Julian Mack:

Software-XPeditionen – eine gelungene Verbindung aus Expeditionssicht und Extreme Programming?

veröffentlicht in:
Mayr, H.C. et al. (Hrsg.): Software-Management 2000: Fachtagung 2.-3. November 2000 an der Philipps-Universität Marburg. OCG-Schriftenreihe Bd. 149. Wien: Österreichische Computer Gesellschaft, 2000. S. 45-59.

Rückfragen und Anmerkungen nimmt der Autor gerne unter der EMail-Adresse julian.mack@hamburg.de entgegen.

SOFTWARE-XPEDITIONEN – EINE GELUNGENE VERBINDUNG AUS EXPEDITIONSSICHT UND EXTREME PROGRAMMING?

Julian Mack[1]

Abstrakt
Veränderungen im Umfeld von Softwareentwicklungs-Vorhaben sowie eine zunehmende technische, soziale und organisationale Verflechtung erfordern flexible Herangehensweisen der professionellen Softwareentwicklung. In diesem Beitrag wird ein neuer Ansatz – die Software-Expedition *– vorgestellt, der Anleihen aus dem Expeditionswesen auf den Bereich der professionellen Softwareentwicklung transferiert und mit dem Ansatz des Extreme Programming von Beck (vgl. [2]) verbindet. Es werden dabei Gemeinsamkeiten und Unterschiede zwischen der Software-Expedition und dem Ansatz von Beck aufgezeigt und Wege für eine hilfreiche Verbindung beider Ansätze skizziert.*

1. Einführung

Mit diesem Beitrag möchte ich Sie einladen, mich drei Schritte auf dem Weg zu einem veränderten Verständnis von professioneller Softwareentwicklung zu begleiten. Ich betrachte dabei die Entwicklung von Anwendungssystemen, die in ein soziales und organisationales Umfeld eingebettet ist. Charakteristisch für diese Form der Softwareentwicklung ist, daß sie umfangreiches Wissen der Beteiligten über den Anwendungsbereich erfordert, welches vorab oder während des Entwicklungsprozesses angeeignet werden muß. Gleichzeitig unterliegt der Kontext solcher Vorhaben einer hohen Veränderungsdynamik, z.B. durch veränderte Ziele, Fusionen oder Restrukturierungen, die meist als „Störungen“ (vgl. [13]) wahrgenommen werden und entsprechend gehandhabt werden müssen.
Im ersten Schritt werde ich begründen, warum eine Organisation solcher Softwareentwicklungs-Vorhaben in Form von Projekten nur bedingt weiterhilft, wenn die Vorhaben einer hohen Veränderungsdynamik ausgesetzt sind. Die in diesem Kontext oft zu beobachtenden Probleme haben mit den Leitvorstellungen des klassischen Projektmanagements zu tun. Im zweiten Schritt führe ich meinen Ansatz der Software-Expedition ein und versuche zu zeigen, weshalb dieser Ansatz die mit Veränderungen einhergehenden Schwierigkeiten besser adressiert als ein Projektansatz. Ebenso wie das Projekt ist auch die Expedition ein von Menschen organisiertes Vorhaben, jedoch stehen hier andere Aspekte (auf die ich noch eingehen werde) im Vordergrund als bei Projekten.
Schließlich zeige ich im dritten Schritt, wie sich die Expeditionssicht als konzeptioneller Rahmen für Softwareentwicklung mit dem Extreme Programming von Beck [2] als technischer Konkretisie-

[1] Univ. Hamburg, Fachbereich Informatik, Vogt-Kölln-Straße 30, D-22527 Hamburg, julian.mack@acm.org

rung verbinden läßt, um professionelle Softwareentwicklung unter sich verändernden Bedingungen zu unterstützen. Dabei besteht mein Anliegen darin, einen Diskussionsbeitrag zu einem besseren Verständnis der Softwareentwicklungs-Praxis zu leisten.

2. Professionelle Entwicklung von Anwendungssoftware

Gegenstand meiner Ausführungen im Rahmen dieses Artikels ist die *professionelle Entwicklung von Anwendungssoftware, die in einen organisationalen Kontext eingebettet ist*. Diese Form der Softwareentwicklung weist verschiedene Spezifika auf, welche sie von anderen Formen der Projektarbeit unterscheidet. Folgen wir [7], so ist Anwendungssoftware

> „ein Mittel zum Zweck, um fachliche Aufgaben in einem oder mehreren *Anwendungsbereichen* zu erledigen. Dazu werden vorhandene oder neue Problemlösungsstrategien durch Software realisiert. Anwendungssoftware modelliert einen *Gegenstandsbereich* der realen Welt und orientiert sich an einem *Einsatzkontext*, der fest sein kann (dedizierte Software) oder allgemein gehalten wird (Standardsoftware). Anwendungssoftware dient der Steuerung von technischen Prozessen oder der Unterstützung von Arbeitsprozessen. Bei ihrer Entwicklung ist auch die Gestaltung der Interaktion zwischen den Anwendungsfachleuten und dem eingesetzten Anwendungssystem zu beachten."

Unter Anwendungssoftware verstehe ich demnach Softwaresysteme, welche in einen sozialen und / oder organisationalen Kontext eingebettet sind und nur vor diesem Hintergrund verstanden (weiter-) entwickelt, eingeführt, angepaßt, eingesetzt und verändert werden können. Die Entwicklung solcher Software erfolgt üblicherweise in einem professionellen Arbeitskontext. In Anlehnung an [22] greift der professionelle Praktiker in seinem Arbeitshandeln auf ein umfangreiches implizites Erfahrungswissen zurück, welches er im Laufe seiner Berufspraxis erworben hat. Er fällt dabei täglich diverse Entscheidungen, für die er keine expliziten Regeln angeben kann, da diese im Laufe der Zeit für ihn selbstverständlich geworden sind.

Die von mir betrachteten Vorhaben können als *komplex* bezeichnet werden: „Komplexe Probleme unterscheiden sich von den komplizierten Problemen dadurch, dass zwar auch viele verschiedene, stark verknüpfte Einflussgrössen die Problemsituation auszeichnen, deren Interaktion sich aber laufend ändert. Hauptcharakteristikum komplexer Probleme ist also Dynamik, ein Eigenleben, das Auftreten immer neuer Muster und Konstellationen" ([8], S. 15). Die Einbeziehung des sich ändernden Kontextes macht eine sukzessive Erschließung und Revision der Anforderungen und eine intensive, interdisziplinäre Zusammenarbeit zwischen den beteiligten Organisationseinheiten und Personen erforderlich. Erschwert wird dies durch die zunehmende Verflechtung der Unternehmensanwendungen: Softwareentwickler treffen immer häufiger auf bereits bestehende Systemlandschaften, in welche sie neue Komponenten integrieren oder bereits bestehende Komponenten verändern sollen.

Diese Umstände stellen hohe Anforderungen an die Flexibilität der Organisation solcher Vorhaben. Diese können jedoch nur bedingt von den klassischen Leitvorstellungen des Projektmanagements erfüllt werden, welche darauf beruhen, ex ante einen umfassenden und detaillierten Plan zu erar-

beiten, der anschließend unter stabilen Randbedingungen in die Tat umgesetzt wird.[2] In der Konsequenz dieser statischen Vorstellung geht es während der Projektdurchführung also nur noch darum, Abweichungen von diesem Masterplan durch entsprechende Steuerungsmaßnahmen zu korrigieren.
In [15] habe ich versucht, einen Weg zu skizzieren, welcher die hohe Veränderungsdynamik solcher Vorhaben direkt anspricht. Er besteht darin, ein neues Leitbild für die Softwareentwicklung vorzuschlagen und eine entsprechende Vorgehensweise zu entwickeln. Provokant gefragt: Was würde passieren, wenn wir uns von den Leitvorstellungen des »Projektes« verabschieden und stattdessen Softwareentwicklung als eine »Expedition« betrachten und betreiben würden? Ich werde im folgenden die damit verbundenen Vorstellungen als »Expeditionssicht« charakterisieren.
Ein solcher Vorschlag wirft zunächst viele Fragen auf, die einer näheren Erläuterung bedürfen. In diesem Artikel werde ich dazu einige Antworten anbieten, die v.a. die verfahrenstechnischen Konsequenzen einer solchen Vorgehensweise betreffen. Dabei werde ich mich insbesondere mit dem Verhältnis zwischen dem Extreme Programming (im folgenden „XP“ abgekürzt; vgl. [2]) – einer neuen Programmiermethode für kleine Teams, die sich einer hohen Veränderungsdynamik ausgesetzt sehen – und meiner Expeditionssicht (vgl. [15]) auseinandersetzen.

3. Project Years, Process Years, Production Era – was kommt danach?

Folgen wir der „Encyclopedia of Software Engineering“, so können wir zwischen drei verschiedenen Entwicklungsperioden der Softwaretechnik mit je spezifischen Schwerpunktsetzungen für die Forschung und Entwicklung unterscheiden (vgl. [16]): Die „Project years“ zwischen Anfang der 1970er bis Mitte der 1980er Jahre standen im Zeichen des aufkommenden Projektmanagements: Die zunehmende Komplexität der zu entwickelnden Softwaresysteme führte zu größeren Projektteams, die sich nicht mehr nebenbei organisieren ließen, sondern die Einführung von Projektmanagement-Methoden erforderlich machten. Der Zeitraum ab Mitte der 1980er Jahre bis Ende der 1990er Jahre kann als Epoche der „Process years“ charakterisiert werden, in dessen Zentrum intensive Bemühungen um eine Verbesserung des Entwicklungsprozesses standen. Seit etwa Mitte der 1990er Jahre bahnt sich ein Übergang in die Epoche der „Production Era“ an, die seitdem andauert. Diese Zeit ist v.a. geprägt durch das Aufkommen der Objektorientierung in vielen Bereichen der Softwareentwicklung, die Entwicklung und Anwendung hochintegrierter und verteilter Softwaresysteme sowie die rapide Verbreitung von Technologien im Umfeld des Internet.
Zentrale Axiome[3] des Projektmanagements, welche mit diesen drei Epochen einhergehen, sind u.a.[4]:

---

[2] Der Begriff »Projekt« stammt aus dem Französischen (2. Hälfte 17. Jh.) und bedeutete ursprünglich „Plan, Entwurf, Vorhaben, Absicht, als Entwurf vorliegende und im Entstehen begriffene Unternehmung, Bauvorhaben“; vgl. [21], S. 1047. Wir sprechen auch von »Projektion«, wenn wir die Vorstellung bezeichnen, einen Plan einer Schablone gleich auf einen Gegenstand zu projizieren – in der Absicht, ihn anschließend gemäß der Projektion zu formen / zu gestalten.

[3] Zur Erinnerung: Ein Axiom ist ein „als absolut richtig anerkannter Grundsatz“ bzw. „eine gültige Wahrheit, die keines Beweises bedarf“; vgl. [5], S. 172.

[4] Diese Leitvorstellungen haben ich aus der umfangreichen Projektmanagement-Literatur herausgearbeitet; vgl. dazu exemplarisch z.B. [10], [12], [14] und [27], auf Software Engineering bezogen z.B. [1], [17], [23] oder [25].

- Ein gegebenes Problem läßt sich stets rational-logisch durchdringen und ist somit grundsätzlich einer Lösung zugänglich (Stichworte: Definition der Problemstellung, Herauslösung des Problems aus dem Problemkontext, Beschreibung von Problemhierarchien und Kausalbeziehungen).
- Die Lösung eines gegebenen Problems läßt sich umfassend im Vorwege planen und mit Hilfe eines schematischen Problemlösungsprozesses realisieren (Stichworte: Planung der Work-breakdown-structure, Streben nach einem standardisierten, vorhersagbaren und optimalen Prozeß, wie z.B. bei Phasenmodellen oder im Capable Maturity Model angestrebt).
- Der Lösungsprozeß läßt sich mit Hilfe des Regelkreis-Ideals steuern (Stichworte: Software-Metriken, Abgleich zwischen Soll- und Ist-Werten, Projektsteuerung nach dem Regelkreis-Modell).
- Von den beteiligten konkreten Personen (Entwickler, Projektleiter, Auftraggeber, Anwender etc.) kann in für den Lösungsprozeß geeigneter Weise abstrahiert werden, und diese Personen können als austauschbare Ressource behandelt werden (Stichworte: Ersetzbarkeit / Austauschbarkeit von Entwicklern, funktionale Rollenbeschreibungen, formalisierte Aufgaben- und Arbeitsteilung).
- Softwareentwicklung läßt sich nach dem Vorbild der industriellen Produktion organisieren, in deren Verlauf die Software als ein Produkt hergestellt wird (Stichworte: Trennung zwischen Herstellung und Einsatz von Software, Produktivitätsmessungen, Software-Wartung, Planung als Produktionsvorschrift).[5]

Während sich diese Vorstellungen für andere Formen der Projektarbeit als tragfähig erwiesen haben, helfen sie nur bedingt weiter, wenn es darum geht, professionelle Softwareentwicklung in einem instabilen Umfeld zu betreiben. So ist z.B. die Vorstellung der Herauslösung eines Problems aus dessen Kontext wenig hilfreich, wenn viele Kontextbedingungen miteinander vernetzt sind und die Integration eines „Produktes“ in dieses Netzwerk sich kaum abschätzbar auf die Gesamtorganisation auswirkt. Zudem wird eine als Produktionsvorschrift verstandene Planung ad absurdum geführt, wenn sich die Grundlagen und Randbedingungen der Planung laufend ändern. Schließlich wird dadurch auch viel Potential verschenkt, wie z.B. Möglichkeiten, gemeinsame Lernprozesse zu vollziehen, sich selbst zu organisieren und Teamgeist zu entwickeln – Faktoren, die sich in einer höheren Motivation und geringeren Fluktuation widerspiegeln und sich so positiv für das Vorhaben auswirken.

Eine zentrale Erkenntnis der bisher umfangreichsten Untersuchung über Softwareentwicklungs-Vorhaben im deutschen Sprachraum war, daß „ein linearer Zusammenhang zwischen »Planung« und Projekterfolg kaum auszumachen [war]: Wir trafen auf Projekte, die trotz fehlender formalisierter Planung recht erfolgreich abgeschlossen wurden; dem standen Projekte gegenüber, die trotz (oder wegen) detaillierter Planung nur zu unbefriedigenden Ergebnissen führten oder gar abgebrochen wurden“ ([26], S. 158). Stattdessen zeigte sich, daß eher „ein Bezug hergestellt werden [konnte] zwischen Projekterfolg und der Bewältigung von Anpassungserfordernissen, die sich im Lauf des Entwicklungsprozesses stellten“ ([26], S. 158). Diese Erkenntnisse sind auch heute noch

[5] Mit einem solchen Produktions-Verständnis setzt sich u.a. Floyd (vgl. [6]) kritisch auseinander.

zutreffend, wie ich anhand einer eigenen Interviewbefragung feststellen konnte.[6]
Dort, wo Software-Vorhaben in die Krise geraten, läßt sich häufig die Anwendung von Maßnahmen der industriellen Produktionslehre beobachten. Dazu zählen u.a.: Fehler ausmerzen; einen reibungslosen Betrieb gewährleisten; möglichst sofort und hart eingreifen, wenn Mitarbeiter sich undiszipliniert verhalten; Mitarbeiter wie austauschbare Teile von Maschinen behandeln; Experimente ausschließen; Verfahren standardisieren (vgl. [3], S. 8). DeMarco und Lister weisen darauf hin, daß sich solche Maßnahmen für die Softwareentwicklung nicht eignen: „Um kreative, geistig arbeitende Mitarbeiter effektiv anzuleiten, müssen Sie Maßnahmen ergreifen, die fast diametral den oben genannten gegenüberstehen“ ([3], S. 9). Die Erkenntnis, daß Softwareentwicklung (auch) ein kreativer Prozeß ist, der unverplante und unbeschränkte Freiräume für kreative Problemlösungen erfordert, setzt sich in der Fachliteratur nur langsam und punktuell durch (vgl. [19], [20]).
Die m.E. entscheidende Frage ist: Wie können wir in Zukunft unseren Anspruch aufrecht erhalten, leistungsfähige Softwaresysteme für die vielfältigen Wertschöpfungsprozesse von Unternehmen zu entwickeln, wenn sich gleichzeitig das Umfeld der Entwicklungsvorhaben laufend verändert? Was können wir unternehmen, um den beteiligten Menschen in sich verändernden Entwicklungsprozessen eine hilfreiche Orientierung zu geben?

4. Software-Expeditionen als professionelle Vorgehensweise in einem instabilen Umfeld

Eine mögliche Lösung besteht für mich darin, den Prozeß der Softwareentwicklung auf eine völlig andere, neuartige Weise anzuschauen. Diese veränderte Betrachtungsweise führt einerseits dazu, bessere Beschreibungsmöglichkeiten dessen zu bekommen, was in der Praxis täglich zu beobachten ist, andererseits eröffnet der veränderte Blickwinkel neue Möglichkeiten, den Prozeß der Softwareentwicklung auf die veränderten Umweltbedingungen (instabiles Umfeld, zunehmende technische, soziale und organisationale Vernetzung) auszurichten und entsprechend aktiv zu gestalten.
Wenn man sich auf das Gedankenexperiment einläßt, Softwareentwicklung als eine »Expedition« im Sinne einer Forschungsreise aufzufassen, so gewinnt man schnell Abstand von der Vorstellung eines rational-logischen Zugangs zur Problemstellung und zum Lösungsprozeß. Stattdessen wird die Aufmerksamkeit auf die handelnden Menschen gelenkt, die in einem unbekannten Gebiet ein bestimmtes Ziel verfolgen. Um einen Bezugspunkt für diesen Kontext zu schaffen, lege ich die Bedeutung des Begriffs »Expedition« wie folgt fest:

> Eine *Expedition* ist eine zeitlich begrenzte Reise, die von mehreren Menschen zur Wahrnehmung einer komplexen Aufgabenstellung mit einer bestimmten Zielsetzung durchgeführt wird. Dabei betritt das Expeditionsteam ein ihm in wenigstens einem Aspekt unbekanntes Gebiet. In ihrem Verlauf ist die Expedition verschiedenen sich ändernden Umwelteinflüssen und schwer kalkulierbaren Risiken ausgesetzt. Das Expeditionsteam versucht dabei sich stets einen größt-

[6] Im Rahmen meiner Promotionsarbeit habe ich im Frühjahr 2000 insgesamt 21 ein- bis zweistündige Interviews geführt. 17 Interviews wurden mit Entwicklern, Projektleitern, Beratern und Projektmanagern aus dem Bereich der Softwareentwicklung geführt, vier weitere Interviews mit Expeditionsleitern.

möglichen Handlungsspielraum zu erhalten. Hinsichtlich der Gesamtheit aller Faktoren, die eine Expedition ausmachen, ist jede Expedition für sich genommen einzigartig.[7]

Mit dem Begriff »Expedition« werden also eher die Erkundung unbekannter Gebiete, Abenteuer, Unwägbarkeiten und Risiken, aber auch verantwortliches Handeln, Risikobewußtheit und Teamgeist assoziiert – Aspekte, die auf viele Softwareentwicklungs-Vorhaben in der Praxis zutreffen. Dies bedeutet keinesfalls die Gleichsetzung von Softwareentwicklung mit Chaos oder eine Idealisierung von Risiken. Stattdessen öffnet die Expeditionsmetapher stärker den Blick für die in der Praxis anzutreffenden Phänomene: vage Zielsetzungen, wechselnde oder hinzukommende Anforderungen, Work-arounds, Verzögerungen, das Betreten technologischen „Neulands“, damit einhergehende Risiken, Überforderung des Teams durch andauernde Überlastung etc. – das Bild der Expedition führt uns die Konsequenzen solcher Phänomene sehr viel deutlicher vor Augen, als es das Projektmanagement zu leisten vermag.

Drei wesentliche Aspekte solcher »Software-Expeditionen« sind (vgl. [15]):

- ein Verständnis des Anwendungsgebiets als *Territorium* der Softwareentwicklung;
- eine Organisation des Softwareentwicklungsprozesses, welche der *Exploration* dieses Territoriums dient (dabei bleibt die Exploration nicht auf eine passive Beobachtung beschränkt, sondern wird vielmehr als aktiver und kontinuierlicher Erkundungsprozeß verstanden);
- eine Auffassung von Softwareentwicklung als *Prozeß der situativen Reorientierung*, d.h. als ein Prozeß der schrittweisen Neubewertung und Entscheidung (je nach aktueller Situation und Position) über das weitere Vorgehen des Expeditionsteams.

Der erste Aspekt öffnet den Entwicklungsprozeß stärker für die beteiligten Personen und Gruppen der Anwendungsorganisation. Betrachten wir das Anwendungsgebiet als Territorium der Softwareentwicklung, sehen wir ein strukturiertes, von gleichlaufenden und konkurrierenden Kräften gestaltetes Gelände. In diesem Gelände soll die zu entwickelnde Software entstehen und eingesetzt werden. Wenn ich den Begriff des Territoriums auf Softwareentwicklung beziehe, so meine ich damit den konkreten Anwendungskontext der Softwareentwicklung im speziellen sowie die Anwendungsorganisation im allgemeinen, mit welcher die Beteiligten zu tun haben. Beide Aspekte sind für das Handeln, Orientieren und Entscheiden der Beteiligten wichtig, wie in Abb. 1 dargestellt. Die ausschließliche Betrachtung nur eines Aspektes führt zu einem unvollständigen Bild des Territoriums und kann sich als entsprechend gefährlich für die Beteiligten erweisen.

Wichtige „Elemente“ des Territoriums für das Handeln, Orientieren und Entscheiden der Beteiligten sind u.a. abzulösende Altsysteme, die sonstige Systemlandschaft, einzusetzende bzw. eingesetzte Software-Entwicklungswerkzeuge, konkurrierende Vorhaben, organisationsinterne Qualitätsstandards, historisch gewachsene Strukturen und Geschäftsprozesse, die räumliche Situation, beteiligte oder betroffene Unternehmensstandorte, Tochter-, Schwester- und / oder Mutterunternehmen. Dar-

[7] Der Einfachheit halber werde ich mich im folgenden auf „Gebirgsexpeditionen“ als Vorbild beziehen, da diese in einem stark heterogen strukturierten Gelände stattfinden, welches durch vielfältige Umwelteinflüsse und -ereignisse (Wetter, Lawinen etc.) verändert wird. Dieses Vorbild beschränkt sich nicht auf die Tätigkeit des Bergsteigens, sondern umfaßt allgemein die Vorbereitung und Durchführung von Reisen in bzw. durch eine Gebirgslandschaft. Damit grenze ich mich bewußt von [9] ab, der sich explizit auf Höhenbergsteigen bzw. Solo-Felsklettern bezieht, welches m.E. als Bild für Softwareentwicklung zu kurz greift und problematisch ist, weil es Risiken idealisiert; vgl. dazu [11], S. 349.

über hinaus ist das Territorium der Anwendungsorganisation selbst in eine Umwelt eingebettet. Wirksame Einflüsse sind hier z.B. Gesetze, Vorschriften und zu beachtende Normen, Behörden sowie Kunden und Konkurrenten der Anwendungsorganisation.
Das Expeditionsteam agiert in diesem Gelände aufgrund der ihm verfügbaren Informationen und in Abhängigkeit von der jeweiligen Position (ihrem Standort) und Situation (den Standortbedingungen). Entscheidungen eines Software-Expeditionsteams (z.B. Neupriorisierung der für die nächste Systemversion zu realisierenden Softwareeigenschaften) lassen sich vor dem Hintergrund der aktuellen Situation (z.B. Probleme mit der Einführung eines neuen Entwicklungstools) und Position (z.B. die Zeit drängt und kann aus politischen Gründen nicht verlängert werden) dieses Teams besser nachvollziehen. Neben dem Team agieren noch weitere Menschen und Gruppen in diesem Territorium, wie z.B. Entscheidungsträger, formelle und informelle Auftraggeber, Fürsprecher und Gegner des Vorhabens, Anwender und sogenannte Key user oder Mitarbeiter aus den Abteilungen und sonstigen Organisationseinheiten der Anwendungsorganisation.

Abb. 1: Die Anwendungsorganisation als Territorium betrachtet.

Der zweite Aspekt beschreibt, wie die an Software-Expeditionen beteiligten Personen dieses Territorium erkunden: Bezogen auf Softwareentwicklung ist mit »Exploration« die schrittweise Erkundung, Konkretisierung und Klärung der Problemstellung, des Anwendungsgebietes, der mit dem Vorhaben verbundenen Ziele und Aufgaben sowie der Randbedingungen und Erwartungen im Zusammenhang mit der Entwicklung eines Softwaresystems gemeint. Diese Exploration sensibilisiert die Beteiligten dafür, daß jede Anwendungsorganisation (auch die eigene) prinzipiell einzigartig ist und viele unbekannte, für Außenstehende kaum nachvollziehbare Besonderheiten aufweist. Dies

trifft insbesondere dann zu, wenn Externe (Berater, Entwickler, IT-Spezialisten) mit Vorhaben einer fremden Anwendungsorganisation beauftragt werden. Viele in der Praxis anzutreffende Probleme beruhen auf einem mangelnden Verständnis der Beziehungen und Prozesse der betreffenden Anwendungsorganisation. Ein solches Verständnis ist jedoch eine entscheidende Erfolgsbedingung für die erfolgreiche Integration von Softwaresystemen in ein organisationales Umfeld.

Wenn professionelle Softwareentwickler den Anspruch erheben, Software für Organisationen zu entwickeln – und nicht umgekehrt die Organisation den Belangen der Software unterzuordnen –, sind vielfältige individuelle und soziale Lernprozesse in der Auseinandersetzung mit den Gegebenheiten der Anwendungsorganisation die Konsequenz. Diese aus der Exploration resultierenden Lernerfahrungen werden zu einem integralen Bestandteil von Software-Expeditionen. Die damit verbundene Exploration erfolgt – im Gegensatz zu konventionellen Ansätzen – sukzessive im Laufe des Entwicklungsprozesses, erforderliche Änderungen können auch noch spät einbezogen werden.

Der dritte Aspekt beschreibt, wie sich das Expeditionsteam im Gelände orientiert und vorgeht (vgl. Abb. 2). Er weist darauf hin, daß neben dem technischen Entwicklungsprozeß, in dessen Rahmen vielfältige Neubewertungen der aktuellen Lage durchgeführt und Entscheidungen getroffen werden müssen, auch auf organisationaler Ebene ständig solche Reorientierungen notwendig sind. Der Begriff »situativ« bedeutet, daß eine Neubewertung der aktuellen Position und des einzuschlagenden Kurses „durch die (jeweilige) Situation bedingt“ ([5], S. 1262) stattfindet (vgl. dazu auch [24]).

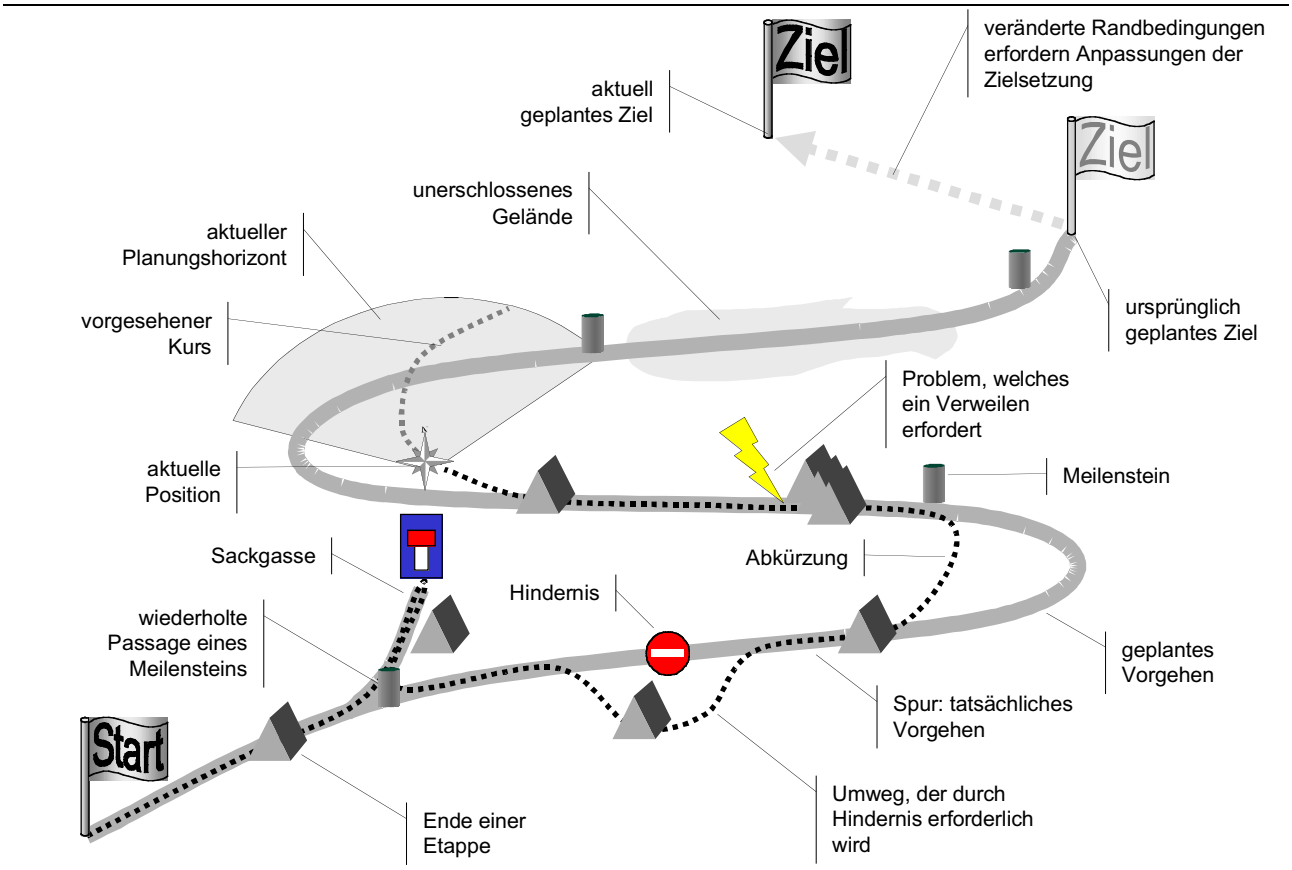

Abb. 2: Das Vorgehen einer Expedition in einem Territorium.

Hier läßt sich ebenfalls eine hohe Affinität zwischen dem Vorgehen von Expeditionen und der Praxis der Softwareentwicklung feststellen: So ergaben Interviews, die ich im Rahmen meiner Forschungstätigkeit geführt habe, daß die Planung zu Beginn eines Vorhabens sich oft auf die groben Eckpunkte und den ungefähren Verlauf des Vorhabens beschränkt. Im Rahmen dieser Grobplanung ist oft viel wichtiger, wieviel Zeit das Team zur Erreichung der Ziele des Vorhabens benötigt, als auf welchem konkreten Weg diese Ziele erreicht werden sollen. Auch bei Expeditionen steht der zeitliche Rahmen stärker im Vordergrund als die detaillierte Festlegung der Wegstrecke: Vom zeitlichen Rahmen hängt nämlich wesentlich die Proviantierung und Ausrüstung der Expedition ab (umgekehrt bestimmt auch die Last der mitgeführten Materialien, wie die groß Reichweite eines Expeditionsteams ist). Eine Konkretisierung des weiteren Vorgehens erfolgt typischerweise erst während des Vorhabens, und zwar für die jeweils unmittelbar bevorstehenden Aktivitäten. Ein typischer Planungshorizont für eine solche Konkretisierung sind zwei bis vier Wochen. In diese Feinplanung werden aktuelle Veränderungen und Umstände einbezogen. Die meisten meiner Interviewpartner waren der Auffassung, daß ein Planungshorizont für die Feinplanung von mehr als etwa einem Monat keinen Sinn macht, da der Aufwand für die Feinplanung in Relation zu dem Gewinn an Orientierung aufgrund der vielen Veränderungen unverhältnismäßig hoch wird.
Gleichzeitig wurde in den geführten Interviews immer wieder deutlich, daß die Befragten im Laufe der Zeit eine „innere Landkarte“ entwickelten, also ein mentales, räumliches Modell von den Elementen des Vorhabens. Dieses spiegelte sich auch in ihrer Sprache wider, wie an den folgenden Ausdrücken exemplarisch sichtbar wird: Vorbe*reiten* und Vor*gehen*; wissen, wo man *steht*; die *Kurve* kriegen; auf die *Schiene* setzen; in einem *Boot* sitzen; Fehler *einkreisen*; einer Sache *auf den Grund gehen*; in die richtige *Richtung* arbeiten; ein Problem *taucht auf*. Ein anderes Beispiel aus meinen Interviews, welches von der Notwendigkeit einer gemeinsam erarbeiteten Planung handelt: „Nur wenn sie das, was sie *beigetragen* haben, wenn sie das wiederfinden *da vorne*, werden sie auch *losmarschieren* mit der Flagge, und werden dafür sorgen, daß das auch umgesetzt wird.“ Die von den Befragten verwendeten Ausdrücke aus dem Bereich der Landschaftsbeschreibung und des Reisens verdeutlichen anschaulich, wie der Prozeß der Exploration des Territoriums und der Reorientierung mit der Entstehung einer mentalen Vorstellung über das Vorhaben einher geht.
Diese Anpassungsleistung stellt neue Anforderungen an die Expeditionsteilnehmer im Hinblick auf ihr Improvisationstalent, ihre soziale und kommunikative Kompetenz und ihre Problemlösungsfähigkeiten. Gleichzeitig wird eine verfahrenstechnische Grundlage notwendig, die ausreichend Flexibilität zur situativen Reorientierung zur Verfügung stellt. Weiter unten werde ich zeigen, daß XP gute Voraussetzungen für eine solche Unterstützung von Software-Expeditionen mitbringt.
Aus diesen Überlegungen ergeben sich verschiedene praktische Konsequenzen für die Gestaltung der Vorgehensweise von Software-Expedition, von denen ich die wichtigsten im folgenden nenne:

- *Teamorientierung*: Das Team der Expeditionsmitglieder und dessen Handeln im Territorium stehen im Mittelpunkt des Geschehens. Die Zusammenstellung und Entwicklung des Teams muß dafür geeignet organisiert werden.
- *Selbstorganisation des Expeditionsteams*: Das Expeditionsteam ist für die Organisation der Softwareentwicklung selbst verantwortlich. Dazu werden ihm die erforderlichen Ressourcen und organisatorischen Freiräume zugestanden.

- „*Light weight software development*“: Analog zu Messners Idee für seine Hidden-Peak-Besteigung (Gasherbrum I)[8]: minimal nötige Logistik, Verzicht auf vermeintlich notwendige Ausrüstung, kleinstes Team, wenige „Basislager“. Hier werden die Praktiken des XP relevant.
- *Routenmanagement*: Vorgehen in Etappen[9] (zeitbezogene Abschnitte) anstelle von Meilensteinen (wegbezogene Abschnitte) und ständige Neubewertung und Reorientierung der Route.
- *Ressourcenmanagement*: Eine als Ressource verstandene Planung. Beschränkung auf wesentliche Fähigkeiten und Fertigkeiten sowie auf verfügbare Ressourcen („das, was man mitnehmen bzw. tragen kann“).
- *Risikomanagement*: Aktives Risikomanagement steht bei Software-Expeditionen im Spannungsfeld aus erforderlicher Redundanz zur Erhöhung der Arbeitsqualität und größtmöglicher Flexibilität für eine rasche Reaktion auf Veränderungen. Dies führt u.a. zu tendenziell kleinen Teams.

In diesem Zusammenhang habe ich die konzeptionelle Brauchbarkeit des Extreme Programming im Hinblick auf die Eignung als Methode für „light weight development“ untersucht und dabei festgestellt, daß diese Methode auf unterschiedlichen Ebenen eine hohe Affinität zu meinem Ansatz der Software-Expedition aufweist.

5. Von Expeditionen zu XPeditionen – XP als technische Grundlage der Software-Expeditionen

Folgen wir [2], so stellt XP eine nützliche Kollektion aufeinander bezogener Methoden und Hilfsmittel bereit, die sich besonders in einem rasch verändernden Umfeld bewährt haben. Beck selbst betont, daß es sich dabei um eine Zusammenstellung seit langem bekannter Werte, Prinzipien und Praktiken handelt, die in extremer Weise konsequent durchgeführt bzw. angewendet werden.

Als Grundlage und Ausgangspunkt des XP dient die »Driving Metaphor«, welche beschreibt, wie man das Fahren eines Autos erlernt (vgl. [2], S. 27f). Im Gegensatz zu der bei Projekten weit verbreiteten Praxis, dem Auto einfach eine Richtung auf das Ziel hin vorzugeben und es so orientiert losfahren zu lassen, kommen wir im Straßenverkehr ohne ständige, kleinere Steuerungskorrekturen schnell von der Fahrbahn ab. Bereits dieses Bild weist eine hohe Affinität zu meiner Expeditionssicht auf, wenngleich die Expeditionssicht deutlicher betont, daß es nicht notwendigerweise vorgegebenen Wege („Straßen“) gibt, sondern mögliche (effektive und effiziente) Wege im Verlauf der Expedition erst erkundet bzw. erschlossen und beschritten werden.

Auch die von XP propagierten Werte »Kommunikation«, »Einfachheit«, »Feedback« und »Mut«[10] sind für Expeditionen zentral (vgl. [2], S. 29ff): Fehlerhafte oder mangelnde Kommunikation kann

[8] Vgl. [18].

[9] Aus der ursprünglich militärischen Bedeutung des Begriffs »Etappe«, nämlich »Warenlager« entwickelte sich die Bedeutung »Tagesmarsch«, da Versorgungsdepots für die Armee in Abständen von Tagesmärschen angelegt wurden; vgl. [21], S. 301.

[10] Bei der nachfolgend verwendete deutschen XP-Terminologie beziehe ich mich auf die Wiki-Website unter der URL: `http://c2.com/cgi/wiki?GermanXpTerminology`

das Expeditionsteam in die Katastrophe führen. Einfachheit (und damit einhergehende Robustheit) ist ein wichtiges Kriterium bei der Ausrüstung von Expeditionen sowie bei der Navigation im Gelände (Landmarken), außerdem spielen einfache Lösungen bei der Überwindung von Schwierigkeiten und Hindernissen eine wichtige Rolle. Feedback ist wesentlich für die Kursbestimmung und Routenwahl – und damit für das situative Handeln der Expeditionsmitglieder insgesamt. Mut ist ein wesentlicher Motor für Expeditionen – Übermut ist dagegen oft ursächlich für das Scheitern von Expeditionen.

Aus der Grundmetapher und dem damit verbundenen Wertesystem leitet Beck die wesentlichen Prinzipien des XP ab, nämlich »unmittelbares Feedback«, »Einfachheit anstreben«, »inkrementelle Veränderungen«, »Veränderung wollen« und »Qualitätsarbeit« (vgl. [2], S. 37ff).

Auch hier zeigen sich deutliche Affinitäten zur Expeditionssicht. So ist für Expeditionen eine ausreichend schnelle und zuverlässige Rückkopplung mit der Umwelt lebensnotwendig: z.B. kann das Expeditionsteam ohne zuverlässige Wetterinformationen von einem plötzlichen Wetterumschwung überrascht werden und somit starken Gefährdungen (z.B. Lawinengefahr) ausgesetzt sein.

Weit ab von jeglichen Ausrüstungsfirmen ist das Expeditionsteam auch auf die Einfachheit der Lösung von auftretenden Problemen angewiesen: defekte Ausrüstungsgegenstände müssen leicht vor Ort wieder instandgesetzt werden können, da keine Reparaturwerkstatt in der Nähe ist; Lösungen müssen mit wenigen universell einsetzbaren Mitteln und Werkzeugen herstellbar sein, da man spezielle, komplexe Spezialwerkzeuge selten zur Hand hat. So steigt der Wert der Ausrüstungsgegenstände mit ihrer universellen Verwendbarkeit (wie z.B. das Offiziersmesser der Schweizer Armee). Das bedeutet allerdings nicht, daß Expeditionen per se technophob seien: Viele Expeditionen (z.B. unter Wasser oder in die Polarregionen) werden heute mit Hilfe von Hochtechnologie überhaupt erst möglich oder zumindest weniger riskant als zuvor! Man denke dabei z.B. an GPS-Navigation, Satellitenkommunikation, High-Tech-Kunstfasergewebe oder Ultraleicht-Materialien.

Desgleichen finden wir bei Expeditionen inkrementelles Vorgehen an verschiedenen Stellen wieder: So erfolgt z.B. die gesamte Orientierung im Gelände inkrementell, ebenso wie die Auswahl des nächsten Wegabschnitts. Neu eintretende Situationen oder Ereignisse machen im allgemeinen eine Neubewertung der Lage und ggf. eine entsprechende Änderung des bisher eingeschlagenen Kurses erforderlich. In meinen Augen ist die Expedition daher als professionelle Organisationsform weit besser als Projekte auf eine konsequente Integration von Wandel und eine entsprechende Anpassung an neue / veränderte Randbedingungen eingestellt. Dabei bestimmt die Qualität des eigenen Vorgehens wesentlich den Erfolg der Expedition mit: unzureichende Navigation kann schnell das Expeditionsteam gefährden – und jeder Teilnehmer hat ein persönliches Interesse am Erfolg der Expedition.

Auch die weiteren Prinzipien des XP weisen viele Ähnlichkeiten zur Vorgehensweise bei Expeditionen auf: Viele Situationen während einer Expedition erfordern es z.B., zu improvisieren und mit Hilfe »gezielter Experimente« ungewohnte Lösungsideen zu entwickeln. »Lernen lehren« ist dabei ein zentrales Moment, Situationen erfolgreich zu bewältigen. Die XP-Prinzipien »auf Sieg spielen« und »Verantwortung übernehmen« unterstützen die Bewältigung solcher Situationen: Eine schwierige Situation wird vom Expeditionsteam als Herausforderung begriffen, und nach einer Lösung gesucht, deren Risiko vom Team gemeinsam verantwortet werden kann. Eine »offene, auf-

richtige Kommunikation« ist für den Bewältigungsprozeß ebenso wichtig, wie ein »ehrliches Messen«, d.h. eine aufrichtige Bewertung der Situation. Dabei ist für Expeditionen wie für Softwareentwicklungs-Vorhaben gleichermaßen wichtig, »die Instinkte des Teams zu nutzen, nicht dagegen zu arbeiten«. Der Planung kommt dabei die Stellung einer wichtigen Ressource (keiner Vorschrift) zu: um die Planung »an lokale Gegebenheiten anpassen« zu können, muß sie ausreichend Spielräume offenhalten.

Das Prinzip der »geringen Anfangsinvestitionen« spielt ebenfalls an verschiedenen Stellen von Expeditionen eine wichtige Rolle, z.B. im Bereich der Risikoabwägung: das Gesamtrisiko des Scheiterns einer Expedition steigt mit dem eingesetzten Material und der Anzahl der beteiligten Menschen. Diese Überlegung hat R. Messner bewogen, zunehmend kleinere Expeditionen durchzuführen (vgl. [18], S. 131): „Bei meinen Touren waren Ausrüstungs- und Kostenminimierung vielfach auch wesentliche Ideenauslöser. Die Idee lag darin, Planung und Realisation zu erleichtern, was in erster Linie durch Gewichtsersparnis zu erreichen war“. So erhöht z.B. der Verzicht auf schwere Sauerstoff-Flaschen bei der Besteigung von Achttausendern die Beweglichkeit und Schnelligkeit der Bergsteiger, außerdem verbrauchen sie durch die Gewichtsreduktion weniger Energie und erhöhen so ihre individuellen Überlebenschancen beträchtlich. Dies stimmt mit dem XP-Prinzip »mit leichtem Gepäck reisen« überein: „Eine Expedition, die zehn Tonnen ins Basislager verschiebt […], kostet eine Menge Zeit und v.a. Geld. Wenn ich mit einem Hunderstel an Gewicht den gleichen, vielleicht sogar schnelleren Erfolg erzielen kann, ist die Großexpedition nicht mehr zu rechtfertigen“ ([18], S. 102).

Wie sich diese Prinzipien auf die vorgeschlagenen Praktiken des XP auswirken, wird in [2] ausführlich beschrieben. Daher beschränke ich mich an dieser Stelle auf einen Vergleich einiger ausgewählter XP-Praktiken mit meinem Konzept der Software-Expedition und verzichte ansonsten auf eine eingehende Auseinandersetzung mit weiteren XP-Praktiken.

Bleiben wir beim Bild der Bergexpedition: die wohl auffälligste XP-Praktik mit einer Übereinstimmung zur Expeditionssicht ist das »Programmieren in Paaren«, also die partnerschaftliche Bearbeitung einer Aufgabe zu zweit an einem Rechner. Ihre Entsprechung ist die Zweierseilschaft bei Bergexpeditionen. Dabei bilden zwei Personen ein Sicherungssystem am Berg: während der eine Partner den anderen absichert, steigt der zweite Partner voran. Anschließend wechseln die Rollen. Durch diese minimale Redundanz wird bei geringem Einsatz (zwei Personen) eine vergleichsweise hohe Qualität der Sicherung und somit eine deutliche Reduktion des Risikos erzielt. Eine solche Form der Absicherung findet sich auch beim »Testen« und bei der »fortlaufenden Integration« in XP wieder: Das was an Code „sicher“ läuft wird in die aktuelle Fassung des Systems auf dem Integrationsrechner integriert. Die Strategie »erst Testklassen, dann zu testende Klassen entwickeln« entspricht dabei der Absicherung mit Haken und einem Sicherungsseil zwischen den Partnern.

Schließlich kommen viele Ideen der Planung im XP Überlegungen von Expeditionsplanungen recht nahe: „Software development is always an evolving dialog between the possible and the desirable. The nature of this dialog is that it changes both what is seen to be possible and what is seen to be desirable“ ([2], S. 55). Planung im XP geht davon aus, daß sich nicht alle Faktoren (Kosten, Zeit, Qualität, Umfang) gleichzeitig im Voraus festlegen lassen; maximal drei Faktoren beeinflussen die verbleibenden Faktoren. Von seiten des Auftraggebers besteht üblicherweise der Wunsch möglichst

alle Faktoren zu seinen Gunsten festzulegen; umgekehrt möchte auch der Auftragnehmer die Faktoren für sich günstig bestimmen. Um diesen Zielkonflikt zu lösen, handeln beide Parteien regelmäßig einen für die aktuelle Situation (z.B. den Zeitraum bis zum nächsten Release) günstigen Kompromiß aus. Dabei sind sich die Beteiligten über die prinzipiellen Grenzen der Planung im klaren. Zu diesem Spannungsfeld zwischen wünschenswertem und möglichem Ergebnis der Planung schreibt Messner ([18], S. 128) für seine Expeditionen: „Pläne sind nur Pläne. Sie können meist in der Durchführung nicht eingehalten werden. […] Die Bedingungen (Wetter, Schneebedingungen) und Umstände (Gesundheit) sind nicht vorherberechenbar. Trotzdem, ohne Plan geht es auch nicht. Weder der Nanga-Parbat-Gipfel noch der Nordpol können ohne dezidierte Logistik erreicht werden. Die Improvisation vor Ort bleibt aber ebenso wichtig."

6. Konsequenzen für Praxis und Forschung der Softwareentwicklung

Bis hier sollte deutlich geworden sein, daß eine Orientierung von Softwareentwicklung am Leitbild der Expedition sowie der konzeptionelle Rahmen der Software-Expedition in Verbindung mit XP als verfahrenstechnischer Grundlage, eine brauchbare Ausgangsbasis zur Beantwortung meiner Eingangsfrage bildet, wie professionelle Softwareentwicklung in einem veränderlichen Umfeld betrieben werden kann und welche Orientierungshilfen den handelnden Personen mitgegeben werden können.

In diesem Beitrag habe ich dargelegt, daß eine hohe Affinität zwischen dem Ansatz des Extreme Programming und meinem Ansatz der Software-Expedition besteht. Die Expeditionssicht gibt eine Orientierung durch das Leitbild der Expedition, die über die »Driving-Metaphor« deutlich hinausreicht und somit den Möglichkeitenraum für Softwareentwicklung weiter faßt als XP. Gleichzeitig erweitert die Expeditionssicht XP um Aspekte der situativen Reorientierung und Exploration, wie sie bisher – in Umfang und Intensität – für XP nicht vorgesehen waren. Darüber hinaus erscheint die Expeditionssicht als generative Metapher geeignet zu sein, um weitere XP-Praktiken zu finden. Umgekehrt gibt XP den Software-Expeditionsteilnehmern das erforderliche technische Werkzeug an die Hand.

Für die Praxis der Softwareentwicklung lassen sich aus der Expeditionssicht Kriterien zur Ausrüstung von Software-Expeditionen ableiten: So sollten die nötigen Werkzeuge zur Entwicklung von Anwendungssystemen u.a. leicht anwendbar, robust und universell einsetzbar sein. Auch werden verschiedene dysfunktionale Maßnahmen des Projektmanagements in ihren Konsequenzen transparenter, wie z.B. das Auswechseln von Teammitgliedern oder das Aufstocken des Teams, um Terminschwierigkeiten zu beheben.

Auch der Prozeß der Teamzusammenstellung verändert sich bei Software-Expeditionen: Er leitet sich aus dem symmetrischen Prozeß des „Anheuerns" ab, bei dem nicht nur der Expeditionsleiter entscheidet, ob der Kandidat mitdarf, sondern auch der Kandidat für sich entscheidet, ob er die mit dem Vorhaben einhergehenden Herausforderungen mit tragen möchte. Durch ein solches gegenseitiges Commitment wird eine hohe Verbindlichkeit und damit eine hohe Stabilität der Teamzusammensetzung gefördert. Damit werden wichtige Voraussetzungen für die von DeMarco und Lister propagierten „eingeschworenen Teams" geschaffen (vgl. [3], S. 141ff).

Die Software-Expedition führt auch weg von einem betont phasenorientierten Vorgehen und hin zu einem etappenweisen Vorgehen, das auch Umwege und Abkürzungen verständlich macht. Dabei setzt die Software-Expedition stärker auf Prototyping als Kommunikationsmedium zwischen Anwendern und Entwicklern. In diesem Zusammenhang orientieren sich die Beteiligten auch stärker als bisher an dem jedem Team eigenen Entwicklungstempo. Schließlich betont die Software-Expedition ein aktives Risikomanagement, welches zu einem integralen Bestandteil der Aktivitäten einer Software-Expedition wird und u.a. mit Hilfe der XP-Idee realisiert wird, in Paaren zu programmieren.

Darüber hinaus verliert XP durch seine Einbettung in den konzeptionellen Rahmen der Software-Expedition etwas von seinem „Extremismus“ – einem Aspekt, der viele Menschen auf den ersten Blick unangenehm berührt und Abwehrreaktionen hervorruft: Nicht jeder Mensch ist extremen Dingen / extremem Verhalten gegenüber aufgeschlossen, da Extremismus in unserer Gesellschaft mit negativen Wertevorstellungen belegt ist (man denke z.B. an politische Extremisten). Expeditionen dagegen sind „spannend“, machen neugierig und werden emotional tendenziell positiv besetzt.

Für die Forschung erhalten wir durch diesen Ansatz eine neue Sichtweise, welche das Potential hat, viele in der Praxis zu beobachtende soziale und organisationale Phänomene besser als bisher zu beschreiben. Dies führt zu einem insgesamt tiefergehenden Verständnis von Softwareentwicklung als sozialem und organisationalem Prozeß der Praxis. Außerdem bietet der Ansatz der Software-Expedition neue Anregungen und Anstöße für bisher ungenutzte Denkrichtungen auf dem Weg zu einer veränderungs-orientierten Softwaretechnik. Für die Softwaretechnik-Forschung eröffnen sich dadurch neue Möglichkeiten, über den (nunmehr situativen) Prozeß der Softwareentwicklung konzeptionell und theoretisch nachzudenken.

Kombiniert bergen XP und Software-Expeditionen ein hohes Synergiepotential für die Beantwortung der Fragestellung, Software in einem sich dynamisch verändernden Kontext professionell zu entwickeln. Durch diesen Ansatz werden neue Denkrichtungen, Spielräume und Möglichkeiten für die Gestaltung und Weiterentwicklung einer an der Praxis orientierten Softwaretechnik eröffnet. Aus diesem Grund schlage ich vor, in diesem Zusammenhang eher von Software-Expeditionen als von Softwareprojekten zu sprechen / zu schreiben. Um der fruchtbaren Verbindung von Software-Expeditionen mit XP Rechnung zu tragen, könnte man zukünftig sogar von Software-XPeditionen sprechen. Der Ansatz lädt dazu ein, die von Denert in [4] angesprochene Kluft des Software Engineering zwischen Wissenschaft und Wirtschaft zu verringern, indem er eine praxisorientierte und fallorientierte Softwaretechnik-Forschung fördert. In diesem Rahmen kann an verschiedenen Stellen der Softwaretechnik „Neuland“ erschlossen werden, z.B. im Bereich der Möglichkeiten und Grenzen des Einsatzes von Software-Expeditionen.

7. Danksagungen

Ich danke Frau Prof. Dr. Christiane Floyd und meinen Kollegen vom Arbeitsbereich Softwaretechnik für die umfangreiche Unterstützung sowie die vielen Diskussionen, welche meine Arbeit vorangetrieben haben. Außerdem danke ich Frank Westphal für einen „extrem“ fruchtbaren Dialog über XP und die Software-Expedition. Schließlich möchte ich der Dehrmann Lauterbach Wolf Unter-

nehmensentwicklung Hannover für die vielen Anregungen danken, die mich inhaltlich bis an den Horizont meiner Vorstellungskraft getrieben und mir damit den Weg für die Expeditionssicht erst eröffnet haben: …der Weg ist das Ziel!

8. Referenzen

[1] BALZERT, H. (1996): Lehrbuch der Software-Technik. Bd. 1. Heidelberg, Berlin, Oxford: Spektrum Akad. Verl.
[2] BECK, K. (1999): Extreme Programming Explained. Reading, MA: Addison Wesley Longman.
[3] DEMARCO, T., LISTER, T. (1991): Wien wartet auf Dich! München, Wien: Hanser.
[4] DENERT, E. (1993): Software-Engineering in Wissenschaft und Wirtschaft: Wie breit ist die Kluft? Informatik-Spektrum, Bd. 16, Heft 5, S. 295-299.
[5] DUDEN (1994): Das Große Fremdwörterbuch. Mannheim, Leipzig, Wien, Zürich: Dudenverlag.
[6] FLOYD, C. (1994): Software Engineering – und dann? Informatik-Spektrum, Bd. 17, 2/1994, S. 29-37.
[7] FLOYD, C., ZÜLLIGHOVEN, H. (1999): Softwaretechnik. In: Rechenberg, P., Pomberger, G. (Hrsg.): Informatik-Handbuch. 2. aktual. u. erw. Aufl. München, Wien: Hanser. S. 763-790.
[8] GOMEZ, P., PROBST, G. J. B. (1995): Die Praxis des Ganzheitlichen Problemlösens. Bern, Stuttgart, Wien: Haupt.
[9] HIGHSMITH, J. (2000): Adaptive Software Development. New York, NY: Dorset House.
[10] KEßLER, H., WINKELHOFER, G. (1999): Projektmanagement. 2., korrig. Aufl. Berlin, Heidelberg: Springer.
[11] KRAKAUER, J. (2000): In eisigen Höhen. München: Piper.
[12] KRAUS, G., WESTERMANN, R. (1995): Projektmanagement mit System. Wiesbaden: Gabler.
[13] KUHNT, B. (1998): Softwareentwicklung als systemische Intervention in Organisationen. Diss. Univ. Zürich.
[14] LITKE, H.-D. (1993): Projektmanagement. 2., überarb. u. erw. Aufl. München, Wien: Hanser.
[15] MACK, J. (1999): Softwareentwicklung als Expedition. In: Ges. f. Informatik (Hrsg.): Informatik-Tage 1999. Leinfelden-Echterdingen: Konradin.
[16] MARCINIAK, J. J. (ed.) (1994): Encyclopedia of Software Engineering. Vol. 1 & 2. New York, NY: J. Wiley & Sons.
[17] MCDERMID, J. (ed.) (1991): Software Engineer's Reference Book. Oxford, UK: Butterworth-Heinemann.
[18] MESSNER, R. (1996): Berge versetzen: Das Credo eines Grenzgängers. 2. Aufl. München, Wien, Zürich: BLV.
[19] MOLZBERGER, P., ZEMANEK, G. V. (Hrsg.) (1985): Software-Entwicklung: Kreativer Prozeß oder formales Problem? Seminar des German Chapter of the ACM am 20.3.1985 in Neubiberg. Stuttgart: Teubner.
[20] NAUR, P. (1985): Programming as Theory Building. Microprocessing and Microprogramming, Vol. 15, pp. 253-261.
[21] PFEIFER, W. (Hg.) (1999): Etymologisches Wörterbuch des Deutschen. 4. Aufl. München: DTV.
[22] SCHÖN, D. (1983): The Reflective Practitioner. New York: Basic Books.
[23] SOMMERVILLE, I. (1995): Software Engineering. 5th edition. Harlow, UK: Addison-Wesley.
[24] SUCHMAN, L. (1987): Plans and situated actions. Cambridge, New York, Melbourne: Cambridge Univ. Press.
[25] THAYER, R. (1997): Software Engineering Project Management. In: Dorfman, M., Thayer, R. (eds.): Software Engineering. Los Alamitos, CA: IEEE Computer Society Press. pp. 358-371.
[26] WELTZ, F., ORTMANN, R. G. (1992): Das Software-Projekt. Frankfurt / M., New York: Campus.
[27] WINKELHOFER, G. (1999): Methoden für Management und Projekte. 2. überarb. Aufl. Berlin, Heidelberg: Springer.