3.5 Projektentwicklung/Phase 2: Konzepte entwickeln

3.5.1 Einführung

Das Ziel der Konzeptphase ist es, mehrere unterschiedliche Grobentwürfe, so genannte *Varianten* bzw. *Konzepte*, anzufertigen. Diese sollen anschließend miteinander verglichen werden. Am Ende wird entschieden, welcher der Grobentwürfe weiter verfolgt wird.

Zu Beginn der Konzeptphase liegen vor:

- IST-Zustand
- Pflichtenheft/SOLL-Zustand

Am Ende liegen vor:

- Grobentwürfe, vergleichbar
- Vergleich und Bewertung
- Entscheid

3.5.2 Ideen finden

Ausgehend von den Randbedingungen, die im Pflichtenheft genannt sind, werden zuerst möglichst verschiedenartige Lösungsmöglichkeiten ausgearbeitet. Dabei nutzt man die Freiheiten, die einem das Pflichtenheft bietet, weitgehend aus. So weit die Theorie.

3.5.2.1 Problem: keine Ideen Die größte Schwierigkeit kann darin bestehen, dass man in diesem Stadium nur eine einzige Möglichkeit sieht, das Projekt zu realisieren. Wenn man dann aufgibt, kann es sein, dass man zum Schluss nur zwei sehr ähnliche Varianten erhält, die sich lediglich in Details („nehme SPS der Firma A oder SPS der Firma B“) unterscheiden.

3.5.2.2 Kreativitätstechnik Brainstorming Deshalb ist es sinnvoll, eine oder mehrere *Kreativitätstechniken* einzusetzen. Das sind Verfahren, die es einer Gruppe ermöglichen, Ideen zu finden und zu kombinieren. Die bekannteste Kreativitätstechnik ist das *Brainstorming*:

- Alle Gruppenmitglieder sitzen zusammen vor einer Schreib- und Anzeigefläche (Bildschirm, Flipchart, Tafel).
- Ein Gruppenmitglied hat die Möglichkeit zum Schreiben.
- Das Thema wird an den oberen Rand der Fläche geschrieben.
- Jedes Gruppenmitglied nennt spontan Begriffe oder Sätze, die ihm zum Thema einfallen.
- Die schreibende Person schreibt möglichst in Echtzeit mit.
- Niemand kommentiert oder bewertet eine Äußerung von jemand anderem. Das ist extrem wichtig, weil sonst die Anzahl der Einfälle schnell zurückgeht (das ist empirisch belegt).
- Sobald keine neuen Einfälle mehr passieren, kann das Brainstorming beendet werden.

Erst *nach* dem Ende des Brainstormings sichtet man den Inhalt der Anzeigefläche. Jetzt erst kann man Einfälle sortieren, gruppieren und zusammenfassen.

3.5.2.3 Ideen kopieren Eine weitere Möglichkeit, zu neuen Varianten zu gelangen, besteht darin, andere Projektbeteiligte (Abnehmer, Zulieferer, Nutzer, ...) einzubeziehen und sich von ihren Ideen inspirieren zu lassen.

Oder man sichtet bestehende Lösungen gleichartiger oder ähnlicher Projekte und orientiert sich an ihnen.

3.5.2.4 Kreativitätstechnik Morphologischer Kasten Hat man zwei sehr unterschiedliche Lösungswege, kann man eventuell über eine dritte Variante nachdenken, die beide kombiniert. Das führt zur Kreativitätstechnik des *Morphologischen Kastens*. In dieser Technik benennt man zunächst die Unterschiede der bisherigen Lösungen. Zum Beispiel benutzt die eine Lösung einen Mikrocontroller, der in C programmiert ist. Die andere Lösung benutzt einen Mini-PC mit Python-Programm. Die beiden Unterschiede sind also die Hardware-Plattform einerseits (zwei verschiedene) und die verwendeten Programmiersprachen (auch zwei verschiedene) andererseits. Diese Unterschiede kann man nun als beiden Dimensionen einer zweidimensionalen Tabelle verwenden (siehe Tabelle 1). Weitere Varianten erhält man nun dadurch, dass man ...

	Hardware m.uC	Hardware m.PC	Hardware mit ...
Programmiersprache C	Var.1: uC in C prog.		
Programmiersprache Python		Var.2: PC in Python	
Programmiersprache ...			

Tabelle 1: Morphologischer Kasten

a) die fehlenden beiden Kombinationen füllt (Mikrocontroller in Python programmieren, falls das geht; PC in C programmieren, falls möglich)

b) durch neue Spalten (=Hardware-Plattformen) oder neue Zeilen (=Programmiersprachen) die Tabelle vergrößert

Auch hier sollte man – wie beim Brainstorming – zuerst Ideen (und seien sie noch so abseitig) sammeln und *erst später* ungeeignete Kombinationen aus der Wertung nehmen.

3.5.3 Hilfen zum Finden von Varianten

Sowohl bei der Software als auch bei der Hardware kann man drei Bezugsquellen unterscheiden:

- Vorhanden: Verursacht keine oder verringerte Kosten. Man muss aber nehmen, was man hat (geringe Flexibilität) und eventuell anpassen. Wichtig ist: Kann man im Reparaturfall (Software hat Fehler, Hardware hat Fehler) den Fehler selbst beheben (spricht für Open-Source-Hard- und Software)? Außerdem: Gibt es Software-Updates? Meist gibt es im Unternehmen Kenner dieser Soft- oder Hardware.

- Wird zugekauft: Verursacht mittlere Kosten, man kann unter mehreren Modellen auswählen (mittlere Flexibilität); man erhöht aber vielleicht die Abhängigkeit von einem Zulieferer. Auch hier ist eventuell eine Anpassung nötig, die möglicherweise vom Zulieferer geleistet wird. Im Fehlerfall kann man sich (je nach Absprache) an den Zulieferer wenden. Updates werden (auch je nach Absprache) vom Zulieferer bereitgestellt. Bei Open-Source-Software ist beides häufig gewährleistet.

- Wird selbst gefertigt: Verursacht höhere Kosten. Man kann die Hard- oder Software jedoch exakt so zuschneiden, wie man sie braucht. Eine Abhängigkeit von einem Zulieferer besteht nicht. Allerdings muss man im Reparaturfall den Fehler selbst beheben. Auch Updates müsste man selbst erstellen.

In allen drei Fällen kann man unterscheiden, wieviel man bekommt:

- Open-Source-Soft- und Hardware: Wird durch unbezahlte Experten einer freiwilligen Entwicklergemeinschaft (*community*) erstellt. Jeder hat das Recht (je nach Lizenz), die Hard- oder Software einzusehen (Quelltext oder Entwurf), sie beliebig weiterzuentwickeln, abzuändern, die geänderte Fassung weiterzugeben usw. In manchen Fällen spalten sich Teile der Community ab und entwickeln das Produkt anders weiter: Es entsteht ein *fork* (=ein Seitenzweig) der Entwicklung, meistens mit einem neuen Namen.

Manche Open-Source-Software wird durch ein Unternehmen entwickelt. Daran ist zunächst nichts Besonderes; es kann dann aber der Fall eintreten, dass das Unternehmen – angeregt durch den Erfolg des Produkts – damit Geld verdienen möchte. Auch dann entsteht ein *fork*: Auf der einen Seite steht die Bezahlversion, die durch das Unternehmen fortentwickelt wird (meistens unter dem Namen der bisherigen Open-Source-Version); auf der anderen Seite steht die Open-Source-Variante, die dann in der Regel unter neuem Namen von einer *community* gesichert und ebenfalls weiterentwickelt wird.

- Proprietäre Soft- und Hardware: Wird durch bezahlte Experten eines Unternehmens erstellt. Das Unternehmen hat alle Rechte (außer, es wurde anders vereinbart) und kann das Produkt abkündigen und vom Markt nehmen. Dann gibt es nach einer gewissen Zeit keine Ersatzteile, keine Fehlerbehebung, keine Sicherheits-Updates und keine neuen Versionen mehr (vgl. alte Versionen bestimmter prorietäter Betriebssysteme!!!). Für ein System aus mehreren Elementen (Hardware, Betriebssystem, Steuerungssoftware) kann das das Ende bedeuten.

  Eine besondere Schwierigkeit tritt dann auf, wenn eine proprietäre Software auch noch ein **proprietäres Datenformat** benutzt. Dazu gehören etwa undokumentierte Dateiformate und undokumentierte Datenbank-Tabellen. Dann können mit jeder Umstellung des Produktes (neue Version, Abkündigung) durch das Unternehmen große Probleme auftreten. Es kann dazu führen, dass nicht nur unser System nicht mehr verwendbar ist, sondern auch unsere bisherigen Daten nicht mehr. In der Folge müssen alle Daten von Hand abgeschrieben und neu eingegeben werden: Eine solche *Datenmigration* kann sehr teuer und zeitraubend sein (manche Firmen benutzen nur deshalb völlig veraltete proprietäre Software jahrzehntelang weiter!).

Was kann man noch unterscheiden? Zunächst die Häufigkeit der benutzten Produkte:

- Universal-Hard- oder Software, die man anpasst. Ein Beispiel sind Standard-PCs, Standard-Betriebssysteme und Büro-Standard-Software. Jeder meint sie zu kennen und damit umgehen zu können. Expertise ist weit verbreitet. Allerdings haben sie viel zu viele Möglichkeiten, von denen man nur wenige braucht. Und um sie für ein Projekt nutzen zu können, muss man sie doch wieder von Hand anpassen, z. B. durch Makros bei Büro-Standard-Software. Universal-Hard- oder Software ist aufgrund der Anbieter-Konkurrenz meistens sehr preisgünstig oder sogar (bei Software) kostenlos.

- Branchen-Hard- oder Software. Ein Beispiel sind SPS, Steuerungssoftware oder z. B. Software für Steuerberater. Viele in der Branche kennen sie und können damit gut umgehen. Die Möglichkeiten sind groß, aber nicht größer, als es die Branche benötigt. Eine Anpassung ist meistens nicht nötig, weil die Spezialaufgaben der Branche bereits fest programmiert sind. Branchen-Hard- oder Software ist in der Regel nicht günstig, aber bezahlbar. Open-Source-Software-Produkte gibt es hier oft nur in einer oder zwei Ausführungen.

- Individual-Hard- oder Software. Das sind Lösungen, die auf einen Kunden oder ein Produkt zugeschnitten sind. Nur die Entwicklergruppe kennt sich damit aus – und natürlich die Nutzer. Eine Anpassung ist nicht nötig. Individual-Hard- oder Software ist in der Regel sehr teuer, wenn sie nicht aus Universal- oder Branchen-Elementen zusammengebaut ist. Ein Open-Source-Software-Produkt gibt es hier nur dann, wenn ein Unternehmen dies aus Lizenzgründen freigibt.

Eine letzte Unterscheidung liegt in der Art der Architektur:

- Monolithische Hard- oder Software: Aus vergleichsweise kleinen Bestandteilen wird ein kompakter Block gebaut. Zum Beispiel wird aus Chips und diskreten Bauelementen eine Steuerung gebaut. Oder es wird (aus Funktionen) ein kompaktes C-Programm geschrieben. In sich kann man diesen Block gut verstehen, er ist unabhängig benutz- und wartbar.

  Solch einen Block kann man nur als Ganzes benutzen oder nicht benutzen. Wichtig sind Schnittstellen nach außen, damit man doch noch Teile der Hard- oder Software umnutzen kann.

- Modulare Hard- oder Software: Wenige große Module werden zu einem System zusammengesetzt. Zum Beispiel wird aus Netzteil, SPS, Klemmen und einem Gehäuse ein Schaltschrank zusammengesetzt. Oder es wird aus einem PC mit Betriebssystem, einem Webserver, einer SQL-Datenbank und etlichen Zeilen in der Programmiersprache PHP ein LAMP-System erstellt (Linux-Apache-MySQL-PHP). Die einzelnen Bestandteile muss man dann nicht komplett verstehen, aber das Zusammenwirken zwischen ihnen.

  Diese Variante hat sich in den letzten Jahren immer weiter verbreitet. Man spricht dabei oft von DevOps, weil auch ein Zusammenspiel von Entwicklung (=*development*) und Systemadministration (=*operations*) dabei hilfreich ist.

  Man kann im Prinzip jeden Bestandteil einzeln nutzen. Selbst das Austauschen einzelner Bestandteile (z. B. MySQL gegen MariaDB oder umgekehrt) ist oft möglich.

  Andererseits ist bei dieser Art von Hard- oder Software besonders auf Sicherheitsaspekte zu achten, weil hier das Zusammenwirken der Bestandteile von außen leicht angegriffen werden kann.

3.5.4 Vergleichbarkeit: Was gehört zu einem Grobentwurf

Hat man genügend Ideen gesammelt, kann man mit dem Ausarbeiten der Entwürfe beginnen. Das Wichtigste an einem Grobentwurf ist natürlich der grobe Lösungsansatz selbst. Zusätzlich müssen die Grobentwürfe vergleichbar gemacht werden, indem man zu jedem die gleichen Fragen beantwortet, zum Beispiel diese:

a) Wie hoch sind die Kosten dieser Variante (einmalige Kosten, Betriebskosten pro Jahr)?

b) Wie lange dauert es, sie zu realisieren?

c) Welche Hardware wird benutzt?

d) Welche Software wird benutzt?

e) Welche Daten werden benutzt/erzeugt?

f) Welche Netzwerkumgebung wird benötigt?

g) Wann ist die Lösung benutzbar (Verfügbarkeit / Hochverfügbarkeit)?

h) Wie lange ist sie benutzbar (Nachhaltigkeit)?

i) Wie einfach sind Erweiterungen realisierbar (Fähigkeit zum Update)?

j) Wie einfach kann man die Lösung bei Erfolg ausbauen (Skalierbarkeit)?

k) Wie sieht es aus mit der Datensicherheit dieser Variante?

l) Wie lange dauert die Einarbeitungszeit der Nutzer?

m) Welche Vorteile hat die Variante?

n) Welche Nachteile die Variante?

3.5.5 Vergleichen und Ausschluss von Varianten

Während man die Grobentwürfe ausarbeitet, kann es sein, dass schon früh deutlich wird, dass die eine oder andere Variante nicht geeignet ist. Dann kann man sie schon mitten in dieser Phase ausschließen. Oder man merkt, dass zwei Varianten sehr ähnlich sind. Die kann man zu einer zusammenfassen. Zum Schluss sollten auf jeden Fall mindestens zwei bis vier Varianten übrig bleiben, für die man dann den Grobentwurf bis zum Ende ausarbeitet. Sie müssen dann sorgsam verglichen werden; anschließend fällt die Projektgruppe die Wahl einer Variante.

Zum Schluss erfolgt dann – entweder durch die Projektgruppe oder durch den Kunden – die endgültige Auswahl einer Variante, damit endet die Konzeptphase.