Gottfried Wilhelm
Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Praktische Informatik
Fachgebiet Software Engineering

Adaption und Visualisierung von Informationsfluss-Modellen nach System Dynamics als Weblösung für JIRA

Bachelorarbeit

im Studiengang Informatik

von

Alper Yeniyayla

Prüfer: Prof. Dr. Kurt Schneider
Zweitprüfer: Dr. Jil Klünder
Betreuer: M. Sc. Fabian Kortum

Hannover, 18.11.2019

Erklärung der Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig und ohne fremde Hilfe verfasst und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet habe. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Hannover, den 18.11.2019

_____

Alper Yeniyayla

Zusammenfassung

Damit in Softwareentwicklungsteams die Projektarbeit reibungslos erfolgen kann, ist eine gute Kommunikation von Nöten. Ist diese nicht gewährleistet kann es zu Verzögerungen oder gar dem Scheitern eines Projektes kommen. Um eine hochwertige Kommunikation zu gewährleisten, können bestehende Strukturen und Arbeitsverläufe mit der FLOW-Methode modelliert und als Ganzes betrachtet werden, um sie damit transparenter zu machen. Jedoch lassen sich dadurch dynamische Komponenten wie Kommunikationsintensivität, krankheitsbedingte Ausfälle oder dem Wegfallen von Kommunikationskanälen nicht modellieren, da der FLOW-Analyst mit dem FLOW-Modell nur die statische Gesamtstruktur darstellen kann. Mit dieser Arbeit soll diese Aufgabe erleichtert werden, indem ein weiteres Werkzeug bereitgestellt wird, welches bereits vorhandene FLOW-Modelle des FlowExplorers in die System Dynamics Terminologie umwandelt und die genannten dynamischen Faktoren simulierbar macht. Dabei wird die FLOW-Methode analysiert, um sie sinnvoll in die System Dynamics Terminologie zu überführen und anschließend mittels JIRA-Plugin automatisch zu konvertieren sowie anzuzeigen und zu simulieren. Dadurch werden die Kommunikationsstrukturen greifbarer, indem mittels explorativer Simulation realer Szenarien Schlüsse hinsichtlich der Strukturen gezogen werden können. Mittels der Simulation der dynamischen Komponenten konnten Problemfälle erkannt werden, die bisher durch die zeitinvariante Betrachtung verborgen blieben und sinnvolle Gegenmaßnahmen getroffen werden, um die Auswirkung dieser zu minimieren. Dadurch ist es möglich Kommunikationsstrukturen so anzupassen, dass sie sowohl effektiver als auch störsicherer sind, welches sich enorm auf den Erfolg eines Projektes auswirkt.

Abstract

Good communication between team members is mandatory when it comes to trouble-free work processes. If such communication is not maintained then running projects may be delayed or even failed by meeting the deadline without completion. In order to compensate for this, the FLOW-Methodology was invented so model communication structures and work processes can be modelled and viewed as a whole to make them more transparent. However, this is not enough, as dynamic influences such as fluctuating communication intensiveness, absence due to illness or even lost communication channels cannot be taken into account since the FLOW-Analyst can only view the static structures inside the FLOW-model as a whole. With this bachelor thesis another tool will be provided to tackle these issues by converting already existing FLOW-Models created by the FlowExplorer into the System Dynamics terminology and making the above mentioned factors simulatable. For this task the FLOW-Methodology will be analysed to transfer the syntax and semantics of FLOW into System Dynamics so an automated converter embedded in a JIRA-Plugin can convert models and make them simulatable. With the simulation of the mentioned dynamic components the viewed structures are less abstract and can be optimized due to the explorative simulation of real scenarios. By simulating these dynamic components it is possible to identify issues, which were hidden in the static time-invariant approach, so countermeasures can be applied to minimize the effects. In the end the FLOW-Analyst will be able to rearrange communication structures, so that they can communicate more effectively and more robust in arising communication issues which will greatly impact the success of a project.

Inhaltsverzeichnis

1 Einleitung

1.1 Problemstellung

Mit der FLOW-Methode ist es möglich Kommunikationsstrukturen als Informationsflussmodelle aus Interviews zu ermitteln und sie als Ganzes zu modellieren. Somit wird ein Überblick über die bestehenden Strukturen geschaffen und es wird ersichtlich wie Informationen zwischen projektinvolvierten Personen fließen um wichtige Knotenpunkte zu besprechen. Jedoch lassen sich dynamische Komponenten, wie den krankheitsbedingten Ausfall von Personen, den Ausfall von Kommunikationskanälen oder die Kommunikationsintensitäten zwischen Personen über die jeweils verwendeten Kanäle deswegen nicht modellieren. Dadurch bleibt das Modell in quantitativer Hinsicht sehr abstrakt und es lassen sich nur beschränkt Schlüsse ziehen, um die Flüsse anzupassen. Durch diese Einschränkung bleiben Engpässe aufgrund von schlechten Strukturen unentdeckt welches letztlich dazu führt, dass Teams Projekte nicht in der geplanten Zeit fertigstellen können und deswegen scheitern.

1.2 Lösungsansatz

Für die Berücksichtigung dieser Faktoren müssen die Informationsflussmodelle simulierbar gemacht werden. Die auf System Dynamics basierende Simulation soll mit FLOW-Modellen aus dem FlowExplorer arbeiten. Da diese mit System Dynamics inkompatibel sind, werden sie zuerst in das System Dynamics XMILE-Format umgewandelt und um eine dynamische Komponente erweitert. Der Simulator generiert auf Grundlage der angewählten Modellverhalten und Kommunikationsintensitäten Formeln, die das Flussverhalten beschreiben und simuliert diese mit der SD.js Bibliothek. Somit können reale Szenarien explorativ durch das An- und Ausschalten verschiedener Verhaltensmuster und dem Setzen der Kommunikationsintensitäten erforscht werden um Schlüsse für die Anpassung der Kommunikationsstrukturen aus den Ergebnissen zu ziehen. Damit der Konverter und Simulator auch effektiv dort genutzt werden können, wo sie gebraucht werden, werden sie in das bereits existierende JIRA-Plugin ”Prodynamics” integriert, um die Sammlung an Werkzeugen für Team-Dynamik zu ergänzen. Dadurch können die Kommunikationsintensitäten direkt aus dem ”Prodynamics”-Plugin entnommen und in die Simulation eingefügt werden. Durch die Pluginimplementation werden sowohl der Simulator als auch der Konverter in HTML und CSS designt und mittels Javascript programmiert.

1.3 Struktur der Arbeit

Kapitel 2 widmet sich den Grundlagen zu Informationsflussmodellen, FLOW, System Dynamics, JIRA und verweist auf relevante Arbeiten. Anschließend werden in Kapitel 3 Konzepte vorgestellt, wie die statischen FLOW-Modelle in dynamische System Dynamics Modelle umgewandelt werden und wie die verschiedenen Modellverhalten als Flussverhalten realisiert werden können. In Kapitel 4 werden die genannte Konzepte in Form des Konverters und des Simulators in Programmcode implementiert und an das Prodynamics-Plugin angebunden. Kapitel 5 befasst sich mit den aufgetretenen Problemen und wie sie gelöst wurden. Eine abschließende Betrachtung in Kapitel 6 fasst die Arbeit zusammen und liefert einen Zukunftsausblick.

2 Grundlagen

In diesem Kapitel werden die Grundlagen der Methoden und Programme erläutert, die in dieser Arbeit Verwendung finden. Dabei werden ebenfalls die Rahmenbedingungen der jeweiligen Thematiken in Bezug auf diese Arbeit erläutert. Abgeschlossen wird das Kapitel mit verwandten Arbeiten auf denen, für eine erfolgreiche Umsetzung, aufgebaut wurde.

2.1 FLOW: Informationsflüsse

InfoFLOW (”Informationsflussoptimierung für Softwareprojekte”, kurz ”FLOW”) ist die systematische Auseinandersetzung mit Informationsflüssen in der Softwareentwicklung. Dafür wurde die FLOW-Methode entwickelt, um diese Flüsse und ihre Strukturen anhand drei zentraler Elemente zu modellieren [1]. Im Konzept und späteren Verlauf der Arbeit werden nur Informationsflussmodelle betrachtet, die zur Modellierung von Kommunikationsstrukturen verwendet werden. Dabei werden bestimmte Annahmen darüber getroffen, wie die Modelle (speziell in der FLOW-Notation) aussehen müssen, damit eine semantisch logische Konvertierung durchgeführt werden kann.

2.1.1 Die FLOW-Notation

Die FLOW-Notation umfasst Informationsspeicher, Informationsflüsse und Aktivitäten. Informationsspeicher werden in Form von Dokumenten oder Personen dargestellt und Informationen als Pfeile zwischen Speichern und Aktivitäten. Personen und Dokumente werden durch ein Smiley bzw. einem Dokumentensymbol gekennzeichnet, wenn sie jeweils eine einzelne Person oder ein einziges Dokument referenzieren. Bei einer Gruppe an Personen oder einer Sammlung an Dokumenten wird das jeweilige Symbol mehrfach, versetzt angeordnet, übereinander wiederholt. Informationen werden in zwei verschiedene Kategorien klassifiziert, fest oder flüssig. Sie gelten als fest, sofern sie die folgenden Bedingungen erfüllen:

1. langfristig abrufbar
2. wiederholt abrufbar
3. für Dritte verständlich[1]

Flüssige Informationen sind Informationen, die nicht fest sind.

[1] Dritte müssen nicht notwendigerweise Unbekannte sein, sondern können Personen aus dem Kontext des Modells wie z.B. des Projektes oder des Unternehmens sein.

Analog dazu, werden Personen als flüssige und Dokumente als feste Informationsspeicher klassifiziert [1].

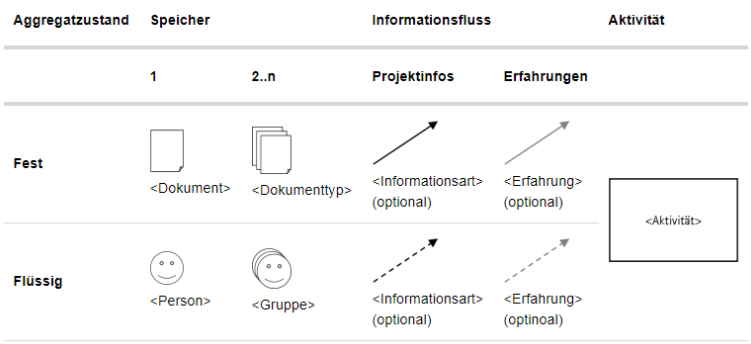

Abbildung 2.1: Syntax und Semantik von FLOW [2]

2.1.2 Der FLOW-Explorer

Für die Erstellung eines FLOW-Modells wurde im Rahmen einer vorhergegangenen Bachelorarbeit von Höppner[3] der FlowExplorer programmiert. Mit diesem ist es möglich, aus Interviews hergeleitene Projekt- und Teamstrukturen zu modellieren und interaktiv anzupassen. Des Weiteren erlaubt das Programm mehrere FLOW-Modelle zusammenzuführen, die aus verschiedenen Interviews entstanden sind sowie 'Bad Patterns' aufzudecken und eine rudimentäre Informationsflussanalyse durchzuführen.

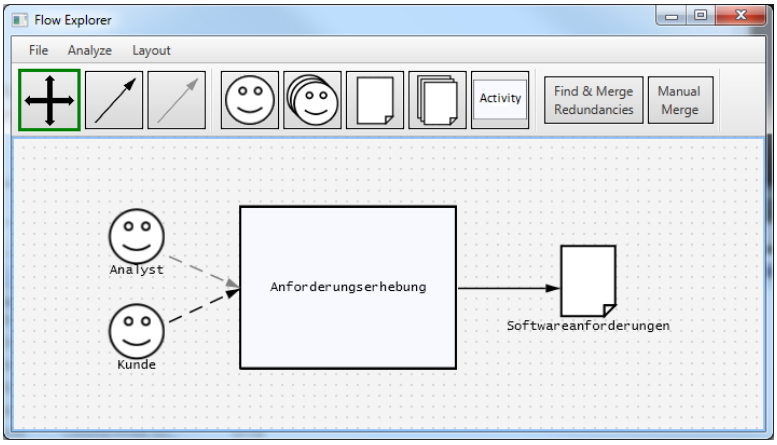

Abbildung 2.2: Anforderungserhebung einer Software als FLOW-Modell

Der FlowExplorer wird im weiteren Verlauf dieser Arbeit verwendet um die FLOW-Modelle zu erstellen und sie als XML-Dateien abzuspeichern, welche durch den Konverter um die dynamische Komponente erweitert und an den Simulator übergeben werden. Dabei werden Annahmen und Voraussetzungen getroffen um, eine erfolgreiche Konvertierung eines initial statischen FLOW-Modells in ein dynamisches System

Dynamics Modell zu überführen.

2.1.3 Kommunikationsindex als Informationsflussmaß

Um die Effektivität von Kommunikation gewichten zu können, damit der Informationsfluss quantifiziert werden kann, wird für den Simulator der Kommunikationsindex in Bezug mit den Gewichtungen nach Klünder [4] als Metrik verwendet. Dieser Index setzt sich zusammen aus der Summe der Produkte der Kanalintensitäten zwischen zwei Personen und der Gewichtung des Kanals.

$$c_{ij} := \sum_{k \in K} int_{ij,k} * w_k \tag{2.1}$$

Die Intensitäten, welche auf einer Skala von 0 (keine Kommunikation) bis 4 (maximale Kommunikation) reichen, werden aus Umfragen mit den Personen bestimmt und je Kanal in einer eigenen symmetrischen Matrix[2] festgehalten. Die in diesem Zusammenhang verwendeten Kommunikationskanäle[3] und deren Gewichtungen werden direkt aus der Definition übernommen und lauten wie folgt:

Kanal	**Wert**
Face-to-Face	4
Video-Chat	3
Chat	2
Telefonat	2
E-Mail	1

Tabelle 2.1: Gewichtung der Kommunikationskanäle [4]

Da in Kapitel 3 und 4 distinkte Information vorausgesetzt wird, kann synonymisch für die Stärke des Informationsflusses der Kommunikationsindex angenommen werden, denn die Effektivität eines Kommunikationskanals ist proportional zu der Menge an Informationen die darüber ungestört fließen können. Dadurch lassen sich die Indizes addieren oder subtrahieren (sinngemäß Steigen/Sinken des Informationsflusses), um den aktuellen Informationsfluss über eine Person zu ermitteln, womit in der Simulation gerechnet werden kann. Dies wird bei der Betrachtung des folgenden Beispiels ersichtlich:

[2] Durch die subjektive unterschiedliche Bewertung der beteiligten Personen werden die Werte gemittelt. Füllt ein Teammitglied den Umfragebogen nicht aus, wird nur der Wert seines Partners verwendet.

[3] Der Face-to-Face Kanal wird auch im gewissen Kontext als Meeting bezeichnet. Gemeint ist hier jedoch immer die direkte Kommunikation ohne Zwischenmedium wie Chats, Telefonaten oder ähnlichem.

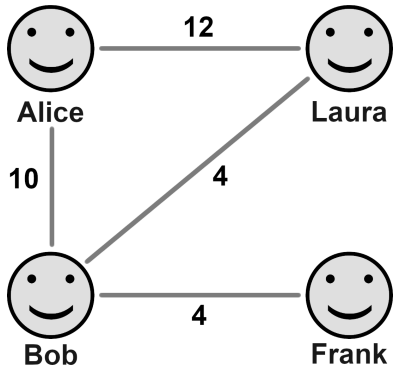

Abbildung 2.3: Kommunikationsnetzwerk mit vier Personen

Gegeben ist ein Kommunikationsnetzwerk bestehend aus Alice, Laura, Bob und Frank, wobei die Werte an den Kanten die jeweiligen Kommunikationsindizes der verbundenen Personen wiedergeben. Nicht vorhandenen Kanten bedeuten, dass zwischen den Personen keine Kommunikation stattfand und der Kommunikationsindex dementsprechend null beträgt. Aufgrund der Proportionalität zwischen Kommunikationsindex und der Informationsflussmenge, können die verbundenen Kanten einer Person aufaddiert werden. Daraus errechnet sich ein Wert der angibt wie viel Informationen i über eine Person in absoluter Menge fließen.

$$
\begin{aligned}
i_{Alice} &= c_{Alice,Laura} + c_{Alice,Bob} = 12 + 10 = 22 \\
i_{Bob} &= c_{Alice,Bob} + c_{Bob,Laura} + c_{Bob,Frank} = 10 + 4 + 4 = 18 \\
i_{Laura} &= c_{Alice,Laura} + c_{Bob,Laura} = 12 + 4 = 16 \\
i_{Frank} &= c_{Bob,Frank} = 4
\end{aligned}
$$

Modelliert dies den Normalzustand, so lässt sich beispielsweise bei personellen Ausfällen die Zusatzbelastung anhand der neuverteilten Informationsmenge ablesen. Damit das bearbeitete Projekt nicht in Verzug gerät, muss der Ausfall zeitnah mit den entsprechenden Verhaltensmustern abgefedert werden. Die fehlende Kommunikation könnte beispielsweise gleichmäßig auf die benachbarten Knoten, oder proportional, je nach Belastung, aufgeteilt werden. Dies ist essenziell für die Simulation und soll als Kernelement dafür dienen, explorativ Szenarien durchzuspielen, um zu sehen, welches Muster angewandt werden könnte oder ob personelle Strukturänderungen vorgenommen werden müssen.

2.2 System Dynamics

Die System Dynamics Methodik wurde von Jay W. Forrester Mitte der 1950er Jahre entwickelt um komplexe Systeme mit Hilfe von Stocks, Flows und Feedback Loops über die Zeit qualitativ und quantitativ zu beschreiben [5]. Damit sollten komplexe, nicht-lineare Systeme greifbarer gemacht werden, indem ein Modell eines Systems

abgeleitet wird um mit einer Simulation Rückschlüsse auf gute bzw. verbesserungswürdige Systemverhaltensweisen zu ziehen. System Dynamics wird im Rahmen dieser Arbeit als Simulationsframework verwendet, um die Informationsflüsse der Kommunikationen zu simulieren und zu visualisieren.

2.2.1 Stocks, Flows und Auxiliaries

Zu fast jedem System Dynamics Modell gehören die drei essenzielle Bausteine Stocks, Flows und Auxiliaries. Diese speichern Bestände (”Stocks”), stellen die Ein- und Ausgangsflüsse (”Flows”) und steuern initiale Bestände bzw. Raten von Ein- und Ausgangsflüssen (”Auxiliaries”).

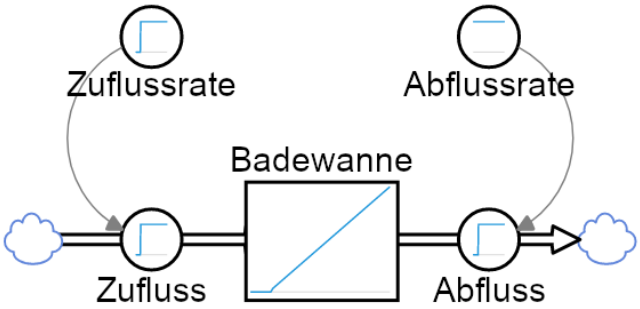

Abbildung 2.4: System Dynamics Modell eines Wasserflusses (adaptiert von [9])

Stocks speichern Systemzustände (als Bestände) und können über die eingehenden und ausgehenden Flüsse diesen Bestand vermehren oder verkleinern. Fließt mehr in einen Stock als rausfließt, so akkumuliert sich der Bestand und der Wert steigt an. Je nach Modellsemantik kann dieser Wert gegen einen Sättigungswert konvergieren oder divergieren. In Abbildung 2.4 ist das Element ”Badewanne” ein Stock, wobei der Wasserpegel, welchen den Bestand darstellt, zuerst null beträgt, da der Wasserhahn noch nicht aufgedreht ist und anschließend langsam steigt, da mehr Wasser einfließt als durch den Ablauf abfließt. Stock Bestände können als Differentialgleichung 1. Ordnung wie folgt ausgedrückt werden [7]:

$$stock(t) = stock(t_0) + \int_{t0}^{t} (inflow(t) - outflow(t))dt \tag{2.2}$$

Simulatoren lösen diese Differentialgleichungen numerisch mit Hilfe von verschiedenen Algorithmen, welche im Modell spezifiziert werden können, wobei der Integrationsschritt die Genauigkeit der Berechnung beeinflusst. Durch die numerische Lösung kann es zu Problemen kommen, wenn in Formeln Trigger für diskrete Zeitpunkte gesetzt sind, welches später in Kapitel 4 näher erläutert wird.

Damit Stocks ihren Bestand variieren können, sind Flows notwendig, da diese beschreiben, wie stark sich der Bestand verändert und können als Materialfluss/Warenfluss verstanden werden. Im obigen Beispiel ist ”Zufluss” der Eingangsfluss in den ”Badewanne”-Stock und ”Abfluss” dessen Ausgangsfluss.

Da Flows nur durch algebraische Ausdrücke und nicht durch Konstanten definiert werden dürfen, ist das dritte Element nötig, um die Werte von Flows und Stocks zu steuern. Dies geschieht mit Hilfe von Auxiliaries, welche im Gegensatz zu Stocks keinen Bestand haben, sondern einen Ausgangswert, der den Wert des Auxiliaries nicht verändert, wenn ein Zeitschritt simuliert wurde. Durch diese Eigenschaft werden sie dafür verwendet, um neuen Bestand/neue Ware in ein Modell einzuführen. Des Weiteren ist es möglich den Wert eines Auxiliary auf den Bestand eines Stocks zu referenzieren, ohne den Bestand eines Stocks zu verändern, dafür aber den Wert des Auxiliaries. ”Zuflussrate” und ”Abflussrate” sind die Auxiliaries im Beispielmodell, welche die Flussmengenbegrenzung der ”Zufluss”- und ”Abfluss”-Flüsse steuern.

2.2.2 Modelle mit XMILE

Um das Dateiformat für Modelle für System Dynamics Software zu vereinheitlichen, wurde von Diker und Allen ein Standard namens ”XML Interchange Language for System Dynamics” (kurz ”XMILE”[4]) definiert [6]. System Dynamics Software wie Vensim, iThink und STELLA implementieren diesen Standard um die Kompatibilität von Modellen über verschiedene System Dynamics Software zu gewährleisten, da vorher jede Software ihr eigenes proprietäres Dateiformat verwendete. Der Konverter und Simulator wird diesen umfangreichen Standard nur im Rahmen zur Bewältigung der Aufgabenstellung implementieren, damit das Simulationsframework SD.js, das konvertierte Modell simulieren kann.

2.2.3 OpenSource durch SD.js

Für die Implementation als JIRA-Plugin ist es notwendig, sowohl den Konverter als auch den Simulator in Javascript zu schreiben. Da eine portable Ansicht von Modellen bisher weitestgehend nur über die Darstellung und Simulation des Modells auf den Servern der System Dynamics Programmbetreiber möglich ist, ist eine Unabhängigkeit erstrebenswert.

Das System Dynamics Simulationsframework SD.js[5] erlaubt mit seiner liberalen MIT-Lizenz diese Unabhängigkeit. SD.js wurde fast ausschließlich von Robert Powers in Typescript programmiert und ist hervorragend geeignet, um die Simulationsgrundlage für die Kommunikationsstrukturen zu liefern. Verwendet wird die SD.js 5.2.7, welche direkt von der sdlabs.io heruntergeladen wurde und aufgrund kleiner Zeichensatzfehler sehr leicht modifiziert wurde.

[4] Online-Dokumentation: http://docs.oasis-open.org/xmile/xmile/v1.0/csprd01/xmile-v1.0-csprd01.html
[5] SD.js Produktseite: https://sdlabs.io

2.3 Projekt Management System JIRA

JIRA[6] ist eine webbasierte Projektmanagentsoftware von Atlassian, welche zur Unterstützung agiler Entwicklungsmethoden in der Softwareentwicklung verwendet wird. Atlassian stellt dabei ein SDK bereit, die es erlaubt Plugins für JIRA zu kompilieren. Durch das Plugininterface können die Erweiterungen eingebunden werden, um Teams noch weiter zu unterstützen. Die aktuelle JIRA-Version ist 8.4.0 und die aktuelle SDK-Version ist 8.0.16, verwendet wird aber die JIRA-Version 7.9.0 und die SDK-Version 6.3.10, da es ansonsten zu Inkonsistenzen zwischen dem Prodynamics-Plugin und JIRA kommen kann, welche erst vor der Fertigstellung des Plugins aufgelöst werden.

2.3.1 Prodynamics für zusätzliches Team-Feedback

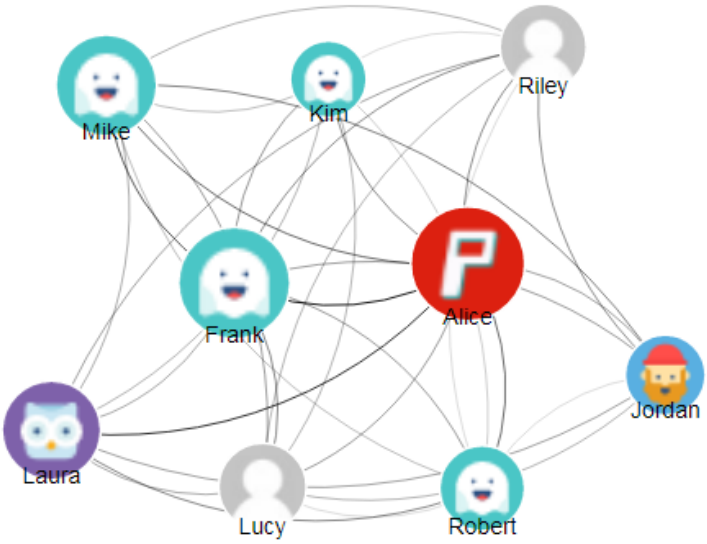

Abbildung 2.5: Visualisierung eines Kommunikationsnetzwerkes mit Prodynamics aus den mitgelieferten Datensätzen

Eingebunden durch das Plugininterface wird das ”Prodynamics”-Plugin [12] welches Werkzeuge für Team-Feedback und -Dynamik bereitstellt. Programmiert wird das Plugin vom Fachgebiet für Software Engineering am Praktischen Institut für Informatik an der Leibniz Universität Hannover und ist momentan noch nicht öffentlich zugänglich, da sie noch nicht publiziert wurde. Mit dem Prodynamics-Plugin ist es möglich Stimmungen, Produktivität, Anwesenheit und diverse andere Faktoren von Team-Mitgliedern und Projekten zu erfassen und auszuwerten. Aus diesem Grund wird der Konverter und Simulator auch über das bereits existierende Plugin eingebunden, da diese eine Ergänzung zu Prodynamics darstellen. Dabei wird aus Gründen der Zeitersparnis sowohl der Konverter als auch der Simulator erst als eigenständige Webseite programmiert und nachträglich in das Plugin eingefügt.

[6] Atlassian JIRA Produktseite: https://www.atlassian.com/de/software/jira

2.4 Verwandte Arbeiten

Höppner verfolgte im Rahmen einer Bachelorarbeit unter anderem die Informationsflussanalyse von FLOW-Modellen [3]. Realisiert wurde die Funktionalität durch einen Algorithmus, der die Personensymbole bei niedrigem Informationsfluss grün und bei stärkerem Fluss rot färbt. Die Auslastung wird anhand der eingehenden und ausgehenden Kanten relativ zur am maximal ausgelasteten Person gemessen. Da jede Kante allerdings gleich gewichtet ist, lassen sich dadurch nur bedingt quantitative Aussagen über das Flussverhalten treffen. Auf diesen Punkt wird diese Arbeit aufbauen, indem ein besseres Konzept zur Analyse von Informationsflussbelastung vorgestellt und implementiert wird. Dies geschieht mittels des Kommunikationsindex [10], welcher stellvertretend als Informationsflussmenge eingesetzt wird.

Der oben genannte Kommunikationsindex wird im Rahmen der Dissertation von Klünder [4] genannten Kommunikationskanäle und verwendeten Gewichtungen besagter Kanäle verwendet. Dies erlaubt eine Auffächerung der verwendeten Kanäle samt der Kommunikationsintensität. Dadurch lassen sich die Informationsflüsse für die Simulation quantifizieren, um qualitative Aussagen über das Modell bei verschiedenen Szenarien treffen zu können, welches essenziell für die erfolgreiche Bewältigung dieser Bachelorarbeit ist.

Kortum et. al [12] entwickelten zur Unterstützung von Teams das JIRA-Plugin Prodynamics welches viele Werkzeuge bietet, um Team Dynamics Faktoren zu betrachten und zu dokumentieren. Dieses Plugin ist essentiell für diese Arbeit, da sowohl der Konverter als auch Simulator dort implementiert werden, um das Plugin mit weiteren Werkzeugen zu erweitern. Das bereitgestellte Template wird dabei zur Implementation verwendet.

3 Konzept

In diesem Kapitel werden die Konzepte vorgestellt, wie die statischen FLOW-Modelle um die dynamische Komponente erweitert und in die System Dynamics Terminologie übersetzt werden können. Dabei werden zuerst Voraussetzungen ermittelt und Annahmen im Verwendungskontext getroffen, um eine sinnvolle Konvertierung zu ermöglichen. Anschließend werden die FLOW-Elemente in System Dynamics Modellkomponenten abstrahiert und die Formeln für die verschiedenen Modellverhalten hergeleitet. Zum Schluss werden noch jeweils ein Programmentwurf vorgestellt welcher im nächsten Kapitel implementiert wird.

3.1 Konvertierung von statischen nach dynamischen Modellen

3.1.1 Annahmen und Voraussetzungen

Die von Natur aus statischen FLOW-Modelle können nur in die System Dynamics Terminologie überführt werden, wenn eine syntaktisch und semantisch korrekte Konvertierung stattfindet. Wie bei der FLOW-Modellierung können von unterschiedlichen FLOW-Analysten als auch bei er System Dynamics Modellierung verschiedene Modelle entstehen, die jedoch die selbe Ausgangssituation darstellen sollen. Dadurch entsteht eine gewisse Varianz bezüglich der Modellstruktur und diese kann dazu führen, dass der Konverter nachher die Elemente zwar algorithmisch korrekt übersetzt, allerdings der Simulator auf Grund von unbekannten Strukturen die Modellverhalten nicht korrekt anwenden kann. Um dieses Problem zu eliminieren werden entsprechende Voraussetzungen ermittelt und Annahmen über das spätere Modell getroffen.

Gleichmäßiger bidirektionaler Informationsfluss durch Kommunikationssymmetrie

Flüsse in System Dynamics sind stets gerichtet, jedoch bereitet dies im Hinblick auf Informationsflüsse von Kommunikationsstrukturen Probleme, denn nach Watzlawick et. al gelten unter anderem folgende Axiome [11]:

1. Man kann nicht nicht kommunizieren.
2. Zwischenmenschliche Kommunikationsabläufe sind entweder symmetrisch oder komplementär.

Da Informationsflüsse zwischen Personen betrachtet werden, eliminiert das erste beider Axiome die gerichteten System Dynamics Flows, welche die Informationsflüsse darstellen sollen. Es muss der Informationsfluss zwischen zwei Personen in beide Richtungen betrachtet werden, darum werden stellvertretend für ungerichtete Flüsse, gerichtete Flüsse in beide Richtungen angenommen. Durch die Betrachtung des Tagesdurchschnitts der Informationsflüsse in der Simulation (siehe Kapitel 3.2.1) kann der Informationsfluss ebenfalls als Durchschnitt betrachtet werden. Dies wird gestützt durch das zweite Axiom, da es sich beinahe immer um symmetrische Kommunikationspartner handelt, weil Teamstrukturen in der Softwareentwicklung betrachtet werden. Die gewählte Metrik untermauert diese Annahme, da sie die gemittelten Intensitäten zwischen zwei Kommunikationspartnern betrachtet.

Syntaktische und semantische Vorgaben für die Eingabe-Modelle

Durch die Einsatzmöglichkeit von FLOW in verschiedenen Bereichen ist es nötig festzusetzen, welche Einschränkungen benötigt werden. Zwar fließen in nahezu jedem FLOW-Modell Informationen zwischen Personen und Aktivitäten, jedoch werden in der Simulation nur Modelle von Kommunikationsstrukturen sinnvoll simuliert. Um die erwähnte Varianz zu auszuschließen, wird für die ordnungsgemäße Funktionalität des Simulators eine eingeschränkte FLOW-Syntax und Semantik verwendet.

Da Kommunikationsstrukturen simulierbar gemacht werden sollen, wird der Konverter nur Modelle korrekt verarbeiten können, die diese darstellen. Alle anderen Eingabemodelle werden fehlerhaft konvertiert. Um die Modelle eindeutig zu beschreiben, müssen diese wie folgt modelliert werden:

Als Aktivitäten werden die Kommunikationskanäle verstanden, über welche die Personen kommunizieren, von denen Informationsflüsse in den Eingang der Aktivität gehen. Dabei werden die verwendeten Kommunikationskanäle der Partner mit einem Komma getrennt als Text der Aktivität gesetzt. Sollte bei der Kommunikation jedoch nicht klar sein, welche Kanäle genutzt wurden, kann stellvertretend nur ”Unbekannt” gesetzt werden um dies zu kennzeichnen. Aufgrund des angenommenen bidirektionalen Informationsflusses, ist es nicht nötig, den Ausgang einer Aktivität zurück zu den kommunizierenden Personen zu verbinden. Zwar würde dies in der ursprünglichen Anwendungsweise von FLOW Sinn machen, jedoch würden sich dadurch die Anzahl der Kanten eines Kommunikationsmodell ohne syntaktischen Mehrwert verdoppeln und zu einer visuellen Überladung des Modells führen.

In der Abbildung 3.1 kommunizieren Frank und Bob nur per Chat, Bob, Alice und Laura per FaceToFace (Meeting), und Alice und Laura verwenden FaceToFace und Telefonanrufe. Damit bei einem größeren Modell für ein Meeting nicht Personen einzeln in eine Aktivität geführt werden müssen, ist es möglich mit dem Gruppensymbol einfach die beteiligten Personen mittels Komma zu trennen. Semantisch macht dies, verglichen mit einzelnen Verbindungen, einen großen Unterschied, da Gruppensymbole andere Modellverhalten implementieren sollen um die Menge an Simulationsszenarien zu erhöhen welche in Kapitel 3.2.1 detailliert erläutert sind.

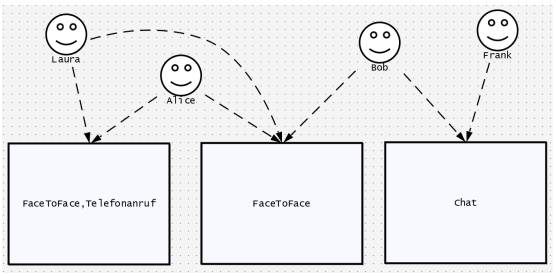

Abbildung 3.1: Beispiel eines einfachen Kommunikationsmodells

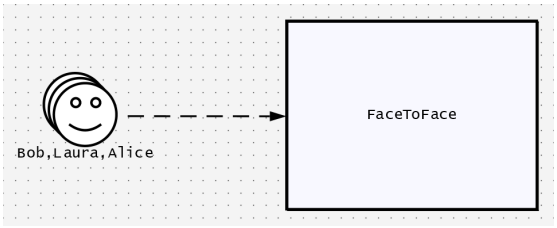

Abbildung 3.2: Kommunikation mittels Gruppensymbol

Für eine erfolgreiche Simulation des Modells ist es noch wichtig die Informationsflüsse zu lenken bzw. über andere Knoten fließen zu lassen, sofern dieser ausfällt oder überbelastet ist. Eine automatische Ermittlung der infragekommender Partner ist dabei von Nachteil, denn dies ist mit der gegebenen FLOW-Syntax nicht praktikabel umzusetzen, da verschiedene Faktoren in die Ermittlung einfließen können. Stattdessen gibt der Modellierer die alternativen Flussmöglichkeiten vor, welche bei den verschiedenen Modellverhalten genutzt werden können. Realisiert wird das ganze durch die Auswahl von Stellvertretern welche bei Ausfall des Kommunikationspartners einspringen. Abbildung 3.3 beschreibt so eine Flussmöglichkeit.

Im Normalfall kommunizieren Max und Sandra über den FaceToFace-Kanal, jedoch springt Kim für Sandra ein, sofern Sandra krank sein sollte und übernimmt solange ihren Teil der Kommunikation, bis sie wieder anwesend ist. Durch die Auswahl verschiedener Einstellungsmöglichkeiten soll es im Simulator später möglich sein, einen passenden Stellvertreter auszuwählen, welches es erlaubt, mehrere Stellvertreter beim Modellieren anzugeben. Personengruppen die mittels Gruppensymbole dargestellt sind und in eine Aktivität laufen, dürfen dabei keine Stellvertreter haben, oder als Stellvertreter fungieren, da diese, wie bereits im Gruppensymbol vorhin erwähnt, andere Modellverhalten aufweisen sollen.

Verstärkt wird die Einheitlichkeit der Modellierung mit Hilfe von mehreren Prüfung vor der Konvertierung welche in Kapitel 4.1.2 näher erläutert werden.

Der Fokus der Analyse liegt in dieser Arbeit auf der flüssigen Kommunikation, weshalb feste Informationsspeicher (Dokumente) vorerst nicht unterstützt werden. Zum einen eignet sich die statische Notation der FLOW-Notation nicht dafür den zeitlichen Zugriff (schreiben & lesen) auf feste Informationsspeicher zu dokumentieren, zum an-

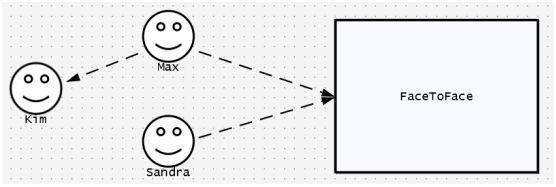

Abbildung 3.3: Stellvertreter durch direkte gerichtete Verbindung

deren soll verstärkt die alternativen Flussverhalten beobachtet werden um realistische Strukturänderungen deuten und anwenden zu können.

3.1.2 Abstraktion der FLOW-Elemente nach System Dynamics

Damit neben der tabellarischen Auflistung der Simulationsergebnisse das System Dynamics Modell in der optischen Modellansicht auch relevante Daten wiedergibt, wird bei der Abstraktion großer Wert darauf gelegt durch die grafische Visualisierung eine andere Perspektive zum besseren Verständnis der Ausgangslage zu bieten.

Personen und Gruppen als Stocks

Durch die Betrachtung der Informationsflüsse über Kommunikationsteilnehmer liegt es nahe, Personen und Gruppen direkt durch Stocks zu repräsentieren, da der Bestand den Betrachtungswert darstellen kann.

Die Ein- und Ausgangsflows könnten trivialerweise mit den eingehenden und ausgehenden Informationen übersetzt werden, jedoch macht das aus optischen und kommunikativen Gründen nicht viel Sinn. Zum einen wird durch die Annahme der symmetrischen bidirektionalen Kommunikation in Kapitel 3.1.1 immer die selbe Menge sowohl in und aus einem Knoten fließen, zum anderen würde sich der Bestand eines Stocks deswegen nicht verändern was dazu führen würde, dass die visuelle Darstellung des Stocks nur eine gerade Linie mit dem Wert null zeigt.

Weder beim ausfallenden Stock, noch bei benachbarten Stocks[1] würde sich der Ausfall bemerkbar machen, da über den Stock keine Informationen mehr fließen und somit der Bestand folglich konstant null bleibt. Aus diesem Grund werden die Personen zwar direkt zu Stocks abstrahiert, jedoch wird nicht der Informationsfluss, sondern der Informationsfluss pro Tag als Bestand etabliert. Dieser Bestand wird gesteuert durch einen Eingangs- und einen Ausgangsflow pro Stock und wird anfänglich auf den maximalen Informationsfluss des Knotens gesetzt. Bei etwaigem Ausfall eines Knotens, oder dessen benachbarten Knotens, werden die Ein- und Ausgangsflows dementsprechend mit Triggern versehen, die bei bestimmten Ereignissen den Bestand reduzieren bzw. erhöhen und in Kapitel 3.2.2 hergeleitet wurden.

[1] Mit benachbarten Stocks bzw. Knoten sind die übersetzen Personen des FLOW-Modells gemeint, die miteinander kommunizieren und nun als Stock dargestellt werden.

Gruppen werden ebenfalls mit dem selben Schema übersetzt, und erhalten einen eigenen Stock, werden jedoch später intern als Menge von Personen betrachtet.

Aktivitäten und ihre eingehenden Flüsse als Auxiliaries

Damit die Flows von Stocks auch mit Werten versorgt werden können, werden die Kommunikationsaktivitäten passend mit Auxiliaries übersetzt. Eine Auffächerung der Aktivität in Unter-Auxiliaries, um die einzelnen Kanäle besser zu repräsentieren, ist kontraproduktiv, da der Anwender diese in der Modellansicht aufgrund visueller Überladung nicht zu sehen bekommt. Stattdessen werden die Kommunikationswerte der einzelnen Kanäle aufaddiert als Wert des Auxiliaries gesetzt. Anschließend können die Auxiliaries in den Flows der, an der Kommunikation beteiligten Personen, nach Bedarf referenziert werden. Durch die Setzung der Trigger in den Flows, ist es in den Auxiliaries nicht notwendig diese mitzuerfassen, stattdessen werden sie lediglich die besagten Kommunikationswerte aufsummieren.

Informationsfluss über Stellvertreter

Die direkten Informationsflüsse zwischen Personen, welche im Konverter als Stellvertreter interpretiert werden, werden per se nicht sichtbar in System Dynamics Elemente übersetzt aber dennoch, zusammen mit anderen Werten, im XMILE-Modell gespeichert, damit der Simulator im Nachhinein darauf zurückgreifen kann. So können später alternative Flussverhalten ausgewählt werden um die Abwesenheit von Teammitgliedern abzufedern.

3.1.3 Der Konverter

Damit die Funktionen in der täglichen Arbeitsumgebung von Entwicklern und Projektmanagern zugänglich werden, werden sowohl der Konverter als auch der Simulator in eine Unterseite des Prodynamics-Plugin für JIRA implementiert. Da der Konverter nur die Aufgabe hat, die FLOW-Modelle zu laden, umzuwandeln und zum Download anzubieten wird ein großer Wert darauf gelegt, diese Oberfläche so einfach wie möglich zu gestalten.

Eine Anpassung der umgewandelten Modellstruktur durch die SD.js Bibliothek ist bisher nicht möglich, deshalb muss das Laden der Modelle, das Auswählen der Einstellungen und das anschließende Herunterladen schnell durchführbar sein, damit nach dem Editieren des FLOW-Modells eine Simulation mithilfe des neuen konvertierten Modells weitergeführt werden kann. Der Konverter wird das XML des FLOW-Modells in das System Dynamics XMILE Dateiformat umwandeln und um die Informationen bezüglich der Personen, Informationsflüsse, Aktivitäten und Stellvertreter erweitern, damit später dort die Informationen für die Simulation gespeichert werden können.

Abbildung 3.4 zeigt ein Mockup des Konverters im Einsatzgebiet von JIRA und verfolgt eine einfache Formularstruktur, welche von oben nach unten abgearbeitet werden soll.

Zuerst werden die Modellierungsvoraussetzungen aufgezählt, damit der User einen Überblick bekommt, welche Voraussetzungen die Modelle erfüllen müssen, damit diese zur Konvertierung zugelassen und anschließend korrekt übersetzt werden. Wenn ein zu konvertierendes Modell aus dem FlowExplorer vorliegt, kann es mit Hilfe des 'Upload'-Buttons hochgeladen werden. Eine Fehlerprüfroutine startet direkt nach Upload um zu verifizieren, ob es sich dabei um ein gültiges FlowExplorer-Modell handelt und die oben genannten Modellierungsvoraussetzungen nicht verletzt sind. Je nach Erfolg der Routine, wird in dem Status-Label ein passender Status angezeigt, um den Nutzer darüber in Kenntnis zu setzen, dass das Modell erfolgreich geladen werden konnte oder eine Fehlermeldung, welche eine passende Beschreibung zur schnellen Fehlerlokalisierung beinhaltet.

Ist das Modell erfolgreich hochgeladen und fehlerfrei erkannt, so werden Informationen zu dem Modell in der Infotabelle angezeigt, damit der User sichergehen kann, dass das korrekte Modell ausgewählt wurde. Es wird der Name der hochgeladen Datei, die Größe, das Änderungsdatum und die Anzahl der einzelnen Flow-Elemente aufgelistet.

Nachdem sichergestellt wurde, dass es sich um das korrekte Modell handelt, können für die Konvertierung verschiedene Optionen gesetzt werden, welche unter Umständen nachträglich nicht mehr im Simulator angepasst werden können. Das Mockup bietet dabei an, dass FLOW-Modell so zu modifizieren, dass die Koordinaten der gesetzten FLOW-Elemente skaliert werden, da die Sytem Dynamics Elemente, bedingt durch die SD.js, ansonsten zu weit auseinander liegen können und dementsprechend die Beschriftungen nicht mehr lesbar sind. Des Weiteren wird noch die Option angeboten, unbekannte Flüsse mit einem Standardwert zu initialisieren, welcher aber nachträglich im Simulator noch angepasst werden kann.

Wenn der Nutzer so weit ist, dass alle Optionen zufriedenstellend gesetzt sind, so bieten sich nun am Ende des Formulars zwei Optionen an um mit dem umgewandelten Modell weiterzuarbeiten. Das Modell kann hierbei heruntergeladen werden oder direkt in den Simulator geladen und nach einer Simulation dort später ebenfalls heruntergeladen werden.

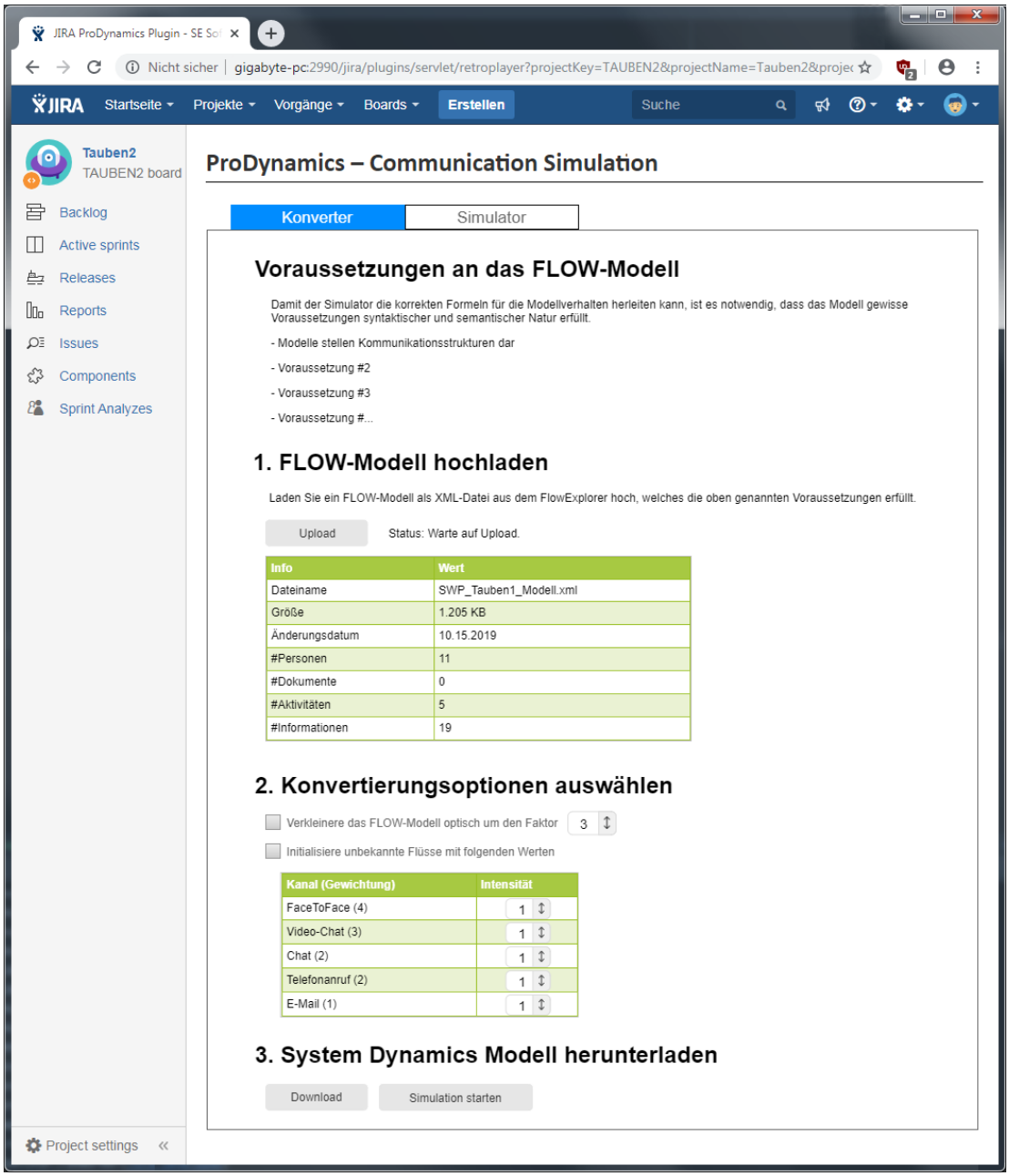

Abbildung 3.4: Mockup des Konverterdesigns

3.2 Simulation der dynamischen Modelle

3.2.1 Annahmen und Voraussetzungen

Wie bei der Konvertierung müssen auch bei der Simulation gewisse Annahmen getroffen werden um Ungewissheiten zu erkennen und zu eliminieren. Die in diesem Kapitel genannten Themen bauen auf den bereits etablierten Voraussetzungen des Konverters auf und führen diese im Kontext der Simulation weiter. Zusammen bestimmen sie den Kontext der Simulation in Bezug auf die Kommunikation zwischen Personen, alternative Flussverhalten und im Allgemeinen die Granularität der Simulationstiefe. Für eine valide Simulation sollte daher die Tiefe nicht zu seicht sein, da ansonsten keine genaueren Schlüsse über alternative Modellstrukturen geschlossen werden können, und andererseits auch nicht zu tief, da hier immer noch der Faktor Mensch involviert ist und die Eingabedaten der Umfragen bezüglich der Intensitäten schwammig sein können oder von den gegebenen alternativen Flussverhalten abgewichen werden können.

Betrachtung des Tagesdurchschnitts

Die von den Teammitgliedern ausgefüllten Umfragebögen können je nach Belieben neu erhoben werden, jedoch ergibt sich durch die Anbindung an JIRA ein guter Richtwert indem die Umfragen wöchentlich oder nach jedem Sprint erhoben werden können wie es bei den Testdaten der Fall ist.

Zudem wird angenommen, dass bei krankheitsbedingtem Ausfall die Betroffenen für alle Krankheitstage vollständig ausfallen. Diese Annahme widerspricht sich zwar mit realen Teams, jedoch muss hier die Verhältnismäßigkeit der Simulationsgranularität gewahrt werden. Eine ausfallende Person könnte immer noch einen kleinen Beitrag leisten oder die Arbeit effizient umverteilen, jedoch ist dieses Verhalten von Person zu Person, und Team zu Team unterschiedlich, weshalb ein kompletter Ausfall angenommen wird. Daraus folgert sich, dass die alternativen Flussverhalten und sonstigen Optionen auf täglicher Basis greifen sollten. Fällt beispielsweise jemand aus, so könnte, je nach Einstellung, die Information erst einen Tag liegen bleiben, weil sie eventuell nicht wichtig ist, und erst am nächsten Tag von einem Stellvertreter weiterverarbeitet werden.

Ein weiterer Vorteil ergibt sich für den Nutzer, da dieser nun genug Spielraum hat, um verschiedene Szenarien durchspielen zu können ohne dabei zu tief einzusteigen. Durch den genannten Faktor Mensch ist es unrealistisch eine sehr genaue Simulation durchzuführen, welcher mit der Realität übereinstimmt, weshalb versucht wird, verschiedene Verhalten bei einer angemessenen Tiefe zu bieten, um die gleichen Tendenzen wie in der Realität zu ermitteln.

Distinkte Information und redundante Informationsflüsse

Eine weitere wichtige Voraussetzung für eine erfolgreiche Simulation ist die der distinkten Information und wird anhand des folgenden Beispiels mit Hilfe von Abbildung 3.5 deutlich[2]. Gegeben sind drei untereinander kommunizierende Personen Bob, Laura und Paul. Bob leitet nun die Information i_x an Laura und Paul separat weiter und es finden Informationsflüsse statt. Nun möchte Laura Paul von der selben Information unterrichten und kommuniziert diese dementsprechend weiter, jedoch hat dieser bereits die selbe Information von Bob bereits empfangen und durch den erneuten Empfang der Information ergibt sich kein Mehrwert, viel mehr hingegen ein Nachteil. Paul und Laura haben ein Teil ihrer Kommunikationskapazität aufgewendet, um die redundante Information zu kommunizieren.

Um diesem Problem entgegen zu wirken, wird für jede Kommunikation inhaltlich verschiedene Information vorausgesetzt, damit bei den Simulationsergebnissen die Informationsflussmengen nicht künstlich erhöht werden. Zudem bietet die FLOW-Notation nur die Möglichkeit Informationsflüsse zwischen fest und flüssig zu klassifizieren, jedoch nicht über den tatsächlichen Inhalt oder Kategorie.

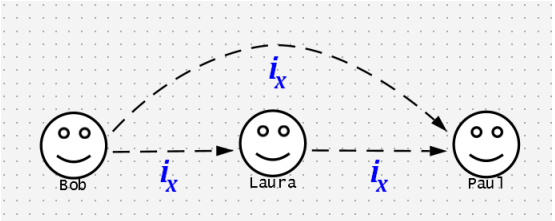

Abbildung 3.5: Redundante Informationsflüsse und nicht distinkte Informationen

Normalisierte Informationsflussmengen

Durch die Verwendung der Kommunikationsindizes [10] als Informationsflussmenge müssen die Informationsflüsse vor der Simulation normalisiert werden. Dies ist dadurch bedingt, dass die Umfragen für die Kommunikationsintensitäten sich auf die gesamte Kommunikation beziehen und die Werte nicht ohne weiteres für die Auxiliaries übernommen werden können. Wenn zwei Personen FaceToFace miteinander kommunizieren und in einer anderen Aktivität mit anderen Mitgliedern ebenfalls über den selben Kanal kommunizieren, so würde der der Kommunikationsindex für beide Kommunikationen verwendet werden. Das ist jedoch nicht korrekt, da so die doppelte Menge an Informationen fließt, obwohl die Umfrageergebnisse die Gesamtheit abbildet. Deswegen werden die Kommunikationsindizes in Bezug auf die Anzahl der Vorkommnisse normalisiert, indem sie durch die jeweiligen Anzahlen der Vorkommnisse geteilt werden.

[2] Die direkten Verbindungen zwischen Personen sollen als Informationsflüsse im klassischen Sinne von FLOW verstanden werden.

3.2.2 Fluss- und Modellverhalten bei personellem Ausfall

Damit bei personellem Ausfall, Deadlines in Projekten immer noch eingehalten werden können, müssen ggf. Teammitglieder die Arbeit von anderen übernehmen, damit das Projekt nicht in Verzug gerät. Aus diesem Grund ist es essentiell, die wegfallenden Ressourcen zu kennen und durch Simulationen potenzielle Umleitungen zu ermitteln, um Informationsflüsse aufrecht zu halten. Deshalb werden für die Simulation verschiedene Fluss- und Modellverhalten benötigt, um diese simulieren zu können. Dabei sollen die Verhalten miteinander, sofern möglich, kombinierbar sein, um mehr Optionen bieten zu können.

Informationen bleiben eine gewisse Anzahl von Tagen liegen

Nicht jede Information ist nicht von so einer hohen Wichtigkeit, dass sie zu jeder Zeit fließen muss, denn oft ist es sogar so, dass Informationen einen Tag liegen bleiben können, bevor sie weiterfließen. Dies kommt unter anderem durch einen kritischen Pfad in einem Projekt zustande, der taktgebend für das gesamte Projekt ist. Verzögert sich dort etwas, oder Informationen bleiben liegen, so verzögert sich das ganze Projekt, daraus resultiert, dass viele Prozesse erst fertiggestellt werden können, solange sie das Projekt nicht verzögern. Diese Verzögerung ist variabel und hängt stark vom Projekt ab, weshalb der Simulator eine Option implementiert, Informationen für eine Anzahl an Tagen zu verzögern um dieses Gegebenheit darzustellen.

Keine Umverteilung bei Ausfall

Bleibt noch genug Zeit bis zur Deadline oder es existiert ein Puffer um Ausfälle hinzunehmen, so könnten personelle Ausfälle komplett ignoriert werden, solange sie den Verzug des Projektes nicht gefährden. Stattdessen stauen sich die Informationen eines Knotens bis sie bei Weiterführung der Arbeit nacheinander abgearbeitet werden.

Stellvertreter übernimmt, Flussanteil wird verteilt

Jedoch stellt das Aussitzen eines ausfallenden Teammitglieds nur eine Seltenheit dar, weshalb ein Kollege, mit ähnlicher Erfahrung[3], für die Periode einspringt die Arbeit übernimmt. Dies ist auch praktikabel, da eine Person für einen kurzen Zeitraum eine Mehrbelastung stemmen kann, ohne ihre eigentliche Arbeit zu vernachlässigen. Aus diesem Grund wird dies, neben dem vorhin genannten alternativen Flussverhalten, als Option implementiert, wobei der Nutzer jedoch später die Simulationsergebnisse genau studieren muss um festzustellen, ob die dadurch entstandene Mehrbelastung tatsächlich tragbar ist oder nicht.

[3] Mit Erfahrung ist hier das Themengebiet gemeint. Ein Programmierer kann aufgrund fehlender Grafikkenntnisse nicht für einen Grafiker einspringen.

Stellvertreter übernimmt, Flussanteil wird gleichmäßig verteilt

Da ein Stellvertreter nur für eine kurze Zeit die Arbeit übernehmen soll und langfristig keine Alternative zur abwesenden Person darstellt, wird noch ein weiteres Flussverhalten angeboten. Die Idee dahinter ist, dass ein Stellvertreter zwar Arbeit übernehmen kann, jedoch nicht die selbe Kapazität besitzt wie die Person die sie vertritt. Deshalb wird der Informationsfluss über den Stellvertreter auf alle an der Kommunikation beteiligten Mitglieder inklusive dem Stellvertreter verteilt. So werden weniger Informationen über den Stellvertreter fließen, da ein Teil die Kollegen übernommen haben. Daraus ergibt sich für Kollegen die vorher weniger an der Kommunikation beteiligt waren, dass sie nun mehr gefordert werden und die die mehr beteiligt waren nun weniger, da der Informationsfluss der abwesenden Person gleichmäßig verteilt wurde.

Für beide Verhalten in denen Stellvertreter zum Einsatz kommen wird immer derjenige ausgewählt, dessen Informationsfluss, zum Zeitpunkt des zu vertretenden Ausfalls, am niedrigsten ist. Dadurch wird eine gleichmäßige Belastung der Mitglieder angestrebt.

3.2.3 Der Simulator

Die Anforderungen an den Simulator unterscheiden sich grundlegend zu denen des Konverters. So wird beim Simulator der Hauptfokus darin gelegt, dass angepasste Modellverhalten direkt sichtbar sein müssen, die Simulation schnell durchläuft und aussagekräftige Ergebnisse liefert. Neben einer detaillierten Ergebnisausgabe in Tabellenform, gibt es noch die visuelle Ansicht des Modells die nicht als Mittel zum Zweck gesehen werden soll. Deshalb ist es wichtig, dass die dargestellten Werte sinnvoll ausgewählt werden wie sie in Kapitel 3.1.2 beschrieben sind.

Vereint wird das ganze durch das Mockup in Abbildung 3.6. Durch die Teilung der Oberfläche des Simulators in zwei vertikale Bereiche kann im linken das visuelle Modell angezeigt werden und in dem anderen die Anpassungen zum Modell. Dabei ist der Steuerungsbereich in der rechten Ansicht in ein Tab-Control unterteilt, damit nicht benötigte Einstellungen elegant versteckt werden können. Im besagten Bereich können die Anwesenheitszeiten der Personen, die verschiedenen Modellverhalten an- und ausgeschaltet, die Kommunikationsintensitäten zwischen den einzelnen Personen angepasst und die Simulationsergebnisse angesehen werden. Zudem gibt es noch drei weitere Buttons um ein Modell in den Simulator zu laden, das angepasste Modell mit den Kommunikationsintensitäten herunterzuladen und die Simulation mit den eingestellten Parametern zu starten.

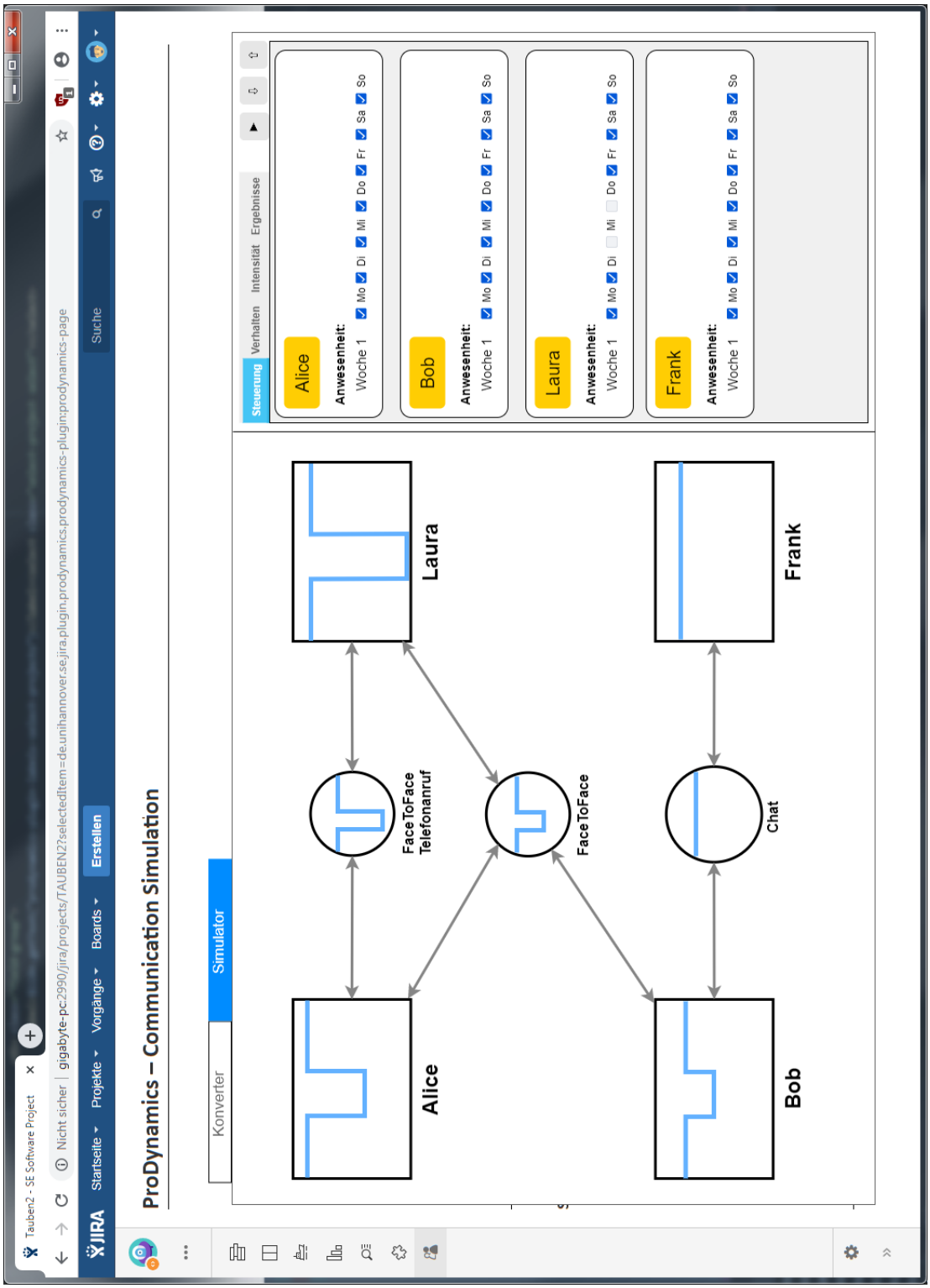

Abbildung 3.6: Mockup des Simulatordesigns

3.2.4 Ergebnisse der Simulation

Da die visuelle Ansicht des Modells nur eine gewisse Menge an Informationen zeigen kann, ist eine detaillierte Ergebnisausgabe notwendig. Die ausgegebenen Werte sollten aufschlussreich sein und dem Ziel der System Dynamics Methodologie, nämlich der ”Policy Analysis”, zugutekommen [7]. Mit Hilfe der Policy Analysis soll das aktuelle Verhalten des Modells begutachtet und angepasst werden um Änderungen am Modell vorzunehmen, die es im Rahmen der Möglichkeiten, robuster und sicherer gegenüber Ausfällen macht.

Mit den gegebenen Eingabewerten und der Interaktion dieser innerhalb des Modells werden folgende Ergebnisse das genannte Ziel ermöglichen:

Gesamtmenge an Informationen - Dient als Referenz wie viele Informationen im Normalzustand fließen würden, wenn keine Störung vorliegt.

Tatsächliche Menge an fließenden Informationen - Sollte eine Störung vorliegen und dadurch Informationen verloren gehen, oder diese durch Stellvertreter neuverteilt werden, so kann hier die tatsächliche Menge an fließenden Informationen betrachtet werden.

Umverteilte Informationen - Hiermit kann beobachtet werden, wie viel der Informationen nun auf alternativen Flüssen zwischen Personen fließen.

Verlorene Informationen - Fallen Personen aus und es springen keine Stellvertreter ein, um die Kommunikation noch zu erhalten, so gehen diese Information verloren.

Personbezogene Werte - Um zu erfahren, wie viel eine Person kommuniziert hat, werden die vorangegangenen Ergebnisse ebenfalls einzeln für jede Person aufgelistet. Hinzukommen die ersten beiden Werte auf Tagesbasis sowie der minimale und maximale Tagesfluss über den gesamten Simulationszeitraum.

Durchschnitt aller Personen - Um eine Person mit dem Durchschnitt im Modell vergleichen zu können, werden die ersten vier genannten personenbezogene Ergebnisse gemittelt und ausgegeben, womit das durchschnittliche Niveau des Modells berücksichtigt werden kann.

4 Implementation

Die im vorangegangenen Kapitel vorgestellten Konzepte werden nun einzeln implementiert, um sie anschließend im JIRA-Plugin zusammenzuführen. Dabei wird erläutert warum Komponenten auf die gewählte Weise implementiert worden sind. Zuerst werden die vorgestellten Mockups umgesetzt und anschließend die Prozesse, die vor dem Konvertieren bzw. Simulieren stattfinden. Nach den Kernfunktionalitäten folgen die jeweiligen Ausgaben und abgeschlossen wird das Kapitel mit der Implementation in das Prodynamics-Plugin.

4.1 FLOW nach XMILE

4.1.1 Aufbau

Durch die in Kapitel 3.1.3 angesprochene einfache Oberfläche ist das Design des Konverters sehr schlicht gehalten und können vom Mockup ohne Probleme übernommen werden, indem sie mit den entsprechenden HTML-Elementen abgebildet werden. Da auf der Seite des Konverters keine neuen Controls erzeugt werden, werden lediglich die Ein- und Ausgabecontrols mit IDs und Funktionen verlinkt.

Der Upload-Button öffnet ein FileDialog Fenster in dem ein FLOW-Modell ausgewählt werden kann. Dieses Modell wird in den Speicher des Webbrowsers geladen, denn der Konvertierungsvorgang findet nicht auf dem Webserver, sondern im Browser statt. Grund dafür ist, dass mit kleinen Dateien gearbeitet wird, welche wenig Rechenaufwand erfordern und dies auch ohne Probleme und zügig auf dem Benutzercomputer läuft. Zudem erfordert die einzige externe Bibliothek SD.js ebenfalls wenig Rechenleistung, weshalb alles clientseitig implementiert wird, um unnötigen Traffic mit dem Server zu vermeiden. Folglich wird ebenfalls ein Interface eingespart, um den Clienten mit dem Server zu vernetzen und der Code muss nur an einer Stelle gewartet werden.

Durch die Verwendung von Javascript im Webbereich gibt es Funktionen um Document Object Modelle[1] (kurz DOM) zu bearbeiten. Geparster HTML- und XML-Code werden in DOM-Objekte intern übersetzt um schnell und einfach im Programmcode zwischen den einzelnen Knoten zu springen und zu arbeiten. Da die FLOW-Modelle und die XMILE-Modelle im XML-Format vorliegen, werden die geladenen FLOW-Modelle einfach mit der DOMParser-Klasse geparst und es werden einige Informationen über das Modell präsentiert, damit der Nutzer sicherstellen kann, dass das gewünschte Modell geladen wurde. Nach dem vermeintlichen Upload startet eine Fehlerprüfroutine.

[1] https://de.wikipedia.org/wiki/Document_Object_Model

4.1.2 Fehlerprüfung

Damit auch sichergestellt werden kann, dass sowohl syntaktisch als auch semantisch korrekte Modelle in den Konverter geladen werden, wird nach dem Upload direkt eine Fehlerprüfroutine ausgeführt.

Diese dient dazu, den Benutzer zu informieren, ob das geladene Modell mit den Voraussetzungen konform ist und gibt dementsprechend eine Statusmeldung aus. Dabei kann die Statusmeldung die Validität des Modells bestätigen oder eine Fehlermeldung ausgeben die dem Benutzer dabei hilft den Fehler zu beheben.

Die XML-Struktur eines vom FlowExplorer produzierten Modells ist dabei wie folgt aufgebaut, wobei nicht relevante Attribute ausgelassen worden sind:

```
<Process>
  <Person id text isGroup />
  <Activity id text />
  <Information id sourceID targetID />
  <Document id text isGroup/>
</Process>
```

Abbildung 4.1: XML-Struktur eines Modells aus dem FlowExplorer

Jedes FLOW-Element hat dabei einen entsprechenden Tag, ein eindeutiges id-Attribut und bis auf den Information-Tag haben alle Elemente ebenfalls ein text-Attribut wobei letzteres den sichtbaren Text bei Personen, Dokumenten und Aktivitäten darstellt. Information-Tags haben dafür die sourceID- und targetID-Attribute um zu kennzeichnen, von welchem Elemente aus Informationen zu einem anderen Element fließen. Personen und Dokumente haben noch das isGroup-Attribut welches angibt, ob es sich um eine einzelne Person oder eine Gruppe von Personen bzw. um ein einzelnes Dokument oder eine Dokumentensammlung handelt. Hauptsächlich werden nur die fünf Attribute verwendet um die Validität des Modells sicherzustellen.

Dabei wurden die bereits vorgestellten Restriktionen des FLOW-Modells einzeln in Javascript-Code übersetzt um das Modell danach zu durchsuchen. Die einzelne Übersetzung verfolgt dabei das Ziel, die Routine sehr einfach warten zu können und bei Weiterentwicklung der Software einfach modifizieren zu können. Zwar könnten viele der Schleifen, zum Iterieren der Elemente zusammengelegt, und somit die Überprüfungsgeschwindigkeit relativ gesehen dramatisch erhöht werden, jedoch ist dies, absolut gesehen, nur eine kleine Verbesserung im Durchlauf der Routine die auf Kosten der Lesbarkeit des Codes geht. Zur besseren Veranschaulichung wie die Restriktionen im Code voneinander getrennt werden wird dies anhand des Codeausschnitts in Abbildung 4.2 erörtert.

```
function preCheckModelErrors()
{
  var persons = xmlFlowModel.querySelectorAll("process > Person");
  var activities = xmlFlowModel.querySelectorAll("process > Activity");
  var documents = xmlFlowModel.querySelectorAll("process > Document");
  var informations = xmlFlowModel.querySelectorAll("process > Information");

  [...]

  // Gruppensymbol als Person verwendet
  for (var i = 0; i < persons.length; i++)
    if (!persons[i].getAttribute("text").includes(",") && persons[i].
    getAttribute("isGroup") == "true")
      return ShowStatus("Es existiert eine Person als Gruppensymbol.", true)
    ;

  [...]

  // Mehrere Personen mit dem selben Namen
  for (var i = 0; i < persons.length - 1; i++)
  {
    for (var j = i + 1; j < persons.length; j++)
    {
      if (persons[i].getAttribute("text").toLowerCase() == persons[j].
    getAttribute("text").toLowerCase())
        return ShowStatus("Es sind mehrere Personen mit dem selben Namen
    vorhanden.", true);
    }
  }

  [...]
}
```

Abbildung 4.2: Codeausschnitt der preCheckModelErrors-Funktion

Zuerst werden alle FLOW-Elemente aus dem DOM-Objekt einzeln herausgeparst und danach folgen die Codeblöcke zur Überprüfung des jeweiligen Kriteriums, wobei zwei zur Veranschaulichung ausgewählt wurden.

Zur Überprüfung, ob ein Gruppensymbol als einzelne Person verwendet wird, werden alle Elemente der NodeList persons iteriert und geprüft, ob das Attribut isGroup true gesetzt ist und ob mehrere Namen durch ein Komma im text-Attribut separiert werden. Schlägt die Überprüfung fehl, so wird eine Fehlermeldung im Status-Label ausgegeben und aus der Funktion herausgesprungen.

Da Personen im Modell nur einmal vorkommen dürfen, wird im nächsten Codeblock geprüft, ob eine Person mit dem selben Namen doppelt auftaucht. Dabei findet die Überprüfung mittels des text-Attribut statt, da für mehrere Personen mit dem selben Namen das id-Attribut nicht übereinstimmt.

Wäre die Funktion performancetechnisch relevant und müsse in einer bestimmten Zeit ablaufen, so ließen sich beispielsweise die beiden äußeren Schleifen zur Iteration der persons NodeList zusammenlegen. Jedoch läuft die Funktion, mit Einbezug der ande-

ren Prüfungen, auch auf leistungsärmeren Computern so zügig, dass hier der Fokus auf Lesbarkeit und Wartbarkeit des Codes gelegt wurde. Durch diesen Fokus ist es möglich zu erkennen, welche Überprüfung in einem Codeblock stattfindet, um diese, bei grundlegenden Veränderungen im Konverter oder Simulator, zu modifizieren oder neue hinzufügen.

Nachdem die Validität des Modells geprüft und bestätigt wurde, geht es auch nach der Auswahl der Konvertierungseinstellungen direkt in den Konvertierungsvorgang.

4.1.3 Konvertierungsvorgang

Begonnen wird der Konvertierungsvorgang indem das DOM-Objekt für die Ziel XMILE-Datei erzeugt wird. Anschließend werden die einzelnen Untersektionen des XMILE-Modells erstellt, welche sich in header, sim_specs, model_units und model gliedern. Zudem wird noch eine eigene Untersektion flow_section angehangen welche nicht in der XMILE-Dokumentation vorgesehen ist.

header

Im Header-Tag sind Metainformationen zum Modell enthalten wie Namespaces, Informationen über die Software, welches das XMILE-Modell generiert hat und noch weitere. Dieser Bereich ist uninteressant, da die konvertierten Modelle wie bereits erwähnt nur im Simulator des Plugins Anwendung finden und auch dort nur vernünftig funktionieren. Deshalb werden die Informationen im Header-Tag mit immer den selben Informationen gefüllt welche aus dem Quellcode entnehmbar sind.

sim_specs

Hier werden die Einstellungen zum Simulationszeitraum aufbewahrt. Es existieren drei Elemente, die den Startzeitpunkt, den Endzeitpunkt und den Zeitschritt zwischen jeder Iteration definieren. Wie in Kapitel 2.2.1 anfänglich erwähnt, werden die Integrale, welche die Bestände eines Stocks ausdrücken, numerisch mit Näherungswerten gelöst. Dabei ist der Integrationsschritt identisch mit dem Zeitschritt der hier angegeben wird.

Da im Simulator Trigger verwendet werden, um bei bestimmten Zeitpunkten Flüsse umzuverteilen, ist die Wahl eines passenden Integrationsschrittes äußerst kritisch. Durch die Interpretation von Kommazahlen im float- bzw. double-Format können einige Zahlen nicht dargestellt werden. Wird beispielsweise ein Zeitschritt von 0.1 genommen, so schlägt später die Abfrage fehl, ob der aktuelle Zeitpunkt 1.2 ist, da die internen Rundungsfehler den Trigger nicht auslösen.

Um diesem Fehlverhalten zuvorzukommen, werden sowohl im Start- und Endzeitpunkt sowie dem Zeitschritt nur Integer verwendet. Da die SD.js Bibliothek jedoch die Bestände in der visuellen Ansicht interpoliert um alle diskreten Zeitpunkte miteinander

verbinden zu können, werden die Zeitpunkt alle mit dem Faktor zehn multipliziert. Daraus folgt, dass die optische Modellansicht starke Sprünge von einen auf den nächsten Tag haben, so wie es die verschiedenen Modellverhalten vorsehen. Bei der Konvertierung werden hier die Standardwerte von einer Woche eingesetzt, welche später im Simulator angepasst werden können.

model_units

Da im Modell nur Informationen fließen und nicht mehrere Einheiten an Beständen, wird dieser Bereich der Vollständigkeit halber mit Standardwerten initialisiert und können dem Quellcode entnommen werden.

flow_section

Damit im Simulator nicht bei jedem Laden des selben Modells die Kommunikationsintensitäten neu gesetzt werden müssen, wird eine Untersektion namens flow_section angehangen. Diese gliedert sich in unknowns, members, communication_values, und communications.

Alle Informationen des zu konvertierenden FLOW-Modells werden in diese Unterbereiche gespeichert. Zuerst werden die Konvertierungseinstellungen ausgelesen, die der User nach dem Upload getätigt hat. Ist die Initialisierung der unbekannten Kanäle mit den gegebenen Werten gewünscht, so werden im unknowns-Bereich die Kanäle mit den entsprechenden Werten initialisiert statt mit null. Soll das Modell optisch verkleinert werden (die Elemente näher zusammenrücken), so werden die Koordinaten, die beim Parsen der restlichen Bereiche erzeugt werden, durch den Faktor geteilt.

Im members-Bereich werden die Personen aus dem FLOW-Modell übernommen und zusammen mit ihrem Namen, Koordinaten und Stellvertretern eingespeichert. Folgend im communications_values-Bereich werden für jede Person die Kommunikationsintensitäten mit den restlichen Personen gespeichert. Beendet wird die flow_section-Untersektion mit dem communications-Bereich. Die Aktivitäten werden geparst, indem die Kanäle und Teilnehmer ermittelt werden und entsprechend eingefügt.

Gefüllt wird die flow_section indem das FLOW-Modell schrittweise geparst und parallel die entsprechenden Einträge erzeugt werden. Das Parsen erfolgt zum größten Teil mit QuerySelector-Funktionen, welche CSS-Selector-Strings akzeptieren, um entsprechende Einträge zurückzugeben. Eine detaillierte Beschreibung, wie im einzelnen Elemente nun ausgelesen und hinzugefügt und in welcher Reihenfolge aufgebaut werden, ist im Quellcode enthalten.

model

Die model-Sektion gliedert sich noch jeweils in die variables- und views-Untersektion.

variables - Dank der Vorgehensweise, die im Konzept vorgestellt wurde, kann dieses problemlos implementiert werden. Pro Person bzw. Gruppe wird ein Stock erstellt und mit dem Bestand null initialisiert und mit einem Eingangs- und Ausgangsfluss speziell für den Stock verknüpft, welche ebenfalls erzeugt werden. Anschließend wird für jede Kommunikationsaktivität ein Auxiliary erstellt und nach den teilnehmenden Personen und Kommunikationskanälen, bedingt durch die SD.js[2], benannt.

Für die Simulation wird dieser Bereich jedoch vollständig neu erzeugt, da intern eine andere Repräsentation verwendet wird, um leichter die Umleitungsflüsse zu berechnen und beim Erzeugen dieser es einfacher ist, parallel dazu Stocks, Flows und Auxiliaries zu erstellen als zu ermitteln, welche bereits existieren und welche noch benötigt werden. Dennoch wird dieser Bereich erzeugt, um nach dem Laden des Modells die Ansicht nicht leer zu lassen, weil dies den Eindruck erwecken könnte, dass das Modell nicht geladen wurde.

views - Im Gegensatz zum vorgegangenen Bereich wird diese Sektion nicht bei jeder Simulation neu erzeugt. Diese Sektion ist dazu da, um verschiedene visuelle Ansichten zum Modell zu präsentieren und implementiert die im Konzept vorgestellte Modellansicht. Dabei werden die Elemente aus der variables-Sektion referenziert und mit Koordinaten versehen, um sie zueinander zu platzieren. Zwar werden die Verknüpfungen zu den Elementen aus der variables-Sektion kurzzeitig ungültig, wenn der Bereich für die Simulation neu erzeugt wird, jedoch ist dies kein Problem, da die neuen Elemente die selben Namen haben und das Modell erst geladen wird, wenn alles erzeugt wurde. Die Art und Weise wie die Elemente durch den Programmcode an das DOM gehangen werden, ist sehr ähnlich zur vorangegangen Sektion.

4.1.4 Ausgabe des XMILE-Modells

Nach Abschluss der Konvertierung wird dem Nutzer die Möglichkeit geboten, das konvertierte Modell herunterzuladen oder direkt in die Simulation zu geben. Dabei wird das DOM-Objekt, welches während bei der Konvertierung aufgebaut wurde, durch die XMLSerializer-Klasse in einen String serialisiert und zum Download angeboten oder die Ansicht zum Simulator gewechselt und das Modell als Eingabe genommen.

4.2 Simulator

4.2.1 Aufbau

Wie beim Konverter kann das Mockup übernommen werden und wird grafisch genauso implementiert jedoch ist sie im Gegensatz dazu dynamisch, da neue Controls erzeugt werden und die Modellansicht aktualisiert wird. Die Modellansicht links wird automatisch von der SD.js generiert sobald eine Simulation startet und rechts befinden sich

[2] Die SD.js unterstützt keine Aliase für Modellelemente weshalb alle Elemente einen unterschiedlichen Namen haben müssen. Mehr dazu in Kapitel 5.1.

die Controls zur Steuerung der Simulation. Dabei gliedert sich die Steuerungsseite in die Tabs 'Steuerung', 'Verhalten', 'Intensität' und 'Ergebnisse'. Auf gleicher Höhe werden rechts Buttons angedockt um die Simulation zu starten, das angepasste Modell herunterzuladen und ein neues Modell zu laden. Jedes der Elemente ist im Javascript-Code mit Events verknüpft um bei Knopfdruck die jeweilige Seite anzuzeigen oder die gewünschte Funktion auszuführen.

4.2.2 Modell laden und Simulation vorbereiten

Damit eine Simulation gestartet werden kann müssen vorerst ein Modell geladen und entsprechende Einstellungen getätigt werden. Geladen wird das Modell über den Button mit dem nach oben gerichteten Pfeil, welcher ein FileDialog-Fenster zur Auswahl des zu ladenden Modells öffnet. Ist ein gültiges Modell geladen, welches aus dem Konverter kam, so wird die Modellansicht links automatisch durch die SD.js mit den neuen Stocks, Auxiliaries und Connectoren aktualisiert.

Des Weiteren werden die im Modell enthaltenen Personen in den Tab 'Steuerung' hinzugefügt. Dies geschieht, in dem, die vom Konverter hinzugefügte, flow_section im XMILE-Modell ausgelesen wird, um gespeicherten Simulationszeitraum zu ermitteln und die Checkboxen der Personen zur Anwesenheitssteuerung zu erzeugen.

Nachdem die Personen hinzugefügt worden sind, wird im Tab 'Intensität' die Eingabefelder zur Steuerung der Kommunikationsintensitäten erzeugt. Die Intensitäten werden aus der communication_values-Sektion des XMILE-Modells gelesen und entsprechend in die Felder eingetragen.

Beim ersten Laden des Modells sind diese Felder alle null, da der Konverter sie mit diesem Wert initialisiert. Damit jedoch nicht beim Neuladen des Modells nicht die Werte erneut eingetragen werden müssen, kann mit Hilfe des Buttons mit dem nach unten gerichteten Pfeil heruntergeladen werden. Dieses heruntergeladene Modell beinhaltet nun die eingetragenen Intensitäten.

4.2.3 Simulationsvorgang

Wenn das Modell geladen ist und die Kommunikationsintensitäten eingetragen sind, sind die Vorbereitungen für das Modell fast abgeschlossen. Bevor die Simulation gestartet wird, muss noch der Simulationszeitraum und die Verhaltensweise bei Ausfall von Kommunikationspartnern gesetzt werden. Da die Verhaltensweisen, wie im Konzept erwähnt, auf täglicher Basis wirken, ist der Simulationszeitraum in Wochen bemessen und kann mit Hilfe des Eingabefeldes gesteuert werden.

Die Umverteilung wird mittels Betätigung der Checkbox "Verteile Informationsfluss bei Ausfall von Kommunikationspartnern" aktiviert und schaltet weitere Optionen frei, welche nun zur visuellen Wahrnehmung nicht mehr grau gefärbt sind.

Zum einen ist es nun möglich anzugeben, wie lange Informationen liegen bleiben, bis der Fluss umgelenkt wird, zum anderen ist es möglich anzugeben, welches Flussver-

halten verwendet werden soll. Dabei werden die im Konzept vorgestellten Flussverhalten mittels Checkboxen im Radiostil angeboten, da aufgrund ihres Flussverhaltens nicht kombinierbar sind.

Nach Auswahl der passenden Simulationsverhalten wird durch die Betätigung des Simulieren-Buttons, dargestellt durch das Play-Symbol, gestartet.

Beim Start der Simulation wird zuerst das geladene XMILE-Modell geklont, damit bei einer neuen Simulation nicht erst die getätigten Veränderungen im Modell zurückgesetzt werden. Anschließend werden die Kommunikationsintensitäten normalisiert, damit später im Modell die korrekte Menge an Informationen fließen. Dies geschieht, indem ermittelt wird, wie oft eine Person mit einer anderen Person über den selben Kanal kommuniziert und der Intensitätswert durch die Anzahl der Vorkommnisse geteilt wird.

Darauffolgend werden die Stocks, Flows und Auxiliaries alle im XMILE-Modell gelöscht um sie während der Simulation neu zu erzeugen. Grund dafür ist, dass es einfacher ist, beim Simulieren diese Elemente parallel zu erzeugen, statt abzufragen ob diese bereits existieren und sie damit zu verknüpfen.

Des Weiteren wird ein interne ein Objekt namens 'persons' angelegt um dort Informationen zu den einzelnen Personen als Javascript-Klasse zu speichern. Dies hat den Hintergrund, dass es im Programmcode effektiver ist, sich durch Klassen zu navigieren, als mit DOM-Objekten zu arbeiten und dort Informationen zu speichern.

In das persons-Objekt werden nun die einzelnen Personen und Gruppen aus der flow_section-Sektion ausgelesen und eingespeichert, wobei bei jeder neuen Person automatisch die Elemente für das Stock und den Ein- und Ausgangsflüssen erzeugt, und an das XMILE-Modell angehangen wird, wobei die Werte die darüber fließen erst später gesetzt werden. Einzig die Werte der Stocks der Personen werden mit dem maximalen ungestörten Informationsfluss initialisiert. Ergänzt werden die Personen im persons-Objekt durch die verfügbaren Stellvertreter und angegebenen Arbeitszeiten. Die Gruppen werden dabei außen vor gelassen, da bei der Zuweisung der Werte für die Flüsse abgefragt wird, ob eine Person in einer Gruppe ist oder nicht. Nichtsdestotrotz werden die Gruppen in das persons-Objekt aufgenommen um die Stocks, Flows und Auxiliaries zu erzeugen.

Bevor die Umleitungen gesetzt werden können, wird das Modell erst ohne Umleitung, nur mit personellen Ausfällen simuliert. Dadurch werden die Informationsflussmengen der Modellpersonen berechnet um später den idealen Stellvertreter auswählen zu können.

Für die Simulation ohne Umleitungen müssen zunächst die Werte für die erzeugten Stocks, Flows und Auxiliaries gesetzt werden indem über jede Kommunikation iteriert wird. Anschließend wird für jedes Paar aus der Kommunikation der jeweiligen Informationsfluss ermittelt indem die normalisierten Werte aus der communication_values-Sektion ausgelesen werden. Sofern bei der Kommunikation in beide Richtungen Intensitätswerte vorliegen (beide Personen haben die Umfragebögen ausgefüllt), wird der Wert gemittelt. Sollte in eine Richtung keine Werte vorliegen (nur eine Person hat den Umfragebogen ausgefüllt), so wird nur der aus der anderen Richtung genommen.

Nach der Ermittlung der Informationsflusswerte werden die Tage ermittelt, an denen mindestens einer des Paares aus der Kommunikation abwesend ist, da beim Paar nur Informationen fließen können, sofern beide anwesend sind.

Durch die Implementation der Stocks als Informationsflussmenge pro Tag, ist es notwendig, mehrere Auxiliaries zu erzeugen, da nicht eins sowohl für das sichtbare Auxiliary der Kommunikation als auch für den Ein- und Ausgangsfluss der Stocks verwendet werden kann. Die Flüsse der Stocks werden nur bei Veränderungen beschaltet, wobei hingegen die sichtbaren Auxiliaries für die Kommunikation durchgängig den aktuellen Kommunikationsfluss anzeigen.

Folglich wird für das Auxiliary der Kommunikation in der Modellansicht ein nicht sichtbares Auxiliary erzeugt um dort die Werte, wie sie in dem sichtbaren Auxiliary benötigt werden, einzubringen. Das versteckte Auxiliary schaltet dabei seinen Ausgang auf null, sofern mindestens einer des Kommunikationspaares abwesend ist, und auf den Wert des normalisierten Informationsflusses, sofern beide anwesend sind und wird anschließend auf den Ausgang des sichtbaren Auxiliaries der Kommunikation aufaddiert in dem es dort verlinkt wird. Das Setzen des Auxiliarywertes wird mit Hilfe einfacher Trigger realisiert, die angeben, dass bei einem gegebenem Zeitpunkt ein bestimmter Wert angenommen wird, ansonsten, bei allen anderen Zeitpunkten, ein anderer Wert geschaltet wird.

Für die Flüsse der Stocks werden ebenfalls versteckte Auxiliaries erzeugt, jedoch schalten diese ihren Ausgang durchgängig auf den Wert des normalisierten Informationsflusses, auch wenn einer der Kommunikationspartner abwesend ist. Grund dafür ist, dass die Auxiliaries in die Ein- und Ausgangsflüsse geschaltet werden, jedoch nicht mittels einfachem Aufaddieren wie bei den sichtbaren Auxiliaries sondern durch Trigger gesteuert werden. Im Eingangsfluss wird mit Hilfe eines Triggers, welcher immer bei Wiederantritt der Anwesenheit einer Person nach einem Ausfall schaltet, der Wert entsprechend in den Stock gegeben. Der Trigger im Ausgangsfluss wird analog zu dem des Eingangsflusses realisiert, jedoch schaltet dieser bei Beginn der Abwesenheit einer Person, um den Bestand des Stocks, den Informationsfluss pro Tag, zu reduzieren.

Sobald alle Kommunikationen iteriert worden sind, ist die Vorbereitung für die Simulation ohne Umleitungen abgeschlossen. Nun wird die Simulation, für den Nutzer unsichtbar, durchgeführt um die Informationsflussmengen über die Personen zu jedem Zeitpunkt zu kennen. Wurde die Option ausgewählt Informationen umzuverteilen, so werden nun die Flüsse für das gewählte Verhalten berechnet.

Da das Erzeugen und Setzen der System Dynamics Elemente ähnlich zu dem bereits Beschriebenen ist, wird speziell nur auf die Umverteilung eingegangen. Eine detaillierte Beschreibung der Erzeugung der Elemente ist im Quellcode vorhanden und kann dort eingesehen werden.

Stellvertreter übernimmt, Flussanteil wird verteilt

Durch die vorangegangene Simulation ohne Umleitung stehen nun die Informationsflussmengen über die Personen bereit um infrage kommende Stellvertreter täglich neu berechnen zu können. Dies wird realisiert indem der Simulationszeitraum durchlaufen und über die Menge an abwesenden Personen im Modell iteriert wird. Informationsflüsse von Personen die nicht länger abwesend sind als die Dauer die Informationen liegen bleiben, gehen dabei verloren. Bei Überschreitung des Zeitraumes wird über jede Kommunikation iteriert bei der die abwesende Person ein Mitglied davon ist, um zu ermitteln ob ein Stellvertreter anwesend ist.

Unterschieden wird dabei zwischen Aktivitäten an denen nur Personen, repräsentiert durch einzelne Personen im FLOW-Modell, teilnehmen oder eine Gruppe, repräsentiert durch das mehrfache Personen Symbol im FLOW-Modell, teilnehmen. Dies hat den Hintergrund, dass bei Gruppenaktivitäten der Stellvertreter immer innerhalb der Gruppe ermittelt werden soll, wobei hingegen bei Kommunikationen ohne Gruppensymbol Stellvertreter nicht Mitglied der ursprünglichen Kommunikation sein müssen, wodurch mehr Szenarien in der Simulation möglich sind.

Personenaktivitäten - Es wird über die anwesenden Mitglieder der Kommunikation iteriert und geprüft, ob für diese Stellvertreter hinterlegt worden sind, um diese für den Ausfall auszutauschen. Dabei wird der Stellvertreter mit dem zum Ausfallzeitpunkt geringsten Informationsflusses ausgewählt, welches durch die vorangegangene Simulation ermittelt werden kann. Sollte kein Stellvertreter anwesend sein, so gehen die Informationen verloren.

Die Berechnung des umzuverteilenden Informationsflusses erfolgt indem die normalisierten Werte aus der communication_values-Sektion zwischen dem Ausfall und den Kommunikationsmitgliedern über die verwendeten Kanäle aufaddiert werden. Jedoch würden so einige Kanten doppelt gezählt werden, wenn noch ein Kommunikationsmitglied ausfällt, weshalb Kanten, bei denen ein anderes Mitglied der Kommunikation ausfällt, halbiert werden. Der restliche Wert des Flusses wird berücksichtigt, wenn der Stellvertreter für diesen Ausfall berechnet wird, da dieser ebenfalls die Hälfte verwendet.

Nach der Berechnung werden nun vom Stellvertreter zu den anwesenden Aktivitätsmitgliedern Kanten erzeugt und die Informationsmenge gleichmäßig auf die Kanten verteilt.

Gruppenaktivitäten - Für Gruppenaktivitäten ist die Verteilung der Informationen identisch zu der der Personenaktivitäten. Jedoch wird nicht über die anwesenden Mitglieder iteriert um Stellvertreter zu prüfen, sondern über die anwesenden Mitglieder der Kommunikation, da für Gruppenaktivitäten immer die Mitglieder der Gruppe als Stellvertreter berücksichtigt werden auch wenn diese im FLOW-Modell nicht gesetzt worden waren.

Sind die Stellvertreter für eine Personenaktivität oder Gruppenaktivität gesetzt, so müssen die Simulationswerte, auf die die Berechnung der Stellvertreter zurückgreift, neu berechnet werden, da Stellvertreter und Aktivitätsmitglieder nun einen veränder-

ten Informationsfluss haben. Dies geschieht nach jeder Aktivität, die für eine ausfallende Person iteriert wird, in der Stellvertreter eingesetzt werden.

Stellvertreter übernimmt, Flussanteil wird gleichmäßig verteilt

Da dieses Flussverhalten analog zum vorangegangenen Verhalten ist, wird nur gesondert auf die Unterschiede eingegangen die mit Auswahl dessen einhergehen.

Die Umverteilung findet bei diesem Verfahren nicht durch den Stellvertreter per se statt, sondern durch alle Aktivitätsmitglieder. Dabei übernimmt der Stellvertreter die Position des Mitglieds, jedoch wird die Kommunikation auf alle Paare der Kommunikation aufgeteilt.

Bei einer Aktivität mit fünf Mitgliedern und und einem Ausfall, würde bei der ersten Verhaltensweise vier Kanten zwischen Stellvertreter und den restlichen anwesenden Aktivitätsmitgliedern entstehen.

Jedoch würde bei der gleichmäßigen Verteilung zwischen jedem Paar eine Kante erzeugt werden, analog zum vollständigen Graphen. Dadurch ergibt sich bei fünf Aktivitätsmitgliedern und einem Ausfall, berechenbar durch die Dreieckszahl, zehn Kanten auf denen die Informationen verteilt werden.

Dies hat zufolge, dass der minimale Informationsfluss der Aktivität grundsätzlich angehoben wird, da die Informationen nun gleichmäßig auf alle Mitglieder verteilt worden sind.

4.2.4 Ausgabe von Ergebnissen

Damit neben der optischen Ansicht der Informationsflüsse auch andere Simulationsergebnisse wie die Menge an umverteilten Informationen eingesehen werden kann, wird nach der Simulation zum Tab 'Ergebnisse' gewechselt. Dort werden die Ergebnisse aufgelistet, die bei der Simulation ermittelt werden konnten. Dabei werden einige aus den Simulationsergebnissen der SD.js ausgelesen, und einige werden während der Bestimmung der Stellvertreter und Umleitungen erfasst, die nicht aus den Ergebnissen der SD.js auslesbar sind.

4.3 JIRA-Plugin

4.3.1 Einbindung an das Prodynamics-Plugin

Da nun sowohl Konverter als auch Simulator vollständig einzeln konzipiert und implementiert worden sind, müssen diese in das Prodynamics-Plugin zusammengefügt werden, welches ein Template bereitstellt um das ganze zu implementieren.

Die anzuzeigende Webseite im Pluginmodul ist dabei in der SystemDynamics.vm-Datei gespeichert und muss nur noch modifiziert werden. Dabei wird der vorhandene Inhalt der Pluginmodulseite im benötigten Umfang gelöscht und durch die des Konverters und Simulators ersetzt. Zum Umschalten werden zwei Checkboxen oben an der Seite angeboten, welche mit dem Javascript-Code verlinkt werden.

Dieser wird jedoch nicht über die Ressourcenverwaltung von JIRA eingebunden, da es dort, wie es in Kapitel 5.2 erläutert wird, zu Problemen kommt. Der Code des Konverters und des Simulators kann problemlos als script-Tag oberhalb des eingefügten Inhalts eingetragen werden.

Jedoch muss die SD.js extern über einen anderen Webserver geladen werden, da sie, bei selbiger Realisierung wie dem Konverter- und Simulatorcode, von JIRA falsch geparst wird und nicht funktionsfähig ist.

Zuletzt werden die CSS-Styles beider Programme in eine neue Datei zusammengelegt und anschließend über die JIRA Ressourcenverwaltung mit Hilfe der atlassian-plugins.xml eingebunden, da dies problemlos funktioniert und der eleganteste Weg ist, Ressourcen einzubinden.

5 Probleme

5.1 Diverse Workarounds durch die SD.js

Leider mussten durch die Unausgereiftheit der SD.js diverse Workarounds in kauf genommen werden, um diese Arbeit in dem Rahmen realisieren zu können. So unterstützt die SD.js in der aktuellen Version keine Aliase um Objekte in der Modellansicht den gleichen Text zu geben, jedoch im Code mithilfe eines ID- oder Name-Attribut unterscheiden zu können. Es ist lediglich das Name-Attribut verfügbar, welches gleichzeitig den Anzeigetext stellt. Zwar können mit Zeilenumbrüchen wie '\n' Texte in der Ansicht umgebrochen werden, jedoch unterscheidet die SD.js nicht zwischen Names, die den gleichen sichtbaren Text haben, aber eine unterschiedliche Anzahl an Umbrüchen. Deshalb mussten die Auxiliaries, welche bei der Herleitung der Formeln erstellt wurden, mit einzigartigen Names versehen werden, damit es nicht zu Kollisionen kommen kann und die Auxiliaries in der Modellansicht enthalten aus diesem Grund ebenfalls die Namen der beteiligten Personen, obwohl es die Ansicht unübersichtlicher macht.

Zudem können keine Styles aus der XMILE geparst werden, welches bei schlechter Platzierung der Elemente dazu führen kann, dass die Pfeile durch die Beschriftung eines Stocks oder Auxiliaries laufen. Dies resultiert ebenfalls darin, dass Stellvertreter nicht in der Modellansicht durch farblich andere Verbindungen gekennzeichnet werden können.

5.2 Unverträglichkeit von JIRA mit bestimmten Code

Die SD.js konnte lediglich über drei Wege angebunden werden, wobei die ersten zwei aufgrund JIRA inkompatibel sind. Die Einbindung als Ressource in der atlassian-plugins.xml führte dazu, dass kein ordnungsgemäßes Minifying[1] durch den YUI Compressor durchgeführt werden konnte, was darin resultierte, dass die Bibliothek überhaupt nicht eingebunden wurde. Mit der inline Einbindung über die .vm-Datei im HTML-Code, wird das Minifying der Pluginressourcen übersprungen, jedoch wird die SD.js fehlerhaft von JIRA geparst, was dazu geführt hat, dass ganze Zeilen verrutscht sind und dadurch die Menge an Klammern und Kommentareblöcken inkonsistent wurde, wodurch der Code im Browser nicht mehr ausführbar war.

Deshalb wurde die SD.js letztlich, über einen zusätzlichen Webserver angebunden, welcher die Bibliothek unverändert und ohne Fehler zur Verfügung stellt.

[1] Reduzierung des Codes in syntaktisch kürzeren, aber semantisch equivalenten Code, um Traffic bei der Übertragung zu sparen.

6 Abschließende Betrachtung

6.1 Zusammenfassung

In dieser Arbeit wurde eine ein Programm zur Umwandlung von initial statischen FLOW-Kommunikationsmodellen nach dynamischen System Dynamics Modellen realisiert, indem diese um die dynamische Komponente erweitert wurden. Des Weiteren wurde ein Simulator programmiert, um mit Hilfe der SD.js Bibliothek die umgewandelten Modelle zu simulieren und dadurch explorativ Szenarien durchspielen zu können. So konnten bestehende Kommunikationsstrukturen in FLOW modelliert werden um sie anschließend mit explorativen Szenarien im Simulator durchspielen zu können. Dadurch wurden Kommunikationsstrukturen analysiert, um potenzielle strukturelle Schwachstellen identifizieren zu können, damit diese nach einer Auswirkungen von Störungen minimiert wird.

6.2 Zukunftsausblick

Durch die Anbindung an das Prodynamics-Plugin und den Datensätzen, die es in der JIRA-Datenbank speichert, könnte der Simulator dahingehend erweitert werden, mit Auswahl eines Projektes und Sprintes direkt die Kommunikationsintensitäten aus der Datenbank zu übernehmen. Dadurch würde der Nutzer Zeit sparen, mühsam die Werte manuell eingeben zu müssen.

Des Weiteren könnten die Simulationsergebnisse um alternative Modellstrukturen erweitert werden um eine Vorhersage zu präsentieren, wie die Kommunikationsstrukturen sich verändern bei gewissen Ausfällen verändern könnten um diese besser abfedern zu können.

Literaturverzeichnis

[1] K. Stapel und K. Schneider,
FLOW-Methode - Methodenbeschreibung zur Anwendung von FLOW,
CoRR, abs/1202.5919, 2012

[2] Quelle der Abbildung: `http://www.se.uni-hannover.de/pages/de:projekte_flow`,
Letzter Zugriff: 16. Oktober 2019

[3] M. Höppner,
Interaktiver Editor für Informationsflussmodelle,
Bachelorarbeit, Fachgebiet Software Engineering, Institut für Praktische Informatik, Leibniz Universität Hannover

[4] J. Klünder,
Analyse der Zusammenarbeit in Softwareprojekte mittels Informationsflüssen und Interaktionen in Meetings,
Logos, 2019

[5] J. W. Forrester,
Counterintuitive behavior of social systems. Technology Review 73(3): 52-68,
Massachusetts Institute of Technology, 1971

[6] V. G. Diker, R B. Allen,
XMILE: towards an XML interchange language for system dynamics models. System Dynamics Review 21(4): 351-359,
Wiley Online Library, 2005

[7] B. K. Bala, F. M. Arshad, K. M. Noh,
System Dynamics - Modelling and Simulation,
Springer, 2017

[9] L. B. Sweeney, J. D. Sterman,
Bathtub dynamics: initial results of a systems thinking inventory. System Dynamics Review 16(4): 249-286,
Wiley, 2000

[10] K. Schneider, O. Liskin, H. Paulsen, S. Kauffeld,
Media, mood, and meetings: Related to project success? ACM Transactions on Computing Education 15 (4),
ACM, 2015

[11] P. Watzlawick, J. H. Beavin, D. D. Jackson,
Menschliche Kommunikation. Formen, Störungen, Paradoxien (11): 53-70,
Huber, Bern, 2007

[12] F. Kortum, J. Klünder, K. Schneider,
Behavior-driven dynamics in agile development: The effects of fast feedback on teams. Proceedings of the 2019 International Conference on Software and System Process.,
ACM, 2019