

Denkanstöße, Folge 1

Staunen, jenseits der Logik...

Erbschaften sind eine feine Sache, zumindest meistens. Wir Softwareentwickler nutzen ein ganz außergewöhnliches Erbe für unsere Arbeit: Das Gehirn. Die moderne Neurologie und Psychologie liefern immer wieder neue Erkenntnisse, warum unser Oberstübchen so genial funktioniert, aber auch zum Risikofaktor werden kann.

Die meisten Softwareentwickler wissen zu wenig über das Werkzeug zwischen ihren Ohren, um das Beste daraus zu machen. Dies führt dazu, dass trotz vieler Tools mit allerlei Klick- und Fensterltechnologie und tollen Prozessmodellen bei Softwareprojekten nicht immer das Gewünschte herauskommt.

Die Bedeutung von gehirngerechten und damit menschengerechten Tools, Methoden, Prozessen und vor allem von emotionalen und zwischenmenschlichen Faktoren wird in der "rationalen" Welt der Technik gerne unter den Teppich gekehrt, oder besser gesagt mit Code zugedeckt. So wird viel Produktivität, Innovationskraft und Spaß an der Arbeit vernichtet. Da es sich bei der Produktivität immerhin um den Faktor zehn und mehr handeln kann, lohnt es sich, hier etwas Gehirnschmalz zu investieren.

Was tun? Die erste und wichtigste Maßnahme: Beobachten Sie sich und Ihre Kollegen. Wo knirscht und knackt es in den Projekten? Wo steckt das häufigste Konfliktpotential in der täglichen Arbeit? Wann läuft es besonders gut? Schauen Sie ganz bewusst durch die menschliche statt durch die betriebswirtschaftliche oder technische Brille! Sie werden viele interessante Entdeckungen machen, auch wenn Sie kein Psychologe oder Gehirnforscher sind.

Es wird Ihnen bewusster, wie stark auch die Gehirnfunktionen, die nichts mit analytischem, faktengetriebenen Denken zu tun haben, Ihre Arbeit bestimmen: Emotionen und Intuition.

Finden Sie Wege, um die bisher ungenutzten Schätze unseres Erbes zu heben, aber auch Risiken zu erkennen und zu vermeiden. Wir helfen Ihnen gerne dabei!

Probieren Sie es! Sie werden staunen, was es alles jenseits von Hardware, Software und Logik zu entdecken gibt.

Ich freue mich auf Ihre Denkanstöße unter denkanstoss@microconsult.de.

Peter Siwon

7 Tipps für gehirngerechte Softwareprojekte

Warum steht uns unser Gehirn trotz guter Vorsätze häufig bei der Teamarbeit im Weg? Die sieben häufigsten Fallstricke und erfolgreiche Strategien dagegen stellt Coaching-Experte Peter Siwon vor, der seit Jahren Softwareentwickler in der Projektarbeit begleitet.

Beobachtung: Projektanforderungen werden sehr unterschiedlich interpretiert und umgesetzt.

Ursachen: Die Interpretation und Priorisierung hängt sehr stark von individuellen Erfahrungen und Neigungen der Menschen ab.

Tipp 1: Vereinbaren Sie gemeinsame Regeln für die Definition und Priorisierung von Anforderungen. Sprechen Sie persönlich über wichtige Anforderungen, um alle verbalen und nonverbalen Kommunikationskanäle zu nutzen. Verwenden Sie alles, was Anforderungen in Bilder oder Geschichten verpackt (Anwendungsszenarien, Analogien, Prototypen, etc). Damit lässt sich das Problem zwar nicht vollständig beseitigen, aber mögliche Widersprüche sind leichter erkennbar. Gute Hilfestellung bieten hier Literatur und Trainings zum Thema Requirements Engineering.

Beobachtung: Projektanforderungen werden vergessen.

Ursachen: Jeder Entwickler hat sein Bild der Lösung im Kopf. Die Gefahr ist groß, dass Anforderungen, die nicht ins Weltbild passen, unbewusst ausgeblendet werden. Bei hoher Komplexität versucht unser Gehirn außerdem, durch Vereinfachung und Filterung den Überblick zu behalten. Dabei gehen oft wichtige Details verloren.

Tipp 2: Führen Sie Prüfschritte und Automatismen ein, die sicherstellen, dass Anforderungen auf dem Weg zur Implementierung nicht verloren gehen. Ideal sind Tools, die ein Requirements Management unterstützen.

Beobachtung: Die Dokumentation ist falsch und lückenhaft.

Ursachen: Der menschlichen Neigung, das Dringende vor dem Wichtigen oder das Angenehme vor dem Unangenehmen zu tun, fällt meist die Dokumentation von Software zum Opfer. Diese Tendenz wird dadurch verstärkt, dass Dokumentation zwar gefordert, aber die notwendige Zeit nicht eingeplant wird. Die überhastete nachträgliche Dokumentation beschreibt nicht selten mehr den Wunsch als die Wirklichkeit.

Tipp 3: Machen Sie sich im Team die Bedeutung guter Dokumentation bewusst, planen und fordern Sie dafür Zeit ein und einigen Sie sich auf verbindliche Regeln (Dateiheader, Kommentare im Sourcecode, Namenskonventionen, etc.). Verwenden Sie Werkzeuge, die das Generieren der Dokumentation unterstützen (Struktogrammgeneratoren, CASE-Tools, etc.) und Sie bei stupider Tipp- und Zeichenarbeit entlasten. Verankern Sie die Dokumentation fest in Ihrem Entwicklungsprozess. Überlegen Sie immer, ob sich ein Programmiertrick auf Kosten der Lesbarkeit lohnt.

Beobachtung: Erosion der Softwarearchitektur.

Ursachen: Den beteiligten Entwicklern fehlt eine klare Vorstellung von Verhalten und Struktur der Software bzw. des Systems. Damit sind sie zwangsläufig auf die Methode Versuch und Irrtum bei der Programmierung angewiesen, die nach und nach die Softwarearchitektur zerstört.

Tipp 4: Wählen Sie Darstellungsformen, die Struktur und Verhalten der Software visualisieren und eine Kapselung von Details erlauben. Das menschliche Gehirn ist extrem leistungsfähig, wenn es um die räumliche Orientierung und Erfassung bildlicher Zusammenhänge geht. Hier bieten sich grafische Notationen (UML, Matlab Simulink, LabVIEW, Visual Basic, etc.) an. Gute Tools erlauben es, verschiedene Detaillierungsebenen zu erzeugen, die die Orientierung zusätzlich erleichtern. Außerdem überprüfen sie die Konsistenz von Struktur und Verhalten. Eine intensive methodische und praktische Schulung sowie projektbegleitendes Coaching stellen den schnellen und effektiven Einsatz von Designmethoden und Tools am besten sicher.

Beobachtung: Es gibt Regeln, an die sich keiner hält.

Ursachen: Der Mensch achtet nur Regeln, die er für sinnvoll und vorteilhaft erachtet. Die Durchsetzung von Regeln durch Strafandrohung führt nur zur formal richtigen aber real sinnlosen Umsetzung. Softwerker können sich dazu sehr gut hinter der Komplexität ihrer Software verstecken und lassen das aus ihrer Sicht bürokratische Management ins Leere laufen. Dies gilt vor allem dann, wenn Regeln nicht verbindlich eingefordert oder von allen nur als Papiertiger abgetan werden.

Tipp 5: Eine Regel ist nur so gut wie ihre reale Akzeptanz und die Möglichkeiten, ihre Verbindlichkeit zu garantieren. Deshalb sollte ein repräsentativer Querschnitt durch den Personenkreis der Betroffenen ein ehrliches Mitspracherecht haben, wenn es um die Diskussion und Festlegung von Regeln geht. Keine Regel darf ohne klare Begründung des Nutzens für den, der sie einhalten muss, ausgegeben werden. Sorgen Sie dafür, dass von Anfang an die Meinungsführer mit einer Mehrheit im Boot sitzen. Stellen Sie die Verbindlichkeit der Regeln sicher. Ideal sind dabei toolgestützte Checker, wie sie z.B. zur Prüfung von Programmierregeln eingesetzt werden können (z.B. LINT, Polyspace, Misra-C Checker, Innovator).

Beobachtung: Hoher Fehlerbehebungsaufwand in der Endphase von Projekten.

Ursachen: Viele Fehler aus frühen Projektphasen, wie Anforderungsanalyse oder Design, werden erst in späten Testphasen gefunden. Damit muss der Entwicklungsprozess teilweise neu aufgerollt werden. Häufig sind die Zusammenhänge zwischen Ursache und Wirkung aufgrund der inzwischen abgelaufenen Zeit selbst für den Entwickler schwer nachvollziehbar, oder der zuständige Entwickler ist nicht mehr greifbar bzw. bereits in einem Folgeprojekt gebunden. Diese Randbedingungen erschweren die Fehlersuche, mindern die Motivation und verhindern das effektive und rechtzeitige Lernen aus Fehlern.

Tipp 6: Der Entwicklungsprozess sollte sicherstellen, dass jeder Entwicklungsschritt auch einen Test- oder Prüfungsschritt nach sich zieht, der zeitnahe Korrekturschritte erlaubt. Je komplexer die Software desto wichtiger sind Maßnahmen, die eine zeitliche Nähe von Ursache und Wirkung sicherstellen. Methoden wie Review, Simulation, toolgestützte statische Analysen oder formale Verifikation sind dabei sehr hilfreich.

Beobachtung: Konflikte im Projekt.

Ursachen: Die größte Gefahr sind Konflikte zwischen Menschen, die auf dem Schlachtfeld des Projekts ausgetragen werden. Sehr häufig entstehen diese Konflikte, wenn nach den Ursachen von Problemen gesucht wird. Vordergründig wird hitzig über die Sache diskutiert, aber jeder spürt, dass es sich um Angriffs- oder Verteidigungsreaktionen handelt, die gegen Menschen gerichtet sind. Unter Stress steigt die Wahrscheinlichkeit, dass wir dabei immer mehr zu störrischen oder aggressiven (kindlichen) Verhaltensmustern neigen.

Tipp 7: Suchen Sie immer nach Lösungen und nie nach einem Sündenbock. Jedem sollte klar sein, dass er Verursacher und auch Löser eines Problems sein kann. Die Grundlagen für dieses konstruktive Verhalten sind Toleranz, Anerkennung und Vertrauen. Sie entstehen z.B. dadurch, dass wir uns auch die Zeit für den Menschen außerhalb aller Projekte nehmen. Kaffeepausen, Mittagessen oder ab und zu ein Bierchen oder eine kleine Unternehmung bieten hier gute Gelegenheiten. In schwierigen Situationen sollten Sie sich nicht scheuen, einen internen oder externen Vermittler hinzuzuziehen, der das Vertrauen aller Beteiligten besitzt. Im Falle immer wiederkehrender Konflikte ist die Einbeziehung eines professionellen Coaches ratsam.

Wir freuen uns auf Ihre Denkanstöße unter denkanstoss@microconsult.de.