Projektes verbunden
  - ◆ Die IT-Architektur soll meistens dauerhaft und nachhaltig sein



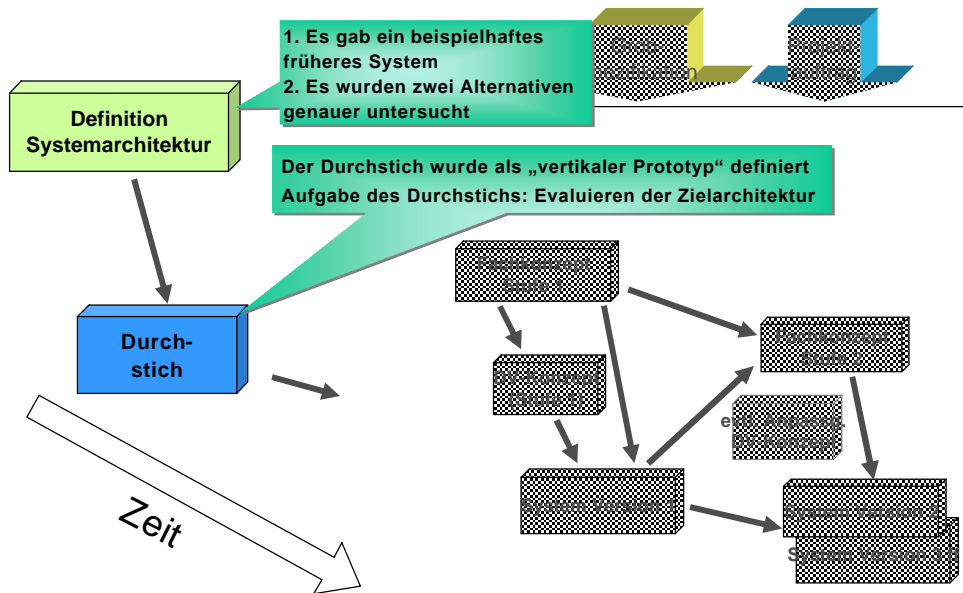
## Bedeutung der IT-Architektur für die Planung

- Die vorgesehene IT-Architektur ist die zweite wesentliche Randbedingung für den Beginn der Projektplanung
  - ◆ Gibt es ein bestehendes System als „Modell“ ?
  - ◆ Gibt es eine Voruntersuchung zur Architektur-Entscheidung ?
- Beispiele für die Charakteristik der IT-Architektur
  - ◆ Wieviele Stufen (Tier) soll die Architektur haben? (J2EE: Meist 4 Stufen)
  - ◆ Soll der Client-Teil der Anwendung nur aus HTML-Seiten bestehen („Thin Client“) oder wird „Fat-Client“-Architektur angestrebt ?
  - ◆ Welche wichtigen Dienstleistungs-Schnittstellen stehen dem zu schaffenden System zur Verfügung ?  
Werden Teilfunktionalitäten durch Standard-Software abgedeckt ?
  - ◆ Welche technische Details sind bereits gelöst (z.B. Middleware-Produkte), in welchen stecken große Risiken ?
- ➔ Ohne eine bestimmte Klarheit in der Architektur ist keine genauere Planung möglich

## IT-Architektur und Kosten

- Die zu wählende Architektur hängt ab von den Anforderungen, aber auch von den für das IT-System vorgesehenen Kosten
  - ◆ "Design (architecture) to cost"
- Beispiel: Ein Kundenverwaltungssystem kann sehr unterschiedlich gestaltet sein
  - ◆ Für 3.000 ? eine intelligente EXCEL-Mappe
  - ◆ Für 30.000 ? eine umfangreichere Access-Lösung
  - ◆ Für 300.000 ? eine große Client-Server- oder J2EE-Lösung
  - ◆ Für 3.000.000 ? ein umfangreiches Kundenportal ....
- Kosten ⇔ IT-Architektur ⇔ Kundenbedürfnisse
  - ◆ ggfs. Verzicht auf eine avisierte IT-Architektur aus Kostengründen
  - ◆ ggfs. Verzicht auf bestimmte Kundenwünsche wg. der IT-Architektur
  - ◆ ggfs. Aufbohren der IT-Architektur wg. bestimmter Kundenwünsche, Beispiele dazu : Security/Transaktionsschutz, zentrale Services (WebServices), Host-Einbindung ...

## Architekturmodell - Projektbeispiel (frühe Phase)



## IT-Architektur: Bekannte Probleme

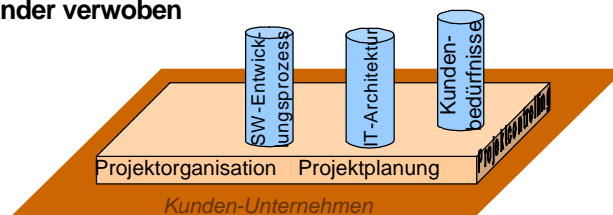
- **Neue Technologie, Sich-Einlassen auf Hersteller-Versprechen**
  - ➔ Bei Neuen Technologien grundsätzlich mit Iterationen arbeiten
  - ◆ "Pioniere" müssen idR immer Lehrgeld bezahlen
  - ➔ ggfs. akzeptabel, wenn dadurch Marktvorteil erzielbar
- **Abhängigkeit von Standard-SW-Implementierung oder Zulieferungen**
  - ◆ Achtung, Berater von Herstellern sind nicht neutral !
  - ➔ Prototypen oder Frühe Versionen ausmachen, notfalls Verträge mit Strafen
  - ➔ Das gilt ebenso für benutzte Middleware etc. (s.o.)
- **Starke Änderungen der Anforderungen**
  - ◆ Z.B. wegen Firmenzukauf, Ausweitung der Geschäfte etc.
  - ➔ Sofortige Klärung über Lenkungsausschuss o.ä., Problem nicht verschleppen !
- **Erst bei Einsatz zeigt sich Instabilität der neuen Architektur**
  - ◆ Wenn last- oder sicherheitskritische Anforderungen:
  - ➔ Gute (!) Experten früh mit einbeziehen (v. Hersteller o.ä., Praktiker !)
  - ➔ Lasttest u.ä. bereits mit früher Version anstreben (Risiko-Minimierung!)

## IT-Architektur: Best Practices

- **Es gibt bereits ein System mit beispielhafter IT-Architektur**
  - ◆ **Stärken-Schwächen-Analyse des vorliegenden Systems - Prüfung, wieweit für neue Kundenanforderungen tragfähig**
  - ◆ **Falls nichts bekannt: Schauen Sie sich um, vieles ist schon vorhanden, ggfs. über Fremdnutzen, externe Beratung u.ä. nachdenken**
- **Definition eines Architektur-Prototyps**
  - ◆ **Knapp halten durch enge Vorgaben (Ziele, Termin, Aufwand,...) - deutlich benennen, was gezeigt / bewiesen werden soll**
  - ◆ **In der Zeit seiner Realisierung können fachliche Erörterungen vertieft werden**
  - ◆ **Sehr ähnlich: Eine frühe erste Version erstellen zur Verifizierung der IT-Architektur**
- **Schnittstellen mit anderen Systemen**
  - ◆ **Keine (wichtigen) Eingangsdaten ohne Eingangsprüfung**
  - ◆ **Frühe Termine anstreben, da oft Schwachpunkt der "Late-Integration"**

## Die dritte Säule: Der SW-Entwicklungsprozess

- **Der SW-Entwicklungsprozess prägt und strukturiert wesentlich den Projektverlauf selbst**
  - ◆ **Die Abschnitte des SW-Entwicklungsprozesses bilden in der Regel die wichtigsten Meilensteine im Projektverlauf**
- **Stichworte dazu**
  - ◆ **Wasserfall-Modell oder Spiral-Modell oder "etwas dazwischen"?**
  - ◆ **Heute gewinnen Iterative Entwicklungsmodelle und Agile Methoden an Bedeutung - sie entsprechen eher den kurzfristigen Anforderungen**
  - ◆ **Das Management muss das SWE-Prozessmodell mit tragen**
  - ◆ **SW-Entwicklungsprozess, IT-Architektur und die Kundenbedürfnisse sind oft eng miteinander verwoben**



## SW-Entwicklungsprozess: Bekannte Probleme

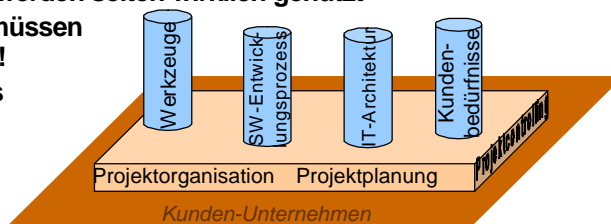
- Vorgehensweise nach Wasserfall-Modell birgt bei Neuen Technologien höhere Risiken, ist unbeweglicher, ergibt oft schlechtere Produktqualität
  - ➔ Bei Neuen Technologien grundsätzlich mit Iterationen arbeiten
- Änderungsanträge ("Change Requests") überschwemmen das Projekt, noch ehe Realisierung fertig
  - ◆ Hohe Änderungsrate zu erwarten ➔ eher iterativ arbeiten
  - ◆ "Time-to-market" beachten: Größeres Team, um schneller zu beenden
- Spiralmodell: Probleme hinsichtlich Vertragsmodell, Gewährleistung etc. bedürfen Veränderungen in Organisation und Denkweise
  - ➔ Wasserfall: Kauf eines Systemumfangs nach Feinkonzept
  - ➔ Spiralmodell: Kauf einer Prozesskompetenz als Dienstleistung, Projekt muss daraus eine Win-Win-Situation schaffen

## SW-Entwicklungsprozess: Best Practices

- Hat sich für den SWE eine Firmenkultur entwickelt und optimiert, so lebt es sich nach dieser in der Regel recht gut
  - ◆ Voraussetzung: "Er funktioniert" und Lenkungsausschuss trägt dies
- Je bekannter die Technik und je klarer die Anforderung / Lösung, umso eher kann auch mit Wasserfall sehr effektiv produziert werden
- Für moderne Technologien (OO, etc.) haben sich eher iterative Verfahren bewährt
- Mit praktikierbarem Spiralmodell steht ein agilerer SW-Entwicklungsprozess zur Verfügung
  - ◆ Kommt der objektorientierten Vorgehensweise sehr entgegen (Generalisierung - Spezialisierung)
  - ◆ geringere technische und kaufmännische Risiken
  - ◆ Höhere Flexibilität z.B. bzgl. Funktionalität, Auslastung etc.
  - ◆ Chance, System mit gewünschter Funktionalität zeitnah zu erhalten

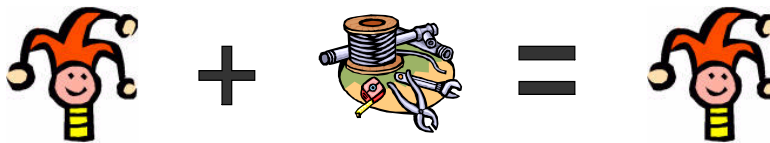
## Die vierte Säule: Werkzeuge für das IT-Projekt

- **Werkzeug** sei hier verstanden nicht nur als Tools sondern auch inclusive der damit erarbeiteten Ergebnisse ("Artefakte")
  - ◆ Also umfasst dies nicht nur die "Tools für die Entwicklung" sondern auch die "Dokumentationsmittel für die Verständigung" zum IT-System
- **Stichworte dazu**
  - ◆ Nur effektive Tools und deren rasche Beherrschung sichern einen wirtschaftlichen Projektprozess
  - ◆ Alle während des SWE-Prozesses erzeugten Artefakte dienen nur der Verständigung über die genaue Form des zu schaffenden IT-Systems, sie müssen also vor allem diesem Ziel dienen
  - ◆ Zu komplexe Tools werden selten wirklich genutzt
  - ◆ Projektergebnisse müssen verstanden werden !
  - ◆ "A fool with a tool is still a fool"
  - ◆ **KISS - Keep It Small and Simple**



## Methoden und Werkzeuge

- "A fool with a tool is still a fool"



- **Werkzeug** ⇔ **Methode**
  - ◆ Methode muss prinzipiell klar sein, ehe das Werkzeug benutzt wird
  - ◆ Man muss es auch "mit der Hand" machen können - das Werkzeug nimmt einem nur die Komplexität und das aufwändige oder wiederholte Arbeiten ab
  - ◆ Das Werkzeug muss durchschaubar bleiben, zumindest für einfache Beispiele
  - ◆ Ein Werkzeug kann den Benutzer "führen", es kann normieren, aber nicht das Methodenverständnis ersetzen

## Lean Production: Vermeidung von Redundanzen

- **Überprüfung von Artefakten**
  - ◆ Helfen diese wirklich, sind sie notwendig für die Weiterarbeit ?
  - ◆ Gibt es unnötige Redundanzen ?  
Sind z.B. Feld-Listen nur an einer Stelle geführt, nicht doppelt ?
  - ➔ Zur Vermeidung helfen z.B. Verweise auf die entsprechende Liste
  
- **Bei iterativer Entwicklung**
  - ◆ Überprüfung am Ende einer Iteration: Welche Artefakte wurden tatsächlich benötigt ? Mit welchen wurde gearbeitet ?
  - ◆ Festhalten, an welchen Stellen zu wenig genau, an welchen möglicherweise zuviel gearbeitet wurde  
➔ Lernen für die nächste Version
  - ◆ Ziel "Verschlankung wo möglich" als eine ständige PM-Aufgabe

## Werkzeuge u. Artefakte: Bekannte Probleme

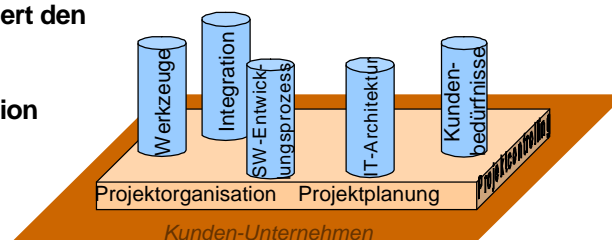
- **Neue Werkzeuge, die noch niemand professionell benutzen kann**
  - ➔ Dringend externe Unterstützung für den Start organisieren
  
- **Durchgängigkeit von Werkzeugen nicht gegeben**
  - ➔ Wird dies früh genug erkannt, wenn möglich Werkzeug ablehnen, falls dies nicht möglich, versuchen, eigene Hilfsmittel zur Überbrückung bzw. Verifikation zu schaffen
  - ➔ Ist Re-Engineering nicht möglich, für iterative Entwicklung ungeeignet
  
- **Ergebnisse werden nicht verstanden, insbesondere auf Kundenseite**
  - ➔ Enormer Risikofaktor für das Projekt, alles dafür tun, um ihn zu verkleinern
  - ➔ Vorführungen, Sprache der Benutzer suchen, an Beispielen verdeulichen ...
  
- **Projektpläne setzen immer wieder neu auf (keine Kontinuität)**
  - ➔ Für regelmäßige Aktualisierungen sorgen, nicht zu fein planen !
  - ➔ Kontinuität geht vor Genauigkeit / Feinheit

## Werkzeuge und Artefakte: Best Practices

- Externen Coach auf Zeit für die Einarbeitung in neues Werkzeug holen
  - ◆ Zum Beispiel auch halböffentliche Vorführungen für das Team
- Frühe Schaffung eines positiven Beispiels (Artefakte aller Art)
  - ◆ Durch erfahrenen oder Chef-Entwickler incl. Feinschliff (QS !)
  - ◆ Durchsprechen im Team, mit Kunden, etc. ... "so sollte es aussehen"
  - ◆ Keine Funktion wird in der IT häufiger benutzt als das Kopieren !
- Auch Protokolle sind für die interne Organisation wichtig (Qualität)
- Nutzung von ständig nachgeführten OpenPoint-Listen für PM
- Bewusstes Projektmarketing durch qualitativ hochstehende Berichte
- Durchschaubarkeit und Klarheit durch Standardisierung fördern
- In regelmäßigen Abständen: Weiterbildungen zur Toolnutzung
  - ◆ durch interne Workshops, externe Schulungen etc. (nach 6 Monaten Benutzung größere Aufnahmebereitschaft für Zusatzfunktionen)

## Die fünfte Säule: Integration in den Betrieb

- Integration umfasst hier sowohl Fragen der technischen Integration als auch der Integration in den betrieblichen Ablauf
  - ◆ ... und damit Fragen, die für den Erfolg des neuen IT-Systems (und damit des Projektziels) von entscheidender Bedeutung sind
- Stichworte dazu
  - ◆ Die Integration muss von Beginn an mit geplant werden
  - ◆ Die isolierte Betrachtung des neuen IT-Systems "nur für sich" mag manchmal berechtigt sein, auf längere Zeit führt sie zum Scheitern
  - ◆ Der "Mehrnutzen" eines neuen Systems und große Teile der "RoI" (Return of Investment) resultieren idR erst durch Integration
  - ◆ Nur Integration sichert den langfristigen Erfolg
  - ◆ EAI - Enterprise Application Integration



## Frühe Fragen zur Integration

- **Geschäftsprozesse, die das neue IT-System betreffen**
  - ◆ Welche Geschäftsprozesse sollen mit dem neuen IT-System abgewickelt werden?
  - ◆ Welche Daten (oder Prozesse) anderer IT-Systeme sind evtl. davon berührt? Was folgt daraus für die Definition der Schnittstellen?
  - ◆ Welche anderen IT-Systeme könnten von Daten oder Prozessen des zu schaffenden IT-Systems profitieren? Folgen für die Schnittstellen?
  - ➔ So ergeben sich erste Aussagen zu den notwendigen Schnittstellen
  
- **Organisatorische Fragen**
  - ◆ Welche der o.g. Systeme werden zu welchen Zeiten zur Verfügung stehen, welche sind also wann mit dem neuen IT-System zu verbinden?
  - ◆ Welche Organisationseinheiten (OEn) sollen mit dem neuen IT-System arbeiten?
  - ◆ Wie werden die betroffenen OEn auf das neue System vorbereitet?
  - ➔ Ein Ziel ist es, einen ersten groben Integrationsplan zu erstellen

## Integration in den Betrieb: Bekannte Probleme

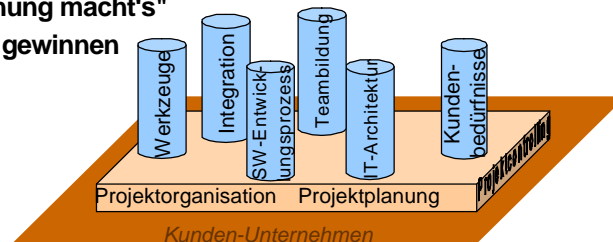
- **Schnittstellen funktionieren nicht**
  - ◆ Frühe Stufen von Integrationstests mit Vorversionen
  
- **Entwicklung und Testarbeiten laufen ohne Beteiligung der Benutzer**
  - ➔ Entsprechende Einbeziehung organisieren, es gibt viele Möglichkeiten !
  
- **Benutzer erhalten keine oder zu geringe Schulung**
  - ➔ Test-Benutzer zu Trainern oder Einführungs-Coaches ausbilden
  
- **Management weiss zuwenig über Auswirkungen des neuen IT-Systems**
  - ➔ Aufklärung, Nutzen erneut herausarbeiten, Promoter im Mgmt. gewinnen
  
- **Neues System wird nicht benutzt**
  - ➔ Benutzern stressfreie Systemnutzung ermöglichen (Fehlerkorrekturen, ohne Termindruck arbeiten, genügend Kapazitäten, korrekte HW, ...)
  - ➔ Nutzen verdeutlichen, Promotoren, Managementebene des Kunden
  
- **Neues System integriert sich nicht in den Betrieb**
  - ➔ Benutzer-Arbeitskreis, ggfs. Change Requests, Optimierung

## Integration in den Betrieb: Best Practices

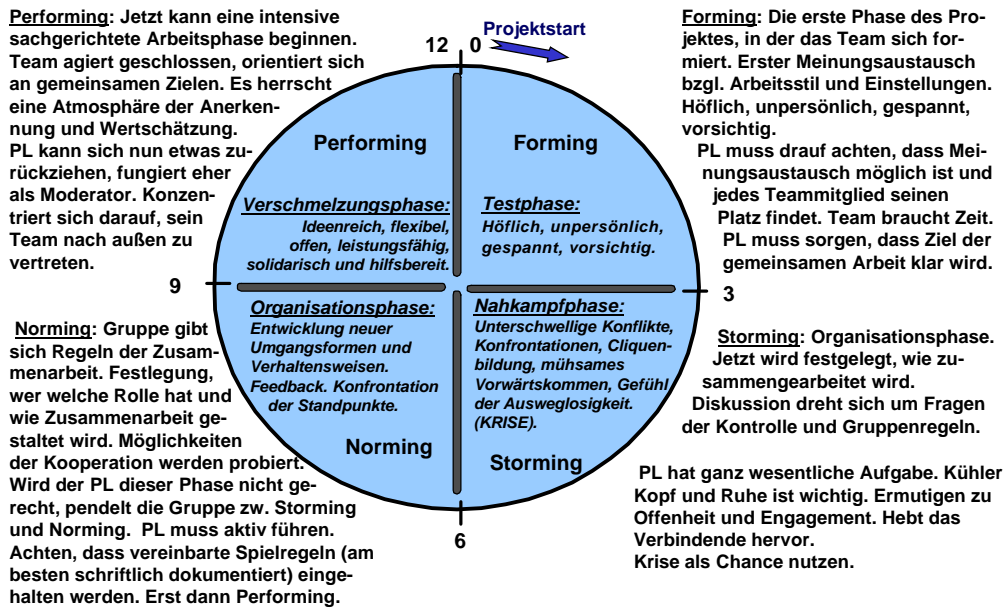
- **Integration mit anderen Systemen früh ausprobieren !**
  - ◆ Der Integrationstest dient ja explizit zum Testen von Schnittstellen mit anderen Systemen
- **Keine "Late-Integration" !**
  - ◆ "Späte Integration": Erst das ganze neue System funktionstüchtig machen, erst danach an andere externe Systeme anbinden
  - ◆ Late-Integration geht fast immer schief, denn auch andere Systeme brauchen den Integrationstest, brauchen Vorlaufzeiten vor Echteintritt ...
- **Mit frühen Versionen Teilnutzen anstreben**
  - ◆ So wird die Integration quasi im Echtbetrieb getestet, ohne schon die volle Funktionalität bieten zu müssen
- **Nicht unbedingt alles fest verkoppeln**
  - ◆ Völlige Integration in direkter Abhängigkeit ist nur solange gut, wie alles 100% funktioniert
  - ◆ Je anfälliger das System, je häufiger spontane Änderungen, je differenzierter der Betrieb, umso kritischer ist die Totalintegration

## Die sechste Säule: Teambildung und Teamentwicklung

- **Teambildung und Teamentwicklung umfasst sowohl die einzelnen Rollen im Projekt als auch den Teambildungsprozess**
  - ◆ Die Rollen sind mit Verantwortlichkeiten verbunden
  - ◆ Das Verhältnis Projektleiter ↔ Projektteam verdient große Aufmerksamkeit
- **Stichworte dazu**
  - ◆ Ohne Rollenverteilung sind die Projektherausforderung oft nicht zu meistern, ohne sie entwickelt sich auch kein richtiges Team-Spiel
  - ◆ Verantwortung übertragen bedeutet auch, jemanden zu vertrauen
  - ◆ "Die (richtige) Mischung macht's"
  - ◆ Nur das Team kann gewinnen



## Die Projektuhr - Vier Phasen der Teamentwicklung



Die 7 Säulen des IT-Projektmanagement,  
Vortrag FG IT-Projektmanagement\_19032004 - Folie 27



© Steinbeis-Transferzentrum IT-Projektmanagement

## Teambuilding: Pflichten, Verantwortung, Vertrauen

- **Starker Zusammenhang aller drei Begriffe**
  - ◆ Verantwortung (für ein Arbeitspaket) wird nur übernommen, wenn Mitarbeiter Vertrauen des PL verspürt
  - ◆ Vertrauen entsteht nur, wenn gewisse Pflichten eingehalten werden
  - ◆ Der PL hat zwar Gesamtverantwortung, muss aber Teile auf die Mitarbeiter übergeben können
  - ◆ PL muss auch LA und Kunden vertrauen, dass sie Ihre Pflichten einhalten
  - ◆ Im positiven Fall verstärken sich die drei Faktoren gegenseitig
- **Alle im Projektteam sollten (müssen ?) ...**
  - ◆ ... Mut haben, Neues zu lernen (über Technik, Menschen, Fachlichkeiten)
  - ◆ ... Verantwortung übernehmen (über Arbeitspakete, fachliche Korrektheit, technische Lösungen, Teil-Budgets, die Selbst-Steuerung)
  - ◆ ... manche Pflichten auf jeden Fall erfüllen (Stunden-Ist-Schreibung, Probleme so früh es geht dem PL mitteilen, uvm.)
  - ◆ ... Lust haben auf die Herausforderungen des Projekts

Die 7 Säulen des IT-Projektmanagement,  
Vortrag FG IT-Projektmanagement\_19032004 - Folie 28



© Steinbeis-Transferzentrum IT-Projektmanagement

## Teambildung: Bekannte Probleme

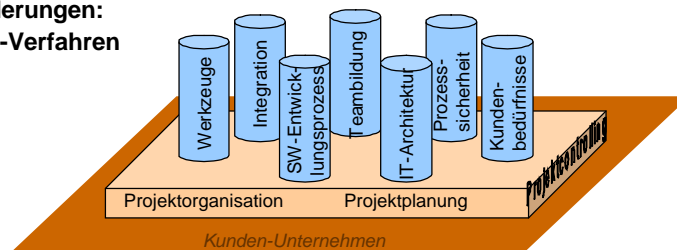
- Einzelne im Team übernehmen keine Verantwortung
  - ➔ Überprüfen der Motivation ("Geschichte"), stimmt das Vertrauen?
- Streitigkeiten im Team führen zu einer Blockade
  - ➔ Stimmt die gemeinsame Zielauffassung, das Projektverständnis?
  - ➔ Ggfs. die Storming-Phase noch nicht überwunden?
- Teammitglieder werden von anderen Teammitgliedern nicht informiert
  - ➔ An die Verantwortung einzelner im Team appellieren, ggfs. vermitteln
- "Verrat am Team" durch PL oder Teammitglied
  - ◆ ...eines der sichersten Mittel, ein Team zu (zer)stören
- Kunde fällt Verständigung mit dem Team schwer
  - ◆ Idealzustand ist, dass sich Kunden-MA selbst als Team-Mitglied sieht
  - ➔ Kunden verdeutlichen, dass nur gemeinsam Win-Win erreichbar
  - ➔ Team verdeutlichen, dass der "Kunde DER König ist", vgl. Kap. 2

## Teambildung: Best Practices

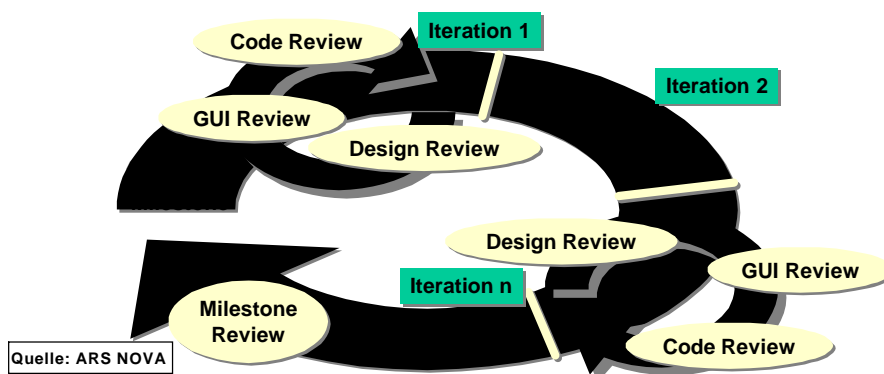
- Teams aus Personen mit unterschiedlichen Fähigkeiten bilden
  - ◆ Ein Team mit MAn der gleichen Art ist oft unkreativ und weniger offen
- Team-Mitarbeiter in eigenen Projekträumen organisieren
  - ◆ Räumliche Nähe schafft viel indirekte Kommunikation
  - ◆ Ohne Nähe kein Storming, aber dann auch kein Performing
- Teambegegnungen nicht nur bei der Arbeit
  - ◆ "Freizeit-Veranstaltungen" schätzt allerdings nicht jeder Mitarbeiter
- Auf äußere Widerstände gemeinsam reagieren
  - ◆ Einer der positiven Faktoren während des Storming
  - ➔ MA in geeigneter Form in Problemstellungen miteinbeziehen
- Anerkennung der Leistungen jedes Einzelnen
  - ◆ ...aber auch der des Teams

## Die siebte Säule: Prozess-Sicherheit und Optimierung

- Den Entwicklungsprozess schnell und sicher vorwärts treiben
  - ◆ Arbeitspakete zügig abschliessen, Qualität sichern, Prozess optimieren
  - ◆ Sicherheit durch einfaches Planen, Ist-Erfassen, Controlling
- Stichworte dazu
  - ◆ Beispiele von effektiven Qualitätssicherungsverfahren
  - ◆ Grundlagen zu Aufgabenplanung, Arbeitspakete und deren Risiken
  - ◆ Ressourcenplanung und deren Risiken
  - ◆ Aufwandsschätzungen, Abnahmen, Meilensteine
  - ◆ Steuerung von Aufwänden, kontinuierliches Controlling
  - ◆ kontrollierte Änderungen: Change-Request-Verfahren



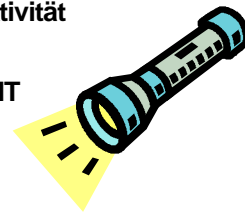
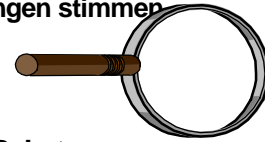
## Qualitätsmanagement im Spiralmodell: Der Review-Zyklus



- Review zeitnah bei jeder Version und zu jedem Artefakt
- Steigende Qualität insbesondere durch steigende Prozesserfahrung
- Frühes Kunden-Feedback zu UI-Storyboard und neuen Versionen
- Reduzierung der Zahl der Artefakte/Dokumente, Konzentration auf diejenigen Artefakte, mit denen wirklich gearbeitet wird

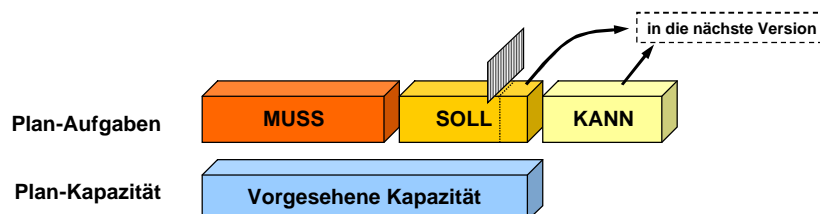
## Risikobetrachtung: Rolle der Arbeitspakete

- Durch Herunterbrechen auf Arbeitspakete (AP) sollen **SÄMTLICHE** Projektarbeiten abgedeckt werden => auf **Vollständigkeit** achten
  - ◆ Risiko innerhalb eines Paketes: Voraussetzungen stimmen nicht, Mittel (z.B. Tools) funktionieren nicht, Skill ungenügend, uvm. => **LUPE**
- Zur Abarbeitung jedes Paketes gehört die Abnahme bis zum **100%igen** Abschluss des Paketes
  - ◆ „die letzten 5 % dauern meistens am längsten“
  - ◆ Abnahme: dem Projektmitarbeiter die Verantwortung für das AP abnehmen
  - ➔ Mitarbeiter damit **BEFREIEN** für die nächste Aktivität
- Risiken für das Gesamtprojekt
  - ◆ Die größten Planungsprobleme entstehen **NICHT** durch diejenigen Pakete, die beschrieben sind, sondern durch diejenigen, die **FEHLEN !!**
  - ◆ Projektleiter braucht die Taschenlampe !!



## Timeboxing: Steuerung der Aufwände über Versionen

- Voraussetzungen zur Steuerungs-Möglichkeit:  
Die Arbeitspakete für eine Version sind in **MUSS-, SOLL- und KANN-Funktionen** kategorisiert -  
(**MUSS-Funktionen** werden für die nächste Version garantiert !)  
und der Anteil der **MUSS-Funktionen** ist deutlich kleiner als 100%
- Bei Mehraufwänden ggfs. Verschiebung von **SOLL-Funktionen** in die nächste Version,  
**ABER** beachten, welche Funktionen zusammengehören !
- Mögliche Vereinbarung: Der vorgesehene Termin wird auf jeden Fall gehalten, notfalls gibt es Streichungen bei **SOLL- und KANN-Funktionen**



## Prozess-Sicherheit: Bekannte Probleme

- **Keine Versionen, sondern Wasserfall-Modell**
- **Aller Anfang ist schwer: Zu viel Zeit, bis erste Version fertig**
  - ◆ Entwickler ermutigen, Erstversion klein halten, ggfs. Experten zuziehen
- **Qualitätssicherungs-Verfahren führen anfangs zu Verunsicherung**
  - ◆ An kleinen Beispielen zeigen, von Effektivität überzeugen
- **Planung ändert sich ständig bzw. ist zu unsicher**
  - ◆ Rollierende Planung, nicht zu fein, Detailplanung ggfs. Delegieren
  - ◆ Ehrliche Ist-Daten-Erfassung und ständiger Abgleich
  - ◆ In der Regel kommt es weniger auf strenge Reihenfolge an (Excel genügt oft statt MS Project)
- **Meilenstein-Termine rutschen**
  - ◆ Wenn es knapp wird, Umfang reduzieren, von Beginn an so planen
- **Zu viele Change-Requests belasten die Ressourcen / Termine**
  - ◆ Kunden Gefährdungspotential klar machen, CRs ggfs. "dämpfen"

## Prozess-Sicherheit: Best Practices

- **Regelmäßige QS-Maßnahmen an kleinen Paketen**
  - ➔ fördert eine Kultur der "sauberen Abschlüsse"
  - ◆ So planen und darauf drängen, dass Pakete abgeschlossen werden
- **Ständige Aktualität der Planung, verbunden mit Ist-Aufwänden**
  - ➔ Zusätzliche Restzeit-Schätzungen ergeben ständig klaren Stand
- **"Vorsichtige" (konservative) Ressourcenplanung, diese mit Ist-Leistungen vergleichen**
  - ◆ Auf feste Kapazitätszusagen drängen
  - ◆ Der PL muss Mittel haben, damit die Zusagen auch eingehalten werden
- **Iterative Vorgehensweise, Erstellung von Versionen**
  - ◆ Spätestens ab Version 2 weiß jeder, wie es funktioniert, worauf zu achten ist, wie geschätzt werden muss usw.
  - ➔ wachsende Erfahrung und Sicherheit
- **Systematisch Risiken einkreisen und verringern**
  - ➔ ... "Lupe" und "Taschenlampe" für den PL

## IT-Projektmgmt.: Aufgaben und Qualifikation

- **Grundlagen von Projekt-Organisation, -Planung und -Controlling**
  - ◆ Standard-"Werkzeuge", die ein PL selbstverständlich beherrschen muss
- **Übersichtswissen IT**
  - ◆ Notwendig bei Architekturentscheidungen, Werkzeuge, Schnittstellen u.ä.
  - ◆ Kompetenter Gesprächspartner für Experten
  - ◆ Unverzichtbar z.B. zum Risikomanagement
- **Erfahrungen in Prozess-Abwicklungen**
  - ◆ Sowohl auf Auftraggeber- (Kunden-)seite, als auch auf Auftragnehmer-Seite lauern viele alltägliche Stolpersteine
  - ◆ Notwendig: Routine, aber auch Mut, Zähigkeit, Konsequenz, "Schläue"
- **Soziale Kompetenzen**
  - ◆ Teamführung einer der wichtigsten Erfolgsfaktoren, Empathie (Kunden ...)
  - ◆ "Richtige" Kommunikation:  
Zuhören, Verstehen, Klar machen, Orientieren, Begeistern
- ➔ **Breite Querschnitts-Fähigkeiten gefordert**