TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Software- und Multimediatechnik, Professur für Softwaretechnologie

Evaluierung der Konsistenz zwischen Business Process Modellen und Business Role-Object Spezifikation

Lars Westermann

Bachelorarbeit

Eingereicht am: 13.10.2019

Inhaltsverzeichnis

Abstract

Gekürzte Version der Einleitung

Die heutigen Methoden zur Erstellung von Software hängen stark von geeigneten definierten Modellen ab, um die Struktur und das Verhalten der Software zu spezifizieren. Die Strukturmodelle müssen an den Verhaltensmodellen ausgerichtet sein, damit das anschließend entwickelte Softwaresystem auch die in den Vorgehensmodellen definierten Geschäftsprozesse umsetzt. Derzeit gibt es keine systematische Möglichkeit, prozessuale Geschäftsprozesse (z.B. in Form von BPMN-Prozessen) in Strukturmodellen der Software zu spezifizieren, um eine solche Konsistenz sicherzustellen. Als erster Ansatz wird dieses Problem in der Sprache der Business Role-Object Specification (BROS) gelöst, indem zeitliche Elemente in eine statische Strukturmodellspezifikation eingefügt werden. Es ist jedoch eine manuelle, komplexe und fehleranfällige Aufgabe, die Konsistenz von BROS mit einer bestimmten prozeduralen Geschäftsprozessen sicherzustellen und zu überprüfen.

In dieser Arbeit wird die Konsistenz zwischen BROS und der prozeduralen BPMN untersucht. Zu diesem Zweck werden die Modellelemente in einem BROS- und einem BPMN-Modell miteinander verglichen, um etwaige Abweichungen in Bezug auf mehrere Konsistenzkonzepte, sogenannte Konsistenzbeschränkungen, zu ermitteln. Basierend auf dieser Analyse werden dem Modellierer Warnungen gegeben, wenn Konsistenzbeschränkungen verletzt werden und wie die Probleme möglicherweise gelöst werden können. Diese Aufgabe wird automatisch über ein Tool ausgeführt werden. Die Proof-of-concept Implementierung nutzt dazu die Modelle des BROS-Editor FRaMED.io und des BPMN-Editor bpnm.io.

1 Einleitung

Größtenteils aus dem MD übernommen

1.1 Motivation

Die heutigen Methoden zur Erstellung von Software hängen stark von geeigneten definierten Modellen ab, um die Struktur und das Verhalten der Software zu spezifizieren. Einerseits werden zur Strukturdefinition häufig UML-Strukturdiagramme verwendet, wie z.B. Klassendiagramme oder Komponentendiagramme. Andererseits werden prozedurale Modelle verwendet, um das Verhalten der Software darzustellen, z.B. BPMN-Diagramme, Sequenzdiagramme oder Petrinetze. Dennoch besteht eine Lücke zwischen diesen beiden Modellperspektiven: Während die Geschäftsprozesse in prozeduralen Modellen modelliert werden, kann die eigentliche Implementierung der Software nicht ohne die Strukturmodelle erfolgen. Die Strukturmodelle müssen daher an den Verhaltensmodellen ausgerichtet sein, damit das anschließend entwickelte Softwaresystem auch die in den Vorgehensmodellen definierten Geschäftsprozesse umsetzt.

1.2 Problemdefinition

Derzeit gibt es keine systematische Möglichkeit, prozessuale Geschäftsprozesse (z.B. in Form von BPMN-Prozessen) in Strukturmodellen der Software zu spezifizieren, um eine solche Konsistenz sicherzustellen. Als erster Ansatz wird dieses Problem in der Sprache der Business Role-Object Specification (BROS) gelöst, indem zeitliche Elemente in eine statische Strukturmodellspezifikation eingefügt werden. Es ist jedoch eine manuelle, komplexe und fehleranfällige Aufgabe, die Konsistenz von BROS mit einer bestimmten prozeduralen Geschäftsprozessen sicherzustellen und zu überprüfen.

1.3 Struktur der Arbeit

In dieser Arbeit wird die Konsistenz zwischen BROS und der prozeduralen BPMN untersucht. Zu diesem Zweck werden die Modellelemente in einem BROS- und einem BPMN-Modell miteinander verglichen, um etwaige Abweichungen in Bezug auf mehrere Konsistenzkonzepte, sogenannte Konsistenzbeschränkungen, zu ermitteln. Basierend auf dieser Analyse werden dem Modellierer Warnungen gegeben, wenn Konsistenzbeschränkungen verletzt werden und wie die Probleme möglicherweise gelöst werden können. Diese Aufgabe wird automatisch über ein Tool ausgeführt werden. Die Proof-of-concept Implementierung nutzt dazu die Modelle des BROS-Editor FRaMED.io und des BPMN-Editor bpnm.io. Um den Wert und die Flexibilität für zukünftige

Entwicklungen zu erhöhen, wird besondere Aufmerksamkeit auf die Erweiterbarkeit des Tools gelegt.

2 Hintergrund

2.1 BPMN

Die Business Process Model and Notation (BPMN) ist ein Standard für die grafische Beschreibung von Geschäftsprozessen. Dabei wird das Verhalten eines Systems mit einer an Flussdiagrammen angelehnten Form beschrieben. Aufgrund der größe des BPMN Standards wird nur ein gekürztes Metamodell auf Basis von Loja et al. 2010 betrachtet.

2.2 BROS

Der neu entwickelte Ansatz der Business Role-Object Specification (BROS) kombiniert die Vorteile der Strukturmodellierung und der Verhaltensmodellierung. Als Grundlage für BROS dient die strukturbasierte Modellierungssprache Compartment Object Role Model (CROM). Diese wird mit Hilfe von Events um den Verhaltensaspekt erweitert. Mit Ausnahme von RoleConstraints wird das vollständige Metamodell aus Schön et al. o.D. betrachtet.

3 Verwandte Arbeiten

Der Bereich der Konsistenzprüfung zwischen Strukturbasierten und Verhaltensbasierten Modellen ist seit Jahren ein gut erforschter Bereich. Ins besondere gibt es ein großes Spektrum an Methoden zum Vergleichen von verschiedenen UML Diagrammen.

3.1 Klassifikationsschema

Die bestehenden Methoden für UML Diagramme wurden von Usman et al. 2008 und Lucas et al. 2009 zusammengestellt. Beide Arbeiten nutzen ein ähnliches Schema:

- **Nature:** Es beschreibt den Fokus der vergleichbaren Modelle. Mögliche Werte sind strukturbasiert, verhaltensbasiert und beides.
- **Diagrams:** Es beschreibt welche konkreten Modelle von der Methode unterstützt werden. Die Arbeiten von Usman et al. 2008 und Lucas et al. 2009 beziehen sich dabei ausschließlich auf UML Diagramme.
- **Consistency Type:** Es beschreibt welche Arten der Konsistenz von der Methode überprüft werden. Dabei wird hauptsächlich unterschieden zwischen *Inter-Modell (vertikale) Konsistenz* (Konsistenz bei verschiedenen Abstraktionsstufen und gleichem Modelltyp), *Intra-Modell (horizontale) Konsistenz* (Konsistenz bei gleicher Abstraktionsstufe und verschienden Modelltypen)und *Evolutionskonsistenz* (Konsistenz der eines Modelles über verschiedende Entwicklungsstufen). Zusätzlich spezifiziert Usman et al. 2008 noch die *semantische-* und die *syntaktische Konsistenz*. Diese Beziehen sich auf die Konsistenz eines Modelles zu seinem Metamodell. Dies wird für die nachfolgende Arbeit als Vorraussetzung angesehen und nicht näher betrachtet.
- **Intermediate Representation:** Es beschreibt ob die Methode eine temporäre Zwischendarstellung benötigt oder nicht.
- **Consitency Strategy:** Es beschreibt die benutzte Validierungsstrategie. Es werden drei verschiedene Strategien genannt und zwar *Analysis* (Auf einem Algorithmus basierend), *Monitoring* (Auf einem Regelsatz basierend) und *Construction* (Auf der Generierung des zu vergleichenden Modelles basierend).
- **Automatable:** Es beschreibt ob die Methode manuell oder automatisiert von einem Programm durchgeführt werden kann. Mögliche Werte sind gut (H), mittel (M) und schlecht (L).
- **Extensibility:** Es beschreibt wie gut die Methode um weitere Konsistenzregeln erweiterbar ist. Mögliche Werte sind gut (H), mittel (M) und schlecht (L).

3.2 Aktuelle Verfahren zur Konsistenzprüfung

Transformation zu *CSP/Object-Z*

- Rasch und Wehrheim (Usman et al. 2008)
- Kim and Carrington (Usman et al. 2008)

Transformation zu *Pertinetze*

- Shinkawa (Usman et al. 2008)
- Liu et al. (Usman et al. 2008)
- Bernardi et al. (Usman et al. 2008)

Anwendung von *Description Logic*

- Satoh et al. (Usman et al. 2008)
- Mens et al. (Usman et al. 2008)
- Simmonds et al. 2004

Transformation in gemeinsames Modell

- Egyed 2001 (ViewIntegra)

Nutzung von UML (zB. *OCL*)

- Briant et al. (Usman et al. 2008)
- Eqyed (Usman et al. 2008)

3.3 Vergleich der bestehenden Verfahren

Vergleich der oben genannten Methoden.

4 Konsistenz zwischen BPMN und BROS

4.1 Konsistenzproblem

Viele der bereits existierenden Methoden lassen sich anpassen und können mit andren Modellen genutzt werden. Anstelle eines UML Klassendiagramms kann ein BROS Modell auf der strukturbasierten Seite und ein BPMN Modell anstelle eines UML-Sequenzdiagramms verwendet werden. Allerdings ist bei dem Vergleich von BPMN und BROS zu beachten das Inkonsistenzen keine strikten Fehler, sondern nur Warnungen an den Modellierer sind. Das liegt an der Möglichkeit ein BROS Modell beliebig anzureichern und das Events eine Abstraktion eines beliebigen Prozesses sein können.

4.2 Konsistenzregeln

Erläuterung der implementierten Regeln anhand von Minimalbeispielen.

4.3 Referenzarchitektur

Im Gegensatz zu anderen Arbeiten wurde zur Überprüfung dieser Regeln kein formales Verfahren auf Basis von zB. *Description Logic* oder *Petrinetzen* genutzt. Dies hat den Vorteil das die Regeln direkt auf den Modellen ausgeführt werden können und nicht erst eine Zwischendarstellung gebaut werden muss. Das hier genutzte Verfahren arbeitet in zwei Stufen auf den Modellen die als *Layered Graph* dargestellt werden. Im ersten Schritt wird ein Matching von Modellelementen aufgebaut. Dies wird iterativ, in Form eines Fixpunkt-Algorithmus, durchgeführt um kaskadierendes Matching zu erlauben. Im zweiten Schritt werden anhand des Matching die Regeln ausgeführt.

5 Implementierung der automatischen Konsistenzprüfung

Um die Umsetzbarkeit und die Funkionalität dieses Ansatzes zu zeigen wurde eine Referenzimplementierung angefertigt.

5.1 Implementierung der Referenzarchitektur

Eine Anforderung war eine mögliche Integration in den bestehenden BROS-Editor FRaMED.io. Durch die Verwendung von Kotlin konnte das Parsing der BROS-JSON Datei direkt übernommen werden. Das Dateiformat von bpmn.io basiert auf XMI und kann nativ von JS geparsed werden.

5.2 Matching der Modelelemente

Um die Verifizierung der Modelle zu ermöglichen muss zunächst ein dazugehöriges Matching aufgebaut werden. Dazu werden Elemente mit Kompatiblen Typen anhand ihres Namens zugeordnet. Zusätzlich kann, aufgrund des Fixpunkt-Algorithmus, ein bestehendes Matching auf andere Elemente übertragen werden. Sollte dies nicht ausreichen kann der Modellierer explizit ein Predefined-Matching einfügen was ein Element Matching erzwingt oder verbietet.

Erklärung einiger Matching-Regeln und ihrer Syntax

5.3 Implementierung der Konsistenzregeln

Dank des bestehenden Matchings lassen sich die genannten Konsistenzregeln einfach implementieren.

Erklärung einiger Verifikationsregeln und ihrer Syntax

5.4 Benutzerinterface

Da das entwickelte Tool eine Webanwendung ist kann es in allen moderneren Browsern benutzt werden die JS aktiviert haben. Das Tool hat ein zweistufiges Interface. Als erster Schritt müssen die Quelldateien der zu überprüfenden Modelle geladen werden. Dazu können die Dateien einfach per Drag'n'Drop in das Tool geladen werden. Das Tool erkennt den Inhalt unabhängig des Namens und läd die entsprechende Datei. Alternativ kann eine manuelle Dateiauswahl genutzt werden oder der Inhalt der Datei in das Textfeld kopiert werden. Sobald jeweils ein valides BPMN- und BROS-Modell geladen wurden, wird die Konsistenzprüfung automatisch gestartet.

Die Ergebnisse der Konsistenzprüfung werden unterhalb der Eingabemaske angezeigt. Zunächst werden statistische Informationen zu den geladen Modellen, des Matchings und der Validierung angezeigt. Unterhalb dieser Statistiken befindet sich die Liste der Validierungsergebnisse. Mit hilfe der Tableiste kann das BPMN- oder BROS-Matching oder auch das geladene Predefined-Matching anzeigen. Das Predefined-Matching kann per Klick auf die Element-IDs direkt bearbeitet werden.

6 Fallstudie

6.1 Anwendung am Beispiel einer Pizzabestellung

Nachdem die verschiedenen Konsistenzregeln aufgestellt und implementiert wurden soll nun die Benutzbarkeit und die Nützlichkeit des Tools gezeigt werden. Dafür wird der Vorgang einer Pizzabestellung betrachtet und mit Hilfe des Tools evaluiert. Die benutzten BPMN- und BROS-Modelle basieren auf den Modellen von Schön et al. o.D.

6.2 Erweiterbarkeit des Ansatzes

Bisher wurden nur Regeln vorgestellt, die die Konsistenz von BPMN-Modellen zu BROS-Modellen prüfen. Da das BROS-Modell gegenüber dem BPMN-Modell angereichert werden kann ist dies auch die häufigste Anwendung. Allerdings können auch einige Regeln in die andere Richtung überprüft werden. Um gleichzeitig die Erweiterbarkeit des Ansatzes und der Implementierung zu zeigen wird eine Regel hinzugefügt, die die BROS-Events und die Existenz der dazugehörigen BPMN-Elemente verifiziert.

7 Schluss

7.1 Zusammenfassung

Ziel dieser Arbeit war die automatisierte Konsistenzprüfung von BPMN- und BROS-Modellen. Dazu wurden anhand bestehender Arbeiten ein neuer Ansatz entwickelt. Die Funktionalität dieses Ansatzes wurde anhand einer Referenzimplementierung und einer Fallstudie gezeigt.

7.2 Wissenschaftlicher Beitrag

- **Nature:** beides
- **Diagrams:** BPMN-BROS
- **Consistency Type:** *Intra-Modell (horizontale) Konsistenz*
- **Intermediate Representation:** Nein
- **Consitency Strategy:** *Monitoring* (Auf einem Regelsatz basierend)
- **Automatable:** gut (H)
- **Extensibility:** gut (H)

Der Ansatz und das entwickelte Tool geben dem Modellierer Warnungen und Hinweise die auf mögliche Inkonsistenzen deuten. Dabei werden explizit keine Handlungsempfehlungen oder Lösungsmöglichkeiten gegeben. Dies liegt in der alleinigen Verantwortung des Modellierers. Des Weiteren wird keine syntaktische und semantische Konsistenzprüfung der Einzelmodelle durchgeführt. Die zu überprüfenden Modelle müssen alleinstehend Konsistent sein. Nur die Korrektheit des Dateiformats wird indirekt überprüft, da es eine Vorraussetzung zum Parsen des Modelles ist.

7.3 Zukünftige Arbeiten

Eine Abgrenzung dieser Arbeit war, das keine Handlungsempfehlungen oder Lösungsmöglichkeiten bei gefundenen Inkonsistenzen gegeben werden. In diesem Feld sollte evaluiert werden welche Inkonsistenzmuster häufig auftreten und ob diese eine Vorhersehbare Lösungsmöglichkeit aufweisen. Auch könnte die Fehlertoleranz des Tools erhöht werden, indem man eine syntaktische und semantische Konsistenzprüfung der Einzelmodelle integriert. Diese ist momentan noch von externen Programmen abhängig oder aufgabe des Modellierers. Da externe Programme meist

auf diese Aufgabe spezialisiert sind müsste zunächst evaluiert werden ob eine integrierte Lösung auch eine gleichbleibende Erkennungsrate besitzt. Um die Benutzbarkeit des Tools weiter zu verbessern kann eine Integration in FRaMED.io erfolgen. Die technische Grundlage ist mit der Referenzimplementierung bereits gegeben. Hierfür müsste noch evaluiert werden wie das Ergebnis der Analyse in die graphische Darstellung integriert werden kann.

Literatur

Egyed, Alexander (2001). „Scalable consistency checking between diagrams-The VIEWINTEGRA approach“. In: *Proceedings 16th Annual International Conference on Automated Software Engineering (ASE 2001)*. IEEE, S. 387–390.

Loja, Luiz Fernando Batista et al. (2010). „A business process metamodel for enterprise information systems automatic generation“. In: *Anais do I Congresso Brasileiro de Software: Teoria e Prática-I Workshop Brasileiro de Desenvolvimento de Software Dirigido por Modelos*. Bd. 8, S. 37–44.

Lucas, Francisco J, Fernando Molina und Ambrosio Toval (2009). „A systematic review of UML model consistency management“. In: *Information and Software Technology* 51.12, S. 1631–1645. ISSN: 0950-5849. DOI: 10.1016/j.infsof.2009.04.009.

Schön, Hendrik et al. (o.D.). „Business Role-Object Specification: A Language for Behavior-aware Structural Modeling of Business Objects“. In: ().

Simmonds, Jocelyn et al. (2004). „Maintaining Consistency between UML Models Using Description Logic“. In: *L'OBJET* 10.2-3, S. 231–244. ISSN: 1262-1137. DOI: 10.3166/objet.10.2-3.231-244.

Usman, Muhammad et al. (2008). „A Survey of Consistency Checking Techniques for UML Models“. In: *2008 Advanced Software Engineering and Its Applications*. IEEE, S. 57–62. DOI: 10.1109/asea.2008.40.