Nelufar Ulfat-Bunyadi

(nelufar.ulfat-bunyadi@paluno.uni-due.de)
ist Wissenschaftliche Mitarbeiterin am Forschungsinstitut „paluno" (The Ruhr Institute for Software Technology) der Universität Duisburg-Essen. Sie promoviert auf dem Gebiet „Kontextanalyse im Requirements Engineering".

Die unterschätzte Bedeutung von Annahmen in Anforderungsspezifikationen

Bei der Definition von Anforderungen an ein System werden auch Annahmen über den Systemkontext getroffen. Beispielsweise wird von Nutzern oder Partnersystemen ein bestimmtes Verhalten erwartet. Diese Annahmen spielen eine wesentliche Rolle. Eine ungültige Annahme kann schlimmstenfalls dazu führen, dass das System im Betrieb versagt, obwohl alle Anforderungen richtig umgesetzt wurden. Ungültige Annahmen werden im Entwicklungsprozess häufig nicht aufgedeckt, weil das System im Rahmen von Qualitätssicherungsaktivitäten hauptsächlich dahingehend geprüft wird, ob es die spezifizierten Anforderungen erfüllt. Ob die Anforderungen selbst auf gültigen Annahmen über den Kontext beruhen, wird dabei nicht geprüft. Aus diesem Grund müssen Annahmen systematisch dokumentiert werden, damit sie im weiteren Systemlebenszyklus einem Review unterzogen und auf ihre Aktualität und Gültigkeit hin geprüft werden können. Damit nachvollziehbar ist, welche Anforderungen von späteren Änderungen in den Annahmen betroffen sind, müssen die Anforderungen mit den ihnen zugrunde liegenden Annahmen verknüpft werden. Dieser Beitrag stellt die Herausforderungen im Umgang mit Annahmen heraus und zeigt auf, dass existierende Entwicklungsprozesse und Methoden die Identifikation, Dokumentation und Überprüfung von Annahmen nicht ausreichend unterstützen.

Was sind Annahmen und warum sind sie wichtig?

Im Rahmen des Requirements Engineering werden nicht nur Anforderungen an ein künftiges System definiert. Bei der Definition der Anforderungen werden auch viele Annahmen über den Systemkontext (vgl. [Poh10]) getroffen (siehe Abbildung 1).

Bei der Entwicklung einer Navigationsanwendung für ein Smartphone beispielsweise wird die Annahme getroffen, dass eine Internetverbindung immer vorhanden ist. Später im Betrieb funktioniert die Navigationsanwendung aber manchmal nicht. Der Grund dafür liegt darin, dass die Annahme, die bei der Entwicklung getroffen wurde, ungültig ist. Es gibt Gebiete mit schlechter Netzabdeckung (z. B. in ländlichen Gegenden oder im Gebirge). Daher ist die Annahme, dass die Internetverbindung immer vorhanden ist, falsch.

Annahmen können sich aus verschiedenen Gründen als ungültig herausstellen. Einerseits können tatsächlich Veränderungen im Kontext dazu führen, dass ursprünglich gültige Annahmen ungültig werden. Andererseits können falsche oder fehlende Informationen über den Kontext zu dem Zeitpunkt, an dem die Anforderungen definiert wurden, die Ursache für ungültige Annahmen sein.

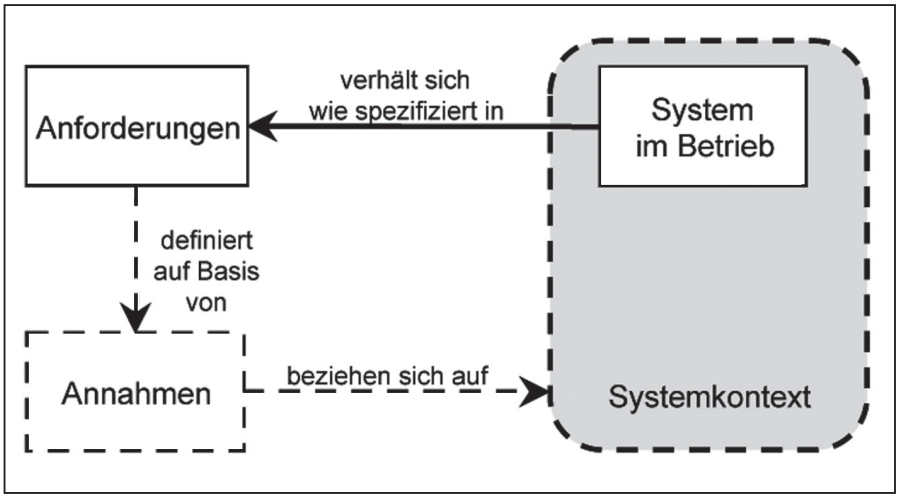

Abb. 1: Beziehungen zwischen Anforderungen, Annahmen und Kontext

In der Vergangenheit haben ungültige Annahmen über den Kontext bereits mehrfach zu kritischen Situationen geführt, in denen die Systeme im Betrieb versagt haben. Der Lufthansa-Flug A320 nach Warschau 1993, der U-Bahn-Unfall in London 1995 und die Ariane 5-Explosion 1996 sind nur einige prominente Beispiele dafür (siehe Tabelle).

System	Anforderungen	Annahmen	Konsequenz	Irrtum
Autopilot der Lufthansa A320 nach Warschau (beschrieben in [Lad93])	R-1: Der Umkehrschub soll freigegeben werden genau dann, wenn das Flugzeug sich auf der Landebahn bewegt.	Das Flugzeug bewegt sich auf der Landebahn genau dann, wenn die Räder sich drehen.	Das Flugzeug schoss über die Landebahn hinweg. Es gab viele Verletzte und einige Tote. Der Autopilot erlaubte es bis 9 Sekunden nach der Landung im Sturm auf nasser Landebahn nicht, dass der Umkehrschub aktiviert wird.	Die Annahme ist ungültig, da sich ein Flugzeug auf der Landebahn vorwärts bewegen kann, auch ohne dass die Räder sich drehen (z. B. bei Aquaplaning).
Trägheitsnavigationssoftware der Ariane 5 (wiederverwendete Komponente der Ariane 4; beschrieben in [Lio96])	R-1: Die Abgleichsfunktion soll die horizontale Beschleunigung zur Verfügung stellen.	Während der ersten 40 Sekunden des Fluges ist die Flugbahn so, dass der Wert, der die horizontale Beschleunigung angibt, den definierten Maximalwert nicht überschreiten kann.	Die Ariane 5 hatte eine horizontale Beschleunigung, die 5 Mal größer war als die horizontale Beschleunigung der Ariane 4. Dieser Wert überschritt somit den Maximalwert und führte zum Eintritt einer Ausnahmebedingung. Daraufhin schaltete sich die Trägheitsnavigationssoftware ab. Dies löste eine Kette weiterer Ereignisse aus.	Die Annahme war ungültig, da die Ariane 5 eine hohe initiale Beschleunigung und eine andere Flugbahn hatte als die Ariane 4.
Software zur elektronischen Steuerung der Handbremse im Fahrzeug (beschrieben in [Lar09])	R-1: Die Handbremse soll gelöst werden genau dann, wenn der Fahrer anfahren möchte.	Die Motordrehzahl erhöht sich genau dann, wenn der Fahrer das Gaspedal betätigt.	An einem heißen Sommertag stieg der Fahrer des Fahrzeugs bei laufendem Motor und angezogener Handbremse aus, um das Tor zu öffnen. Da die Fahrzeugtür offen stand, schaltete sich die Klimaanlage automatisch ein. Die Motordrehzahl erhöhte sich, die Handbremse wurde gelöst, der Fahrer wurde von seinem eigenen Fahrzeug überrollt.	Die Annahme ist ungültig, da die Motordrehzahl sich nicht nur bei Betätigung des Gaspedals erhöht, sondern bspw. auch, wenn sich die Klimaanlage einschaltet.
Bremssystem in U-Bahnen (beschrieben in [HRH01])	R-1: Bei einem roten Signal soll das System eine Bremsung einleiten.	Die nach dem Signal zur Verfügung stehende Strecke reicht aus, damit der Zug zum Stehen kommt.	Die Distanz zwischen den Signalen war kürzer als der Bremsweg der Züge.	Die Annahme ist ungültig, da der Abstand zwischen den Signalen 1918 festgelegt wurde. Damals waren Züge kürzer, leichter und langsamer als Züge im Jahre 1995.

Tab.: Beispiele für ungültige Annahmen

Häufig werden ungültige Annahmen und daraus resultierende Anforderungsfehler deshalb nicht aufgedeckt, weil das System im Rahmen von Qualitätssicherungsaktivitäten hauptsächlich dahingehend geprüft wird, ob es die spezifizierten Anforderungen erfüllt. Ob die Anforderungen überhaupt auf gültigen Annahmen über den Kontext beruhen, wird dabei nicht geprüft.

Aus diesem Grund müssen Annahmen dokumentiert werden. Nur so können sie im weiteren Systemlebenszyklus einem Review unterzogen und auf ihre Aktualität und Gültigkeit hin geprüft werden. Darüber hinaus müssen die Anforderungen mit den ihnen zugrunde liegenden Annahmen verknüpft werden, damit im Fall von späteren Änderungen in den Annahmen nachvollziehbar ist, welche Anforderungen betroffen sind (siehe Tabelle).

Verschiedene Referenzstrukturen für die Spezifikation von Anforderungen sehen auch einen separaten Abschnitt für die

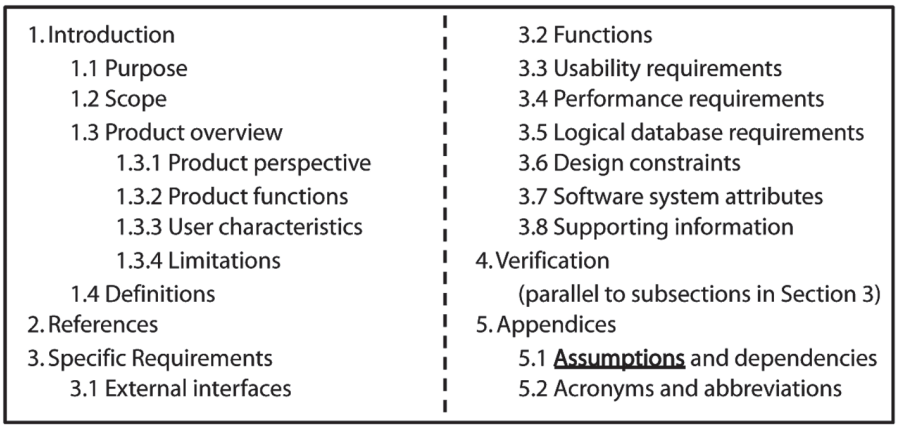

Abb. 2: Beispielhafte Struktur einer SRS (aus [III11])

Dokumentation der Annahmen vor, die den spezifizierten Anforderungen zugrunde liegen (vgl. z. B. die Strukturen für die System Requirements Specification (SyRS) und die Software Requirements Specification (SRS), die in [III11] beschrieben sind, oder das Volère Requirements Specification Template aus [RuR06]).

Herausforderungen für die Praxis

Im Allgemeinen bezeichnet eine Annahme eine Aussage, die als wahr akzeptiert wird, ohne dass ein Beweis dafür vorliegt. Die Besonderheit im Requirements Engineering liegt darin, dass die Entwickler darauf angewiesen sind, dass die Annahmen, die sie über den Systemkontext treffen, wahr bzw. gültig sind, und dass sie keinen Einfluss auf die Gültigkeit der Annahmen haben, d. h. sie können sie nicht „wahr machen“ (vgl. [Ale06]).

Diese Charakteristika von Annahmen führen beispielsweise zu folgenden Herausforderungen in Entwicklungsprojekten:

H1. Es gibt Annahmen, die während der Systemkonzeption nicht mit ausreichender Sicherheit untermauert werden können. Was passiert mit solchen Annahmen?

H2. Wiederverwendung von Systemkomponenten: Können die Annahmen über den Kontext übernommen werden, wenn die Anforderungen 1 zu 1 übernommen wurden?

H3. Annahmen sind nicht immer als solche erkennbar! Was die Entwickler für Fakten halten, kann sich im weiteren Verlauf der Entwicklung als Annahme herausstellen. Wie sollten Entwickler damit umgehen?

H4. Verteilte Entwicklung: Werden Teilsysteme eines Gesamtsystems von unterschiedlichen Abteilungen entwickelt, so trifft jede Abteilung Annahmen über Partnersysteme. Wie müssen sich Abteilungen in Bezug auf die Annahmen abstimmen?

Zu H1.
Während der Systementwicklung werden verschiedene Arten von Annahmen getroffen. Für einige Annahmen ist es möglich, diese bis zur Freigabe der Anforderungsspezifikation aufzulösen und sie in Anforderungen oder Rahmenbedingungen zu überführen (wie in [RuR06] gefordert). Beispiele hierfür sind Annahmen über das Leistungsvermögen hinzugekaufter Komponenten. Durch eine Ausführung der Komponenten beispielsweise im Rahmen einer Simulation kann geprüft werden, ob sie den Erwartungen entsprechen und die Annahmen sich somit als wahr bzw. gültig herausgestellt haben oder nicht.

Für viele andere Annahmen ist dies aber nicht möglich, weil ein derartiger Beweis nicht erbracht werden kann. Beispiele für solche Annahmen sind Annahmen über Benutzergruppen, über den Markt bzw. ein Marktsegment, Annahmen über Konkurrenzprodukte, Annahmen über mögliche Bedrohungen für das System. Häufig bleiben diese Annahmen Annahmen, selbst wenn das System ausgeliefert und in Betrieb ist.

Zu H2.
Wie das Ariane-5-Beispiel zeigt (siehe Tabelle), können bei der Wiederverwendung von Komponenten die Annahmen, die für die Komponente in ihrer ursprünglichen Einsatzumgebung gültig waren, nicht einfach übernommen werden. Selbst wenn die Anforderungen an die Komponente exakt dieselben Anforderungen sind wie im Altsystem, müssen die ursprünglichen Annahmen, die diesen Anforderungen zugrunde lagen, im neuen Kontext auf Aktualität und Gültigkeit geprüft werden.

Zu H3.
Das Ariane 5-Beispiel veranschaulicht auch, dass Annahmen nicht immer als solche erkennbar sind. Im Laufe der Zeit können sich Fakten auch als Annahmen herausstellen. Die in der Tabelle gezeigte Annahme beispielsweise, auf der die Trägheitsnavigationssoftware der Ariane 5 basierte, war für dieselbe Komponente in der Ariane 4 ein Fakt über den Systemkontext. Solche kritischen Fakten über den Systemkontext müssen daher hinterfragt und im Zweifelsfall als Annahmen dokumentiert werden.

Zu H4.
Werden Teilsysteme eines Gesamtsystems von unterschiedlichen Abteilungen und Zulieferern entwickelt, so trifft jede Abteilung bei der Entwicklung eines Teilsystems auch Annahmen über Partnersysteme, z. B. dass sich ein Bussignal so verhält wie in vergangenen Projekten auch. Da jede Abteilung in jedem Projekt aber auch weiterentwickelt, können sich solche Annahmen als ungültig herausstellen. Nur wenn die Partnerabteilung explizit macht, dass sich etwas gegenüber früheren Projekten geändert hat, erfahren die anderen Abteilungen überhaupt von dieser Veränderung.

Häufig ist dies aber nicht der Fall. In einer solchen Projektsituation muss jede Abteilung die von ihr getroffenen Annahmen dokumentieren und mit den von ihr definierten Anforderungen verknüpfen. Darüber hinaus sollte jede Abteilung kritische Annahmen und Fakten den jeweiligen Partnerabteilungen vorlegen, damit diese ungültige Annahmen aufdecken können.

Mangelnde Unterstützung durch Entwicklungsprozesse und Methoden

Aus den beschriebenen Herausforderungen folgt,

- dass Annahmen zunächst identifiziert bzw. aufgedeckt werden müssen, denn viele Annahmen werden von den Entwicklern nicht bewusst getroffen
- dass Annahmen dokumentiert werden müssen, da die meisten Annahmen nicht vor Freigabe der Anforderungsspezifikation aufgelöst werden können

- dass Annahmen geprüft/validiert werden müssen, weil sich der Kontext mit der Zeit verändert oder die Entwickler ggf. an Wissen über den Kontext gewinnen
- dass Annahmen mit Anforderungen verknüpft werden müssen, um bei ungültigen Annahmen die Auswirkungen auf Anforderungen feststellen zu können.

In Bezug auf diese notwendigen Entwicklungsaktivitäten ergeben sich folgende Forschungsfragestellungen:

1. Identifikation von Annahmen:
   - Wo soll nach Annahmen gesucht werden?
   - Welche Anhaltspunkte gibt es für verborgene Annahmen?
   - Welche Techniken eignen sich, um Annahmen zu identifizieren, die sich hinter Anforderungen verbergen?
2. Dokumentation von Annahmen:
   - Auf welche Art und Weise sollen Annahmen dokumentiert werden?
   - Welche Vor- und Nachteile bringen die unterschiedlichen Dokumentationsformen mit sich?
   - In welchem Detaillierungsgrad sollen Annahmen dokumentiert werden?
   - Wie wirken sich Projekt- und Systemeigenschaften auf die Dokumentation von Annahmen aus?
3. Validierung von Annahmen:
   - Gegen wen oder was können Annahmen geprüft werden? Welchen Stakeholdern können Annahmen für eine Validierung vorgelegt werden?
   - Welche Techniken können eingesetzt werden, um Annahmen auf Gültigkeit und Aktualität zu prüfen?
4. Nachverfolgung von Annahmen:
   - Welche Typen von Nachvollziehbarkeitsbeziehungen müssen zwischen textuell bzw. modellbasiert dokumentierten Annahmen und Anforderungen aufgezeichnet werden?
   - Wie kann der Aufwand für das Erstellen, aktuell Halten und Auswerten der Nachvollziehbarkeitsbeziehungen in einem vertretbaren Rahmen gehalten werden?

Die meisten dieser Fragen bleiben in der Literatur unbeantwortet. Existierende Ansätze, die Unterstützung beim Umgang mit Annahmen im Requirements Engineering bieten, fokussieren auf die Dokumentation der Annahmen und ihrer Beziehungen zu Anforderungen (vgl. z. B. [Lar09], [WSB11], [BuF03], [MMDR05], [HLMN06], [EuY09]). Auch Kontextmodellierungsansätze wie [Jac01], [Yu93]/[Yu97], [Bea04] oder Domänenmodellierungsansätze wie [Sut02] können genutzt werden, um Annahmen zu dokumentieren, da Annahmen im Grunde Wissen über den Kontext darstellen.

Allerdings hat sich keiner dieser Ansätze in der Praxis durchgesetzt. Die Ursache liegt darin, dass die gebotene Unterstützung vor dem Hintergrund der im vorherigen Abschnitt beschriebenen Herausforderungen unzureichend ist. Insbesondere die Identifikation bzw. Aufdeckung von verborgenen Annahmen, die vor einer Dokumentation erfolgen muss, wird von existierenden Ansätzen kaum unterstützt.

Aber auch in Bezug auf die Dokumentation, Validierung und Nachverfolgung von Annahmen weisen die Ansätze Defizite auf. Forschung, Industrie und Werkzeughersteller müssen zusammenarbeiten, um Entwicklungsprozesse, Methoden und Werkzeuge so zu erweitern, dass Entwicklern ein systematischer Umgang mit Annahmen ermöglicht wird und Fehler aufgrund ungültiger Annahmen mit den entsprechenden Konsequenzen (siehe Tabelle) vermieden werden können. ■

Referenzen

[AleO6] I. Alexander: "10 Small Steps to Better Requirements", IEEE Software, Vol. 23, Nr. 2, März 2006, S. 19-21.

[Bea04] Bresciani, P.; Perini, A.; Giorgini, P.; Giunchiglia, F.; Mylopoulos, J.: Tropos: An Agent-Oriented Software Development Methodology. In: Autonomous Agents and Multi-Agent Systems, 8 (2004) 3, S.203--236

[BuF03] D. Bush, A. Finkelstein: "Requirements Stability Assessment Using Scenarios", Proc. 11th IEEE International Requirements Engineering Conference (RE 03), IEEE Press, 2003, S. 23-32.

[EuY09] G. Elahi, E. Yu: "Trust Trade-off Analysis for Security Requirements Engineering", Proc. 17th IEEE International Requirements Engineering Conference, IEEE, 2009.

[HLMN06] C.B. Haley, R.C. Laney, J.D. Moffett, B. Nuseibeh: "Using trust assumptions with security requirements", Requirements Engineering (2006) 11:138-151.

[HRH01] J. Hammond, R. Rawlings, A. Hall: "Will It Work?", Proc. 5th IEEE International Symposium on Requirements Engineering, IEEE Press, Los Alamitos, 2001, S. 102-109.

[Iso11] ISO/IEC/IEEE: "International Standard ISO/IEC/IEEE 29148 Systems and Software Engineering – Life Cycle Processes – Requirements Engineering", ISO/IEC/IEEE 29148:2011(E), 2011.

[Jac01] M. Jackson, Problem Frames – Analysing and Structuring Software Development Problems, ACM Press, New York, NY 2001.

[Lad93] P. Ladkin: "More news on the Lufthansa A320 accident in Warsaw", in: The Risks Digest, Forum on Risks to the Public in Computers and Related Systems, Vol. 15, Nr. 30, Dezember 1993.

[Lar09] A. van Lamsweerde: "Requirements Engineering – From System Goals to UML Models to Software Specifications", John Wiley & Sons, West Sussex, 2009.

[Lio96] J. L. Lions: "ARIANE 5 Flight 501 Failure: Report of the Inquiry Board", Juli 1996.

[MMDR05] A. Miranskyy, N. Madhavji, M. Davison, M. Reesor: "Modelling Assumptions and Requirements in the Context of Project Risk", Proc. 13th IEEE International Conference on Requirements Engineering (RE 05), IEEE Press, 2005, S. 471-472.

[Poh10] K. Pohl: "Requirements Engineering – Fundamentals, Principles, and Techniques", Springer, Berlin, Heidelberg, 2010.

[RuR06] S. Robertson, J. Robertson: "Mastering the Requirements Process", 2. Auflage, Addison-Wesley, Amsterdam, 2006.

[Sut02] A. Sutcliffe: "The Domain Theory – Patterns for Knowledge and Software Reuse", Lawrence Erlbaum Associates, Mahwah, New Jersey, 2002.

[WSB11] K. Welsh, K., P. Sawyer, N. Bencomo, "Run-time Resolution of Uncertainty", Proc. 19th IEEE International Requirements Engineering Conference, IEEE Press, 2011, S. 355-356.

[Yu93] E. Yu: An Organisational Modelling Framework for Multiperspective Information System Design. In: J. Mylopoulos et al. (Eds.): Requirements Engineering 1993 – Selected Papers, Tech Report DKBS-TR-92-2, Department of Computer Science, University of Toronto, Toronto, 1993, pp. 66-86.

[Yu97] E. Yu: Towards Modeling and Reasoning Support for Early-phase Requirements Engineering. In: Proceedings of the 3rd International Symposium on Requirements Engineering (RE'97), IEEE Computer Society Press, Los Alamitos, 1997.