Personalisierte Recommender Systems für Software-Produktlinienkonfigurationen[1]

Juliana Arriel[2]

Abstract: *Software-Produktlinien* (SPLs) werden in der Industrie zur Massenproduktion individualisierter Produkte eingesetzt, um Produktionskosten und Time-to-Market zu reduzieren. Die inhärente Komplexität und Variabilität von SPLs führt jedoch zu einer exponentiell anwachsenden Menge an möglichen Produkten. Gerade bei großen SPLs sind daher Skalierbarkeit und Performanz ein Herausforderung und macht spezialisierte Tool-Unterstützung entscheidend, um Entscheidungsträger bei der Produktkonfiguration zu unterstützen. In diesem Zusammenhang war die Konfiguration von SPLs in den letzten Jahren ein wichtiges Forschungsthema. In dieser Arbeit geben wir einen Überblick über die SPL-Konfigurationstechniken und schlagen einen effizienten, durch Recommender-Systeme unterstützten SPL-Konfigurationsprozess vor, um Entscheidungsträgern genaue und skalierbare Lösungen anzubieten. Dazu passen wir moderne, kollaborative Recommender-Algorithmen an den SPL-Konfigurationskontext an. Zusätzlich führen wir visuelle Unterstützung ein, um Entscheidungsträger durch den Konfigurationsprozess zu führen, indem wir ihnen ermöglichen, sich auf eine begrenzte Anzahl von validen und relevanten Teilen des Konfigurationsraums zu konzentrieren. Wir demonstrieren empirisch die Anwendbarkeit der implementierten Algorithmen und Werkzeuge in verschiedenen realen Szenarien.

1 Einführung

Im heutigen kompetitiven Softwaremarkt ist es von größter Wichtigkeit auf die Bedürfnisse einzelnen Kunden einzugehen. Dabei reichen diese Bedürfnisse von funktionalen bis zu nicht-funktionalen Eigenschaften (z.B. Performanz und Kosten) eines Produktes. *Software-Produktlinien* (SPL) haben sich durch ihre systematische Wiederverwendung als effizientes und effektives Mittel für kundenindividuelle Massenproduktion durchgesetzt. SPL beschreiben eine Familie von Softwareprodukten mit einer gemeinsamen Menge von Features. Durch die Kombinationen verschiedener Features wird die Individualisierung von Softwareprodukten für jeden Kunden ermöglicht. Trotz ausführlicher Forschung in den letzten Jahrzehnten bleibt die effiziente Individualisierung von Software weiterhin ein wichtiges Forschungsthema. So müssen zum Beispiel wegen der hohen Variabilität vieler SPL Nutzern durch einfacher und verständlicher Konfigurationsprozess unterstützt werden. Zwar wurden in der Vergangenheit bereits einige *interaktive*, *teil-automatisierte* und auch *automatisierte* Ansätze für den Konfigurationsprozess vorgestellt, allerdings erreichen diese immer noch nicht das volle Potential bei der Unterstützung der Nutzer.

[1] Englischer Titel: "Personalized Recommender Systems for Software Product Line Configurations"

[2] Otto-von-Guericke-Universität Magdeburg, juliana.alves-pereira@ovgu.de

cc by sa

Verstärkt wird dies noch, sobald die Komplexität des Konfigurationsprozess weiter erhöht wird, wie zum Beispiel durch die Einführung von Abhängigkeiten zwischen Features und nicht-funktionalen Eigenschaften.

Typischerweise konfigurieren Nutzer ihr individuellen Produkte über ein Konfigurationswerkzeug, in dem sie nacheinander die gewünschten Features auswählen. In diesem Kontext gibt es viele Ansätze, die Nutzer zu einer validen Konfiguration führen und sicherstellen, das keine Abhängigkeiten der SPL verletzt werden [PCF15]. Allerdings müssen Nutzer bei diesen Ansätzen auch immer wieder Features beachten, die für sie nicht relevant sind [Pe16a]. Dies kann dazu führen, dass die Effizienz des Konfigurationsprozess sinkt, da Nutzer mehr Entscheidungen treffen müssen. Gerade bei hoch konfigurierbaren SPL mit komplexen Abhängigkeiten wird der Konfigurationsprozess langwierig und fehleranfällig. Somit wird zusätzlicher Support notwendig, um Nutzer durch den Konfigurationsprozess zu führen und ihren Fokus auch die für sie relevanten Features zu lenken.

Für einen *teil-automatisierten* Konfigurationsprozess haben Galindo et al. [Ga15] ein dynamisches Entscheidungsmodell mit einer Menge von Fragen und möglichen Antworten vorgeschlagen. Andere Autoren [As14, BE14, TLL14] haben einen Ansatz mit paarweisen Entscheidungen vorgeschlagen, in dem Nutzer jeweils zwei Feature miteinander im Bezug auf ihrer Relevanz für die Erfüllung von nicht-funktionalen Eigenschaften vergleichen. Allerdings können Fragen-basierte Entscheidungsysteme zu vagen und irreführenden Beschreibungen in Fragebögen führen. Hingegen können paarweise Entscheidungen Inkonsistenzen in der Rangfolge der Features verursachen. Wenn Features oder Antworten in den Fragebögen für den Benutzer gleichwertig oder von geringer Relevanz sind, kann der Benutzer bei der Konfiguration nicht weiter unterstützt werden. Da zudem ein einzelnes Feature viele nicht-funktionalen Eigenschaften beeinflussen kann, ist es möglich, dass die Menge und Komplexität an Informationen Benutzer überwältigt und die Konfiguration erschwert.

Für einen *automatisierten* SPL-Konfigurationsprozess existieren Ansätze [Oc17, Oc18], die Constraint Programming und evolutionäre Algorithmen verwenden, um eine Konfiguration, die den Produktanforderungen eines Benutzers entspricht, in einem einzigen Schritt zu generieren. Zwar garantieren exakte Ansätze, wie Constraint Programming eine optimale Konfiguration, aufgrund der rechnerischen Komplexität des Problems haben diese jedoch eine ineffiziente Laufzeit für eine hohen Zahl von Konfigurationsoptionen. Daher wurden heuristische Verfahren, wie unter anderem evolutionäre Algorithmen, eingehender untersucht, um auch für große Konfigurationsräume nahezu optimale Konfigurationen effizient generieren zu können. Ein Nachteil bei der Verwendung von automatisierten Ansätzen ist, dass die Spezifikation mehrerer Anforderungen zu sehr unterschiedlichen Konfigurationen führen kann. So lassen sich beispielsweise bei einem Mobiltelefon nur schwer die beiden Eigenschaften Sicherheit und Kosten gleichzeitig in der selben Konfiguration optimieren. Oft erzeugen automatisierte Ansätze in solchen Situationen mehrere Konfigurationen und überlassen den Benutzern die endgültige Entscheidung. Dabei leiten aktuellen Ansätze

die Benutzer weder bei der Auswahl einer geeigneten Konfiguration, noch bieten sie Unterstützung bei der Festlegung der Benutzerpräferenzen.

Ausgehend von den identifizierten Problemen bisheriger Ansätze besteht der Beitrag dieser Doktorarbeit darin, einen effizienterer Konfigurationsprozess für hoch konfigurierbare SPL vorzuschlagen. Um dieses Ziel zu erreichen, schlagen wir zum ersten Mal ein Verfahren vor, das ein kollaboratives Recommender-System verwendet, das auf Konfigurationen vorheriger Benutzer basiert, um personalisierte Empfehlungen für einen aktuellen Benutzer zu generieren. Darüber hinaus bietet unser System visuelle Unterstützungen, die es Benutzern ermöglicht, sich auf wenige, relevante Informationen aus Teilen des Konfigurationsraums zu konzentrieren. Insgesamt beantworten wir die folgenden Forschungsfragen:

- *RQ1.* Welche Konzepte gibt es zur Unterstützung des SPL-Konfigurationsprozess in der Literatur?
- *RQ2.* Wie lassen sich kollaborative Recommender-Systeme nutzen, um relevante Features anhand von expliziten Nutzerinformationen zu bestimmen?
- *RQ3.* Wie lässt sich der Nutzerkontext in eine personalisiertes kollaborative Recommender-System integrieren?
- *RQ4.* Wie können die Selbstkonfiguration dynamischer SPL durch kollaborative Recommender-Systeme unterstützen?
- *RQ5.* Wie lassen sich die vorgeschlagenen Techniken in ein state-of-the-art Konfigurator integrieren?

Um *RQ1* zu beantworten, führen wir eine *Systematischen Literaturrecherche* (SLR) zum SPL-Konfigurationsprozess durch und klassifizieren einen Korpus von 157 Veröffentlichungen. Basierend auf unseren Erkenntnissen definieren wir eine Reihe offener Probleme in diesem Bereich. Darauf aufbauend untersuchen wir für *RQ2*, wie sich sechs verschiedene, aktuelle kollaborativen Recommender-Algorithmen in den SPL-Konfigurationsprozess integrieren lassen. Zur Untersuchung von *RQ3* erweitern wir diese Algorithmen, so dass sie nicht-funktionale Anforderungen berücksichtigen können. Um *RQ4* zu beantworten schlagen wir einen Tensor-basierten Recommender-Algorithmus vor, der die dynamische Konfiguration von SPL zur Laufzeit durch Modellierung eines N-dimensionalen Tensors *User-Feature-Context* ermöglicht. Bei dem vorgeschlagenen Ansatz werden verschiedene Arten von Kontextdaten als zusätzliche Dimensionen betrachtet. Um die Genauigkeit der vorgeschlagenen Algorithmen abzuschätzen, verwenden wir in unserer empirischen Evaluation mehrere große, reale Konfigurationsdatensätze aus mittleren und großen Produktlinien unterschiedlicher Domänen. Zur Beantwortung von *RQ5* entwickeln wir schließlich einen Open-Source-Konfigurator namens PROFilE. PROFilE ist ein Eclipse-Plug-In, in welchem alle vorgestellten Konzepte und Visualisierungsmechanismen zur Vereinfachung des Konfigurationsprozesses implementiert sind.

2 Aktuelle Forschung zur Konfiguration von Software-Produktlinien

Laut Benavides et al. [Be13a] ist der SPL-Konfigurationsprozess ein aktives Forschungsfeld und wurde in den letzten Jahren sowohl von Praktikern als auch Forschern aufgegriffen. Seit der Einführung von Feature-Modellen in den 90er Jahren durch Kang et al. [Ka90] wurden viele neue Techniken, Werkzeuge und Algorithmen bereitgestellt, um Entscheidungsträger im SPL-Konfigurationsprozess zu unterstützen. Diese Ansätze konzentrieren sich jedoch auf sehr spezifische Bereiche, sodass in der Literatur immer noch keine zusammenhängende Übersicht aller Mechanismen des Konfigurationsprozesses und deren Tool-Unterstützung existieren. Daher führen wir eine systematische Literaturrecherche nach den Richtlinien von Kitchenham und Charters [KC07] durch. Wir identifizieren, klassifizieren und bewerten vorhandene Veröffentlichungen zum SPL-Konfigurationsprozess und geben einen vollständigen Überblick über die in diesem Bereich erzielten Fortschritte. Unsere Literaturübersicht stellt dabei eine Ergänzung zu bereits durchgeführten Studien [BSRC10, Be13a] dar.

Insgesamt vergleichen und klassifizieren wir 157 Primärstudien. Wir betrachten verschiedene Details zum SPL-Konfigurationsprozess, wie z.B. die Beschreibung von Konfigurationsbeschränkungen und die Effektivität und Effizienz vorhandene Konfigurationsansätze. Basierend auf dieser Klassifizierung definieren wir einen Satz von 5 Konfigurationsaktivitäten und 17 Konfigurationsmechanismen [Pe17]. Wir geben einen detaillierten Einblick in die Mechanismen, die in jeder Arbeit verwendet werden, wie diese Mechanismen empirisch bewertet werden, sowie ihre Hauptnachteile. Die Ergebnisse unserer Recherche bestätigen, dass die Forschung in der SPL-Konfiguration immer noch fragmentiert und facettenreich ist. So mussten wir feststellen, dass viele Techniken unabhängig voneinander entwickelt wurden und die gleichen Mechanismen adressieren. Unser Übersicht zeigt, dass *(i)* die Qualität der empirischen Bewertung vorhandener Ansätze verbessert werden muss; *(ii)* ganzheitliche Lösungen zur Unterstützung mehrerer Konfigurationsbeschränkungen fehlen; und *(iii)* die Skalierbarkeit und Effektivität existierender Ansätze verbessert werden muss. Angesichts dieser Erkenntnisse und der zunehmenden Bedeutung dieses Forschungsfeldes ergeben sich viele interessante Forschungsmöglichkeiten. Im folgenden Abschnitt greifen wir diese Erkenntnisse auf, indem wir Recommender-Algorithmen verwenden, um Entscheidungsträger bei der Konfigurationsaufgabe einfach und präzise zu unterstützen.

3 Personalisierte Konfiguration von Software-Produktlinien

Zur Umsetzung eines personalisierten Konfigurationsprozesses schlagen wir die Verwendung von Recommender-Algorithmen vor. Recommender-Algorithmen können Vorschläge enthalten, die einen großen Konfigurationsraum effektiv beschneiden, sodass Benutzer auf Features hingewiesen werden, die ihren Bedürfnissen und Präferenzen am besten entsprechen. Unser Ansatz zielt auf die folgenden Herausforderungen, abgeleitet aus der bestehenden Literatur und aktuellen Industriebedürfnissen:

- Die Zahl der Features und deren Abhängigkeiten beeinflusst sowohl die Erstellungszeit als auch die Qualität einer Konfiguration. Variabilitätsmodelle reale SPL haben oft

viele und komplexe Abhängigkeiten und Beschränkungen von nicht-funktionalen Eigenschaften (z.B. der Linux-Kernel [Si10]). Obwohl einige Konfiguratoren die Einhaltungen von Abhängigkeiten garantieren können, indem sie implizierte Entscheidungen automatisch treffen, kann dieses Verhalten für Benutzer zu unachvollziehbaren Ergebnissen führen. Folglich kann es für einen Benutzer schwierig sein, eine gültige Konfiguration zu erstellen, die alle ihre (nicht-funktionalen) Anforderungen erfüllt.

- Mithilfe der Auswahl von funktionalen Features lässt sich einfach eine gültige Konfiguration bestimmen, die auf die funktionalen Anforderungen eines Produkts abgestimmt ist. Dabei werden jedoch keine nicht-funktionelle Anforderungen berücksichtigt (z.B. langsame Leistung oder zu hohe Kosten). Obwohl es automatische Ansätze für das Generieren von Konfigurationen gibt, die nicht-funktionale Anforderungen optimieren [Oc17, Oc18], geben diese potentiell eine große Anzahl gültiger Konfigurationen zurück. Eine Auswahl zwischen diesen zu treffen, ist für Entscheidungsträger oft schwierig, da sie viele detaillierte technische Informationen über die Features und deren Kontext berücksichtigen müssen.
- Psychologische Studien haben gezeigt, dass Entscheidungsträger bei einer großen Auswahl an Entscheidungen in Bezug auf die Bedürfnisse der Nutzer in der Regel unsicher sind. Laut einer Industrieumfrage [Be13b], geben etwa 60% der Teilnehmer das Verstehen und die Visualisierung von Variabilitätsmodellen als ein Problem an. Eine weitere Umfrage [Ba17] zeigt, dass 17% der Konfiguratoren explizit angeben, dass sie Einschränkungen bei ihrer Visualisierung haben.

Um diese Herausforderungen zu bewältigen, haben wir einen personalisierten Recommender-Algorithmus eingeführt, der relevante Features aus früheren Konfigurationen bestimmt. Der Hauptbeitrag unserer Arbeit besteht darin, den Konfigurationsprozess zu vereinfachen, indem Entscheidungsträger effizient durch den Konfigurationsprozess geleitet werden. Der in Abb. 1 dargestellte Workflow bietet einen Überblick über den vorgeschlagenen Konfigurationsprozess. Der Konfigurationsprozess wird in fünf Hauptaktivitäten untergliedert: *Configure*, *Propagate Decisions*, *Check Validity*, *Claculate Recommendations* und *Visualize*. Als Eingabe wird das Variabilitätsmodell und die (nicht-funktionalen) Produktanforderungen von Benutzern, Kunden und anderen Interessengruppen verwendet. Dasselbe Variabilitätsmodell wird verwendet, um andere Produkte an die Anwendungsszenarien neuer Kunden anzupassen.

Der Konfigurationsprozess beginnt, indem ein Benutzer einzelne Features aus dem Variabilitätsmodell auswählt. Dabei beschränkt eine fokussierte Ansicht die direkte Auswahlmöglichkeit des Benutzers, da diese zunächst nur abstraktere Features anzeigt. Sobald der Benutzer ein Feature aus- oder abwählt, wird *Decision Propagation* angewendet, sodass alle, durch diese Entscheidung und die Abhängigkeiten im Variabilitätsmodell implizierten, Features automatisch aus- oder abgewählt werden [Ja08]. Damit wird verhindert, dass ein Benutzer zwei widersprüchliche Features auswählen kann [Pe16a]. Als nächstes wird geprüft, ob die gegebene Teilkonfiguration alle Abhängigkeiten des Variabilitätsmodells erfüllt sind oder ob noch weitere Features gewählt werden müssen. Ist die Teilkonfiguration bereits

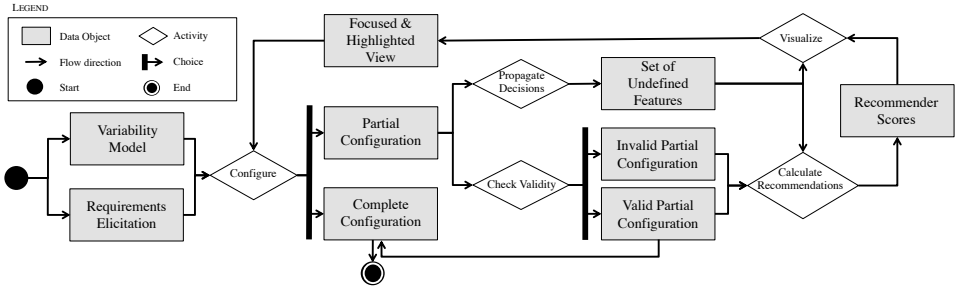

Abb. 1: Graphische Darstellung des Ablaufs des Konfigurationsprozesses.

gültig, so kann der Benutzer den Konfigurationsvorgang abschließen, indem alle bisher unkonfigurierten Features automatisch abwählt werden. Andernfalls muss der Benutzer weitere Features wählen. Dabei zeigt eine hervorgehobene Ansicht dem Benutzer, welche Entscheidungen zu einer gültige Konfiguration führen. Der Prozess endet, sobald eine vollständige Konfiguration erstellt ist. Durch die Anwendung von *Decision Propagation* ist diese immer gültig.

Vor jeder Auswahl des Benutzers bewertet das Recommender-System die Relevanz jedes unkonfigurierten Features (siehe Abschnitt 3.1). Im interaktiven Konfigurationsprozess wird dies als 5-Sterne-Klassifizierung im Konfigurator angezeigt. Im automatischen Konfigurationsprozess wird jeweils das Feature mit der höchsten Relevanz (unter Berücksichtigung der Abhängigkeiten) automatisch ausgewählt. Die Automatisierung unseres Ansatzes erfordert als Eingabe eine Teilkonfiguration mit einem oder mehr ausgewählten Features und mindestens einer vorherigen Konfiguration als Datenbasis. Allerdings steigt die Effektivität unseres Ansatz mit einer umfangreicheren Datenbasis deutlich.

3.1 Vorgeschlagene Recommender-Ansätze

Um Empfehlungen zu berechnen, schlagen wir drei Ansätze vor: *Binary-Based Recommender*, *Context-Based Recommender* und *Runtime-Based Recommender*.

Binary-Based Recommender. Dieser Ansatz beruht auf der einmaligen Verwendung von Daten aus vorherigen Konfigurationen, um personalisierte Empfehlungen für einen aktuellen Benutzer zu generieren. Wir haben sechs *Collaborative Filtering* (CF) Algorithmen für dieses SPL-Konfigurationsszenario angepasst: *Benutzerbasierte CF*, *CF mit Signifikanzgewichtung*, *CF mit Shrinkage*, *CF with Hoeffding Bound*, *Average Similarity (AS)* und *Matrix Factorization (MF)* [Pe16b, Pe18b].

Context-Based Recommender. Unser bisheriger Ansatz weist eine Einschränkung auf: *Die Ergebnisse sind stark abhängig von der Anzahl der bisher konfigurierten Features*. Folglich wird die Empfehlung neuer nützlicher Informationen für Benutzer erschwert. Um diese Einschränkung zu überwinden, kombinieren wir vier Empfehlungstechniken: *context-aware*, *knowledge-based*, *CF-based* und *rule-based*. Ein *context-aware Recommender*

versucht Empfehlungen basierend auf Schlussfolgerungen zu Präferenzen des Benutzers vorzuschlagen. Er verfügt über Kontextwissen über Produktanforderungen, z.B. über den finanziellen Kontext eines Benutzers. Der *knowledge-based Recommender* erstellt eine vollständige Nutzenfunktionen aus vorherigen Konfigurationen. Die Nutzenfunktionen berücksichtigt verschiedene Faktoren, die zur Bewertung einer Konfiguration beitragen, indem die Bedeutung jedes Features für jeden Benutzer gewichtet wird. Der *CF-based recommender* erfasst zusätzlich Ähnlichkeiten zwischen Benutzern auf Grundlage der vorherigen Konfigurationen. Insgesamt haben wir fünf CF-Algorithmen angepasst: *User-Based CF*, *Feature-Based CF*, *User-Based AS*, *Feature-Based AS* und *MF* [Pe18d].

Runtime-Based Recommender. Der bisherige Ansatz bietet keine einfache Möglichkeit, Kontextdaten (d.h. nicht-funktionale Eigenschaften [BSRC10]) in das Empfehlungsmodell zu integrieren [KBV09]. Es wird ein reduktionsbasierter Ansatz verwendet, um das Problem der N-dimensionalen ($User \times Feature \times Contexts$)-Empfehlung auf zweidimensionale ($User \times Feature$)-Empfehlung zu reduzieren. Dies führt zwar zu relevanteren Daten für die Berechnung von Empfehlungen, führt jedoch auch dazu, dass Daten verwendet werden, die nur auf Konfigurationen mit dem gleichen oder einem ähnlichen Kontext basieren. Darüber hinaus sind Reduktionsansätze rechenintensiv, da für jede Kombination von Kontextanforderungen ein Empfehlungsmodell trainiert und getestet werden muss. Bei dynamischen SPL, in dem ein System zur Laufzeit neu konfiguriert wird, empfiehlt sich daher die Verwendung eines mehrdimensionalen Ansatzes. Daher verwenden wir statt einer zweidimensionale Matrix einen mehrdimensionalen Tensor und treffen Featureempfehlungen mithilfe von *Tensor Factorization* (TF), einer N-dimensionalen Erweiterung von MF [Pe18c]. TF basiert auf einem einzelnen wenig rechenintensivem Modell und skaliert auf eine beliebige Menge von Kontextinformationen.

3.2 Analyse der Ergebnisqualität

Zur Bewertung der Qualität der Empfehlung unseres Recommender-Systems, führen wir eine empirische Evaluation durch. Wir vergleichen in unsere Experimenten zehn Recommender-Algorithmen anhand mehrerer realer Konfigurationsdatensätze. Wir präsentieren im Folgenden nur die wichtigsten Ergebnisse unserer Untersuchungen. Alle Details zum Experimentdesign, die empirischen Bewertung der Algorithmen und zusätzliches Material (Variabilitätsmodell, Konfigurationsdatensätze, Ergebnisse) stellen wir auf unserer Webseite[5] zur Verfügung.

1. Können Recommender-Algorithmen den SPL-Konfigurationsprozess in realistischen Konfigurationsszenarien unterstützen? Ja, wir haben gezeigt, dass die implementierten Recommender-Algorithmen relevante Features effektiv identifizieren. Sie liefern sogar bessere Ergebnisse als von Domänexperten in einem interaktiven Konfigurationsprozess. Zudem reduzieren sie den Zeitaufwand und die Fehleranfälligkeit des Prozesses.

[5] http://wwwiti.cs.uni-magdeburg.de/~jualves/PROFilE/

2. Ab welcher Phase der Produktkonfiguration kann ein Recommender-Algorithmus gute Empfehlungen geben? Die von uns untersuchten Algorithmen liefern bereits in der Anfangsphase (d.h. mit 10 % der ausgewählten Features) bessere Empfehlungen als unsere Baseline.

3. Welchen Einfluss haben die implementierten Algorithmen jeweils auf die Qualität der Empfehlungen? Die Wahl des Algorithmus hat einen großen Einfluss auf die Qualität der Empfehlungen. Für Szenarien, in denen lediglich bisherige Konfigurationen als Datenbasis verfügbar sind, empfehlen wir die Verwendung eines der drei Algorithmen: CF-shrinkage, CF-significance weighting oder MF. Sind zusätzlich Kontextinformationen verfügbar, empfehlen wir die Verwendung von kontextbasierten MF und TF. Unsere Daten zeigen, dass der TF-Algorithmus bei kleinen Datensätzen besser funktioniert, da der kontextbasierte MF-Algorithmus die Datenbasis für seine lokale Vorhersagemodelle noch weiter aufteilt. Folglich basieren bei MF diese Empfehlungen auf einer kleinen Anzahl von Konfigurationen, die auf denselben oder einen ähnlichen Kontext beschränkt sind. Bei dynamischen SPL, in dem sich ein System ständig neu konfiguriert, erscheint ebenfalls die Verwendung eines TF-Algorithmus sinnvoll (siehe Abschnitt 3.1).

4. Was sind die Vorteile eines kontextabhängigen gegenüber einem kontextunabhängigen Recommender-Algorithmus bezogen auf die Ergebnisqualität? In unseren Experimenten übertraf die Leistung eines kontextabhängigen Recommender-Algorithmus die eines kontextunabhängigen Algorithmus in allen Phasen des Konfigurationsprozesses deutlich, wobei die Vollständigkeit der Konfiguration außer Acht gelassen wurde. Daher glauben wir, dass kontextabhängige Recommender-Algorithmen für die meisten Anwendungen mit verfügbaren Kontextinformationen kontextunabhängige Algorithmen übertreffen sollten. Welcher dieser beiden Ansätze dominiert, kann jedoch von vielen verschiedenen Faktoren abhängen, wie z.B. der Anwendungsdomäne und den Eigenschaften der verfügbaren Kontextdaten.

5. Wie lange dauert im Durchschnitt die automatische, vollständige Konfiguration durch einen Recommender-Algorithmus? Alle Algorithmen hatten in unseren Experimenten eine praktikable Laufzeit (bis zu 112,5 ms für einen 7-dimensionalen Tensor mit über hundert historischen Konfigurationen).

3.3 PROFilE: Tool-Unterstützung

Im Rahmen unserer Forschung entwickelten wir den open-source SPL-Konfigurator PROFilE. Dieser ist als Erweiterung des SPL-Tools FeatureIDE [Th14] öffentlich verfügbar. PROFilE bietet eine Funktion zum Ausblenden von Teilen des Variabilitätsmodells mit dem Ziel, relevante Informationen in fokussierten Ansichten darzustellen. [Pe16a]. Weiterhin bietet es eine Reihe von miteinander verknüpften Konfigurationsansichten, die dem Benutzer für eine Menge von Produktanforderungen die Auswirkungen seiner Entscheidungen auf andere Features und auf nicht-funktionale Eigenschaften visualisiert [Pe18a]. Darüber hinaus werden relevante Features bewertet (5-Sterne-Klassifizierung) und nach ihrer Wichtigkeit für den Benutzer kategorisiert (Top-10-Merkmale). Wir bewerten die Leistung von PROFilE

in Bezug auf *Effektivität*, *Skalierbarkeit* und *Effizienz* für elf SPL [Pe18a]. Die Ergebnisse unserer Experimente zeigen, dass die Verwendung eines Recommender-Systems: *(i)* den Konfigurationsprozess vereinfacht, *(ii)* die Erfüllung von Anforderungen an das Endprodukts erhöht und *(iii)* die mentale Belastung für die Entscheidungsträger erheblich reduziert.

4 Fazit

Diese Arbeit besteht aus vier Phasen. Erstens, führen wir ein SLR zum SPL-Konfigurationsprozess durch, um die grundlegenden Herausforderungen in diesem Forschungsfeld zu erfassen. Zweitens, stellen wir personalisierte Recomender-Algorithmen für den Konfigurationsprozess vor. Drittens, evaluieren wir die Algorithmen anhand mehrerer realer SPL. Viertens, basierend auf Erkenntnissen aus früheren empirischen Studien ([Co16, Pe13]), erweitern wir einen etablierten Konfigurator mit unseren Recommender-Algorithmen. Darüber hinaus schlagen wir eine Reihe interaktiver und automatisierter visueller Mechanismen vor, die Benutzer bei der Featureauswahl unterstützen und unnötige, sowie ungültige Entscheidungen verhindern. Dadurch können Benutzer: *(a)* sich auf für sie relevante Features fokussieren; *(b)* implizite und explizite Abhängigkeiten zwischen funktionalen und nicht-funktionalen Eigenschaften erkennen; *(c)* auf eine Liste von am meisten empfohlenen Features zurückgreifen; und *(d)* an jedem Punkt im Konfigurationsprozesses die Konfiguration automatisiert vervollständigen lassen. In zukünftige Arbeiten wollen wir erforschen, wie sich die Anzahl der vorherigen Konfigurationen auf die Effektivität der Algorithmen und Visualisierungsmechanismen auswirkt.

Literaturverzeichnis

[As14] Asadi, Mohsen; Soltani, Samaneh; Gasevic, Dragan; Hatala, Marek; Bagheri, Ebrahim: Toward automated feature model configuration with optimizing non-functional requirements. Information and Software Technology (IST), 56(9):1144–1165, 2014.

[Ba17] Bashroush, Rabih; Garba, Muhammad; Rabiser, Rick; Groher, Iris; Botterweck, Goetz: CASE tool support for variability management in software product lines. ACM Computing Surveys (CSUR), 50(1):14:1–14:45, 2017.

[Be13a] Benavides, David; Felfernig, Alexander; Galindo, José A; Reinfrank, Florian: Automated analysis in feature modelling and product configuration. In: Safe and Secure Software Reuse, S. 160–175. Springer, 2013.

[Be13b] Berger, Thorsten; Rublack, Ralf; Nair, Divya; Atlee, Joanne M.; Becker, Martin; Czarnecki, Krzysztof; Wasowski, Andrzej: A survey of variability modeling in industrial practice. In: Proceedings of the Workshop on Variability Modelling of Software-intensive Systems (VaMoS). ACM, S. 7:1–7:8, 2013.

[BE14] Bagheri, Ebrahim; Ensan, Faezeh: Dynamic decision models for staged software product line configuration. Requirements Engineering Journal (REJ), 19(2):187–212, 2014.

[BSRC10] Benavides, David; Segura, Sergio; Ruiz-Cortés, Antonio: Automated analysis of feature models 20 years later: a literature review. Information Systems, 35(6):615–708, 2010.

[Co16] Constantino, Kattiana; Pereira, Juliana Alves; Padilha, Juliana; Vasconcelos, Priscilla; Figueiredo, Eduardo: An empirical study of two software product line tools. In: International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE). Springer, 2016.

[Ga15] Galindo, José A; Dhungana, Deepak; Rabiser, Rick; Benavides, David; Botterweck, Goetz; Grünbacher, Paul: Supporting distributed product configuration by integrating heterogeneous variability modeling approaches. Information and Software Technology (IST), 62:78–100, 2015.

[Ja08] Janota, Mikolás: Do SAT solvers make good configurators? In: International Systems and Software Product Line Conference (SPLC). Springer, S. 191–195, 2008.

[Ka90] Kang, Kyo C.; Cohen, Sholom G.; Hess, James A.; Novak, William E.; Peterson, A. Spencer: Feature-oriented domain analysis (FODA) feasibility study. Bericht CMU/SEI-90-TR-21, Software Engineering Institute, 1990.

[KBV09] Koren, Y.; Bell, R.; Volinsky, C.: Matrix factorization techniques for recommender systems. Computer, 42:30–37, August 2009.

[KC07] Kitchenham, B.; Charters, S: Guidelines for performing Systematic Literature Reviews in Software Engineering. Citeseer, 2007.

[Oc17] Ochoa, Lina; Pereira, Juliana Alves; González-Rojas, Oscar; Castro, Harold; Saake, Gunter: A survey on scalability and performance concerns in extended product lines configuration. In: Proceedings of the Workshop on Variability Modelling of Software-intensive Systems (VaMoS). ACM, S. 5–12, 2017.

[Oc18] Ochoa, Lina; Gonzalez-Rojasa, Oscar; Pereira, Juliana Alves; Castro, Harold; Saake, Gunter: A Systematic Literature Review on the Semi-Automatic Configuration of Extended Product Lines. Journal of Systems and Software, 2018. Accepted.

[PCF15] Pereira, Juliana Alves; Constantino, Kattiana; Figueiredo, Eduardo: A systematic literature review of software product line management tools. In: International Conference on Software Reuse (ICSR). Springer, S. 73–89, 2015.

[Pe13] Pereira, Juliana Alves; Souza, Carlos; Figueiredo, Eduardo; Abilio, Ramon; Vale, Gustavo; Costa, Heitor Augustus Xavier: Software variability management: an exploratory study with two feature modeling tools. In: Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS). IEEE, S. 20–29, 2013.

[Pe16a] Pereira, Juliana Alves; Krieter, Sebastian; Meinicke, Jens; Schröter, Reimar; Saake, Gunter; Leich, Thomas: FeatureIDE: scalable product configuration of variable systems. In: International Conference on Software Reuse (ICSR), S. 397–401. Springer, 2016.

[Pe16b] Pereira, Juliana Alves; Matuszyk, Pawel; Krieter, Sebastian; Spiliopoulou, Myra; Saake, Gunter: A feature-based personalized recommender system for product-line configuration. In: ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences (GPCE). ACM, S. 120–131, 2016.

[Pe17] Pereira, J. A.: Personalized Recommender Systems for Software Product Line Configurations. Dissertation, University of Magdeburg, Germany, 2017.

[Pe18a] Pereira, Juliana Alves; Martinez, Jabier; Gurudu, Hari Kumar; Krieter, Sebastian; Saake, Gunter: Visual Guidance for Product Line Configuration Using Recommendations and Non-Functional Properties. In: ACM Symposium on Applied Computing (SAC). ACM, 2018.

[Pe18b] Pereira, Juliana Alves; Matuszyk, Pawel; Krieter, Sebastian; Spiliopoulou, Myra; Saake, Gunter: Personalized Recommender Systems for Product-line Configuration Processes. Computer Languages, Systems & Structures (COMLAN), 2018.

[Pe18c] Pereira, Juliana Alves; Schulze, Sandro; Figueiredo, Eduardo; Saake, Gunter: N-dimensional Tensor Factorization for Self-Configuration of Software Product Lines at Runtime. In: International Systems and Software Product Line Conference (SPLC). ACM, 2018. to appear.

[Pe18d] Pereira, Juliana Alves; Schulze, Sandro; Krieter, Sebastian; Ribeiro, Márcio; Saake, Gunter: A Context-Aware Recommender System for Extended Software Product Line Configurations. In: Proceedings of the Workshop on Variability Modelling of Software-intensive Systems (VaMoS). ACM, S. 1–8, 2018.

[Si10] Sincero, Julio; Tartler, Reinhard; Egger, Christoph; Schröder-Preikschat, Wolfgang; Lohmann, Daniel: Facing the linux 8000 feature nightmare. In: European Conference on Computer Systems (EuroSys). 2010.

[Th14] Thüm, Thomas; Kästner, Christian; Benduhn, Fabian; Meinicke, Jens; Saake, Gunter; Leich, Thomas: FeatureIDE: an extensible framework for feature-oriented software development. Science of Computer Programming (SCP), 79(0):70–85, 2014.

[TLL14] Tan, Lei; Lin, Yuqing; Liu, Li: Quality ranking of features in software product line engineering. In: Asia-Pacific Software Engineering Conference (APSEC). Jgg. 2. IEEE, S. 57–62, 2014.

Juliana Arriel wurde am 22. August 1989 in Brasilien geboren. Sie hat ihr Informatik Studium 2012 mit einem Bachelor an der Universität Lavras (Brasilien) und 2014 mit einem Master an der Universität Minas Gerais (Brasilien) abgeschlossen. Im Jahr 2018 promovierte sie an der Otto-von-Guericke Universität Magdeburg mit Auszeichnung. Ihre Doktorarbeit wurde mit dem Fakultätspreis für die beste Dissertation ausgezeichnet. Zum jetzigen Zeitpunkt ist sie Postdoc an der Universität Rennes 1 (Frankreich). Der Fokus ihrer Forschung ist automatisierte Software Entwicklung unter Anwendung und Kombination von Methoden aus der Software Analyse, dem Maschinenlernen und der metaheuristischen Optimierung. Sie ist regelmäßig an der Organisation von bedeutenden internationalen Konferenzen beteiligt. Außerdem publiziert sie in und begutachtet für führende Zeitschriften und Tagungen im Bereich Software Entwicklung. Sie ist Mitglied in der Association for Computing Machinery (ACM) und der Arbeitsgruppe DiverSe (Diversity-Centric Software Engineering) an der Universität Rennes 1.