

Bausteine	Motivation	Ausgewählte Bestandteile	Spezielles Knowhow in der Tetrasoft	Besondere Erfahrungen
Nur ein Genie überblickt alles, kleine Geister arbeiten iterativ.				
Iteratives Vorgehen	Vermindert Projektrisiken und erlaubt konkurrenzfähiges «Time to Market».	Jede Iteration kann folgendes beinhalten: Anforderungsaufnahme, Analyse, Design, Implementierung, Test und Auswertung. In frühen Iterationen werden die grössten Projektrisiken gezielt thematisiert.	Wir sind alle unsere Projekte der letzten Jahre iterativ angegangen.	Umfangreiche Aufgaben sind vielschichtig. Man überblickt sie nicht auf Anhieb. Konzeptionelle Fehler können in frühen Iterationen erkannt werden. Anforderungen ändern sich. Lösungen müssen verfeinert werden. Code will optimiert sein. Iterative Ansätze können damit besser umgehen.
Ich muss wissen was ich brauche bevor ich sehe was ich habe.				
Requirements Engineering	Software-Projekte ohne klare Ziele und Anforderungen scheitern häufig.	Ziele formulieren. Stakeholder identifizieren. Anforderungen aufnehmen. Erhebungstechniken Use Cases LoFi Prototyping Anforderungen verwalten.	Mitarbeit im Requirements Engineering für diverse Projekte. Ausbildung mit SAQ Zertifikat.	Stellenwert in Projekten ist oft noch zu klein. Zusammenarbeit mit Stellen ausserhalb des Projektteams ist manchmal schwierig. Das richtige Mass finden. In manchen Projekten fehlen wichtige Anforderungen. Andere Projekte ‚versinken‘ in einer Flut von Detailanforderungen.
Das soll mal einer unter einen Hut bringen!				
Software-Architektur	Lösung soll dem Problem angepasst sein. Lösung braucht ein adäquates Gerüst. Wenn die Architektur steht, kann die Arbeit verteilt werden.	Horizontale Gliederung – Schichten mit zunehmender Abstraktion. Vertikale Gliederung – Isolierte Dienste (Services), die sich durch alle Schichten durchziehen. Variation Points. Abhängigkeiten über «Interfaces» auflösen. Testpunkte für automatisierte Tests in allen Schichten. Metadaten-basierte Produktversionen.	Architekturverantwortung in Kundenprojekten. Erfahrung mit umfangreichen metadaten-basierten Systemen. Asynchrone (ereignis-orientierte Architekturen). Verteilte Systeme	Es gibt selten nur eine richtige Architektur. Man muss sich auch entscheiden können. Wenn man sich für eine Architektur entschieden hat, soll man diese konsequent durchzuziehen.



Bausteine	Motivation	Ausgewählte Bestandteile	Spezielles Knowhow in der TetraSoft	Besondere Erfahrungen
Der Teufel liegt im Detail.				
Software-Design	Gutes Design ist entscheidend für Qualität, Effizienz und Wartbarkeit.	Funktionsblöcke klar identifizieren und abgrenzen. High cohesion, low coupling. Komponenten-Framework. Funktionsblöcke sollen zu Testzwecken durch Stubs und Testtreiber ersetzt werden. Design Patterns	Wir haben in unterschiedlichsten Projekte verschiedenste Designansätze mit Erfolg eingesetzt. Dies bildet einen wesentlichen Teil unseres Erfahrungsschatzes.	Oft zeigt erst die Erfahrung im Projekt, welches Design sich langfristig bewährt. Gutes Design muss robust mit Datenschema-Migrationen umgehen können. Fehler in frühen Produktversionen lassen sich später nur schwer korrigieren.
Das Handwerk will gelernt sein.				
Implementierung	Saubere Programmierung ist immer noch ein wichtiger Qualitätsfaktor	Moderne Programmiersprachen und Tools	C#, .NET, Java, C++, Samlltalk	
Passt das zusammen?				
Configuration Management	Ohne Configuration Management geht es selbst in kleinsten Projekten nicht mehr.	CM Tools	Komplexe Projekte mit parallelen Entwicklungs-Branches.	Die Grenzen von CM-Tools zeigen sich oft erst wenn das Projekt tausende von Quelldateien und viele Megabyte an Daten verwalten muss. Dann noch umzusteigen ist schwierig. Die Ablagestruktur grosser Systeme muss manchmal verändert werden. Hier soll das CM-Tool Unterstützung bieten.
Gestern hat's noch funktioniert				
Change und Bug Management	Änderungen müssen koordiniert und geplant sein.	Change und Bug Management Tools	Erfahrung aus diversen Projekten.	Ist zusammen mit dem Anforderungs Management eine wichtige Basis des Projektmanagements. Change und Bug Reports müssen laufend priorisiert werden. Eng mit dem iterativen Vorgehen gekoppelt.

Bausteine	Motivation	Ausgewählte Bestandteile	Spezielles Knowhow in der TetraSoft	Besondere Erfahrungen
Täglich frisch gebacken.				
Buildumgebung – Daily Build	Das Produkt ist täglich aktuell und vollständig verfügbar.	Vollautomatisierte Build Tools Täglicher Release für automatische Tests.		Basiert auf gutem Configuration Management.
Bei uns läuft es – läuft es auch beim Anwender?				
Deployment	Die Software muss problemlos installierbar sein.	Tools wie InstallShield, Wise Plattformspezifika wie MSI		Softwareupgrade ist meist komplexer als die erste Installation. «Write once – test everywhere». Den unterschiedlichen Zielplattformen Rechnung tragen.
Im Laufe der Zeit sammelt sich Strandgut an.				
Wartung, Refactoring	Die «Unordnung» nimmt zu – es muss stets aufgeräumt werden. Performance Bottlenecks zeigen sich und müssen angegangen werden. Ursprüngliche Designansätze sind hinderlich für die Weiterentwicklung. Eingeplante Flexibilität wird doch nicht benötigt - Vereinfachungen sind möglich. Lösungen sind zu starr- Flexibilität muss eingebaut werden.	Code Refactoring und; Subsystem Redesign. Manchmal ist sogar die Überarbeitung von Teilen der Architektur sinnvoll. Softwaremetriken beleuchten Schwachstellen in Architektur und Design.	Erfahrung aus diversen Projekten.	Zeit dafür ist oft schwer zu finden. Notwendigkeit wird vom Management manchmal nicht anerkannt. Refactoring steigert Lebensdauer und Ertrag des Produktes. Moderne Refactoring Tools bieten wertvolle Unterstützung.
Fehlerhafte Software schadet dem Ruf.				
Test	Nachbesserungen beim Kunden sind teuer. Je früher ein Fehler entdeckt wird, desto kleiner ist der Schaden.	Use Cases ergeben Testfälle. Reviews, Integrationstests, Systemtest, Regressionstests.		Das Testteam muss rechtzeitig rekrutiert und geschult werden.



Bausteine	Motivation	Ausgewählte Bestandteile	Spezielles Knowhow in der TetraSoft	Besondere Erfahrungen
Schnell und zuverlässig.				
Testautomation	<p>Automatische Tests überwachen die Qualität während der gesamten Entwicklungszeit. Regressionstests erkennen unbeabsichtigte Veränderungen im Verhalten des Produkts</p>	<p>Automatisierte Unit-, Integrations und Systemtests. Tests auf verschiedenen Zielplattformen.</p>	<p>Breite Anwendung von Unit Tests ist Standard. Entwicklung der Infrastruktur für automatisierte Systemtests.</p>	<p>Setzt auf den in den Architektur vorgesehenen Testpunkten auf. Umfangreiche automatisierte Tests brauchen viel Unterhaltsaufwand. Automatisierte Systemtests lassen sich leichter in die Breite (über verschiedene Datensätze) als in die Tiefe (unterschiedliche Funktionalität) erweitern.</p>

