Analyse der sozialen Teamstruktur in Softwareprojekten

Johannes Meißner
Research and Development
SK8DLX Services GmbH Jena
johannes.meissner@sk8dlx.de

Frederik Schulz
Project Management
SK8DLX Services GmbH Jena
frederik.schulz@sk8dlx.de

Wilhelm Rossak
Lehrstuhl für Softwaretechnik
Friedrich-Schiller-Universität Jena
wilhelm.rossak@uni-jena.de

Abstract: Um den organisatorischen Kontext von Softwareentwicklungsprojekten abzubilden, haben wir das Goal-orientierte OrCA-Framework entworfen. Damit lassen sich die Abhängigkeiten zwischen Projektbeteiligten, ihren Beiträgen zur Umsetzung sowie den Ergebnissen ihrer Arbeit modellieren. Durch die Integration von OrCA mit dem Teamrollenkonzept von Belbin reichern wir diese Modelle mit Informationen zu persönlichen und sozialen Charakteristika der Projektbeteiligten an. Diese Arbeit zeigt, wie wir diese Modelle formalisieren, um eine Grundlage für eine automatische Analyse der Qualität der Teamzusammenstellung unter sozialen Aspekten zu schaffen.

1 Einführung

Verpasste Deadlines, überzogene Budgets oder ein nicht verwendbares Ergebnis: Softwareprojekte können zu wahren Alpträumen mutieren [Bro87]. Lösungsansätze, um die Produktivität und Verlässlichkeit zu erhöhen, basieren meist auf der Entwicklung neuer Technologien oder Tools, die den Entwicklungsprozess unterstützen sollen. Doch schon Conway merkte an, dass Organisationen nur Systeme produzieren können, deren Aufbau eine Kopie ihrer organisatorischen Struktur ist [Con68].

Mit Hilfe unseres Frameworks *Organizational Context Analysis* (OrCA) kann der organisatorische Kontext und sein wechselseitiger Einfluss mit der Softwarearchitektur erfasst werden [SMR13]. Dazu wird zunächst die *Work-Breakdown-Structure* [BCK03] modelliert: Wer (*Contributors*) beteiligt sich durch welche Aufwände (*Realizations*) an welchen Projektergebnissen (*Deliverables*)? Durch die Einführung von Prädikaten kann diese Struktur zusätzlich mit technischen und organisatorischen Eigenschaften angereichert werden, z. B. Skillprofilen, Entwicklungskosten oder Technologieentscheidungen [SMR13].

Die wichtigste Ressource einer Organisation oder eines Projektes sind aber die Mitarbeiter [Tam89]. In der Softwareentwicklung werden computerbasierte Systeme von Menschen für Menschen umgesetzt [Gog93]. Während die Einteilung von Mitarbeitern zu Projekten häufig auf *funktionalen* Fakten basiert, also etwa Erfahrung, Fähigkeit und Hierarchieposition, werden *nicht-funktionale* Charakteristika, wie der soziale Umgang oder die

persönliche Arbeitsweise, oft ausgeblendet [Con95]. Auch Boehm sah in der Betrachtung von persönlichen Attributen und zwischenmenschlichen Beziehungen die größten Möglichkeiten zur Produktivitätssteigerung innerhalb der Softwareentwicklung [Boe81].

In verschiedenen E-Commerce-Projekten konnten wir die Wichtigkeit dieser „weichen" Faktoren nachvollziehen. Besonders die Zusammenstellung von Teams und die Beziehungen zwischen den Teammitgliedern rückten häufig in den Fokus. Daher haben wir das OrCA-Framework in [MS14] erweitert, so dass soziale Aspekte mit einbezogen werden können. Dazu bauen wir auf dem *Teamrollen*-Modell von Belbin auf [Bel10]: Jedem Mitglied einer Organisation kann eine Rolle zugewiesen werden, die sein typisches Verhalten in einem Team beschreibt. Mit einer Reihe von Hinweisen zielt Belbin darauf ab, durch die Balance und die Diversität dieser Rollen ein leistungsfähiges und gut funktionierendes Team zu formen. In den folgenden Abschnitten zeigen wir an einem Beispiel, wie Organisation und Rollenverteilung mit dem OrCA-Modell erfasst werden können. Anschließend formalisieren wir unser Modell und übersetzen die Hinweise von Belbin in einen Regelsatz. Damit schaffen wir eine Grundlage, um künftig die Eignung von Teamkompositionen automatisiert überprüfen zu können. Diese Arbeit gibt zudem einen kurzen Ausblick auf eine technische Umsetzung dieser automatisierten Analyse.

2 Grundlagen: OrCA und Belbin

OrCA-Modelle basieren auf der *Goal Requirements Language* (GRL) [LY04] und bestehen aus drei Schichten. Die anvisierten Arbeitsergebnisse eines Projekts, also z. B. Softwarekomponenten oder -bibliotheken, werden in der untersten Schicht, den *Deliverables*, erfasst. Zu deren Darstellung bedienen wir uns der *Resource*, einem Element aus der GRL, das eine physische oder logische Entität repräsentiert. Die mittlere *Realizations*-Schicht enthält die Aufwände, die zur Umsetzung dieser Deliverables notwendig sind, etwa Planungs- oder Implementierungsaufgaben. Sie werden durch *Task*-Elemente dargestellt, mit denen in der GRL Aktivitäten repräsentiert werden. Die oberste Schicht der *Contributors* zeigt schließlich, welche Mitarbeiter an der Entwicklung beteiligt sind. Hier greifen wir auf das *Actor*-Element der GRL zurück. Jede Aufgabe in der Realizations-Schicht ist genau einem Contributor und einem Deliverable zugeordnet. Wir modellieren dies mit *Dependency-Links*, mit denen die GRL Abhängigkeiten zwischen Elementen repräsentiert: Ein Projektergebnis hängt ab von einer Reihe von Aufgaben, die jeweils vom zuständigen Projektmitglied abhängen. Die OrCA-Schichten bilden auf diese Weise eine Schablone, um eine Work-Breakdown-Structure mit einem Goal-orientierten Modell abzubilden. Eine solche Struktur kann beliebig um weitere GRL-Elemente erweitert werden, etwa um *Goals*, mit denen Ziele und Motivationen dargestellt werden – die Mächtigkeit der GRL soll nicht eingeschränkt werden.

Über die Elemente der GRL hinaus bietet OrCA mit einem *Predicate*-Element auch die Möglichkeit, das Modell mit Faktoren und Eigenschaften wie Skillprofilen für Contributors oder Technologieentscheidungen für Deliverables anzureichern [SMR13]. Dieses Konzept bietet für OrCA zwar einen großen Mehrwert, es wird in dieser Arbeit aber nicht weiter thematisiert.

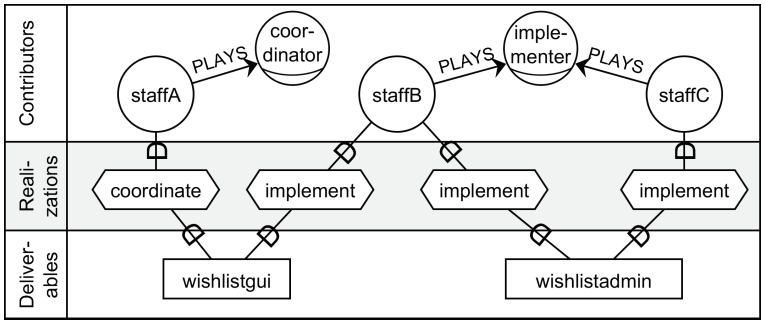

Abbildung 1: Der Contributor staffA mit der Teamrolle coordinator beteiligt sich durch die Planungsaufgabe coordinate am Deliverable wishlistgui (Shopfrontend). Durch die Umsetzung implement ist auch staffB mit der Teamrolle implementer daran beteiligt. Zusätzlich arbeitet staffB gemeinsam mit staffC an der Umsetzung des Backendtools wishlistadmin. Auch staffC wird als typischer implementer eingestuft. Gegenüber der Praxis ist dieses Beispiel stark vereinfacht.

Durch die einfache Erfassung der *sozialen Prädisposition* eines Individuums findet das *Teamrollenmodell* von Belbin [Bel10] weite Verbreitung in der Wirtschaft. Es unterscheidet zwischen neun verschiedenen Teamrollen. Eine kommunikationsorientierte Teamrolle ist beispielsweise der *Coordinator*, dem das Delegieren von Aufgaben auf vertrauensvolle und selbstsichere Art leicht von der Hand geht. Der handlungsorientierte *Implementer* benötigt hingegen klare Anweisungen und setzt diese effizient und zuverlässig in die Tat um. Ein *Plant* fungiert dagegen als innovativer Ideengeber. Um die Rolle einzelner Teammitglieder zu bestimmen, bietet das Konzept unter anderem einen einfachen Fragebogen. Belbin propagiert, dass eine große Diversität dieser Rollen in einem Team zu dessen besserer Performance beiträgt. Es kommt insbesondere nicht nur auf die Individuen, sondern auch auf deren Umgang miteinander an.

Um die beiden Ansätze zu integrieren, ordnen wir jedem OrCA-Contributor seine jeweilige Belbin-Teamrolle zu. Dazu verwenden wir die *PLAYS*-Beziehung und das *Role*-Element, die ebenfalls der GRL entlehnt sind [MS14]. Das anonymisierte Beispiel in Abb. 1 zeigt einen Ausschnitt eines solchen Modells aus der Praxis. Es bezieht sich auf ein Projekt zur Umsetzung einer Wunschlistenfunktion für einen Onlineshop. Die Projektziele setzten sich aus Änderungen im Shopfrontend für den Kunden und der Entwicklung eines Backend-Werkzeugs für die Shopadministratoren zusammen. Das Teilprojekt für das Shopfrontend zeichnete sich durch eine Vielzahl unterschiedlicher Beteiligter aus: Softwareentwickler, Designer und Marketing waren an der Entwicklung beteiligt. Dagegen war das Backendtool zwar eine komplexe, aber rein technische Aufgabe. Demgegenüber ist das abgebildete Modell jedoch stark vereinfacht und bewusst unvollständig. Wir nutzen dies im nachfolgenden Abschnitt, um zu zeigen, dass mit formalisierten Belbin-Regeln solche Defizite leicht erkannt werden können.

3 Regelbasierte Analyse der Teamstruktur

Belbin postuliert eine Reihe von Hinweisen, welche Zusammenstellung von Rollen zu einer guten oder schlechten Teamleistung führt. Dabei betrachtet er auch den Zusammenhang der vorhandenen Rollen mit der Teamgröße. Aus diesen Hinweisen haben wir einen Satz von Regeln extrahiert. Nachfolgend sind drei Beispiele für solche Regeln zu möglichen Rollenkombinationen aufgeführt. Ergänzende Hinweise zu geeigneten Teamzusammenstellungen sind [Bel10] zu entnehmen.

- R_1 Nicht alle Teammitglieder dürfen die Rolle *Implementer* haben. Diese arbeiten zwar gewissenhaft, ihnen fehlt es aber sowohl an Führung als auch an kreativen Ideen.
- R_2 In einem Team mit drei oder weniger Mitgliedern ist ein *Coordinator* überflüssig.
- R_3 Ein Team mit mehr als einem *Plant* (Ideengeber) produziert ein Überangebot an Ideen, die sich gegenseitig ausschalten können.

An dieser Stelle wollen wir unser OrCA-Framework und das Teamrollenkonzept formalisieren. Damit bilden wir die Grundlage für eine automatisierte Analyse unserer Modelle, um Probleme mit der Teamzusammenstellung frühzeitig zu erkennen.

Definition 1 *Es sei* $R_{\mathsf{Belbin}} = \{\mathsf{coordinator}, \mathsf{implementer}, \mathsf{plant}, \ldots\}$ *die Menge aller Teamrollen gemäß des Belbin-Modells in [Bel10].*

Definition 2 *Für ein gegebenes OrCA-Modell* M *sollen weiterhin folgende Definitionen gelten:*

- $\mathsf{CONTR}(M)$ *sei die Menge aller Contributors des Modells* M*;*
- $\mathsf{PLAYS}(M) \subseteq \mathsf{CONTR}(M) \times R_{\mathsf{Belbin}}$ *sei die binäre Relation, die alle Paare von Contributors* c *und von Teamrollen* r *enthält, bei denen* c *und* r *in* M *durch eine* PLAYS*-Beziehung verbunden sind;*
- $\mathsf{TEAM}(D) \subseteq \mathsf{CONTR}(M)$ *sei die Menge aller Contributors des Modells* M*, die als Team gemeinsam an einem Deliverable* D *des Modells* M *arbeiten.*

Die drei oben aufgeführten Regeln R_1 bis R_3 lassen sich mit Hilfe dieser Definitionen als Prädikate formulieren. Das „zu prüfende“ Team wird als Argument übergeben und erfüllt das Prädikat, sofern es der Regel genügt:

$$R_1(\mathsf{TEAM}(D)) \equiv \tag{1}$$
$$\exists t \in \mathsf{TEAM}(D), \exists r \in R_{\mathsf{Belbin}} : (t, r) \in \mathsf{PLAYS}(M) \wedge r \neq \mathsf{implementer}$$
$$R_2(\mathsf{TEAM}(D)) \equiv \tag{2}$$
$$|\mathsf{TEAM}(D)| \leq 3 \rightarrow \nexists t \in \mathsf{TEAM}(D) : (t, \mathsf{coordinator}) \in \mathsf{PLAYS}(M)$$
$$R_3(\mathsf{TEAM}(D)) \equiv \tag{3}$$
$$|\{t \in \mathsf{TEAM}(D) : (t, \mathsf{plant}) \in \mathsf{PLAYS}(M)\}| \leq 1$$

Wir wenden die Definitionen 1 und 2 auf das Beispiel aus Abb. 1 an. Das dort gezeigte Modell eines Wunschlistenprojekts referenzieren wir dabei mit dem Bezeichner W. Es ergeben sich die folgenden Mengen:

$\mathsf{CONTR}(W) = \{\mathsf{staffA}, \mathsf{staffB}, \mathsf{staffC}\}$

$\mathsf{TEAM}(\mathsf{wishlistgui}) = \{\mathsf{staffA}, \mathsf{staffB}\}, \mathsf{TEAM}(\mathsf{wishlistadmin}) = \{\mathsf{staffB}, \mathsf{staffC}\}$

$\mathsf{PLAYS}(W) = \{(\mathsf{staffA}, \mathsf{coordinator}), (\mathsf{staffB}, \mathsf{implementer}), (\mathsf{staffC}, \mathsf{implementer})\}$

Die Erfüllung der Regelprädikate durch die beiden Teams in dieser Beispielkonstellation wird in Tab. 1 wiedergegeben. Das Team für wishlistadmin besteht in unserem Modell bisher nur aus Mitgliedern mit der Rolle implementer, was Regel R_1 nicht genügt. Hier wurde wegen des hohen Aufwands in der Praxis tatsächlich ein größeres und vielfältigeres Team gewählt. Die Auswertung von Regel R_2 zeigt, dass unser Team für die Arbeit an wishlistgui in der vorliegenden, unvollständigen Version des Modells noch zu klein ist, als dass es einen coordinator benötigen würde. In der Praxis war dieser Coordinator für die Organisation der Zusammenarbeit mit allen beteiligten Abteilungen allerdings dringend notwendig. Nach Regel R_3 sollte kein Team mehr als ein plant besitzen, so dass kein Überangebot an Ideengebung entsteht. Dies wird im Modell durch beide Teams erfüllt.

	R_1	R_2	R_3
TEAM(wishlistgui)	✓		✓
TEAM(wishlistadmin)		✓	✓

Tabelle 1: Erfüllung der Regeln R_1, R_2 und R_3 am Beispiel.

Im Umgang mit menschlichen Beziehungen sind Manager von Softwareprojekten aufgrund ihres technischen Hintergrunds oft weniger erfahren [BD94]. Durch die automatisierte Regelauswertung sollen dem Nutzer daher mögliche Fehlerquellen aufgezeigt werden. Die oben gezeigte Formalisierung der Teamregeln ist die Grundlage dieser Automatisierung. Technisch haben wir dies durch eine Portierung des OrCA-Metamodells in eine OWL-Ontologie erreicht. Mit dem etablierten Editor Protégé können OrCA-Ontologien erstellt und bearbeitet werden. Die formalisierten Regeln lassen sich in SPARQL-Querys übersetzen und gegen die Ontologien richten. Dadurch kann bei der Prüfung verschiedener Varianten und Projektkonfigurationen ein schnelles Feedback bezüglich der Teamqualität erreicht werden. Dieser Regelsatz kann projektspezifisch angepasst und erweitert werden. Für die Zukunft planen wir eine Integration dieser ontologischen Analyse mit einem EMF-basierten grafischen OrCA-Editor, der sich in der Entwicklung befindet.

4 Fazit und Ausblick

Unsere Arbeit bietet eine Grundlage für eine Goal-orientierte Dokumentation und die Diskussion von Teamstrukturen in Organisationen. Dabei soll OrCA keinesfalls etablierte Planungsmethoden oder -tools ersetzen. Vielmehr verwenden wir das Framework als

Ergänzung. Eine detaillierte Analyse der Teamstrukturen lohnt sich im industriellen Umfeld nicht für jedes Projekt. Als Daumenregel verwenden wir bei SK8DLX Services einen Mindestaufwand von vier Personenmonaten mit mindestens drei Beteiligten als Voraussetzung für eine solche Planung. Natürlich existieren in der Praxis viele organisatorische Faktoren, die eine optimale Teamaufteilung mitunter unmöglich machen. In diesem Fall lässt sich das Analyseergebnis aber mindestens verwenden, um den Fokus während der Projektdurchführung auf bestimmte Problempotentiale zu richten. Das OrCA-Framework befindet sich in einer Phase der Evaluation und ständigen Weiterentwicklung. Zu unseren nächsten Schritten gehört die Herleitung weiterer Regeln für die Teambildung. Zudem sollen auch die von Belbin postulierten *Sekundären Teamrollen* [Bel10] und funktionale Rollen eines Individuums innerhalb der Organisationshierarchie integriert werden.

Literatur

[BCK03] Len Bass, Paul Clements und Rick Kazman. *Software Architecture in Practice*. Series in Software Engineering. Addison-Wesley, Boston, 2. Auflage, 2003.

[BD94] Sheila Brady und Tom DeMarco. Management-aided software engineering. *IEEE Software*, 11(6):25–32, 1994.

[Bel10] R. Meredith Belbin. *Management Teams: Why they succeed or fail*. Routledge, 3. Auflage, 2010.

[Boe81] Barry W. Boehm. *Software Engineering Economics*. Prentice-Hall advances in computing science and technology series. Prentice-Hall, Englewood Cliffs and N.J, 1981.

[Bro87] Brooks Jr., Frederick P. No Silver Bullet Essence and Accidents of Software Engineering. *Computer*, 20(4):10–19, 1987.

[Con68] Melvin E. Conway. How Do Committees Invent? *Datamation*, 14(4):28–31, 1968.

[Con95] Larry L. Constantine. *Constantine on Peopleware*. Yourdon Press computing series. Yourdon Press, Englewood Cliffs, 1995.

[Gog93] Joseph A. Goguen. Social Issues in Requirements Engineering. In *Proceedings of IEEE International Symposium on Requirements Engineering, RE 1993, San Diego, USA, January 4-6, 1993*, RE'93, Seiten 194–195. IEEE, 1993.

[LY04] Lin Liu und Eric S. Yu. Designing Information Systems in Social Context: A Goal and Scenario Modelling Approach. *Information Systems*, 29(2):187–203, 2004.

[MS14] Johannes Meißner und Frederik Schulz. Social Team Characteristics and Architectural Decisions: a Goal-oriented Approach. In *Proceedings of the Seventh International i* Workshop*, CEUR Workshop Proceedings vol. 1157. 2014.

[SMR13] Frederik Schulz, Johannes Meißner und Wilhelm Rossak. Tracing the interdependencies between architecture and organization in goal-oriented extensible models. *In Proceedings of the Third Eastern European Regional Conference on the Engineering of Computer Based Systems*, Seiten 25–32, 2013.

[Tam89] Mahen Tampoe. Project managers do not deliver projects, teams do. *International Journal of Project Management*, 7(1):12–17, 1989.