Was braucht es für die Langlebigkeit von Systemen? Eine Kritik

Lutz Prechelt
Institut für Informatik, Freie Universität Berlin
prechelt@inf.fu-berlin.de

Abstract: Der Aufruf zum 1. Workshop "Langlebige Softwaresysteme" lässt in seiner Sicht darauf, was forschungsseitig für erfolgreiche Langlebigkeit von Softwaresystemen zu tun ist, vernachlässigte Bereiche erkennen. Dieser Artikel zeigt auf, wo diese liegen, warum die Vernachlässigung problematisch ist und mit welcher Maßnahme eine Beschädigung des resultierenden Forschungsprogramms vermutlich vermieden werden kann.

1 Ziel und Aufbau dieses Artikels

Der Einreichungsaufruf zu diesem Workshop[1] charakterisierte dessen Zweck wie folgt: Im Workshop "sollen die oben geschilderte Entwicklung [dass es zunehmend bedeutsamer wird, der Software-Erosion erfolgreich entgegenzutreten], Erfahrungen hierzu sowie Lösungsansätze [...] beleuchtet werden." Die Sicht der Workshop-Organisatoren darauf, welche Aspekte dafür wichtig sind, beleuchtet im Aufruf die dann folgende Themenliste.

Dieser Artikel geht anhand dieser Themenliste der Frage nach, ob es in dieser Sicht, was für erfolgreiche Langlebigkeit wissenschaftlich zu bearbeiten wichtig ist, "blinde Flecken" gibt, ob also bedeutsame Teilaspekte nicht oder kaum beachtet werden – was die Chancen einer erfolgreichen Bearbeitung natürlich verringern würde.

Ich gehe dabei davon aus, dass (a) der Aufruf die Haltung der Organisatoren gut wiedergibt und (b) die Organisatoren die deutsche Forschungsszene in diesem Sektor gut repräsentieren, so dass gefundene blinde Flecken also Risiken dafür aufzeigen, dass eine gemeinschaftlich definierte größere Forschungsanstrengung in eine nicht optimale Richtung gehen könnte, wenn diese unterrepräsentierten Aspekte nicht mehr Beachtung erhalten als bisher.

Dazu stelle ich als erstes eine einfache Taxonomie relevanter Teilaspekte auf, die so allgemein ist, dass sie im Grundsatz hoffentlich konsensfähig ist (Abschnitt 2). Die Frage, welche Teile davon sich in welcher Gewichtung am stärksten für eine Erforschung aufdrängen, bleibt dabei zunächst offen. Dann vergleiche ich diese Taxonomie mit den Einträgen der Themenliste aus dem Workshop-Aufruf, um deren Orientierung und Gewichtung aufzuzeigen und eventuelle blinde Flecken zu entdecken (Abschnitt 3). Im dritten Schritt führe

[1] http://www.fzi.de/index.php/de/forschung/forschungsbereiche/se/6783

ich eine Kritik dieser gefundenen Gewichtung vor und schlage eine Ergänzung des Forschungsprogramms vor (Abschnitt 4).

2 Problembereiche aus der Vogelschau

Hier eine mögliche Sicht, was insgesamt an Facetten zu bedenken und zu erforschen ist, wenn man die Langlebigkeit von Systemen stärken möchte.

Nicht jedem wird die Einteilung und die Formulierung gefallen – etwa, weil sie vielleicht eine Gewichtung nahelegt, die er oder sie nicht teilt, oder weil manche Themen als nicht softwaretechnischer Art empfunden werden. Ich hoffe aber, dass niemand hierin klaffende Lücken entdeckt oder Einträge, die er oder sie als komplett falsch einstufen würde.

- **1.1 Produktstruktur**: Wissenschaftliche Erkenntnis darüber, was langlebigkeitsgeeignete Software-Architekturen ausmacht.
- **1.2 Entwicklungsmethoden**: Wissenschaftliche Erkenntnis darüber, wie man solche Architekturen im gegebenen Einzelfall entwickelt und dauerhaft erhält. Das betrifft alle Teilaufgaben der Softwaretechnik von der Anforderungserhebung über den Entwurf und die Realisierung bis hin zu Verifikation/Validierung sowie dem Projekt-, Prozess-, Qualitäts-, Anforderungs- und Risikomanagement.
- **1.3 Produktinfrastruktur**: Wiederverwendbare Technologien, um Strukturen nach 1.1 einfach, verlässlich und schnell umsetzen zu können. Dies betrifft geeignete Programmiersprachen, generische Komponenten (wie Betriebssysteme, DBMS, Frameworks, Middleware und andere, deren Natur zum Teil vielleicht noch gar nicht entdeckt ist), sowie bereichsspezifische Komponenten.
- **1.4 Methodeninfrastruktur**: Softwarewerkzeuge zur technischen Unterstützung der Methoden von 1.2.
- **2.1 Investitionsentscheidung für die nötigen Einmalkosten**, z.B. Beschaffung von Werkzeugen und deren Einführung. Dies geschieht bei der jeweils betroffenen Softwareorganisation. Der wissenschaftliche Aspekt daran besteht darin, zu verstehen, wie solche Entscheidungen heute zustande kommen und zu beschreiben, wie sie idealerweise zustande kommen sollten.
- **2.2 Investitionsentscheidung für die nötigen Kosten im Einzelfall**. Das betrifft insbesondere die Bereitschaft, entsprechend qualifiziertes Personal zu bezahlen und diesem die nötige Zeit zum sauberen Verfolgen der Methoden von 1.2 zur Verfügung zu stellen. Wissenschaftlicher Aspekt analog zu 2.1.
- **3.1 Individuelle Möglichkeiten**: Die nötigen Fähigkeiten (Talent) und Fertigkeiten (Ausbildung und praktische Erfahrung) bei den einzelnen handelnden Softwareingenieur/inn/en, um die Grundlagen nach 1.1 bis 1.4 geeignet einsetzen zu können. Wissenschaftliche Fragestellungen in diesem Bereich betreffen die Beschreibung,

welches Maß an Fähigkeiten und Fertigkeiten in welchen Bereichen von 1.1 bis 1.4 nötig wäre, welches Maß an Fähigkeiten heute typisch ist und mit welchen Methoden man die Fertigkeiten optimal trainieren kann.

- **3.2 Individuelle Bereitschaft**: Die nötige Disziplin bei den einzelnen handelnden Softwareingenieur/inn/en, um die Grundlagen nach 1.1 bis 1.4 im konkreten Einzelfall (insbesondere unter Zeitdruck) auch tatsächlich hinreichend vollständig und korrekt einzusetzen. Wissenschaftliche Fragen in diesem Bereich: Wo mangelt es wie sehr an solcher Disziplin? Warum? Wie kann man das reduzieren?

3 Vergleich mit dem Workshop-Aufruf

Hier gebe ich die Themenliste aus dem Workshop-Aufruf wieder und markiere jeweils, welchen Bereichen der obigen Taxonomie ich die Einträge zuordne. Über diese Zuordnung mag man hier und da streiten, das ist nicht schlimm: Für die Zwecke dieses Artikels reicht eine ungefähre Zustimmung aus.

- Anpassbarkeit und Evolution
  - evolutionsfähige Software-Architekturen (1.1)
  - Anpassbarkeit von Anwendungen an wechselnde Plattformen, sich verändernde Anforderungen und Randbedingungen (1.1, 1.2)
  - Koevolution zw. Anforderungen, Architektur und Implementierung (1.1, 1.2)
  - Variabilität auf Anforderungs- und Architekturebene wie z.B. in Produktlinienansätzen (1.1, 1.2)
- Reengineering
  - Verfahren zur Erkennung von Legacy-Problemen und ihrer Ursachen (1.2)
  - Metamodelle für die Integration von Wissen über Software (1.2)
  - modellbas. Reengineering und Refactoring bestehender Systeme (1.2, 1.3, 1.4)
  - Modellextraktion durch Programmverstehen (1.2, 1.4)
- Entwicklungs-, Betriebs- und Wartungsmethoden
  - modellbasierte und architekturzentrierte Methoden und Techniken für (die Integration von) Entwicklung und Betrieb langlebiger Softwaresysteme (1.2, 1.4)
  - Business-IT-Alignment, geschäftsorientierte Evolution der IT (1.2, evtl. 2.2)
  - Lebenszyklus-übergreifendes Anforderungsmanagement (1.2)
  - Roundtrip-Engineering, integrierte Entwicklung auf Modell- und Codeebene (1.2, 1.4)
  - virtuelle Maschinen zur Erzielung von Plattformunabhängigkeit (1.1, 1.3)
  - kooperative Entwicklungsmethoden (1.2, evtl. 3.1/3.2), Kommunikations- und Kooperationswerkzeuge und Entwicklungsumgebungen, die die enge Zusammenarbeit von Domänenexperten, Anwendern und Softwareentwicklern unterstützen (1.4)

- Qualitätsmanagement
  - Qualitätsanforderungen an langlebige Softwaresysteme (1.2)
  - Qualitätsbewertung langlebiger Softwaresysteme (1.2)
  - Qualitätssicherung beim Betrieb langlebiger Softwaresysteme (1.2)

Wie man sieht, deckt der Workshop nur die Bereiche 1.1 bis 1.4 der Taxonomie ab (reine Software-Technik), aber berührt allenfalls zart die Sektoren 2.1/2.2 (Verhältnisse in SW-Organisationen) und 3.1/3.2 (individuelle menschliche Faktoren).

4 Kritik

Dies ist kein Übersichtspapier und deshalb werde ich gar nicht erst versuchen, die folgenden Aussagen mit wissenschaftlicher Literatur zu untermauern, sondern sie ganz als persönliche Ansichten formulieren:

1. Meiner Ansicht nach sind die weitaus wirksamsten Faktoren für die heute zu beobachtenden (und in der Tat heftigen) Langlebigkeitsprobleme in den Bereichen 2.2, 3.1 und 3.2 zu suchen.[2]

2. Selbst gewaltige Durchbrüche in den Bereichen 1.1 bis 1.4 würden daran nur graduell etwas ändern, weil selbst mit der bestmöglichen Technologie ein erheblicher Restwiderspruch zwischen "kurzfristig günstig" und "langfristig günstig" bestehen bleiben wird.[3] Diesen gibt es in so gut wie allen Lebensbereichen, auch den industriellen.

3. Folglich kann eine Wirksamkeit von Fortschritten in den Bereichen 1.1. bis 1.4 am besten sichergestellt werden, indem man zuvor oder zugleich die "Hygienefaktoren" 2.1 bis 3.2 studiert und sich bei der Bearbeitung von 1.1 bis 1.4 bestmöglich an die dort gefundenen (oder erzielten) Bedingungen anpasst.

4. Damit das gelingen kann, muss der Kreis der am Forschungsprogramm beteiligten um Personen ergänzt werden, die an der nötigen empirischen Forschung interessiert und in entsprechenden sozialpsychologischen und mikrosoziologischen Methoden versiert sind. Diese gibt es zwar nicht zahlreich, sie sind aber vereinzelt in diversen Fächern zu finden, z.B. in Informatik, Soziologie, Psychologie, Wirtschaftswissenschaften.

[2] Ich vermute, dass auch die große Mehrheit ganzheitlich denkender Praktiker dem zustimmt.

[3] Als eindrucksvolles Beispiel bietet sich aktuell die modellbasierte Softwareentwicklung an, die trotz ihrer großen potentiellen Vorteile nur sehr schleppend in der Praxis Fuß fasst, was zum Teil daran liegen dürfte, dass (a) der hilfreiche Einsatz nicht einfach ist (3.1) und (b) viele der Vorteile sich erst langfristig materialisieren (2.2, 3.2).