Systems Engineering: neu überdacht

Reinhard Haberfellner und Ernst Stelzmann

Abstract— **Das Systems Engineering-Konzept als systematischer Leitfaden zur Gestaltung von Systemen bzw. Abwicklung von Projekten ist nun in seiner Reifephase. Ausgehend von den Arbeiten von A.D. Hall in den Bell-Laboratories wurde es von Mitarbeitern des Betriebswissenschaftlichen Instituts der ETH Zürich aufgegriffen, erweitert, konkretisiert und auch regelmässig neuen Anforderungen angepasst. Reinhard Haberfellner war Mitglied des ETH-Kernteams. Eine englische Version, die ca. 60 Prozent des ursprünglichen Konzepts beinhaltet, wird dzt. vom MIT-Professor Olivier de Weck bearbeitet. Wir an der TU Graz betrachten es als eine Herausforderung, an diesem Prozess aktiv teilzuhaben und insbes. neuere Entwicklungen, die unter dem Begriff „Agile Methods" publik werden, sorgsam zu verfolgen und wenn möglich in das bestehende Konzept zu integrieren. Das vorliegende Papier gibt Einblick in diese Überlegungen.**

Index Terms — **Agile Methods, Problemlösungszyklus, Projektmanagement, Systems Engineering**

I. Das SE-Konzept

DAS SE-Konzept kann kurz wie folgt charakterisiert werden (Haberfellner *et al.* 2002):
SE ist ein Denkmodell und eine Vorgehensmethodik zur Lösung komplexer Probleme. Diese Methodik soll eine systematische und transparente Gestaltung des Problemlösungsprozesses sowie eine effiziente Führung und Abwicklung des Problemlösungsprozesses ermöglichen. Sie soll die Suche nach kostengünstigen, effizienten, sozial verträglichen, termingerechten Lösungen unterstützen. Die Anwendung dieser Methodik ist grundsätzlich vielfältiger Art. Trotz unterschiedlicher Begriffe in verschiedenen Disziplinen bzw. Anwendungsgebieten ist die Grundidee vielfältig einsetzbar. Die Methodik ist aufgeschlossen für die Nutzung verschiedenster Methoden und Techniken, im Sinne eines Werkzeugkastens.

Das SE-Konzept besteht aus folgenden Komponenten (siehe Abb. 1):

Dem *Systemdenken* als Werkzeug zur Darstellung von Wirkungszusammenhängen, zur Strukturierung und Abgrenzung der Ausgangssituation und der Lösungen.

Dem *Vorgehensmodell* als modular aufgebautes Konzept, das auf einfache Art den praktischen Bedürfnissen und der Grösse eines Projektes angepasst werden kann.

Der inhaltlichen Gestaltung eines Systems (*Systemgestaltung*) und den entsprechenden Methoden und *Techniken* der Systemgestaltung, insbesondere jenen der Situationsanalyse und Zielformulierung, der Lösungssuche und der Bewertung bzw. Auswahl von Lösungen.

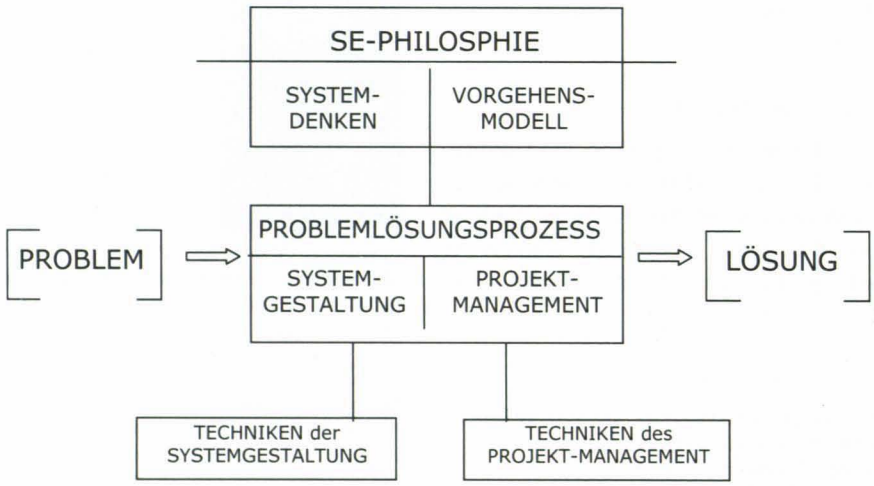

Abb. 1. Das Systems Engineering Konzept

Manuscript received May x, 200y, and accepted August z, 200y, by Prof. Siegfried Vössner.

Dem *Projektmanagement* als Summe von organisatorischen Überlegungen und Massnahmen zur Abwicklung von Projekten und den verschiedenen Methoden und *Techniken*, die deren Einsatz unterstützen sollen, wie z.B. Ressourcenplanung und Budgetierung, die Konfiguration der Projektgruppe, die Kompetenzen des Projektleiters, die Steuerungsgremien, Zeitplanung, Ablaufplanung, Teamarbeit, Konfliktlösung u.a.m..

A. Der Systemansatz

Das Denken in Systembegriffen soll dazu anregen, die Elemente eines Problems und wichtige Einflussfaktoren herauszuarbeiten, zueinander in Beziehung zu setzen und sinnvolle Grenzen für Problemfelder und Lösungen zu finden.

Probleme sollen dabei - vor allem zu Beginn - nicht zu eng gesehen werden, sondern als Bestandteile eines umfassenderen Systems, im Sinne eines ganzheitlichen Denkens. Deshalb ist es sinnvoll, zu Beginn eines Projektes den Horizont zunächst bewusst auszuweiten, um ihn dann aber rasch auf ein bewältigbares Ausmass einzuschränken. Die Blackbox-Betrachtung ermöglicht dabei eine Grobstrukturierung (Überblick wahren), die Unterscheidung verschiedener Betrachtungsaspekte („Brillen") ermöglicht es, differenzierte Standpunkte gegenüber einem Sachverhalt einzunehmen.

Die Idee des ganzheitlichen Denkens ist sowohl auf Probleme, wie auch auf Lösungen anwendbar und lenkt das Augenmerk auch auf Voraussetzungen und Konsequenzen, die erwartet werden können bzw. müssen.

Zusammenfassend und vereinfacht gesagt, sollen folgende Fragen geklärt werden:

- Was ist das System, das wir anschauen? Wie definieren wir es?
- Was sind die Bausteine, Elemente dieses Systems?
- Wie stehen sie zueinander bzw. mit anderen Elementen (Umwelt) in Beziehung?
- Welche(r) Aspekt(e) des Systems sind für uns von Bedeutung? Welche Brille(n) sollten wir benützen?
- Wie grenzen wir unser System von der Umwelt ab? (System: der von uns gestaltete, veränderte Bereich. Umwelt: Jener Bereich, den wir zur Kenntnis nehmen, aber nicht direkt beeinflussen können oder wollen).
- Welche Umwelt ist bzw. Teile der Umwelt sind von Bedeutung?
- Wie sind sie definiert und wie stehen Sie in Beziehung?
- Welchen Teil können/dürfen wir verändern?
- Welche Teile, die Beziehungen aufweisen, haben wir zu beachten?

Ergebnis sollte ein gemeinsames Verständnis darüber sein, worüber wir sprechen, was wir neu gestalten bzw. verändern sollen bzw. dürfen - und was wir als (derzeit?) unveränderbar betrachten.

B. Das Vorgehensmodell

Das SE-Vorgehensmodell besteht aus 4 Komponenten, die sinnvoll miteinender verbunden werden können (siehe Abb. 2):

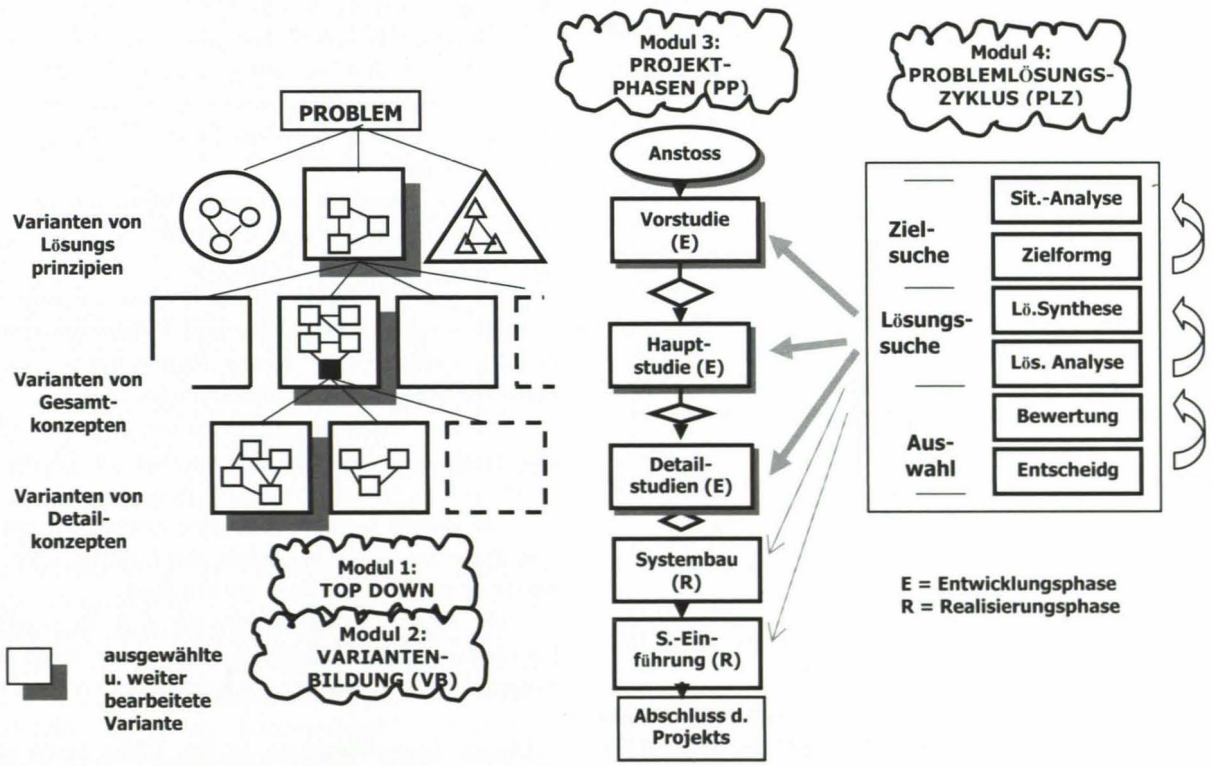

Abb. 2. Das SE-Vorgehensmodell und seine 4 Module

Das Vorgehensprinzip „Vom Groben zum Detail" (Modul 1: top down), womit gemeint ist: Gesamtübersicht vor Detailuntersuchung, Gesamtkonzept vor Detailkonzepten.

Das Prinzip des Denkens in Varianten (Modul 2): Grundsätzlich nicht mit einer einzigen (der "erstbesten") Lösung zufrieden zu geben, sondern nach Alternativen suchen.

Das Prinzip der Gliederung eines Projektes in Projektphasen (Modul 3), demzufolge der Prozess der Systementwicklung und Realisierung nach zeitlichen Gesichtspunkten zu gliedern ist. Damit werden klare Marschhalte, Besinnungs- und Entscheidungspunkte ermöglicht, deren Beachtung für Projektteam und Auftraggeberschaft gleichermassen wichtig ist.

Der Problemlösungszyklus (Modul 4): als sich wiederholende Logik innerhalb der Entwicklungsphasen mit den Schwerpunkten: Zielsuche (Wo stehen wir? Was wollen wir?), Lösungssuche (Welche Möglichkeiten gibt es, dorthin zu kommen?) und Auswahl (welches ist die beste, zweckmässigste?)

C. Zuordnung und Kritik

Das SE-Konzept gehört, wie eine Reihe anderer Methodiken der Gruppe der sog. „plan-driven"-Methods (p) an. Die Unterscheidung zwischen "plan-driven" und "agile" erfolgt in Anlehnung an Boehm *et al.* (2004). Mit „plan-driven" ist gemeint, dass diese ein hohes Mass an Struktur vorgeben und mit der Erwartung verbinden, dass damit auf effizientere Art qualitativ hochstehende Lösungen erarbeitet werden können. Andere Vertreter dieser Gruppe sind z.B. das sog. „Wasserfall"-Modell, das V-Modell, die INCOSE-Methodology oder die IEEE-15288. Der Gedanke des Simultaneous (Concurrent) Engineering ermöglicht es bereits, gewisse agile Eigenschaften in die plan-driven-methods einzuführen.

Diese plan-driven-Methods werden – trotz des unbestrittenen Vorteils, dass sie eine logische Ablaufstruktur in Projekte bringen können – durchaus und zum Teil vehement kritisiert. Insbesondere wirft man ihnen vor, dass sie den (Software)-Entwicklungsprozess unnötig schwerfällig machen, lange Entwicklungszeiten erfordern und im Ergebnis meist doch nicht befriedigen können, weil sie zu einem sehr restriktiven Verhalten gegenüber durchaus berechtigten nachträglichen Änderungen der Spezifikationen führen. Aus diesem Bedürfnis heraus haben sich – beginnend in der Softwareentwicklung - die sog. „agile methods" (a) entwickelt.

II. Agile methods (a) als Alternativen zu den plan-driven methods (p)?

Eine wesentliche Rolle im Zusammenhang mit der Entstehung der Agilen Methoden spielt das sog. Agile Manifesto (bzw. genauer "Manifesto for Agile Software Development"), das in der Folge auszugsweise wiedergegeben werden soll (Beck *et al.* 2001): (Anm.: Übersetzung durch die Verfasser):

„Wir bieten bessere Wege zur Entwicklung von Software an und wollen andere anregen, uns zu folgen. Dabei sind wir zur Auffassung gekommen, dass:

- Personen und deren wechselseitige Beziehungen wichtiger sind als (vorgeschriebene) Prozesse und Werkzeuge

- Funktionstüchtige Software wichtiger ist als umfangreiche Dokumentationen

- Konstruktive Zusammenarbeit mit den Kunden wichtiger ist als langwierige und mühsame Vertragsverhandlungen

- Reagieren auf wechselnde Anforderungen wichtiger ist als das konsequente Verfolgen eines Plans (der eigentlich schon überholt ist)

Obwohl wir den Wert der in obiger Aufzählung jeweils rechts angeführten Aussagen durchaus anerkennen, ordnen wir ihn dem der Aussagen unter, die jeweils links stehen.

Diese generellen Aussagen werden nachstehend präzisiert. Höchste Priorität hat die Zufriedenheit der Kunden. Wir können diese leichter erreichen, wenn

- wir schon sehr bald und kontinuierlich Software abliefern, die der Kunde beurteilen kann und die für ihn brauchbar sind (Prototyping-Ansatz)

- Änderungswünsche nicht als unangenehme Störungen betrachtet werden, sondern gemeinsam besprochen und ggf. akzeptiert werden, auch wenn sie in einem späten Stadium der Entwicklung vorgebracht werden. Durch diese Flexibilität tragen wir dazu bei, unseren Kunden Systeme zur Verfügung zu stellen, die ihre Wettbewerbsfähigkeit fördern

- wir darauf bestehen, dass unsere Kunden mit uns täglich und während der gesamten Dauer des Projekts aktiv zusammenarbeiten. Dabei sollten Sponsoren, Entwickler und Anwender imstande sein, ein konstantes Tempo lange Zeit durchzuhalten

- wir in Projekten motivierte Menschen antreffen und ihnen ein Umfeld und jene Unterstützung geben, die sie für eine erfolgreiche Arbeit benötigen

- wir unseren Teams zutrauen, dass sie ihre Aufgaben schaffen werden und sie dabei nach Kräften unterstützen. Die beste Architektur, die besten Requirements und Designs entstehen in sich selbst organisierenden Teams

- wir uns immer vor Augen halten, dass die effizienteste und effektivste Methode, Information zu übermitteln die persönliche Kommunikation ist (face-to-face)

- uns dessen bewusst sind, dass Einfachheit wichtig und eine Kunst ist, die darin besteht, den Umfang jener Arbeit zu maximieren, die nicht getan werden muss.

- die Teams lernen indem sie sich in regelmässigen Intervallen damit beschäftigen, wie sie noch effektiver werden können – und ihr Verhalten entsprechend anpassen ..."

Dieses Agile Manifesto ist im Klima eines generellen Unbehagens gegenüber den bestehenden, als starr

empfundenen Vorgehenskonzepten entstanden. Gleichzeitig wurde damit die Hoffnung auf raschere Verfügbarkeit von Ergebnissen verbunden, auf Möglichkeiten einer evolutionären Anpassung von Arbeit und Ergebnissen an veränderte Gegebenheiten und Möglichkeiten und nicht zuletzt auf zufriedene Kunden.

Ausserdem ist in diesem Umfeld eine Reihe von sog. Agile Methods entstanden, mit denen versucht wird, die im Agile Manifesto enthaltenen Vorstellungen in die konkrete Projektarbeit zu übertragen. Aus der Vielzahl der Agile Methods sollen einige in der Folge kurz skizziert werden (Wikipedia 17.1.2008):

- *eXtreme Programming (XP)* akzeptiert die Ungewissheit, mit der die Softwareentwicklung verbunden ist. Es folgt einem klaren, strukturierten Vorgehen und stellt Teamarbeit, Offenheit und stetige Kommunikation zwischen allen Beteiligten in den Vordergrund. Kommunikation ist eine Grundsäule dieses Vorgehensmodells. Die Methode berücksichtigt, dass der Kunde die wirklichen Anforderungen an die zu erstellende Software zu Projektbeginn meist selbst noch nicht genau kennt. Damit kann das mit der Realisierung betraute Entwickler-Team gar nicht über alle (technischen) Informationen verfügen, um eine verlässliche Aufwands- und Zeitschätzung zu machen. Im Laufe eines Projektes ändern sich darüber hinaus nicht selten Prioritäten. Zu Beginn geforderte Funktionen der Software werden evtl. in einer anderen Form benötigt oder im Laufe der Zeit sogar hinfällig.

Angefangen mit einer ersten kleinen Version der Software, vergrößert sich der Umfang zusehends. Neue Funktionalitäten werden laufend entwickelt, in die bestehenden Versionen eingefügt und gemeinsam getestet. Um zu der zu entwickelnden Funktionalität zu gelangen, werden gewöhnlich jeweils die Schritte Risikoanalyse, Nutzenanalyse, die Bereitstellung einer ersten ausführbaren Version (exploratives Prototyping) und ein Akzeptanztest durchgeführt. Das Modell lässt sich als agil, iterativ und inkrementell bezeichnen.

Sog. User-Stories (use cases) sind wichtiger Bestandteil der Methodik. Sie beschreiben die Funktionsanforderungen an ein System aus Sicht eines Akteurs und müssen aus deren Köpfen auf Papier gebracht werden. Zu allen User-Stories gibt es ausführliche Tests. Eine User Story ist erst abgeschlossen, wenn alle Tests erfolgreich abgelaufen sind. Der tägliche, kurze Austausch zwischen Entwicklern und Anwendern ist für die agile Methodik üblich.

XP definiert fünf zentrale Werte, die von grosser Bedeutung sind: Kommunikation, Einfachheit, Feedback, Mut und Respekt.

Interessant sind einige der vorgeschlagenen Praktiken des XP: Pair-Programming (2 Programmierer arbeiten gemeinsam, mit verteilten Rollen, die regelmässig getauscht werden: einer ist Driver, der andere Partner); Laufende Integration der einzelnen Komponenten zu einem lauffähigen Gesamtsystem in kurzen Zeitabständen; Testgetriebene Entwicklung bzw. Permanentes Testen; Kurze Iterationen und laufendes Einbeziehen der Kunden; Einfaches Design: die einfachste Lösung, die genau das Gewünschte erreicht, soll angestrebt werden (KISS[1], YAGNI[2]); gemeinsam akzeptierte Standards zur Erleichterung der Teamarbeit u.a.m..

- *Feature Driven Development (FDD)* ist eine Sammlung von Arbeitstechniken, Strukturen, Rollen und Methoden für das Projektmanagement im Rahmen agiler Softwareentwicklung. FDD stellt den Feature-Begriff in den Mittelpunkt der Entwicklung. Jedes Feature stellt einen Mehrwert für den Kunden dar. Die Entwicklung wird anhand eines Feature-Plans organisiert. Eine wichtige Rolle spielt der Chefarchitekt (engl. Chief Architect), der ständig den Überblick über die Gesamtarchitektur und die fachlichen Kernmodelle behält.

FDD-Projekte durchlaufen fünf Prozess-Schritte:

1) Gesamtmodell entwickeln mit dem Ziel, einen Konsens über Inhalt und Umfang des zu entwickelnden Systems sowie das fachliche Kernmodell zu erreichen.

2) Feature-Liste erstellen und Features nach dem Schema Aktion, Ergebnis, Objekt beschreiben,

3) Features planen hins. der Reihenfolge, in der sie realisiert werden sollen. Diese richtet sie sich nach den gegenseitigen Abhängigkeiten der Features, ihrer Komplexität und der Auslastung der Programmierteams. Die Features werden zur weiteren Bearbeitung einzelnen Entwicklerteams zugeteilt.

4) Features entwerfen: Die Entwicklerteams erstellen Sequenzdiagramme für die Features und die Chefprogrammierer verfeinern die Klassenmodelle auf Basis der Sequenzdiagramme. Die Entwickler schreiben dann erste Klassen- und Methodenrümpfe. Schließlich werden die erstellten Ergebnisse inspiziert. Bei fachlichen Unklarheiten können die Fachexperten hinzugezogen werden.

5) Features konstruieren: die Entwickler programmieren die vorbereiteten Features. Dabei werden Komponententests und Code-Inspektionen zur Qualitätssicherung eingesetzt.

Die ersten drei Schritte werden innerhalb weniger Tage durchlaufen. Die Schritte 4 und 5 werden in ständigem Wechsel durchgeführt, weil jedes Feature in maximal zwei Wochen realisiert werden soll.

- *Scrum* (engl. das Gedränge) ist eine Sammlung von Arbeitstechniken, Strukturen, Rollen und Methoden für das Projektmanagement im Rahmen einer agilen Softwareentwicklung. Es enthält wenige Festlegungen. Teams bzw. Entwickler organisieren sich weitgehend selbst und wählen auch die eingesetzten Methoden. Das Vorgehen und die Methoden werden fortlaufend aktuellen Erfordernissen angepasst.

Scrum knüpft an viele Grundannahmen der sog. Schlanken Produktion (engl. lean production) an und überträgt

[1] KISS = Keep it simple and smart (Variante statt smart: stupid)

[2] YAGNI = You Ain't Gonna Need It, d.h. konsequenter Verzicht auf Funktionalitäten die derzeit nicht verlangt werden, aber möglicherweise in Zukunft

Erfahrungen aus der Automobilbranche (Toyota) auf die Softwareentwicklung. Zentraler Aspekt ist bei beiden die ständige Weiterentwicklung der am Prozess Beteiligten, auch der Kunden und Partner, der Herstellungsprozesse, der Arbeitsmittel und Methoden, mit gleichzeitig konstantem Beibehalten der Grundannahmen, die dahinter stehen: die Produktion ständig zu verbessern, um höchste Qualität bei niedrigstem Aufwand zu erreichen.

Zentrales Element von Scrum ist der Sprint, der die Umsetzung einer Iteration bezeichnet. Scrum sieht ca. 30 Tage als Iterationslänge vor. Vor dem Sprint werden die Produkt-Anforderungen des Kunden in einem Product Backlog gesammelt, wobei darunter die Features des zu entwickelnden Produkts zu verstehen sind. Es beinhaltet alle Funktionalitäten, die der Kunde wünscht, zuzüglich technischer Abhängigkeiten.

Es gibt drei klar getrennte Rollen für die Mitarbeiter eines Projekts, die die gleichen Ziele verfolgen sollen:

Der Product Owner legt das gemeinsame Ziel fest, das er und das Team zu erreichen hat bzw. sanktioniert es. Er stellt das Budget zur Verfügung und setzt regelmäßig die Prioritäten für die einzelnen Product-Backlog-Elemente. Damit entscheidet er auch, welche die wichtigsten Features sind, aus denen das Entwicklungsteam eine Auswahl für den nächsten Sprint trifft.

Das Team schätzt die Aufwände für die Entwicklung der einzelnen Backlog Elemente ab und beginnt mit der Implementierung der für den nächsten Sprint machbaren Elemente. Das Team arbeitet selbstorganisiert im Rahmen einer Time Box (dem Sprint), und hat das Recht (und die Pflicht), selbst zu entscheiden, wieviele Elemente des Backlogs nach dem nächsten Sprint erreicht werden müssen, man spricht dabei von sog. commitments.

Der Scrum Master hat die Aufgabe, die Prozesse der Entwicklung und Planung durchzuführen und die Aufteilung der Rollen und Rechte zu überwachen. Er hält die Transparenz während der gesamten Entwicklung aufrecht, und fördert das Zu-Tage-Treten der bestehenden Verbesserungspotentiale. Er ist nicht für die Kommunikation zwischen Team und Product Owner verantwortlich, da diese direkt miteinander kommunizieren sollen. Er steht dem Team zur Seite und soll dafür sorgen, dass das Team produktiv arbeiten kann.

- *Crystal* ist eine ganze Familie von Methoden, die eine Differenzierung hins. der Anzahl der beteiligten Personen und der Höhe der Risiken (sog. Kritikalität) ermöglicht. Die Methoden werden mit Farben benannt: Crystal Clear, Crystal Yellow, Crystal Orange, Crystal Red, Crystal Magenta,

Tabelle 1. Potentielle Eignung von Vorgehensmethoden (Boehm *et al.* 2004)

Kriterium	Eignung von Agile Methods	Eignung von Plan-Driven Methods
Grösse des Projekts	Gut geeignet für eher *kleine Projekte* und Teams. Stützt sich auf sogen. tacit knowledge (in den Köpfen gespeichertes Wissen, nicht explizit dargestellt). Skalierung nach oben damit begrenzt	*Grosse Projekte* und Teams. Downsizing auf kleine Projekte schwierig
Kritikalität (Sicherheit etc.)	Keine Erfahrungen bei Sicherheits-kritischen Produkten vorhanden. Schwierigkeiten zu erwarten bei einfachem design und mangelhafter Dokumentation.	Entwicklung von hoch kritischen Produkten
Dynamik des Umfeldes	Dynamisches Umfeld und einfaches Design, verbunden mit kontinuierlicher Strukturanpassung (refactoring) passen ausgezeichnet zueinander.	Stabile Umwelt gestattet grosse Entwürfe, die dann detailliert ausgeführt werden.
Personal	Erfordert die ständige Anwesenheit von qualifiziertem Personal. Wegen der o.a. Dynamik riskant, wenn nicht verfügbar	Qualifiziertes Personal besonders für die Projekt-Definition erforderlich. Ausarbeitung auch von nicht ganz so hoch qualifiziertem Personal durchzuführen – sofern die Anforderungen stabil bleiben.
Kultur	Gedeiht in einer Kultur, in der Menschen Freude an grossem Gestaltungsfreiraum und Handlungsfreiheit (empowerment) haben.	Gedeiht in einer Kultur, in der Menschen sich gut fühlen, wenn ihre Rollen durch klare Prozeduren und Verhaltensweisen definiert sind.

Crystal Blue. Die einfachste Variante, "Crystal Clear" wird für Teamgrössen von zwei bis sechs Personen empfohlen.

Im Vergleich zu anderen Agilen Methoden wird Crystal von seinen Befürwortern als weniger dogmatisch und formalisiert angesehen. Bei Crystal Clear werden z.B. weder Paarprogrammierung noch customer on site gefordert.

Crystal sieht nicht dauerhafte Methoden für das Team vor, sondern legt sie für jedes Projekt neu fest. Bei einfacheren Projekten kann dies dazu führen, dass viele der auch in XP eingesetzten Agilen Methoden zum Einsatz kommen; bei komplexeren Projekten würden eher komplexere Methoden gewählt.

III. Eignungsbereiche der plan-driven bzw. agile methods

In der Folge soll versucht werden, die „Claims" für die in ihrem Charakter doch recht unterschiedlichen Methoden abzustecken. Dies erfolgt unter der Voraussetzung, dass ein Ansatz nicht ausschliesslich richtig und der andere nicht auschliesslich falsch sein muss, sondern dass es auf den konkreten Zusammenhang ankommt[3].

In Tabelle 1 werden die dominanten Einsatzbereiche der beiden Methodengruppen stichwortartig beschrieben.

IV. Gegenseitige Annäherung möglich?

Im Allgemeinen bestehen zwischen den plan-driven und den agile methods also teilweise grosse Unterschiede, sodass sie auf den ersten Blick miteinander unvereinbar zu sein scheinen. Speziell im Hinblick auf das SE-Konzept soll deshalb der Frage nachgegangen werden, inwieweit dieses tatsächlich im Widerspruch zu agilen Prinzipien steht. Dabei soll identifiziert werden

- wo das SE-Konzept bereits Ansätze zu einer gewissen Agility aufweist bzw. agile Prinzipien unterstützt (A)
- welchen agilen Prinzipien gegenüber sich das SE-Konzept neutral verhält bzw. welche agilen Prinzipien innerhalb des SE-Konzept problemlos möglich sind, auch wenn sie nicht aktiv unterstützt werden (B)
- und wo eine gewisse Unvereinbarkeit zwischen agilen Prinzipien und dem SE-Konzept besteht bzw. wo eine Einbeziehung agiler Prinzipien in das SE-Konzept noch eines Forschungsaufwandes bedarf (C)

A. Bereits bestehende Agility im SE-Modell

Fähigkeit zur Agility ist vor allem in folgenden Eigenschaften des SE-Vorgehensmodells zu erkennen (siehe Abb. 2):

- Die gedankliche Trennung der 4 Module, insbes. des PP- und des PLZ-Moduls ermöglichen eine Anpassung der Methodik an unterschiedliche Projektgrössen: Kleine Projekte brauchen nicht in mehrere Entwicklungsphasen mit zunehmender Konkretisierung (TD) untergliedert werden. Es genügt ein 1-maliges Durchlaufen des PLZ mit unmittelbarem Übergang zur Realisierung (Systembau)
- Wie bei den agile methods gefordert, liegt die Prozess-Verantwortung beim Team. Je nach den Erfordernissen des Projektes, kann der Prozess vom Team selbst angepasst werden, z.B. für unterschiedliche Projektgrößen, wie soeben beschrieben.
- Stetiges Lernen und die Verwendung von Erfahrungen sind im SE-Konzept ebenso wichtig, wie in den agile methods. Das SE-Konzept ist kein stur abzuarbeitender Prozess sondern eine Methodik, die eine Richtschnur darstellen soll, wie Fachwissen, Erfahrung, Methodenwissen, etc. am besten miteinander kombiniert werden können.
- Das Prinzip der Variantenbildung auf 3 Ebenen (links im Bild), verbunden mit Entscheidungen am Ende der jeweiligen Entwicklungsphasen (Vor-, Haupt-, Detailstudien, Bildmitte) liefern Entscheidungssituationen, die eine Korrektur der Marschrichtung gestatten.
- Die rückläufigen Pfeile im PLZ (rechts im Bild) stellen Rückgriffe auf frühere Schritte dar und können als Iterationen verstanden werden.
- Besonders deutlich geht eine gewisse Fähigkeit zur Agility aus der Darstellung in Abb. 3 hervor: Jedes im Detail ausgearbeitet Detailkonzept wird gedanklich in das übergeordnete Gesamtkonzept integriert (nach oben gehende Pfeile). Nicht zufriedenstellende Situationen können und müssen bearbeitet werden auf der Gesamtkonzept- oder Detailkonzept-Ebene. Auch externe Einflüsse, die gar nichts mit dem Projekt zu tun haben, können eine Anpassung des Gesamtkonzepts erforderlich machen (Blitze in Abb. 3).

B. Im SE-Modell problemlos anwendbare Agility

Folgende agile Prinzipien werden vom SE-Modell zwar nicht explizit gefordert oder unterstützt. Sie können aber problemlos angewandt werden, ohne mit dem SE-Konzept in Widerspruch zu geraten.[4]:

- Die personenbezogenen Ansätze, wie z.B. die starke Einbindung der Anwender in die Entwicklungsteams (customer on site), regelmässige Besprechungen in kurzen Zeitintervallen u.a.
- Die Übernahme der Idee der Backlog-Elemente und Sprints (von Scrum)
- Übernahme der Idee der User Stories und des explorativen Prototypings für ausgewählte und besonders diffizil zu gestaltende Module eines Systems
- Übernahme der Gedanken der Einfachheit und Schlankheit von Prozessen: Simplicity (KISS, YAGNI,..), Toyota Lean Production System (Spear et al. 1999)

[3] sog. Situativer Ansatz in der Organisationslehre

[4] Die Integration dieser Prinzipien in das SE-Modell wird Ernst Stelzmann, der Ko-Autor dieses Aufsatzes, im Rahmen seiner gerade laufenden Forschungsarbeit aufzeigen.

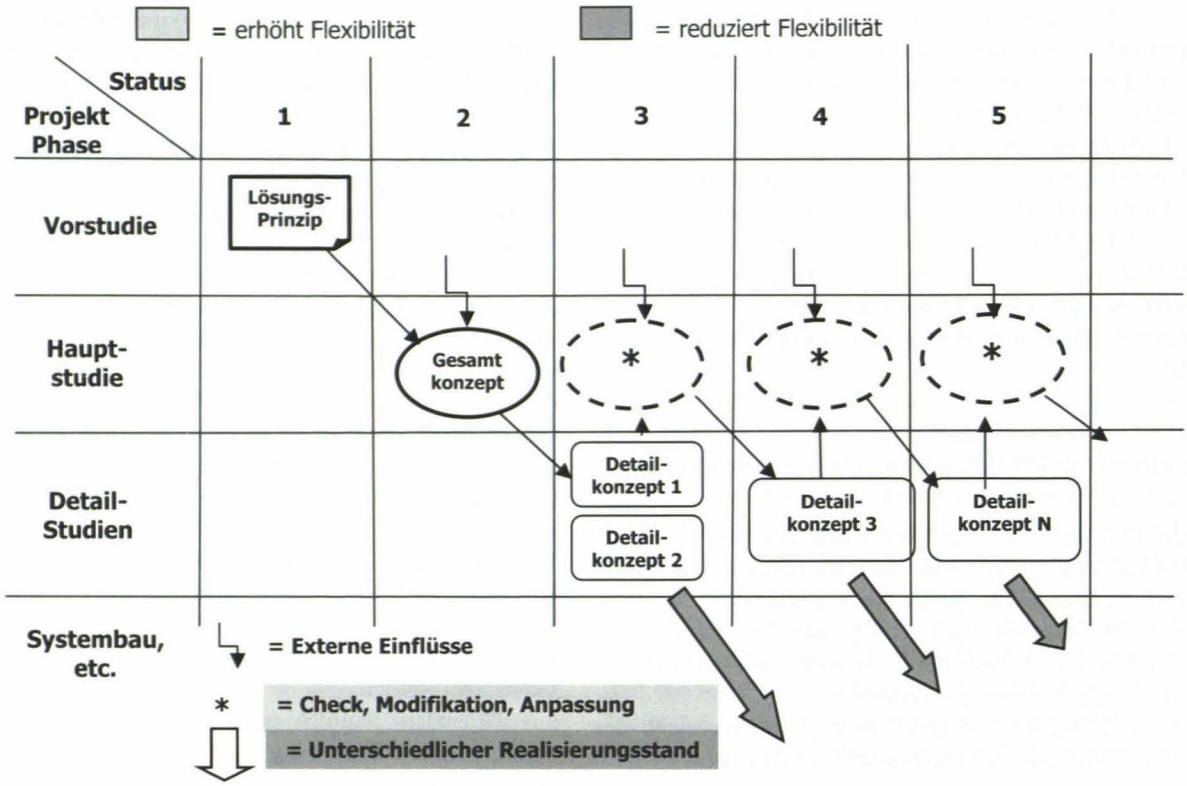

Abb. 3. Dynamik der Gesamtkonzeption mit schrittweiser Integration von Teilergebnissen und der Möglichkeit externer Einflüsse

C. Agility die schwer mit dem SE-Modell zu vereinbaren ist

Bei einigen Prinzipien gibt es grössere Widersprüche zum SE-Modell. Weitere Forschung muss hier nicht nur ergründen, wie diese Prinzipien im SE-Modell angewendet werden können, sondern auch, ob dies überhaupt sinnvoll ist[5]:

- Entwicklung in kurzen Iterationen die immer Kundennutzen liefern müssen: Als einer der wesentlichsten Punkte der agile methods, ist hier zu überprüfen, wie weit es möglich ist, die Systementwicklung iterativ zu gestalten. Die Grundvoraussetzung lautet ja, dass jede Iteration einen gewissen, ständig steigenden Kundennutzen liefern muss, d.h. funktionstüchtig sein muss. Für Software alleine ist dies natürlich viel leichter zu realisieren, als für Systeme (komplexe Produkte, wie z.B. Fahrzeuge). Vor allem in frühen Entwicklungsphasen wird Kundennutzen kaum realisierbar sein. Hier kann aber z.B. mit Simulationsmethoden (HIL) sinnvoll iterativ entwickelt werden, auch wenn die „System-Gesamt-Funktion" und damit der Kundennutzen nicht dargestellt werden kann. In späteren Projektphasen, ist eine iterative Entwicklung mit Kundennutzen in gewisser Weise doch machbar und sinnvoll, vor allem für Projekte in denen Prototypen des Systems entwickelt werden.
- Neutralität gegenüber Änderungen: Hier gibt es umgekehrt Vorteile in frühen und Nachteile in späten Projektphasen. Das SE-Konzept wurde ja entwickelt, um nach einer anfänglichen Aufnahme aller System-Anforderungen und einer möglichst frühzeitigen Ermittlung aller weiteren Umstände, die das System betreffen können, in der Vorstudie ein Lösungsprinzip zu ermitteln und festzulegen. In der Hauptstudie soll dann ein Lösungskonzept ermittelt und festgelegt werden, um sich dann in den Detailstudien auf die Details zu konzentrieren. Es ist also sinngemäß ein Problem, wenn während der Hauptstudie noch eine Änderung des Lösungsprinzips verlangt wird oder während den Detailstudien eine solche des Gesamtkonzepts. Eine möglichst späte Festlegung des Lösungskonzeptes ist nicht immer möglich und zielführend. Welche Strategien hier erfolgversprechend sind oder auf welcher Ebene hier einzugreifen ist (z.B. Modularität in der Systemarchitektur oder verstärkte Anwendung von Software, weil diese ja per Update leicht verändert werden kann) muss erst genauer untersucht werden.

[5] Mit diesen Fragen wird sich Ernst Stelzmann der Ko-Autor dieses Aufsatzes im Rahmen seiner gerade laufenden Forschungsarbeit beschäftigen. Die obige Aufzählung ist deshalb nur beispielhaft zu verstehen und ist noch nicht zu Ende gedacht

V. Conclusion

Das Systems Engineering Konzept hat sich bisher als Methodik verstanden, mit der Projekte ganzheitlich strukturiert, geplant und abgewickelt werden konnten. Es ist dabei ein Rahmenwerk, innerhalb dessen, alle herkömmlichen Projektmanagement- Methoden erfolgreich angewandt werden können. Aufgrund der Unterschiede in der Charakteristik der „Agile Methods" zu den herkömmlichen Methoden erscheint es aber nun fragwürdig, ob auch diese innerhalb eines nach SE strukturierten Projektes erfolgreich angewandt werden können. Dieser Artikel versteht sich als Aufforderung diesen Sachverhalt zu überprüfen.

Weiters stellt er auch die Frage, ob das SE-Konzept selbst, als Methodik, zu starr ist und zuwenig Agilität für die Lösung der Probleme in der heutigen Welt bietet. Hierzu zeigt der Artikel bereits konkret einige Charakteristika des SE-Konzepts auf, die durchaus agile Wesenszüge zeigen und solche, die eine agile Verhaltensweise zulassen, sie also zumindest nicht verhindern. Es wurden aber auch Widersprüche erkannt, die zwischen den Erfordernissen, welche die SE-Methodik an die Strukturierung eines Projektes stellt und den Erfordernissen der Agility stehen. Weitere Forschung soll hier zuerst klären, ob und wie weit es sinnvoll ist, das SE-Konzept an agile Erfordernisse anzupassen. Danach kann schrittweise überprüft werden, welche agilen Vorgehensweisen sich in der praktischen Anwendung als durchgängig erfolgreich erwiesen haben, um das SE-Konzept dahingehend anzupassen.

References

1. Beck K. *et al.* 2001: Manifesto for Agile Software Development http://agilemanifesto.org/
2. Boehm B. and Turner R. 2004: Balancing Agility and Discipline. A Guide for the Perplexed. Addison-Wesley, Boston
3. Haberfellner R. *et al.* 2002: Systems Engineering. Methodik und Praxis. Industrielle Organisation, 11. Auflage, Zürich
4. Haberfellner R. and de Weck O. 2005: Agile SYSTEMS ENGINEERING versus AGILE SYSTEMS engineering. Paper, presented at the INCOSE 2005 World Conference at Rochester NY
5. Spear and Bowen 1999: rigidly specified and highly adaptable. Harvard Business Review, 77(5) 97-106

Reinhard Haberfellner, Dipl.-Ing. Dr.sc.techn. oUProf., Vorstand des Instituts für Unternehmungsführung und Organisation an der TU Graz
Email: reinhard.haberfellner@tugraz.at

Jahrgang 1942, Studium Wirtschaftsingenieurwesen-Maschinenbau an der TU Graz, Promotion an der ETH-Zürich. 1966 - 79 Unternehmensberater am Betriebswissenschaftliches Institut der ETH-Zürich.. Seit 1979 Prof. für Unternehmungsführung und Organisation an der TU-Graz. 1984 - 86 Dekan der Fakultät für Maschinenbau, 1987 - 89 Rektor der TU-Graz. 1995 für 5 Jahre beurlaubt und Vorstandsvorsitzender der STYRIA Medien AG. Ab 2000 zurück an der TU Graz. Autor von 5 Büchern und ca. 50 Fachartikeln. Forschungsinteresse: Unternehmungsstrategien, Erfolgsmerkmale, Systems Engineering.

Ernst Stelzmann, Dipl.-Ing., wissenschaftlicher Assistent am Institut für Unternehmensführung und Organisation an der TU Graz
Email: ernst.stelzmann@tugraz.at

Jahrgang 1979, Studium Wirtschaftsingenieurwesen-Maschinenbau an der TU Graz von 1999 – 2005
2006-2007: entwicklungsbegleitendes Qualitätsmanagement bei der Fa. Magna Powertrain in Lannach
Seit März 2007: wissenschaftlicher Assistent, Forschungsschwerpunkt und Themengebiet für die Dissertation: „Agile Systems Engineering"