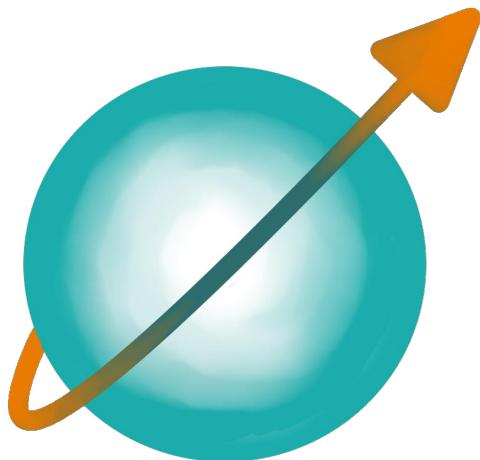


Software-Projekt 2 WiSe 2018/2019
VAK 03-BA-901.02

Testprotokoll: GraphIt
Black-Box-Test



Anthony Mendil	antmen@tzi.de
Bastian Rexhäuser	brexhaeu@tzi.de
Clement Phung	clement1@tzi.de
Jacky Philipp Mach	machja@tzi.de
Jonah Jaeger	jjaeger@tzi.de
Nina Unterberg	nin_unt@tzi.de

Abgabe: 10. März 2019 — Version 1.0

Anmerkung: Weil für manche Tests bestimmte Funktionalitäten funktionieren müssen, bauen die Tests aufeinander auf. Beispielsweise wird beim Testfall **Knoten hinzufügen** davon ausgegangen, dass nach dem Testfall **Sphären hinzufügen** entsprechende Sphären existieren und somit nicht noch vorher erstellt werden müssen. Ebenso wird beispielsweise beim Hinzufügen einer Sphäre nicht explizit erwähnt, dass in der Gui der Button zum Hinzufügen von Sphären betätigt wird. Sofern es nicht anders beschrieben ist, soll im folgenden also davon ausgegangen werden, dass bei den jeweiligen Aktionen des Testers die entsprechenden Buttons in der Kopfleiste der Gui betätigt werden. Des weiteren wird im folgenden nicht jedes mal erwähnt, dass das Auswählen von einzelnen Elementen mithilfe eines Linksklick geschieht, beziehungsweise Umschalten und Linksklicks für mehrere Elemente. Bewegt werden Elemente durchs gedrückt Halten des Rechtsklicks. Vor dem eigentlichen Testen der Anwendung sind im folgenden Bilder der Benutzeroberfläche zu sehen. Bei Testfällen, bei denen alle Tests negativ ausgefallen sind, sind Abbildungen (sofern vorhanden) nach den Tests zu finden und beziehen sich auf alle Testdurchläufe des entsprechenden Testfalls. Andernfalls werden bei jedem Testdurchlauf Abbildungen gezeigt (sofern vorhanden). Auch hat sich im Verlauf bei der Erstellung des Testprotokolls die Standardfarben für Sphären und Symptomen geändert, was bei einigen Abbildungen zu erkennen ist.

Inhaltsverzeichnis

1 Bilder der Benutzeroberfläche	12
2 Ersteller-Modus	14
2.1 Sphären hinzufügen an freien Stellen	14
2.1.1 Notizen	14
2.1.2 Fazit	14
2.2 Sphäre hinzufügen, sodass es zu einer Überschneidung mit einer anderen kommen würde	14
2.2.1 Notizen	14
2.2.2 Fazit	15
2.3 Knoten hinzufügen an freien Stellen in Sphären	15
2.3.1 Notizen	15
2.3.2 Fazit	15
2.4 Knoten hinzufügen an belegter Stelle	15
2.4.1 Notizen	15
2.4.2 Fazit	16
2.5 Knoten hinzufügen außerhalb einer Sphäre	16
2.5.1 Notizen	16
2.5.2 Fazit	16
2.6 Verstärkende Relation zwischen zwei Symptome der selben Sphäre	16
2.6.1 Notizen	16
2.6.2 Fazit	16
2.7 Rückgängig machen (Undo)	17
2.7.1 Notizen	17
2.7.2 Fazit	17
2.8 Wiederherstellen (Redo)	17
2.8.1 Notizen	17
2.8.2 Fazit	18
2.9 Relation zwischen zwei Symptomen, zwischen denen bereits eine Relation existiert	18
2.9.1 Notizen	18
2.9.2 Fazit	18
2.10 Abschwächende Relation zwischen zwei Symptomen unterschiedlicher Sphären	18
2.10.1 Notizen	18
2.10.2 Fazit	19
2.11 Unbekannte Relation zwischen zwei Symptomen unterschiedlicher Sphären	19
2.11.1 Notizen	19
2.11.2 Fazit	19
2.12 Einstellen der Vorlage-Regeln	20
2.12.1 Notizen	20

2.12.2 Fazit	21
2.13 Schließen des Programms ohne den Graphen zu speichern	21
2.13.1 Notizen	21
2.13.2 Fazit	21
2.14 Verstoß gegen die eigenen Vorlage-Regeln	22
2.14.1 Notizen	22
2.14.2 Fazit	22
2.15 Entfernen einer Relation im Ersteller-Modus bei der in den Vorlage-Regeln das Verändern des Styles verboten wurde	22
2.15.1 Notizen	22
2.15.2 Fazit	23
2.16 Speichern der Datei in den eigenen Dateien	23
2.16.1 Notizen	23
2.16.2 Fazit	23
3 Bearbeiter-Modus	24
3.1 Rückgängig machen einer Aktion die im Ersteller Modus durchgeführt wurde und in den Vorlage-Regeln verboten wurde	24
3.1.1 Notizen	24
3.1.2 Fazit	25
3.2 Datei Öffnen	25
3.2.1 Notizen	25
3.2.2 Fazit	26
3.3 Keinen Verlauf im Ersteller-Modus erstellen	26
3.3.1 Notizen	26
3.3.2 Fazit	26
3.4 Mehr Sphären hinzufügen als in den Vorlage-Regeln erlaubt	26
3.4.1 Notizen	27
3.4.2 Fazit	27
3.5 Abschwächende Relation hinzufügen obwohl es in den Vorlage-Regeln verboten wird	27
3.5.1 Notizen	27
3.5.2 Fazit	28
3.6 Einer Sphäre ein Symptom hinzufügen obwohl dies durch die Vorlage-Regeln verboten wird	28
3.6.1 Notizen	28
3.6.2 Fazit	28
3.7 Titel einer Sphäre ändern bei der es in den Vorlage-Regeln verboten wird	28
3.7.1 Notizen	28
3.7.2 Fazit	28
3.8 Style eines Symptoms ändern bei dem es in den Vorlage-Regeln verboten wird	28
3.8.1 Notizen	29
3.8.2 Fazit	29

3.9 Relationsart einer Relation ändern bei der es in den Vorlage-Regeln verboten wird	29
3.9.1 Notizen	29
3.9.2 Fazit	29
3.10 Entfernen einer Sphäre mit Symptomen und Relationen	29
3.10.1 Notizen	29
3.10.2 Fazit	30
3.11 Rückgängig machen des Entfernens einer Sphäre mit Symptomen und Relationen	30
3.11.1 Notizen	31
3.11.2 Fazit	32
3.12 Vergrößern einer Sphäre	32
3.12.1 Notizen	32
3.12.2 Fazit	32
3.13 Verkleinern einer Sphäre	33
3.13.1 Notizen	33
3.13.2 Fazit	33
3.14 Ändern der Farbe einer Sphäre	33
3.14.1 Notizen	34
3.14.2 Fazit	34
3.15 Ändern der Schriftart einer Sphäre	34
3.15.1 Notizen	34
3.15.2 Fazit	35
3.16 Verändern der Schriftgröße einer Sphäre	35
3.16.1 Notizen	35
3.16.2 Fazit	36
3.17 Sphären automatisch anordnen obwohl eine Sphäre nicht bewegt werden darf	36
3.17.1 Notizen	36
3.17.2 Fazit	36
3.18 Sphären automatisch anordnen	37
3.18.1 Notizen	37
3.18.2 Fazit	37
3.19 Rückgängig machen der automatischen Anordnung von Sphären	37
3.19.1 Notizen	38
3.19.2 Fazit	38
3.20 Bewegen einer Sphäre auf eine andere	38
3.20.1 Notizen	38
3.20.2 Fazit	39
3.21 Entfernen eines Symptoms, das Relationen hat	39
3.21.1 Notizen	39
3.21.2 Fazit	40

3.22 Rückgängig machen des Entfernens eines Symptoms mit Relationen	40
3.22.1 Notizen	40
3.22.2 Fazit	41
3.23 Entfernen mehrere Symptome ohne Relationen	41
3.23.1 Notizen	41
3.23.2 Fazit	44
3.24 Vergrößern eines Symptoms	44
3.24.1 Notizen	44
3.24.2 Fazit	45
3.25 Verkleinern eines Symptoms	45
3.25.1 Notizen	45
3.25.2 Fazit	45
3.26 Ändern der Füllfarbe eines Symptoms	46
3.26.1 Notizen	46
3.26.2 Fazit	46
3.27 Ändern der Randfarbe eines Symptoms	46
3.27.1 Notizen	46
3.27.2 Fazit	47
3.28 Ändern der Form eines Symptoms	47
3.28.1 Notizen	47
3.28.2 Fazit	47
3.29 Automatisches Anordnen der Symptome obwohl ein Symptom nicht bewegt werden darf	48
3.29.1 Notizen	48
3.29.2 Fazit	48
3.30 Automatisches Anordnen der Symptome	48
3.30.1 Notizen	48
3.30.2 Fazit	49
3.31 Ändern der Schriftart mehrerer Symptome, wobei dies durch die Vorlage-Regeln nicht bei allen erlaubt wird	49
3.31.1 Notizen	49
3.31.2 Fazit	50
3.32 Ändern der Schriftgröße eines Symptoms	50
3.32.1 Notizen	50
3.32.2 Fazit	51
3.33 Bewegen eines Symptoms in eine andere Sphäre	51
3.33.1 Notizen	51
3.33.2 Fazit	52
3.34 Bewegen eines Symptoms an eine Stelle an der keine Sphäre liegt	52
3.34.1 Notizen	52
3.34.2 Fazit	53

3.35 Verändern der Farbe einer Relation	53
3.35.1 Notizen	53
3.35.2 Fazit	55
3.36 Verändern der Linienart einer Relation	55
3.36.1 Notizen	55
3.36.2 Fazit	55
3.37 Zusammenfassung gleicher Pfeilspitzen	55
3.37.1 Notizen	56
3.37.2 Fazit	56
3.38 Verändern der Relationsart einer Relation	56
3.38.1 Notizen	57
3.38.2 Fazit	57
3.39 Hinzufügen von Ankerpunkten	57
3.39.1 Notizen	58
3.39.2 Fazit	58
3.40 Ankerpunkte ausblenden	58
3.40.1 Notizen	58
3.40.2 Fazit	59
3.41 Ankerpunkte einblenden	59
3.41.1 Notizen	59
3.41.2 Fazit	60
3.42 Entfernen mehrerer Relationen	60
3.42.1 Notizen	60
3.42.2 Fazit	61
3.43 Rückgängig machen des Entfernens mehrerer Relationen mit Ankerpunkten	61
3.43.1 Notizen	61
3.43.2 Fazit	62
3.44 Bewegen eines Symptoms, bei dem eine Relation Ankerpunkte hat	62
3.44.1 Notizen	62
3.44.2 Fazit	63
3.45 Bei angezeigten Ankerpunkten in den Analyse-Modus wechseln	63
3.45.1 Notizen	63
3.45.2 Fazit	66
3.46 Änderung der Beschriftung eines Symptoms	66
3.46.1 Notizen	66
3.46.2 Fazit	68
3.47 Ändern der Sprache des Graphen	68
3.47.1 Notizen	68
3.47.2 Fazit	69
3.48 Bewegen eines Symptoms auf ein anderes Symptom	69
3.48.1 Notizen	69
3.48.2 Fazit	69

3.49 Änderung der Beschriftung eines Symptoms zu einer bereits existierenden Beschriftung	70
3.49.1 Notizen	70
3.49.2 Fazit	71
3.50 Rückgängig machen der Änderung der Beschriftung von Symptomen	71
3.50.1 Notizen	71
3.50.2 Fazit	73
3.51 Änderung der Beschriftung einer Sphäre	73
3.51.1 Notizen	73
3.51.2 Fazit	75
3.52 Änderung der Beschriftung einer Sphäre zu einer bereits existierenden Beschriftung	75
3.52.1 Notizen	75
3.52.2 Fazit	76
3.53 Eine Sphäre mehr verkleinern als es die Symptome erlauben	76
3.53.1 Notizen	76
3.53.2 Fazit	77
3.54 Eine Sphäre mehr vergrößern als es die anderen Sphären erlauben	77
3.54.1 Notizen	77
3.54.2 Fazit	78
3.55 Ausblenden von Elementen	78
3.55.1 Notizen	78
3.55.2 Fazit	79
3.56 Einblenden von Elementen	79
3.56.1 Notizen	79
3.56.2 Fazit	80
3.57 Elemente aus Menge der auszublendenen Elemente entfernen	80
3.57.1 Notizen	80
3.57.2 Fazit	82
3.58 Hervorheben von Elementen	82
3.58.1 Notizen	82
3.58.2 Fazit	82
3.59 Ein Element aus Menge der hervorzuhebenden Elemente entfernen	83
3.59.1 Notizen	83
3.59.2 Fazit	83
3.60 Exportieren als Protokoll	85
3.60.1 Notizen	85
3.60.2 Fazit	86
3.61 Anzeige des Verlaufs innerhalb der Applikation	86
3.61.1 Notizen	86
3.61.2 Fazit	89

3.62 Verlauf filtern	89
3.62.1 Notizen	89
3.62.2 Fazit	90
3.63 Graph ändern und Verlauf neu laden	90
3.63.1 Notizen	90
3.63.2 Fazit	91
3.64 Anzeige der kompletten Übersicht	91
3.64.1 Notizen	92
3.64.2 Fazit	92
3.65 Relation in der Übersicht auswählen	92
3.65.1 Notizen	92
3.65.2 Fazit	93
3.66 Anzeigen ausschließlich verstärkender Relationen	93
3.66.1 Notizen	93
3.66.2 Fazit	94
3.67 Textsuche	94
3.67.1 Notizen	94
3.67.2 Fazit	95
3.68 Zoom Auswahl	95
3.68.1 Notizen	95
3.68.2 Fazit	97
3.69 Zoom Bar	97
3.69.1 Notizen	98
3.69.2 Fazit	99
3.70 Zoom Kontext	99
3.70.1 Notizen	99
3.70.2 Fazit	100
3.71 Ändern der Relationsart einer Relation über die Übersicht	101
3.71.1 Notizen	101
3.71.2 Fazit	103
3.72 Änderung der Füllfarbe eines Symptoms über die Übersicht	104
3.72.1 Notizen	104
3.72.2 Fazit	106
3.73 Löschen einer Sphäre über die Übersicht	106
3.73.1 Notizen	107
3.73.2 Fazit	108

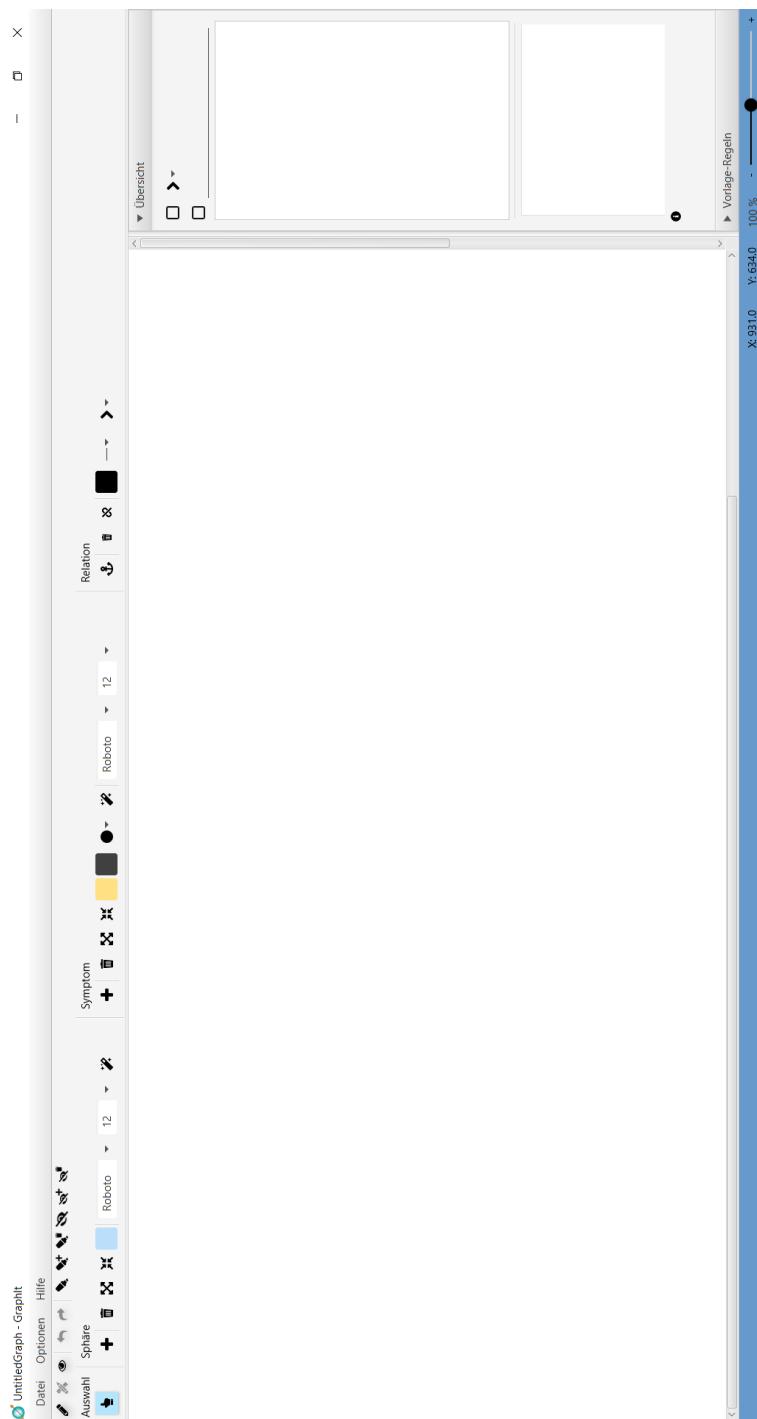
4 Analyse-Modus	110
4.1 Korrektes Berechnen und Anzeigen des Umfangs, des Vernetzungsindex und des Strukturindex	110
4.1.1 Notizen	110
4.1.2 Fazit	112
4.2 Vorgänger von einem Symptom	112
4.2.1 Notizen	112

4.2.2	Fazit	114
4.3	Nachfolger von einem Symptom	114
4.3.1	Notizen	114
4.3.2	Fazit	116
4.4	Nachfolger und Vorgänger eines Symptoms	116
4.4.1	Notizen	116
4.4.2	Fazit	118
4.5	Kürzester Weg von einem Symptom zu einem anderen (Weg existiert)	118
4.5.1	Notizen	118
4.5.2	Fazit	120
4.6	Kürzester Weg von einem Symptom zu einem anderen (Weg existiert nicht)	120
4.6.1	Notizen	120
4.6.2	Fazit	122
4.7	Alle Wege von einem Symptom zu einem anderen	122
4.7.1	Notizen	122
4.7.2	Fazit	124
4.8	Pfeilketten	124
4.8.1	Notizen	124
4.8.2	Fazit	126
4.9	Konvergente Verzweigungen	126
4.9.1	Notizen	126
4.9.2	Fazit	128
4.10	Divergente Verzweigungen	128
4.10.1	Notizen	128
4.10.2	Fazit	130
4.11	Verzweigungen	130
4.11.1	Notizen	130
4.11.2	Fazit	132
4.12	Zyklen	132
4.12.1	Notizen	132
4.12.2	Fazit	134
5	Import, Export, Ändern der Sprache und Hilfe	134
5.1	Vorlage exportieren	135
5.1.1	Notizen	135
5.1.2	Fazit	135
5.2	Vorlage importieren	135
5.2.1	Notizen	135
5.2.2	Fazit	135
5.3	GXL exportieren	135
5.3.1	Notizen	135
5.3.2	Fazit	136

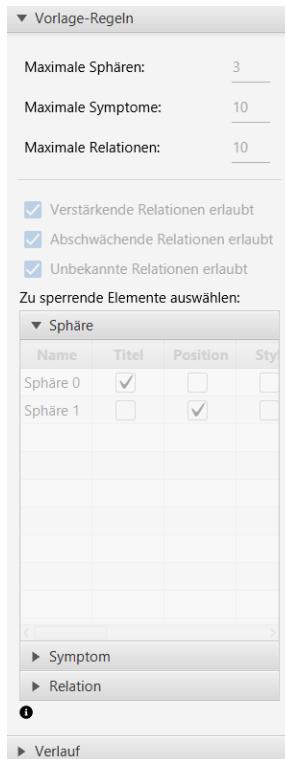
5.4	GXL importieren	136
5.4.1	Notizen	136
5.4.2	Fazit	136
5.5	Exportieren als PDF	136
5.5.1	Notizen	136
5.5.2	Fazit	136
5.6	Drucken	136
5.6.1	Notizen	137
5.6.2	Fazit	137
5.7	Sprache der Benutzeroberfläche ändern	137
5.7.1	Notizen	137
5.7.2	Fazit	137
5.8	Sprache des Graphen und der Benutzeroberfläche ändern	137
5.8.1	Notizen	137
5.8.2	Fazit	137
5.9	Anzeigen des Benutzerhandbuchs	138
5.9.1	Notizen	138
5.9.2	Fazit	138
5.10	Über uns	138
5.10.1	Notizen	139
5.10.2	Fazit	139
6	JUnit Tests	140
6.1	Analysis Test	140
6.2	GXLio Test	153
6.3	OOFio Test	178

1 Bilder der Benutzeroberfläche

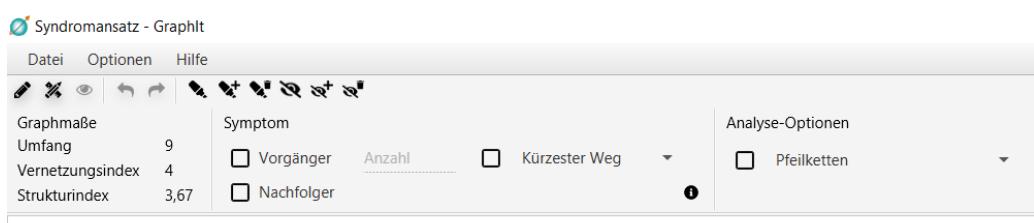
Die Benutzeroberfläche im Ersteller-Modus:



Der einzige Unterschied zur Benutzeroberfläche im Bearbeiter-Modus ist der, dass es in der Seitenleiste zusätzlich zu den Vorlage-Regeln den Verlauf gibt, wobei aber die Vorlage-Regeln nicht editierbar sind sondern nur zum Lesen der Regeln dient:



Was den Analyse-Modus von den anderen unterscheidet, ist die Kopfleiste und dass es keine Vorlage-Regeln gibt:



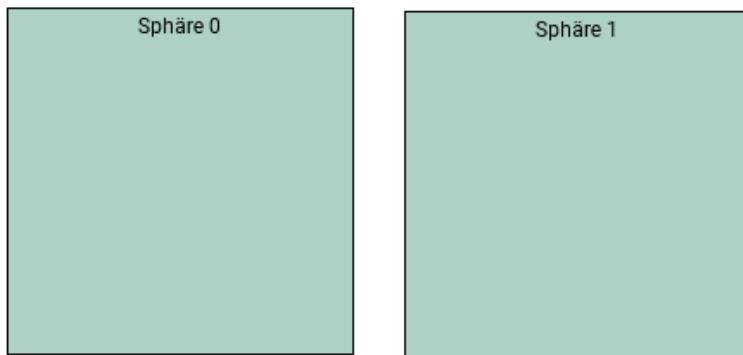
2 Ersteller-Modus

2.1 Sphären hinzufügen an freien Stellen

Der Ersteller erstellt zuerst zwei Sphären an verschiedenen Positionen. Eine links und dann eine rechts.

2.1.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 21.02.2019 um 17:04
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 19:10
Test negativ.



2.1.2 Fazit

Die Software arbeitet wie erwartet.

2.2 Sphäre hinzufügen, sodass es zu einer Überschneidung mit einer anderen kommen würde

Der Ersteller versucht eine dritte Sphäre an einer Position hinzuzufügen, an der es zu einer Überschneidung mit einer der bereits erstellten Sphären kommen würde (nicht direkt auf der Sphäre). Hierbei ist zu erwarten, dass dies nicht erfolgreich ist und in der Fußleiste eine entsprechende Meldung angezeigt wird.

2.2.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 21.02.2019 um 17:15
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 19:12
Test negativ.

2.2.2 Fazit

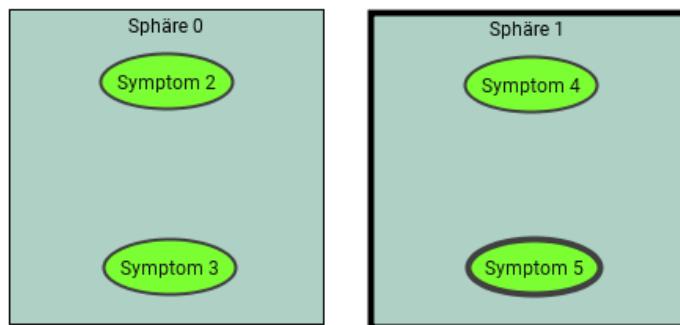
Die Software arbeitet wie erwartet. Die Abbildung ist identisch zur vorherigen.

2.3 Knoten hinzufügen an freien Stellen in Sphären

Es werden den zwei existierenden Sphären jeweils zwei Symptome an verschiedenen Positionen hinzugefügt. Erst in die linke Sphäre oben mittig und unten mittig und dann in die rechte Sphäre oben mittig und unten mittig.

2.3.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 21.02.2019 um 17:20
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 19:14
Test negativ.



2.3.2 Fazit

Die Software arbeitet wie erwartet.

2.4 Knoten hinzufügen an belegter Stelle

Der Ersteller versucht an einer Stelle innerhalb einer Sphäre ein Symptom hinzuzufügen, an der bereits ein anderes Symptom ist. Hierbei ist analog zu Sphären zu erwarten, dass dies nicht erfolgreich ist und in der unteren Leiste eine entsprechende Meldung angezeigt wird.

2.4.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 21.02.2019 um 17:30
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 19:17
Test negativ.

2.4.2 Fazit

Die Software arbeitet wie erwartet. Die Abbildung ist identisch zur vorherigen.

2.5 Knoten hinzufügen außerhalb einer Sphäre

Der Ersteller versucht an einer Stelle außerhalb einer Sphäre ein Symptom hinzuzufügen. Zu erwarten ist, dass dies nicht erlaubt wird. Durch eine Meldung in der Fußleiste sollte dies angezeigt werden.

2.5.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 21.02.2019 um 17:39
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 19:17
Test negativ.

2.5.2 Fazit

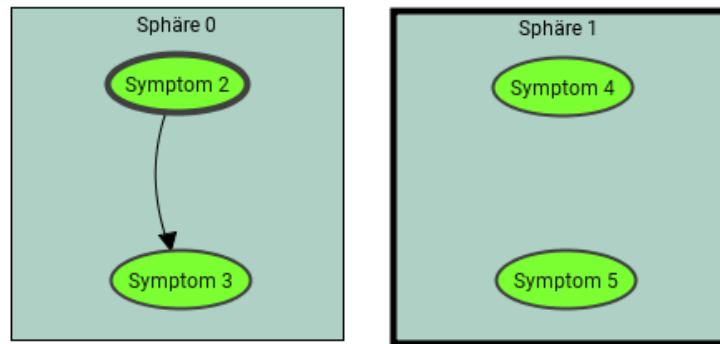
Die Software arbeitet wie erwartet. Die Abbildung ist identisch zur vorherigen.

2.6 Verstärkende Relation zwischen zwei Symptome der selben Sphäre

Der Ersteller zieht eine verstärkende Relationen von Symptom 2 nach Symptom 3.

2.6.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 21.02.2019 um 17:39
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 19:18
Test negativ.



2.6.2 Fazit

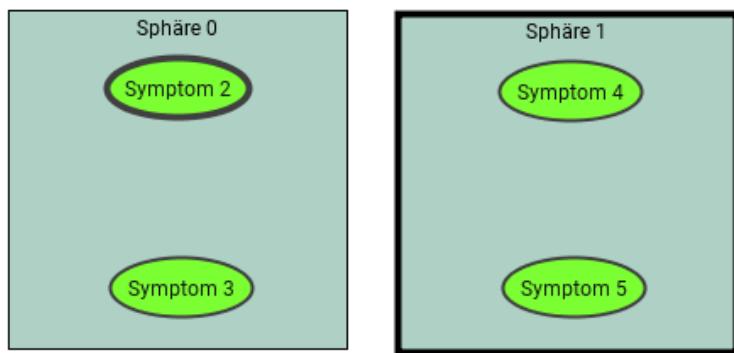
Die Software arbeitet wie erwartet.

2.7 Rückgängig machen (Undo)

Der Ersteller betätigt den Button zum Rückgängigmachen der letzten Aktion.

2.7.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 21.02.2019 um 17:45
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 19:19
Test negativ.



2.7.2 Fazit

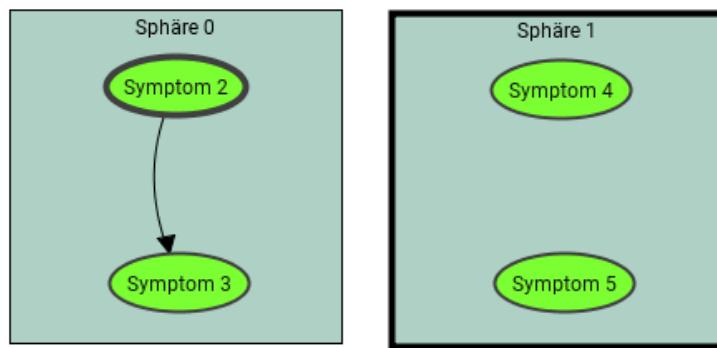
Die Software arbeitet wie erwartet.

2.8 Wiederherstellen (Redo)

Der Ersteller betätigt den Button zum Wiederherstellen der zuletzt rückgängig gemachten Aktion.

2.8.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 21.02.2019 um 17:53
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 19:20
Test negativ.



2.8.2 Fazit

Die Software arbeitet wie erwartet.

2.9 Relation zwischen zwei Symptomen, zwischen denen bereits eine Relation existiert

Der Ersteller versucht jeweils eine verstärkende, eine unbekannte und eine abschwächende Relation.

2.9.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 21.02.2019 um 18:12
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 19:25
Test negativ.

2.9.2 Fazit

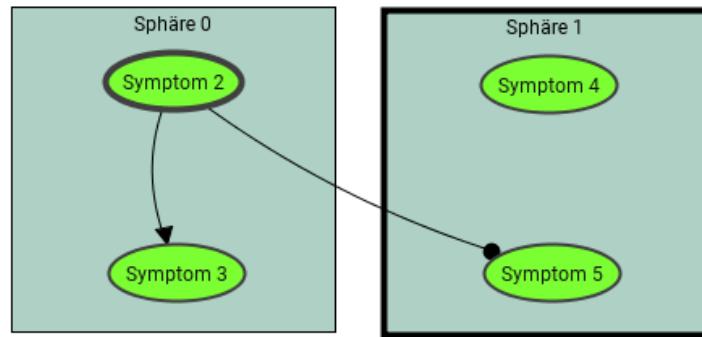
Die Software arbeitet wie erwartet. Die Abbildung ist identisch zur vorherigen.

2.10 Abschwächende Relation zwischen zwei Symptomen unterschiedlicher Sphären

Der Ersteller zieht eine abschwächende Relationen von Symptom 2 nach Symptom 5.

2.10.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 21.02.2019 um 17:39
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 19:27
Test negativ.



2.10.2 Fazit

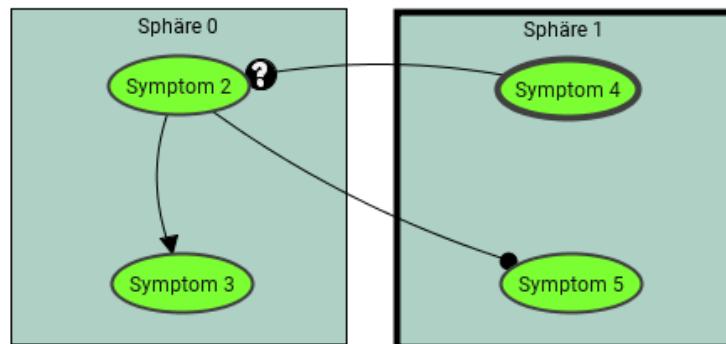
Die Software arbeitet wie erwartet.

2.11 Unbekannte Relation zwischen zwei Symptomen unterschiedlicher Sphären

Der Ersteller zieht eine unbekannte Relationen von Symptom 4 nach Symptom 2.

2.11.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 21.02.2019 um 18:01
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 19:29
Test negativ.



2.11.2 Fazit

Die Software arbeitet wie erwartet.

2.12 Einstellen der Vorlage-Regeln

Der Ersteller öffnet in der rechten Seitenleiste die Vorlage Regeln. Er stellt ein, dass es nur maximal drei Sphären, zehn Symptome und zehn Relationen geben darf, sowie dass unbekannte und abschwächende Relationen nicht erlaubt sind.

Anschließend stellt er ein, dass der Titel sowie der Style von **Sphäre 0** nicht verändert werden darf und die Position von **Sphäre 1** nicht verändert werden darf. Ebenso dürfen **Sphäre 1** keine Symptome hinzugefügt werden und keine Symptome von ihr gelöscht werden. Trotzdem stellt der Ersteller ein, das maximal 4 Symptome in **Sphäre 0** und maximal 3 Symptome in **Sphäre 1** erlaubt sind.

Bezüglich der Symptome wird eingestellt, dass der Titel von **Symptom 5**, die Position von **Symptom 3** und der Style von **Symptom 2** nicht verändert werden dürfen.

Abschließend stellt er noch ein, dass die Position der Relation von **Symptom 2** nach **Symptom 5**, der Style der Relation von **Symptom 4** nach **Symptom 2** und die Relationsart der Relation von **Symptom 2** nach **Symptom 3** nicht verändert werden dürfen.

2.12.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 21.02.2019 um 18:01

Test negativ.

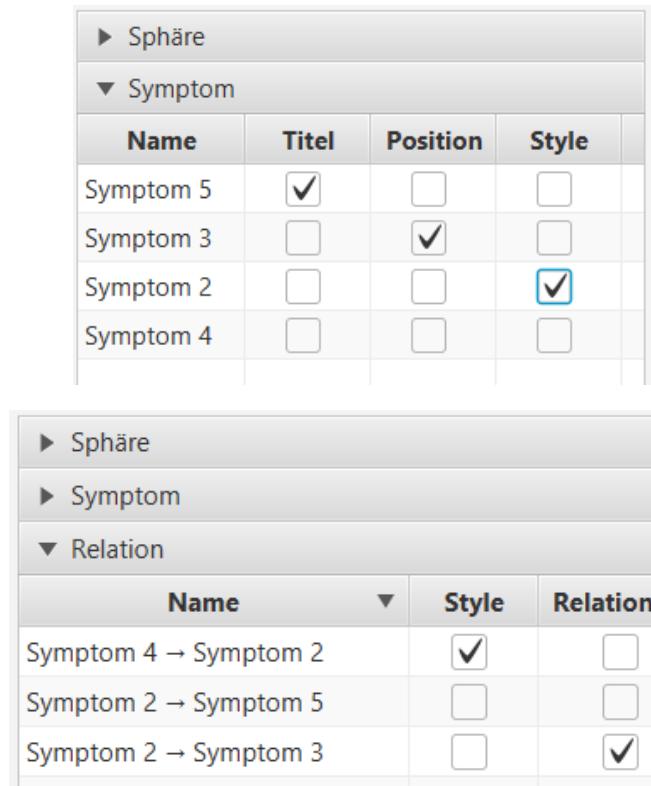
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 19:34

Test negativ.

▼ Vorlage-Regeln

Maximale Sphären:	3
Maximale Symptome:	10
Maximale Relationen:	10
<input checked="" type="checkbox"/> Verstärkende Relationen erlaubt	
<input type="checkbox"/> Abschwächende Relationen erlaubt	
<input type="checkbox"/> Unbekannte Relationen erlaubt	

▼ Sphäre						
Name	Titel	Position	Style	Löschen/Hinzufügen von Symptomen	Maximale Menge von Symptome	
Sphäre 0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	4	
Sphäre 1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	3	



Name	Titel	Position	Style
Symptom 5	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Symptom 3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Symptom 2	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Symptom 4	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Name	Style	Relationsart
Symptom 4 → Symptom 2	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Symptom 2 → Symptom 5	<input type="checkbox"/>	<input type="checkbox"/>
Symptom 2 → Symptom 3	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Da der Erzeuger gegen seine eigene Regeln verstößen kann, müssen entsprechende Funktionalitäten im Bearbeiter-Modus getestet werden.

2.12.2 Fazit

Die Software arbeitet wie erwartet.

2.13 Schließen des Programms ohne den Graphen zu speichern ☐☐☐

Der Ersteller schließt das Programm ohne vorher seinen Graphen zu speichern. Anschließend startet er das Programm erneut. Hierbei ist zu erwarten, dass der Graph identisch zu dem vor dem Schließen des Programms ist.

2.13.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 21.02.2019 um 18:42

Test negativ.

2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 14:38

Test negativ.

2.13.2 Fazit

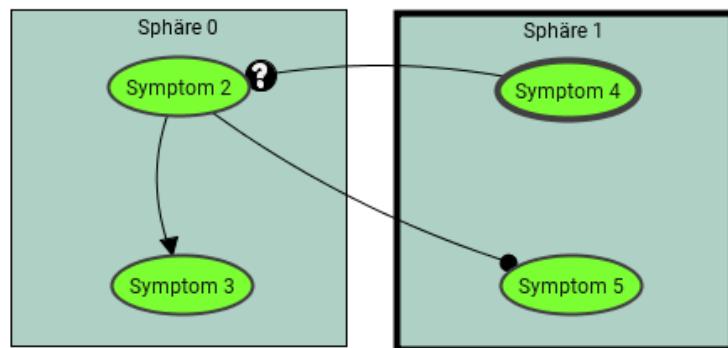
Die Software arbeitet wie erwartet.

2.14 Verstoß gegen die eigenen Vorlage-Regeln ☑☐☐

Der Ersteller versucht Sphäre 1 zu bewegen. Obwohl das Bewegen dieser Sphäre in den Regeln verboten wird, sollte diese Aktion erfolgreich sein, weil der Ersteller gegen seine eigenen Regeln verstößen darf. Analog zu diesem Testfall sollte der Tester versuchen auch gegen die anderen in den Vorlage-Regeln festgelegten Einschränkungen zu verstößen.

2.14.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 21.02.2019 um 18:52
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 19:43
Test negativ.



2.14.2 Fazit

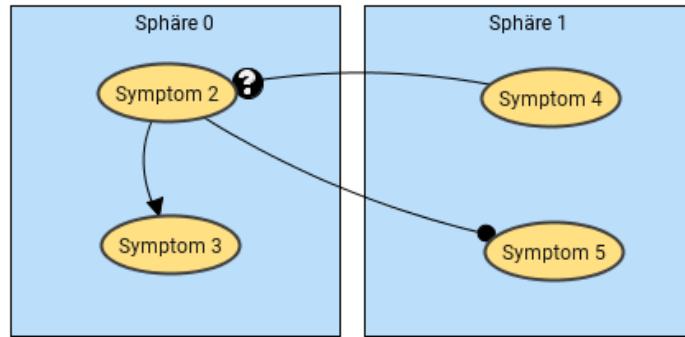
Die Software arbeitet wie erwartet.

2.15 Entfernen einer Relation im Ersteller-Modus bei der in den Vorlage-Regeln das Verändern des Styles verboten wurde ☑☐☐

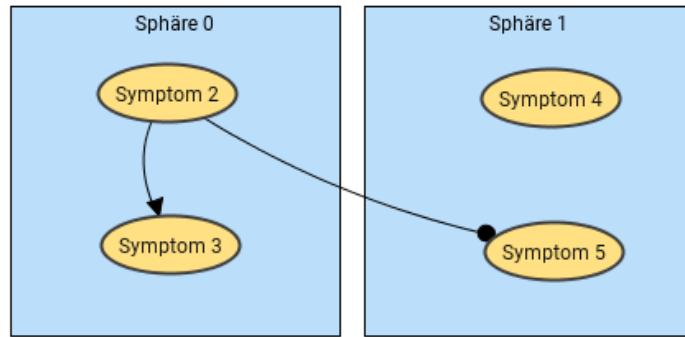
Der Ersteller versucht die Relation von Symptom 4 nach Symptom 2 zu entfernen. Obwohl dies aufgrund der Vorlage-Regeln verboten wird, ist zu erwarten, dass die Relation entfernt wird, weil die Regeln nicht im Ersteller-Modus gelten sollen.

2.15.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 22.02.2019 um 13:56
Test positiv, weil das Entfernen der Relation nicht erfolgreich war. Die Abbildung ist identisch zur vorherigen.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 19:43
Test negativ.
Vor dem Entfernen der Relation:



Nach dem Entfernen der Relation:



2.15.2 Fazit

Die Abfrage des jetzigen Modus entsprach nicht dem geplanten. Nachdem der Boolesche Wert negiert worden ist, funktioniert dies nun wie erwartet.

2.16 Speichern der Datei in den eigenen Dateien

Der Ersteller speichert über Datei und Speichern unter die soeben erstellte Vorlage inklusive der Regeln.

2.16.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 21.02.2019 um 19:00

Test negativ.

2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 19:45

Test negativ.

2.16.2 Fazit

Die Software arbeitet wie erwartet. In dem Verzeichnis das zum Speichern ausgewählt wurde ist die entsprechende Datei zu finden.

3 Bearbeiter-Modus

Im Ersteller-Modus getestete Funktionalitäten, die auch im Bearbeiter-Modus zur Verfügung stehen, werden im Bearbeiter-Modus nicht nochmal explizit getestet.

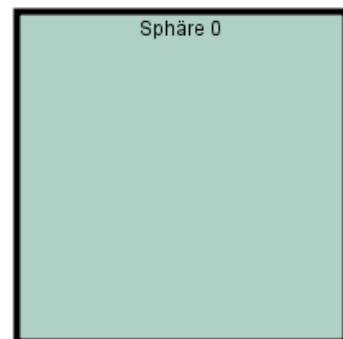
3.1 Rückgängig machen einer Aktion die im Ersteller Modus durchgeführt wurde und in den Vorlage-Regeln verboten wurde.....☒□

Hierfür wird vorerst im Ersteller-Modus eine Sphäre erstellt und in den Vorlage Regeln festgelegt das sie nicht bewegt werden kann (analog zu Testfall 1.10). Anschließend wird die Sphäre im Ersteller-Modus bewegt und direkt in den Bearbeiter-Modus gewechselt, wo der Button zum Rückgängigmachen der letzten Aktion betätigt wird. Hierbei ist zu erwarten, dass das Bewegen der Sphäre nicht rückgängig gemacht wird, weil das Bewegen dieser Sphäre in den Vorlage-Regeln verboten wurde.

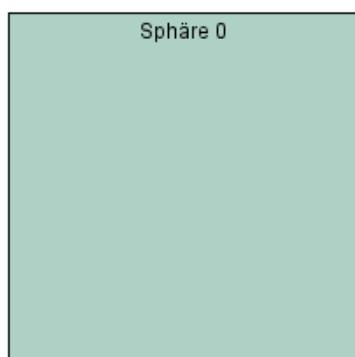
3.1.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 22.02.2019 um 11:39
Test positiv, weil die Sphäre beim Betätigen des Buttons zum Rückgängigmachen bewegt wurde.

Vorher:



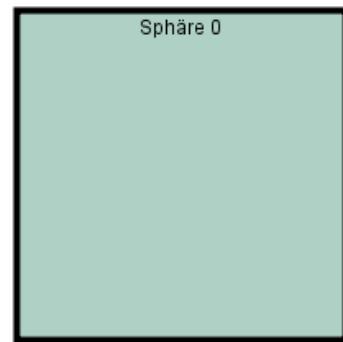
Nachher:



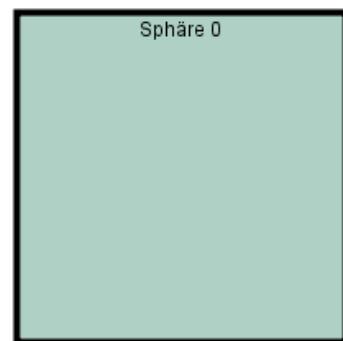
2.Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 19:47

Test negativ.

Vorher:



Nachher:



3.1.2 Fazit

Nach dem ersten Test wurde geändert, dass nach jeden Wechsel des Modus die **Action History** geleert wird. Nach dem zweiten Test arbeitet die Software wie erwartet.

3.2 Datei Öffnen.....

Der Ersteller drückt auf **Datei** und **Datei öffnen** und wählt im entsprechendem Verzeichnis die im Ersteller-Modus gespeicherte Datei aus. Zu erwarten ist, dass der Graph identisch zum im Ersteller-Modus erstellten ist.

3.2.1 Notizen

1.Test: Getestet von: Anthony Mendil. Datum: 22.02.2019 um 10:32

Test negativ.

2.Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 19:49

Test negativ.

3.2.2 Fazit

Die Software arbeitet wie erwartet.

3.3 Keinen Verlauf im Ersteller-Modus erstellen ☐☐☐

Im Ersteller-Modus soll der Verlauf bei der Erstellung des Graphen nicht aufgezeichnet werden. Deshalb ist im Bearbeiter-Modus, sofern man einen soeben erstellten Graphen geladen hat, zu erwarten, dass in der Seitenleiste der Verlauf noch keine Einträge hat.

3.3.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 22.02.2019 um 10:32
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 19:52
Test negativ.



3.3.2 Fazit

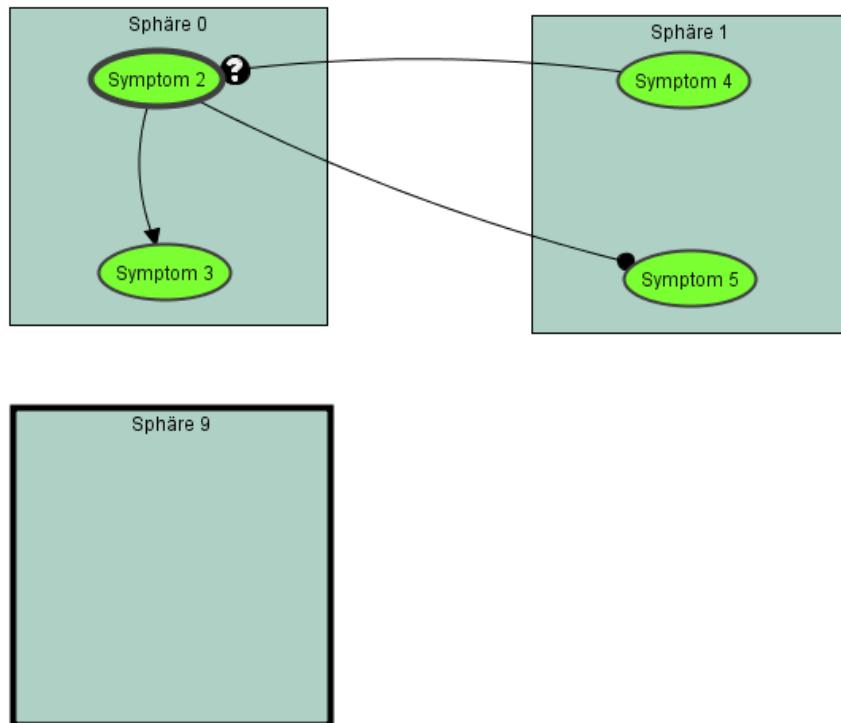
Die Software arbeitet wie erwartet.

3.4 Mehr Sphären hinzufügen als in den Vorlage-Regeln erlaubt ☐☐☐

Der Bearbeiter setzt zuerst eine Sphäre links unter Sphäre 0 und dann eine Sphäre rechts unter Sphäre 1. Hierbei wird erwartet, dass das Hinzufügen der ersten Sphäre erfolgreich ist, das der zweiten jedoch nicht, weil sonst die Anzahl an maximalen Sphären in den Vorlage-Regeln überschritten wird.

3.4.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 22.02.2019 um 14:18
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 19:53
Test negativ.



3.4.2 Fazit

Die Software arbeitet wie erwartet.

3.5 Abschwächende Relation hinzufügen obwohl es in den Vorlage-Regeln verboten wird.....☒☒☒

Der Bearbeiter wählt den Relationstyp abschwächend aus und versucht von Symptom 3 nach Symptom 4 eine Relation zu ziehen. Zu erwarten ist, dass dies aufgrund der Vorlage-Regeln nicht erlaubt wird.

3.5.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 22.02.2019 um 15:06
Test negativ. Die Abbildung ist identisch zur vorherigen.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 19:55
Test negativ. Die Abbildung ist identisch zur vorherigen.

3.5.2 Fazit

Die Software arbeitet wie erwartet.

3.6 Einer Sphäre ein Symptom hinzufügen obwohl dies durch die Vorlage-Regeln verboten wird

Der Bearbeiter versucht Sphäre 1 rechts mittig ein Symptom hinzuzufügen. Hierbei wird erwartet, dass dies aufgrund der Vorlage-Regeln nicht erlaubt wird.

3.6.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 22.02.2019 um 16:08
Test negativ. Die Abbildung ist identisch zur vorherigen.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 20:01
Test negativ. Die Abbildung ist identisch zur vorherigen.

3.6.2 Fazit

Die Software arbeitet wie erwartet.

3.7 Titel einer Sphäre ändern bei der es in den Vorlage-Regeln verboten wird

Der Bearbeiter versucht den Titel von Sphäre 0 zu ändern. Hierbei wird erwartet, dass dies aufgrund der Vorlage-Regeln nicht erlaubt wird.

3.7.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 22.02.2019 um 16:13
Test negativ. Die Abbildung ist identisch zur vorherigen.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 20:02
Test negativ. Die Abbildung ist identisch zur vorherigen.

3.7.2 Fazit

Die Software arbeitet wie erwartet.

3.8 Style eines Symptoms ändern bei dem es in den Vorlage-Regeln verboten wird

Der Bearbeiter versucht die Füllfarbe von Symptom 2 zu ändern. Hierbei wird erwartet, dass dies aufgrund der Vorlage-Regeln nicht erlaubt wird.

3.8.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 22.02.2019 um 16:38
Test negativ. Die Abbildung ist identisch zur vorherigen.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 20:04
Test negativ. Die Abbildung ist identisch zur vorherigen.

3.8.2 Fazit

Die Software arbeitet wie erwartet.

3.9 Relationsart einer Relation ändern bei der es in den Vorlage-Regeln verboten wird.....

Der Bearbeiter versucht die Relationsart der Relation von **Symptom 2** nach **Symptom 3** zu einer verstärkenden Relation zu ändern. Hierbei wird erwartet, dass dies aufgrund der Vorlage-Regeln nicht erlaubt wird.

3.9.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 22.02.2019 um 16:58
Test negativ. Die Abbildung ist identisch zur vorherigen.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 20:05
Test negativ. Die Abbildung ist identisch zur vorherigen.

3.9.2 Fazit

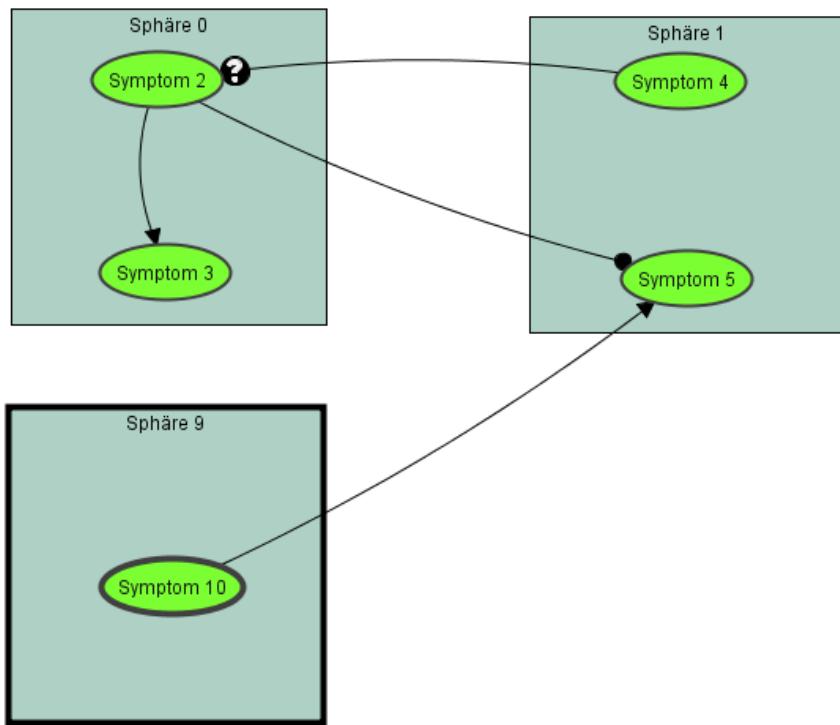
Die Software arbeitet wie erwartet.

3.10 Entfernen einer Sphäre mit Symptomen und Relationen.....

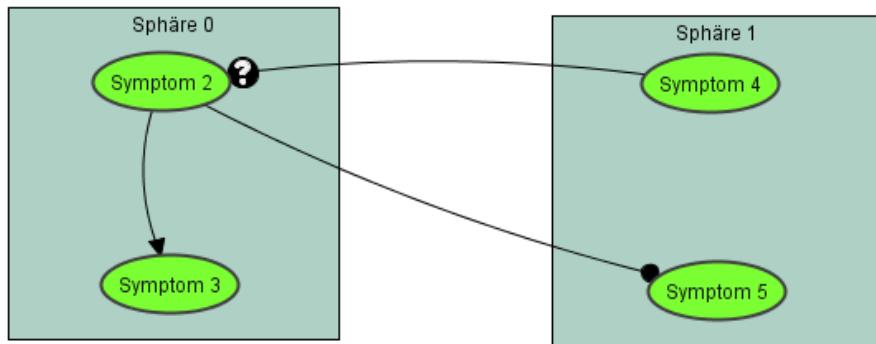
Der Bearbeiter fügt vorerst in **Sphäre 9** ein Symptom hinzu und zieht von ihm eine verstärkende Relation zu **Symptom 5**. Anschließend entfernt er **Sphäre 9**. Es ist zu erwarten, dass mit der Sphären auch alle enthaltenden Symptome und die Relationen der Symptome gelöscht werden.

3.10.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 22.02.2019 um 17:18
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 20:06
Test negativ.
Nach Hinzufügen der beschriebenen Elemente:



Nach Löschen der Sphäre:



3.10.2 Fazit

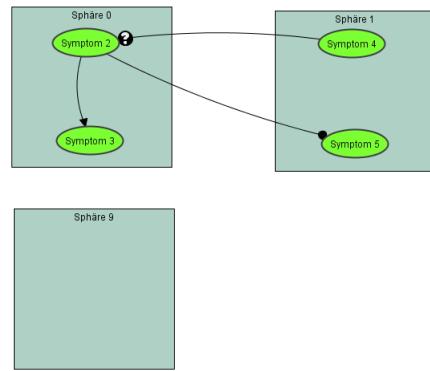
Die Software arbeitet wie erwartet.

3.11 Rückgängig machen des Entfernen einer Sphäre mit Symptomen und Relationen ☒□□

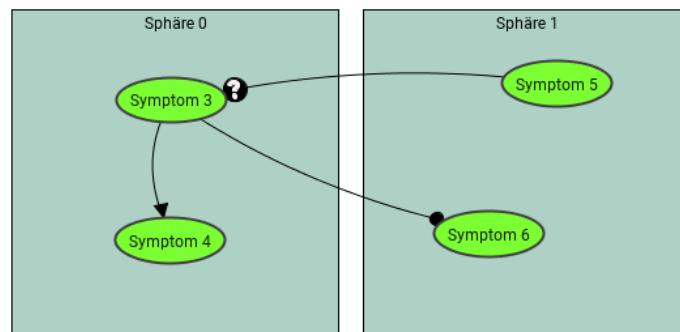
Der Bearbeiter betätigt den Button zum Rückgängigmachen der letzten Aktion. Hierbei wird erwartet, dass die vorherige Sphäre, inklusive der ursprünglichen Symptome und Relationen wieder hinzugefügt wird.

3.11.1 Notizen

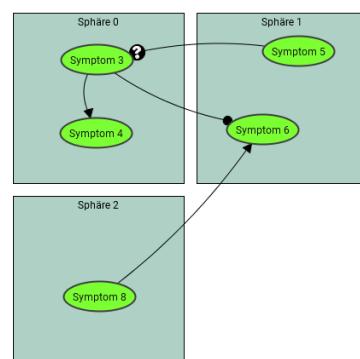
1. Test: Getestet von: Anthony Mendil. Datum: 22.02.2019 um 17:48
Test positiv, weil die ursprünglichen Symptome und Relationen nicht wieder hinzugefügt wurden.



2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 20:08
Test negativ.
Nach Entfernen der Sphäre:



Nach Rückgängigmachen des Entfernens der Sphäre mit Symptomen und Relationen:



3.11.2 Fazit

Die an die Sphären gebundenen Elemente wurden in der Implementierung aus Versehen nicht beachtet. Nachdem diese hinzugefügt worden sind, arbeitet die Software wie erwartet.

3.12 Vergrößern einer Sphäre

Der Bearbeiter betätigt vorerst den Button zum Auswählen von Elementen. Anschließend macht er einen Linksklick auf Sphäre 9 und betätigt den Button zum Vergrößern der Sphäre. Es wird erwartet, dass entsprechende Änderung auch statt findet.

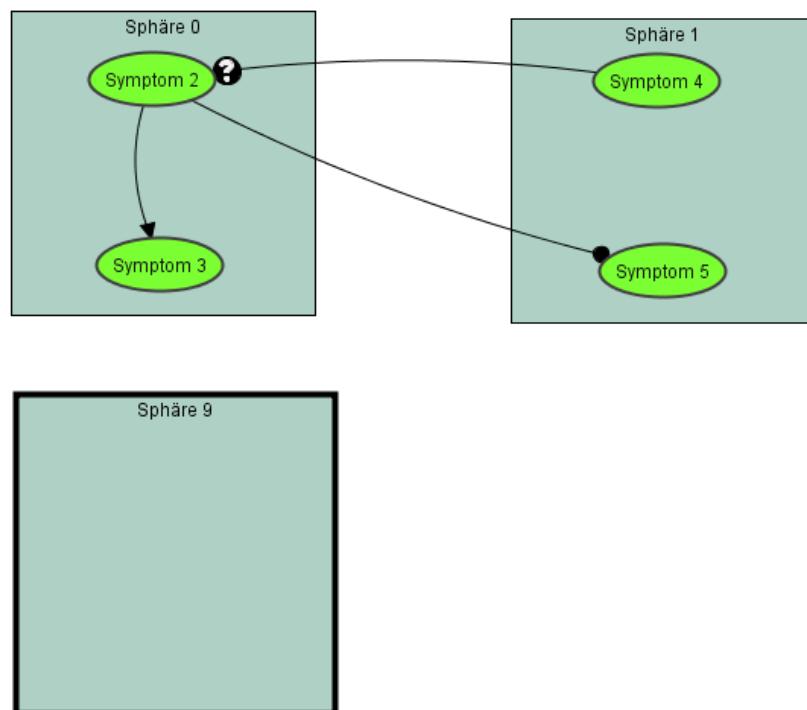
3.12.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 22.02.2019 um 18:28

Test negativ.

2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 20:10

Test negativ.



3.12.2 Fazit

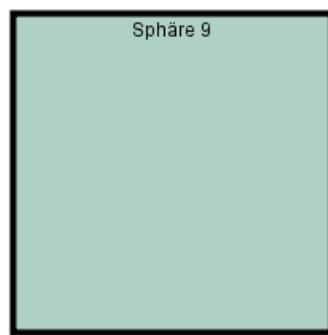
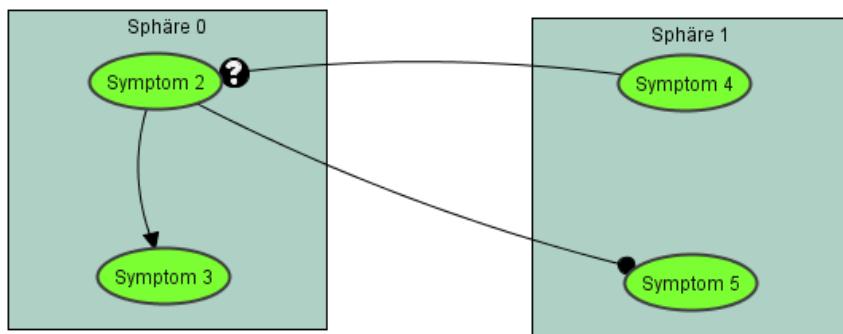
Die Software arbeitet wie erwartet.

3.13 Verkleinern einer Sphäre

Vorerst entfernt der Bearbeiter **Symptom 10**. Dann macht der Bearbeiter einen Linksklick auf **Sphäre 9** und betätigt den Button zum Verkleinern der Sphäre. Es wird erwartet, dass entsprechende Änderung auch statt findet.

3.13.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 22.02.2019 um 18:37
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 20:11
Test negativ.



3.13.2 Fazit

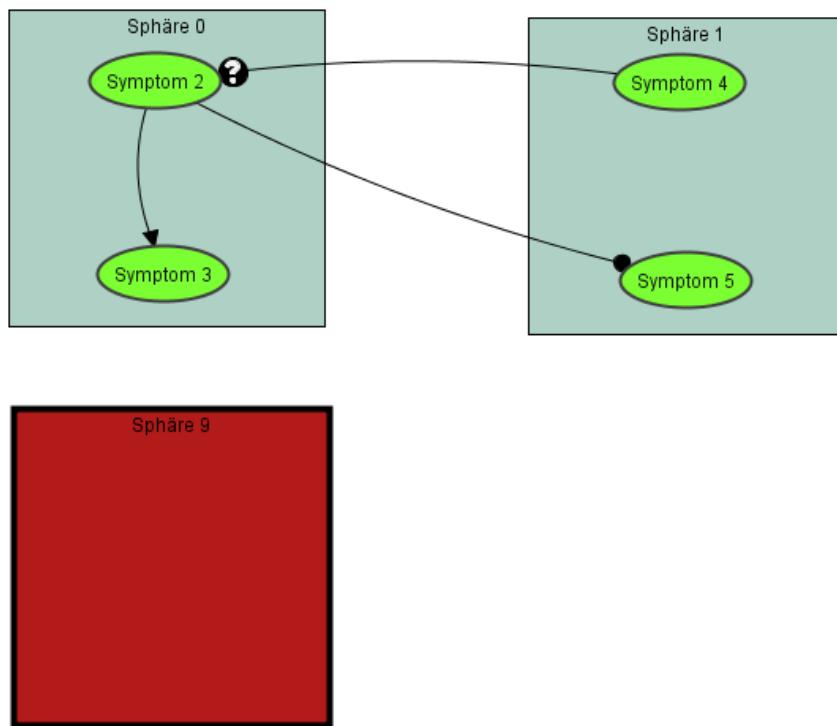
Die Software arbeitet wie erwartet.

3.14 Ändern der Farbe einer Sphäre

Der Bearbeiter macht einen Linksklick auf **Sphäre 9** und ändert die Farbe der Sphäre auf **Backstein** (dunkles rot). Es wird erwartet, dass die entsprechende Änderung auch statt findet.

3.14.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 22.02.2019 um 18:37
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 20:12
Test negativ.



3.14.2 Fazit

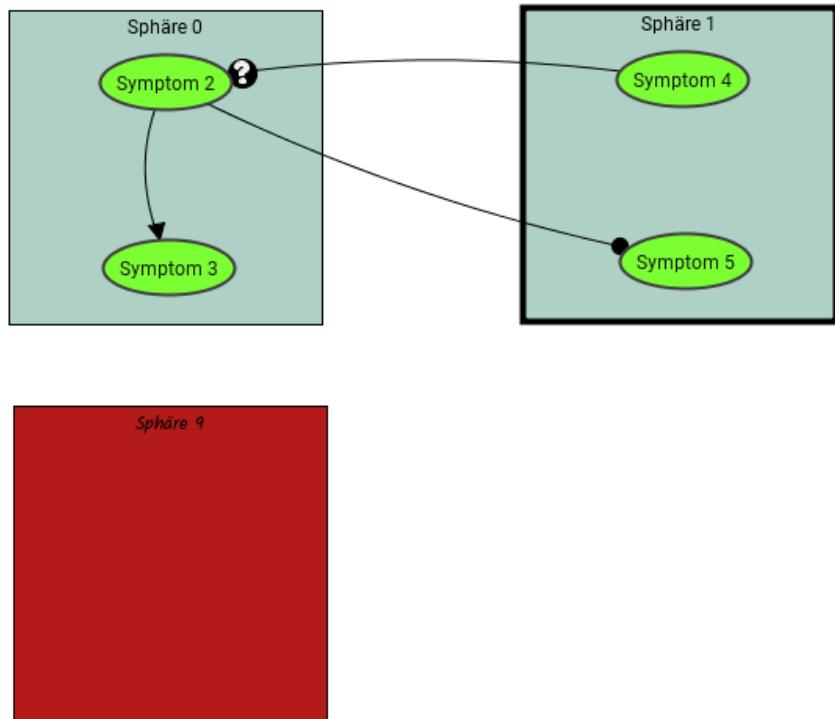
Die Software arbeitet wie erwartet.

3.15 Ändern der Schriftart einer Sphäre ☐☐□

Der Bearbeiter macht einen Linksklick auf **Sphäre 9** und ändert die Schriftart der Sphäre auf **Kalam**. Es wird erwartet, dass die entsprechende Änderung auch statt findet.

3.15.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 22.02.2019 um 18:37
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 20:13
Test negativ.



3.15.2 Fazit

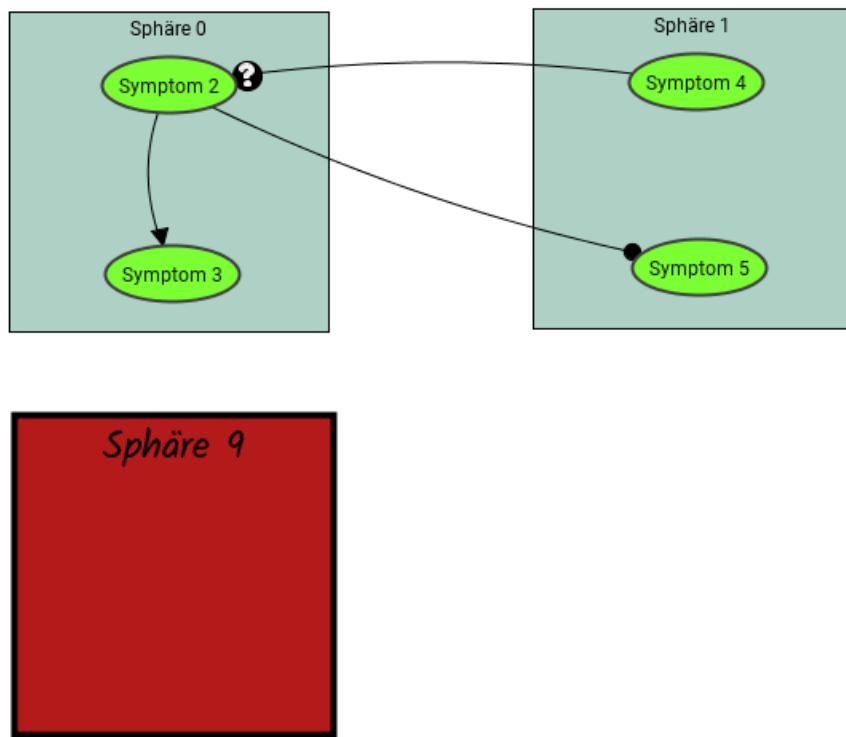
Die Software arbeitet wie erwartet.

3.16 Verändern der Schriftgröße einer Sphäre

Der Bearbeiter macht einen Linksklick auf Sphäre 9 und ändert die Schriftgröße der Sphäre auf 24. Es wird erwartet, dass die entsprechende Änderung auch statt findet.

3.16.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 22.02.2019 um 18:37
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 20:13
Test negativ.



3.16.2 Fazit

Die Software arbeitet wie erwartet.

3.17 Sphären automatisch anordnen obwohl eine Sphäre nicht bewegt werden darf.....

Der Bearbeiter versucht den Button zum automatischen Anordnen der Sphären zu betätigen. Es wird erwartet, dass dies nicht möglich ist, da das Bewegen von Sphäre 1 in den Vorlage-Regeln verboten wird.

3.17.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 22.02.2019 um 18:37
Test negativ. Die Abbildung ist identisch zur vorherigen.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 20:13
Test negativ. Die Abbildung ist identisch zur vorherigen.

3.17.2 Fazit

Die Software arbeitet wie erwartet.

3.18 Sphären automatisch anordnen

Vorerst wird in den Ersteller-Modus gewechselt um die Bewegung von **Sphäre 1** zu erlauben. Anschließend betätigt der Bearbeiter den Button zum automatischen Anordnen der Sphären. Es wird erwartet, dass die Sphären automatisch angeordnet werden und die Symptome und Relationen immer noch korrekt platziert sind.

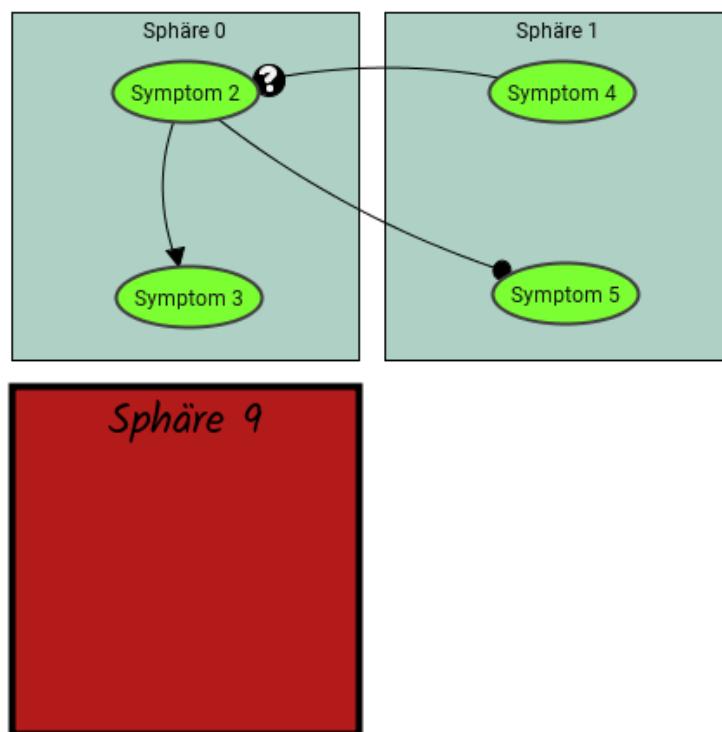
3.18.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 22.02.2019 um 18:37

Test negativ.

2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 20:13

Test negativ.



3.18.2 Fazit

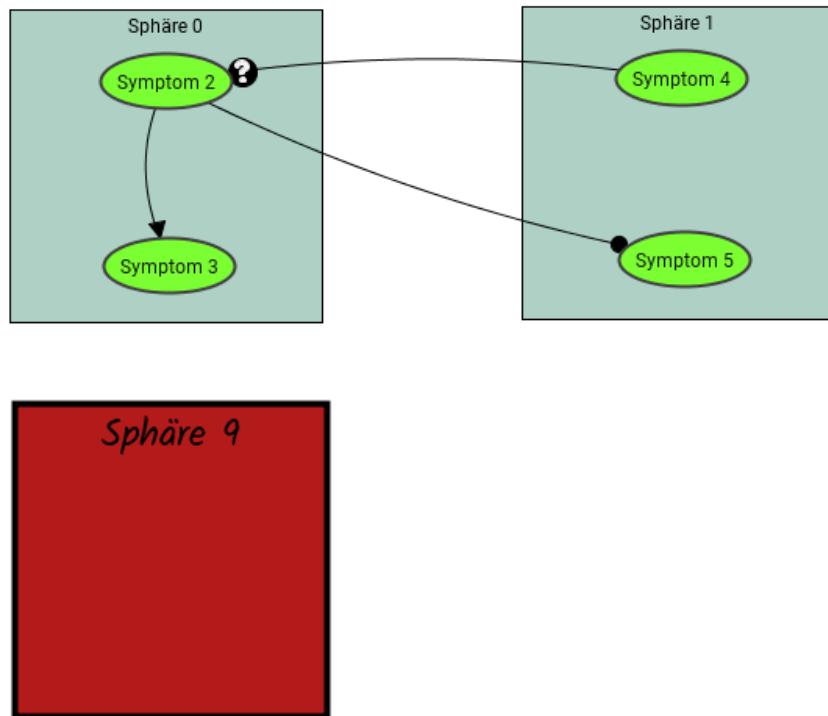
Die Software arbeitet wie erwartet.

3.19 Rückgängig machen der automatischen Anordnung von Sphären ..

Der Bearbeiter betätigt den Button zum Rückgängigmachen der letzten Aktion. Es wird erwartet, dass die Sphären mit ihren Knoten und den Relationen wieder an ihre vorherige Positionen bewegt werden.

3.19.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 22.02.2019 um 18:37
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 20:15
Test negativ.



3.19.2 Fazit

Die Software arbeitet wie erwartet.

3.20 Bewegen einer Sphäre auf eine andere ☐☐□

Der Bearbeiter versucht Sphäre 9 auf Sphäre 0 zu bewegen. Hierbei wird erwartet, dass Sphäre 9 auf ihre ursprüngliche Position zurück bewegt wird und Sphäre 0 unverändert bleibt.

3.20.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 22.02.2019 um 18:37
Test negativ. Die Abbildung ist identisch zur vorherigen.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 20:16
Test negativ. Die Abbildung ist identisch zur vorherigen.

3.20.2 Fazit

Die Software arbeitet wie erwartet.

Symptome

3.21 Entfernen eines Symptoms, das Relationen hat.....☒☒

Der Bearbeiter fügt vorerst in Sphäre 0 ein Symptom hinzu und zieht von ihm eine verstärkende Relation zu Symptom 5 und eine von Symptom 3 zum hinzugefügten Symptom. Anschließend entfernt er das soeben hinzugefügte Symptom nachdem er es mit einem Linksklick ausgewählt hat. Es ist zu erwarten, dass mit dem Symptom auch alle ausgehenden und eingehenden Relationen des gelöschten Symptoms entfernt werden.

3.21.1 Notizen

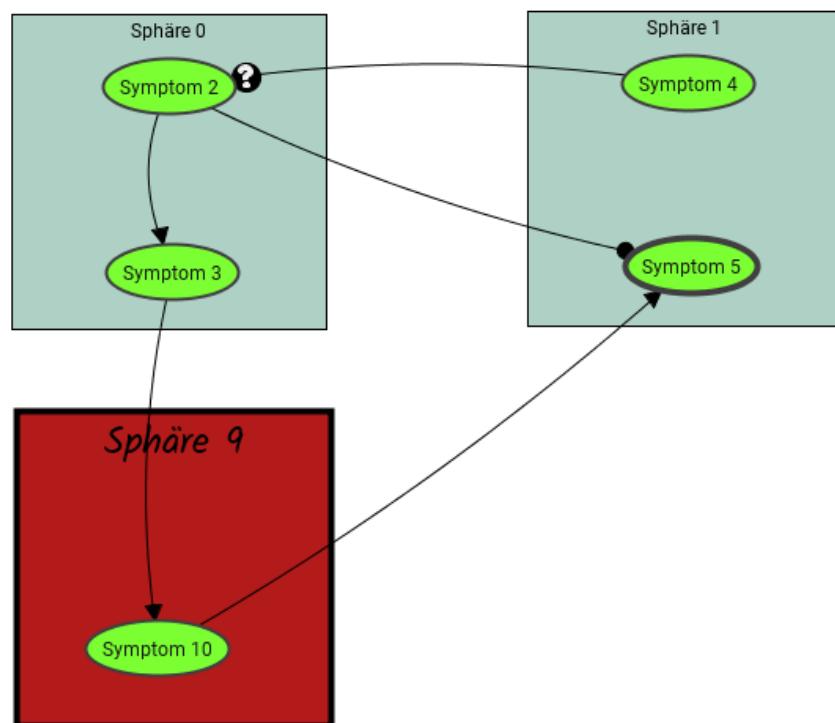
1. Test: Getestet von: Anthony Mendil. Datum: 23.02.2019 um 10:58

Test negativ.

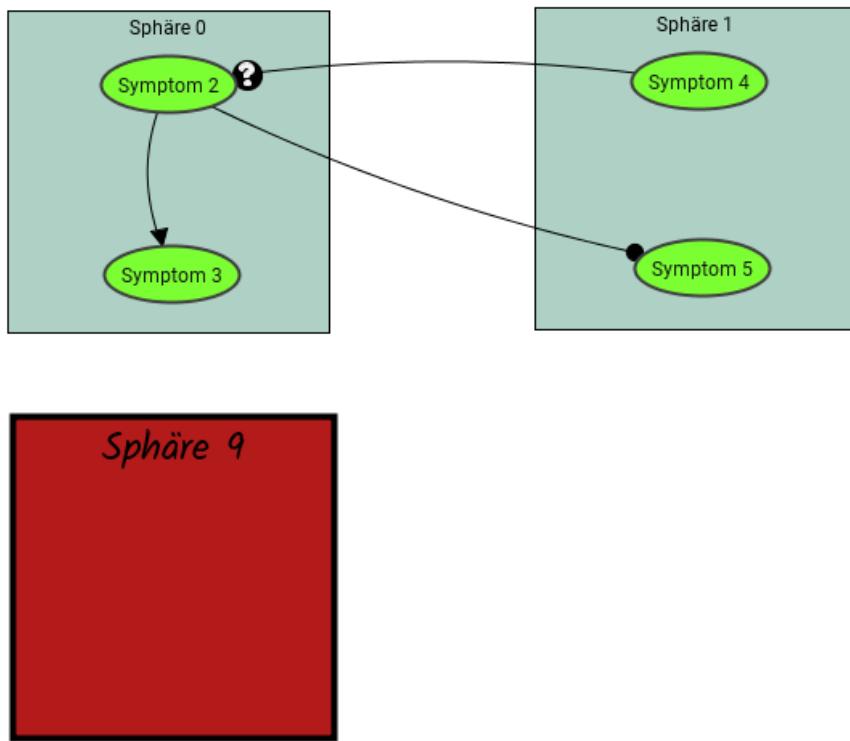
2. Test: Getestet von: Jacky Philipp Mach. Datum: 03.03.2019 um 20:18

Test negativ.

Nach hinzufügen der beschriebenen Elemente:



Nach löschen der Sphäre:



3.21.2 Fazit

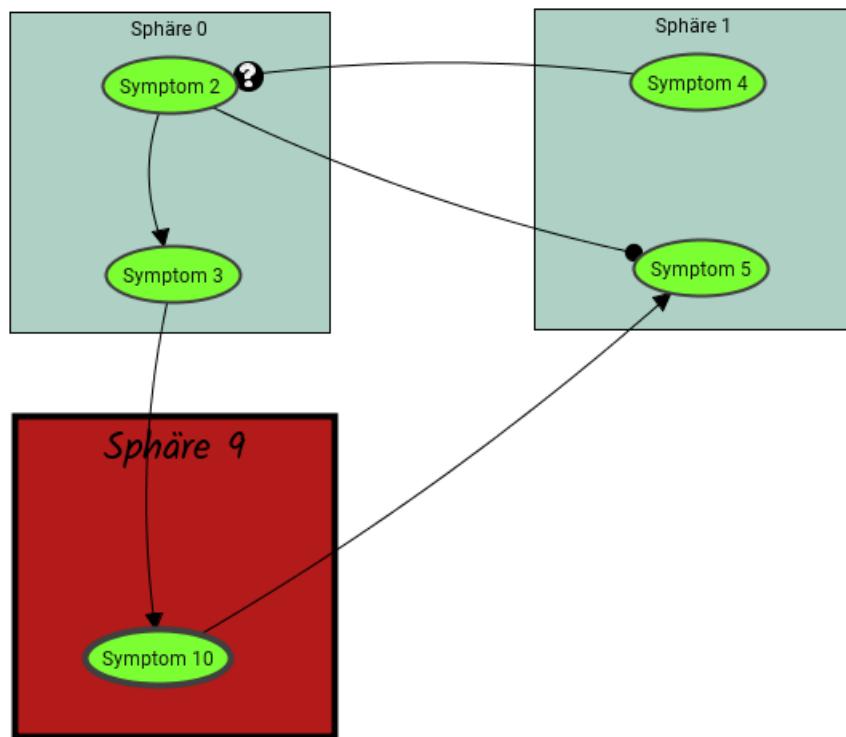
Die Software arbeitet wie erwartet.

3.22 Rückgängig machen des Entfernen eines Symptoms mit Relationen

Der Bearbeiter betätigt den Button zum Rückgängigmachen der letzten Aktion. Hierbei wird erwartet, dass das vorherige Symptom, inklusive der ursprünglichen Relationen wieder hinzugefügt wird.

3.22.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 23.02.2019 um 11:33
Test positiv, weil die ursprünglichen Symptome und Relationen nicht wieder hinzugefügt wurden. Der Graph verändert sich nicht.
2. Test: Getestet von: Anthony Mendil. Datum: 23.02.2019 um 20:18
Test negativ.



3.22.2 Fazit

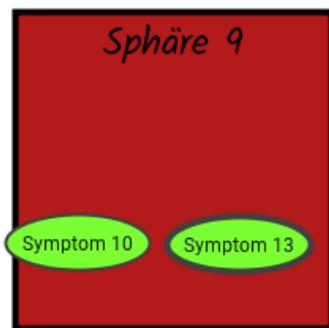
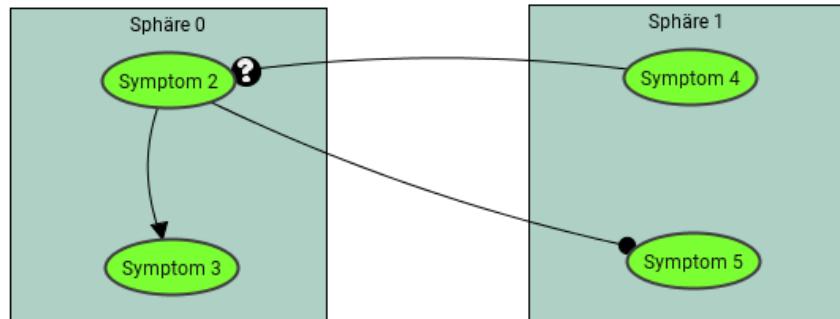
Die Relationen, welche eine eingehende oder ausgehende Relation des zu löschen Symptoms ist wurde in der Implementierung aus Versehen nicht beachtet. Nachdem die Semantik dazu implementiert wurde arbeitet die Software wie erwartet.

3.23 Entfernen mehrere Symptome ohne Relationen ☐□

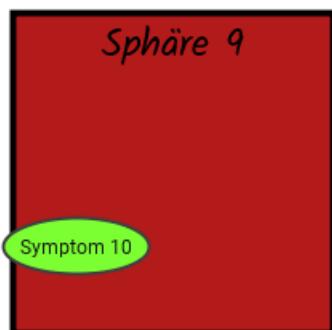
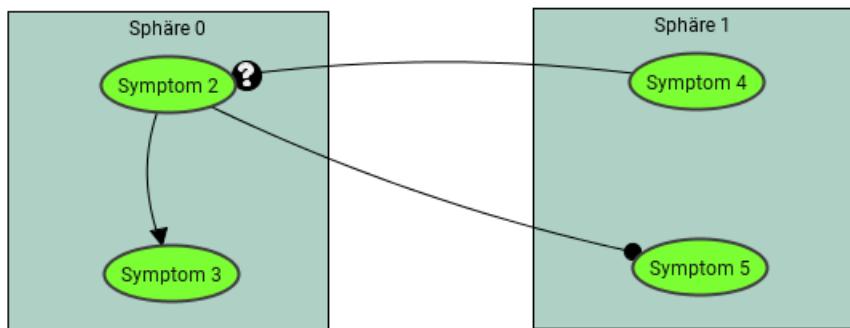
Zunächst entfernt der Bearbeiter alle Relationen von Symptom 10, verschiebt es ein wenig nach links und fügt Sphäre 9 ein weiteres Symptom rechts von Symptom 10 hinzu. Durch gedrückt halten von Umschalten und Linksklicks auf die soeben erwähnten Symptome wählt der Bearbeiter mehrere Symptome aus. Das rechte Symptom ist nach dem Hinzufügen bereits ausgewählt, weshalb auf die soeben beschriebene Weise nur noch das linke Symptom der Auswahl hinzugefügt werden muss. Anschließend betätigt er den Button zum Löschen der Symptome. Hierbei wird erwartet, dass alle ausgewählten Symptome gelöscht werden.

3.23.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 23.02.2019 um 11:39
Test positiv, weil nur eins der ausgewählten Symptome entfernt wurde.
Nach Hinzufügen der beiden Symptome:



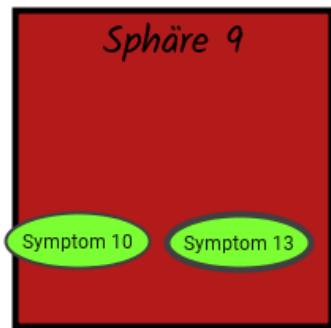
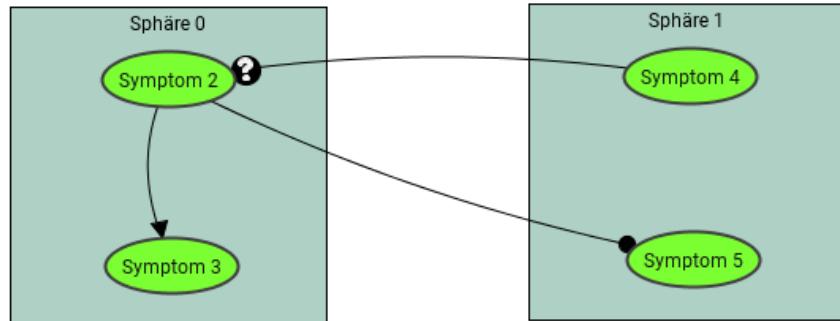
Nach Betätigen des Buttons zum Entfernen der Symptome:



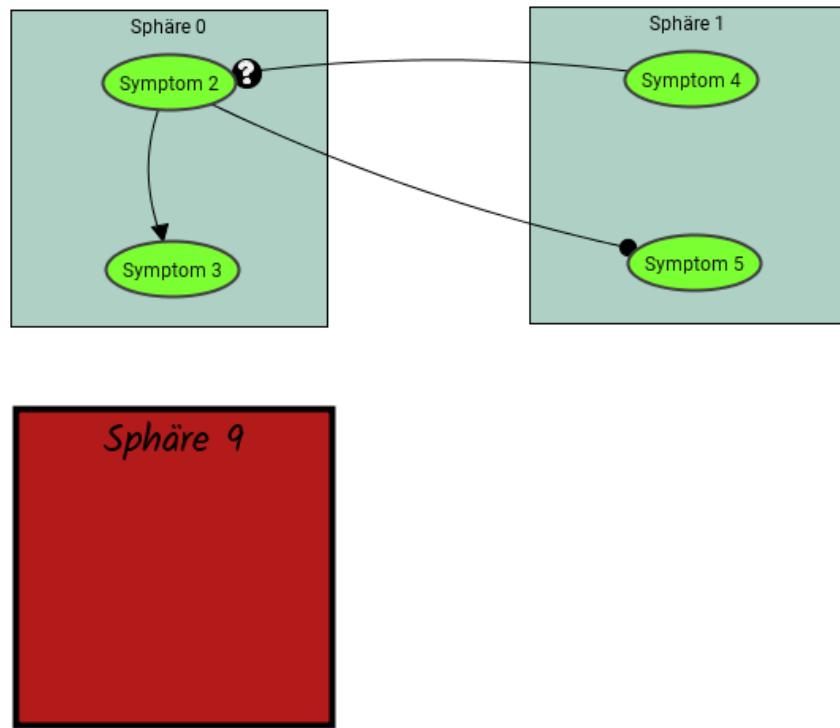
2. Test: Getestet von: Anthony Mendil. Datum: 23.02.2019 um 20:26

Test negativ.

Nach Hinzufügen der beiden Symptome:



Nach Betätigen des Buttons zum Entfernen der Symptome:



3.23.2 Fazit

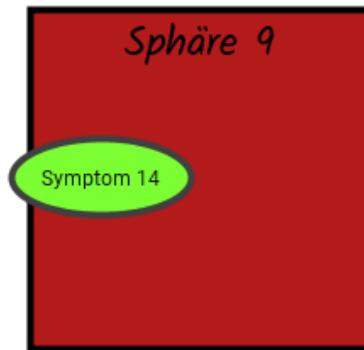
Die Menge der ausgewählten Symptome wurde in der jeweiligen Action nicht richtig iteriert. Nachdem dieser Fehler behoben wurde, arbeitet die Software nun wie erwartet.

3.24 Vergrößern eines Symptoms

Der Bearbeiter fügt links mittig in Sphäre 9 eine Symptom hinzu und betätigt den Button zum Vergrößern des Symptoms zehn mal. Hierbei wird erwartet, dass entsprechende Änderungen an dem Knoten vorgenommen werden.

3.24.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 23.02.2019 um 12:32
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 12:14
Test negativ.



3.24.2 Fazit

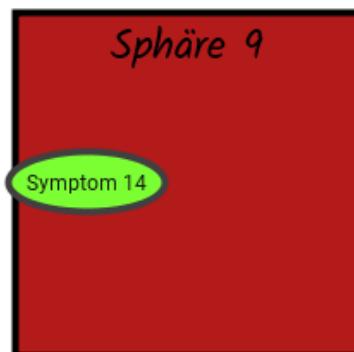
Die Software arbeitet wie erwartet.

3.25 Verkleinern eines Symptoms

Der Bearbeiter wählt Symptom 10 aus und betätigt den Button zum Verkleinern des Symptoms zehn mal. Hierbei wird erwartet, dass entsprechende Änderungen an dem Knoten vorgenommen werden.

3.25.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 23.02.2019 um 12:34
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 12:15
Test negativ.



3.25.2 Fazit

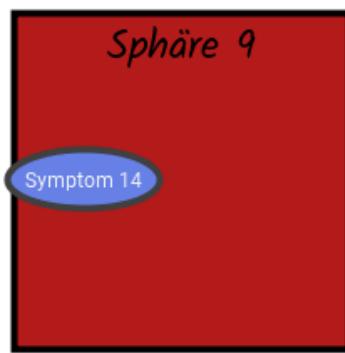
Die Software arbeitet wie erwartet.

3.26 Ändern der Füllfarbe eines Symptoms □□□

Der Bearbeiter wählt **Symptom 10** aus und ändert die Füllfarbe des Symptoms zu Kornblumenblau. Hierbei wird erwartet, dass entsprechende Änderungen an dem Knoten vorgenommen werden.

3.26.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 23.02.2019 um 12:36
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 12:15
Test negativ.



3.26.2 Fazit

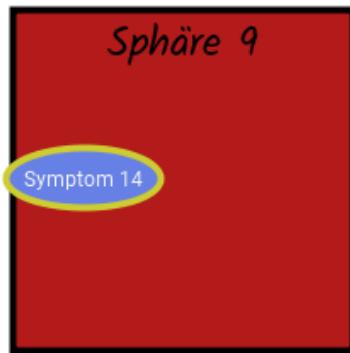
Die Software arbeitet wie erwartet.

3.27 Ändern der Randfarbe eines Symptoms □□□

Der Bearbeiter wählt **Symptom 10** aus und ändert die Randfarbe des Symptoms zu Goldgrüne. Hierbei wird erwartet, dass entsprechende Änderungen an dem Knoten vorgenommen werden.

3.27.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 23.02.2019 um 12:39
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 12:16
Test negativ.



3.27.2 Fazit

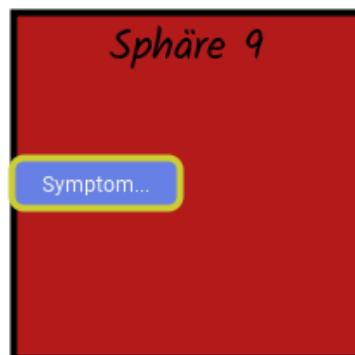
Die Software arbeitet wie erwartet.

3.28 Ändern der Form eines Symptoms

Der Bearbeiter wählt Symptom 10 aus und ändert die Form des Symptoms zu rechteckig. Hierbei wird erwartet, dass entsprechende Änderungen an dem Knoten vorgenommen werden.

3.28.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 23.02.2019 um 14:12
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 12:16
Test negativ.



3.28.2 Fazit

Die Software arbeitet wie erwartet.

3.29 Automatisches Anordnen der Symptome obwohl ein Symptom nicht bewegt werden darf.....

Der Bearbeiter versucht den Button zum automatischen Anordnen der Symptome zu betätigen. Hierbei wird erwartet, dass dies nicht erlaubt wird, da das Bewegen von Symptom 3 in den Vorlage-Regeln verboten wird.

3.29.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 23.02.2019 um 14:18
Test negativ. Die Abbildung ist identisch zur vorherigen.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 12:14
Test negativ. Die Abbildung ist identisch zur vorherigen.

3.29.2 Fazit

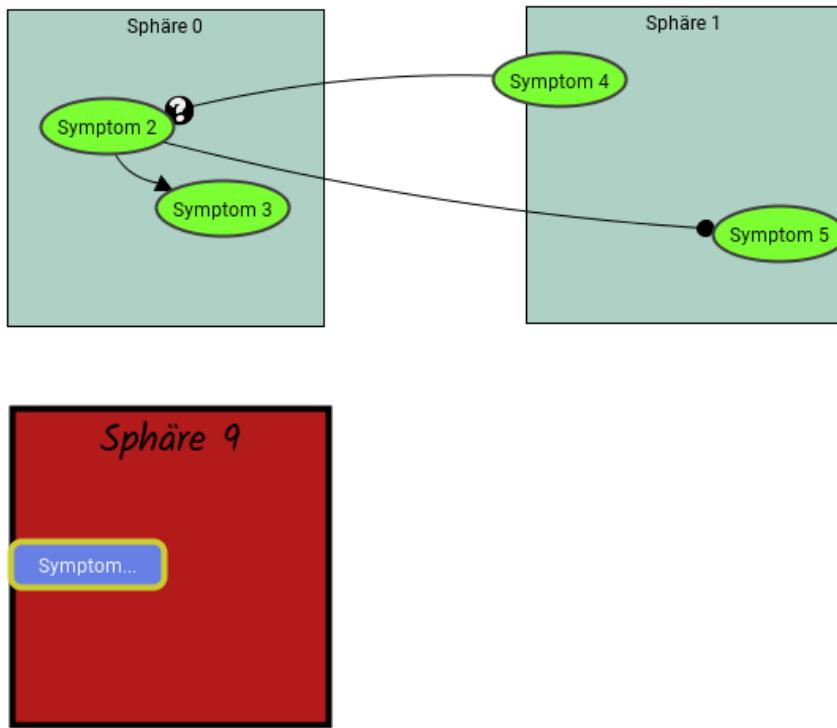
Die Software arbeitet wie erwartet.

3.30 Automatisches Anordnen der Symptome

Vorerst wird in den Ersteller-Modus gewechselt um das Bewegen von Symptom 3 zu erlauben. Anschließend wird wieder in den Bearbeiter-Modus gewechselt. Der Bearbeiter betätigt den Button zum automatischen Anordnen der Symptome. Hierbei wird erwartet, dass die Symptome in den Sphären automatisch angeordnet werden.

3.30.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 23.02.2019 um 14:18
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 12:14
Test negativ.



3.30.2 Fazit

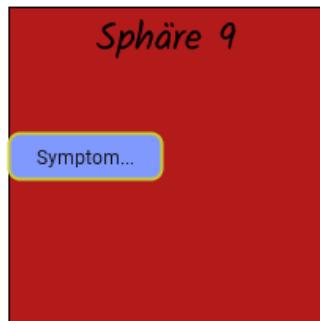
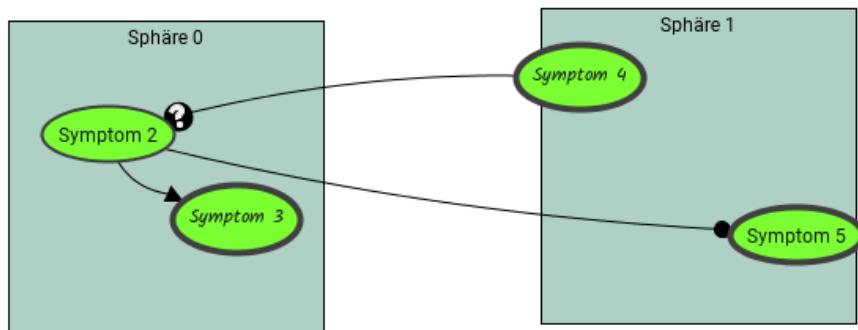
Die Software arbeitet wie erwartet.

3.31 Ändern der Schriftart mehrerer Symptome, wobei dies durch die Vorlage-Regeln nicht bei allen erlaubt wird

Der Bearbeiter wählt Symptom 2, Symptom 4 und Symptom 5 aus und ändert die Schriftart der Symptome zu Kalam. Hierbei wird erwartet, dass die Schriftart von Symptom 2 und Symptom 4 entsprechend verändert wird, während die Schriftart von Symptom 5 gleich bleibt, da deren Änderung in den Vorlage-Regeln verboten wurde (Schriftart wird mit unter Style aufgefasst).

3.31.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 23.02.2019 um 14:46
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 12:16
Test negativ.



3.31.2 Fazit

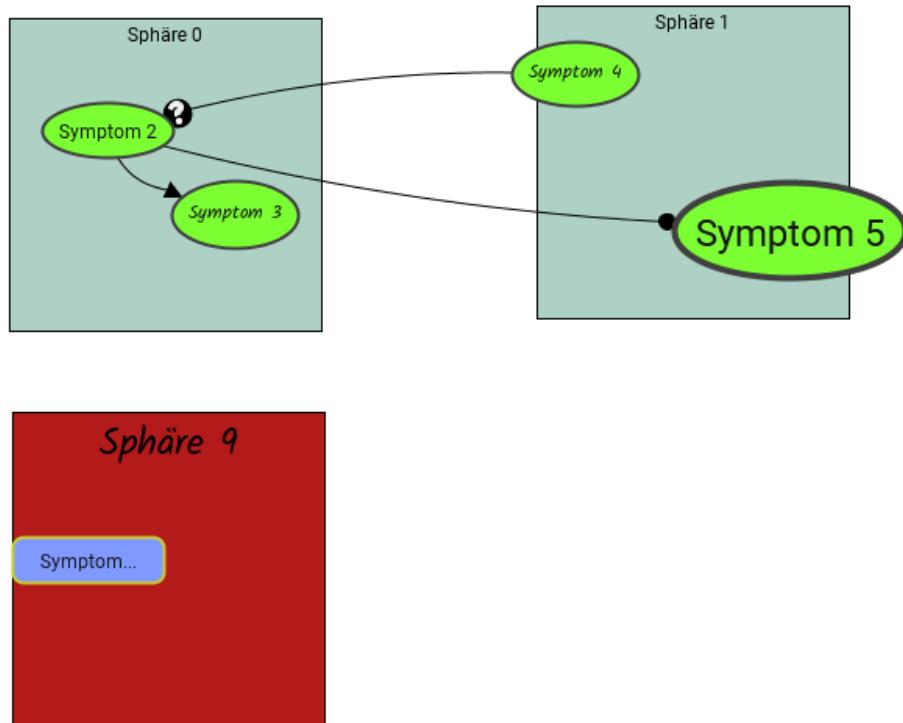
Die Software arbeitet wie erwartet.

3.32 Ändern der Schriftgröße eines Symptoms

Der Bearbeiter wählt Symptom 5 aus und ändert seine Schriftgröße zu 24. Hierbei wird erwartet, die Schrift des Symptoms größer wird und die Größe des Symptoms ebenfalls so wächst, dass die Beschriftung noch komplett angezeigt wird.

3.32.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 23.02.2019 um 14:50
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 12:19
Test negativ.



3.32.2 Fazit

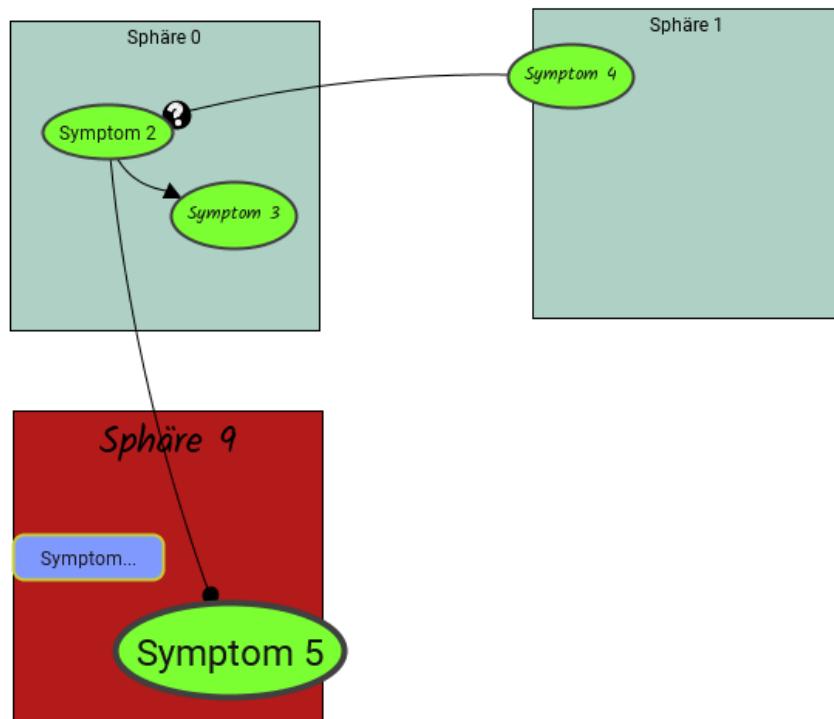
Die Software arbeitet wie erwartet.

3.33 Bewegen eines Symptoms in eine andere Sphäre

Der Bearbeiter bewegt Symptom 5 in Sphäre 9. Hierbei wird erwartet, dass das Symptom nach Sphäre 9 verschoben wird und die eingehende Relation des Symptoms mitwandert.

3.33.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 23.02.2019 um 14:33
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 12:22
Test negativ.



3.33.2 Fazit

Die Software arbeitet wie erwartet.

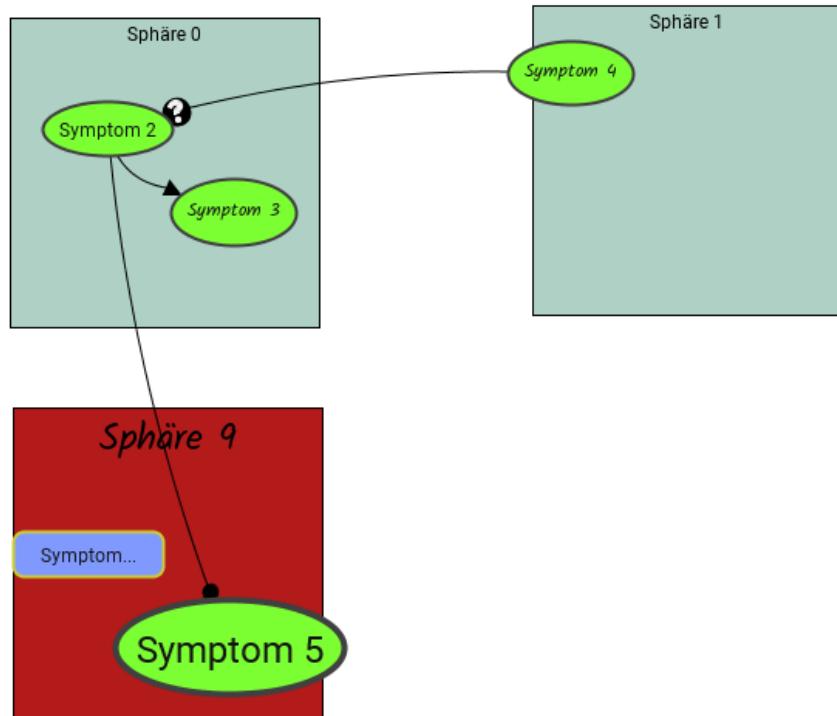
3.34 Bewegen eines Symptoms an eine Stelle an der keine Sphäre liegt.....



Der Bearbeiter bewegt Symptom 5 rechts neben Sphäre 9. Hierbei wird erwartet, dass das Symptom inklusive seiner eingehenden Relation wieder an ihre ursprünglichen Positionen zurück springen.

3.34.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 23.02.2019 um 14:37
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 12:22
Test negativ.



3.34.2 Fazit

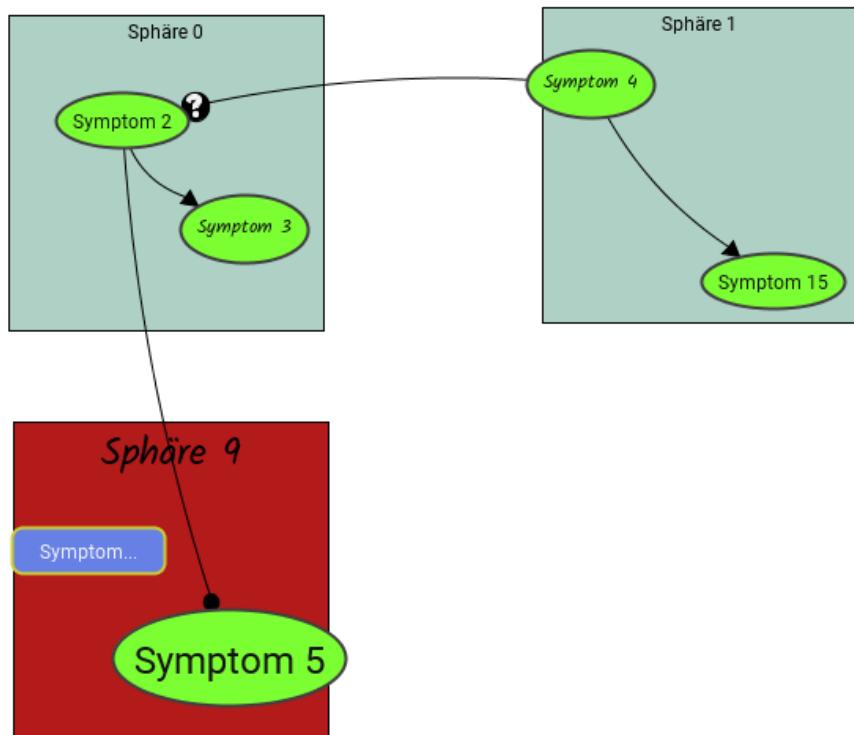
Die Software arbeitet wie erwartet.

3.35 Verändern der Farbe einer Relation ☐☐□

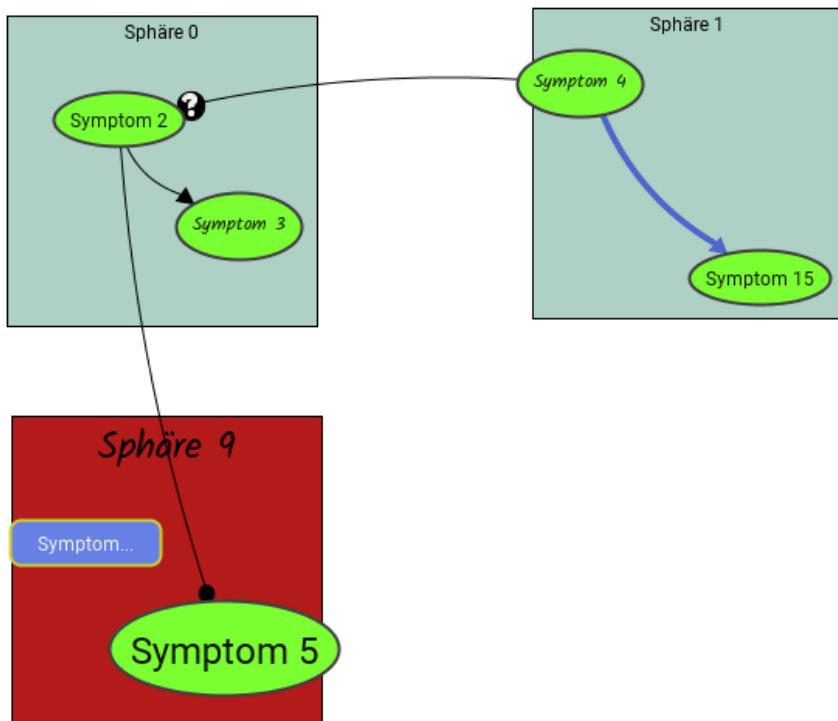
Vorerst fügt der Bearbeiter Sphäre 1 ein Symptom hinzu (Schriftart: Roboto) und zieht eine verstärkende Relation von Symptom 4 zum eben erstellten. Anschließend wählt er die Relation aus und ändert ihre Farbe zu Königsblau. Hierbei wird erwartet, dass entsprechende Änderung stattfindet.

3.35.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 24.02.2019 um 15:28
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 12:24
Test negativ.
Nach Hinzufügen des Symptoms und der Relation:



Nach Ändern der Farbe der Relation von Symptom 4 nach Symptom 15 :



3.35.2 Fazit

Die Software arbeitet wie erwartet.

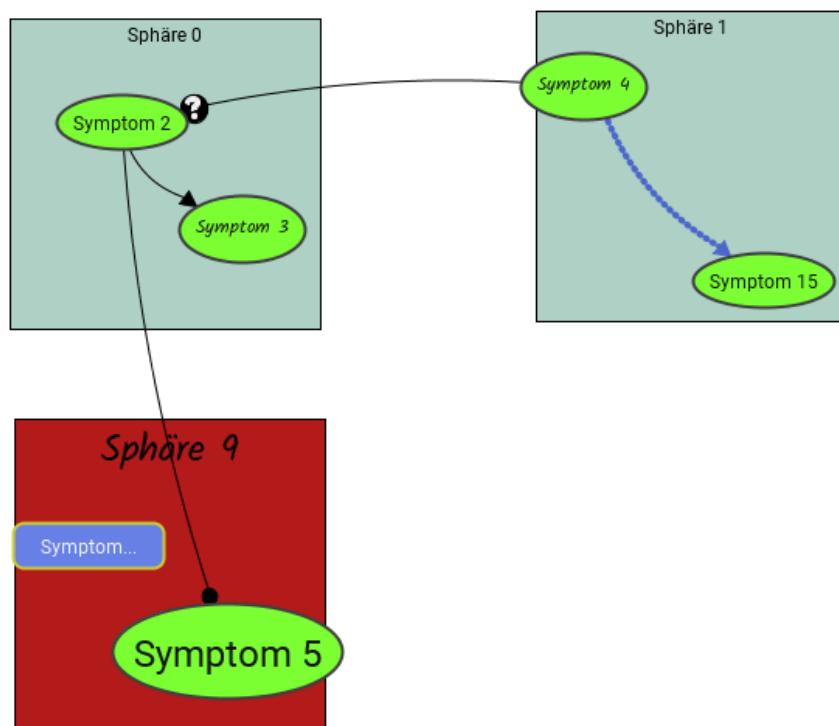
3.36 Verändern der Linienart einer Relation

Der Bearbeiter wählt die Relation von Symptom 4 nach Symptom 15 aus und ändert ihre Linienart zu gepunktet. Hierbei wird erwartet, dass entsprechende Änderung stattfindet.

3.36.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 24.02.2019 um 16:01
Test negativ.

2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 12:25
Test negativ.



3.36.2 Fazit

Die Software arbeitet wie erwartet.

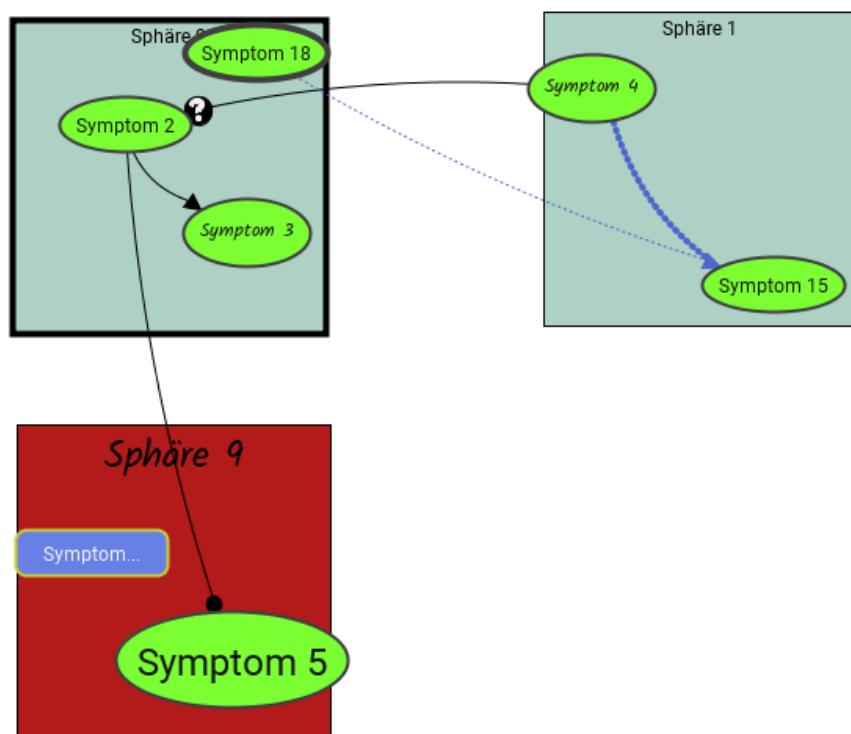
3.37 Zusammenfassung gleicher Pfeilspitzen

Der Bearbeiter fügt vorerst zu Sphäre 0 oben rechts ein Symptom hinzu und zieht von ihm aus eine verstärkende Relation zu Symptom 15. Hierbei wird erwartet, dass die

Pfeilspitzen der Relation mit der Relation von Symptom 18 nach Symptom 15 zusammengefasst wird, da es sich um die gleiche Pfeilspitze handelt.

3.37.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 24.02.2019 um 16:28
Test negativ.
 2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 12:28
Test negativ.
- Nach hinzufügen des Symptoms und der Relation:



3.37.2 Fazit

Die Software arbeitet wie erwartet.

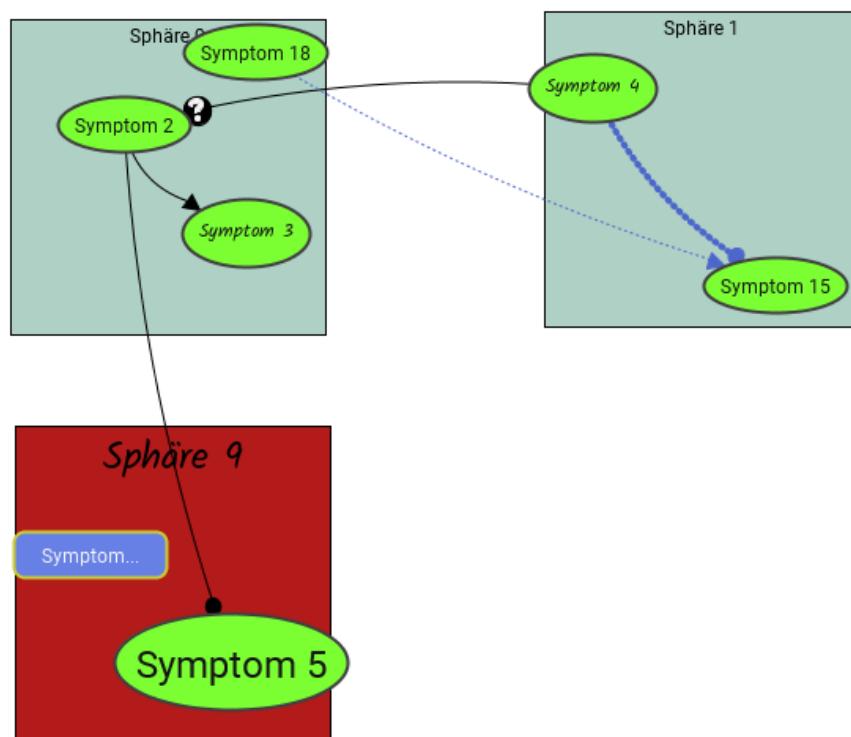
3.38 Verändern der Relationsart einer Relation

Damit folgende Aktion möglich ist, muss vorerst in den Ersteller-Modus gewechselt werden um abschwächende Relationen zu erlauben. Anschließend wählt der Bearbeiter die Relation von Symptom 4 nach Symptom 15 aus und ändert die Relationsart zu abschwächend. Hierbei wird einerseits erwartet, dass der Relationstyp der Relation geändert wird und andererseits, dass die Pfeilspitzen der Relation nicht mehr mit der Relation

von **Symptom 18** nach **Symptom 15** zusammengefasst wird, da es sich nach der Änderung nicht mehr um die gleiche Pfeilspitze handelt.

3.38.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 24.02.2019 um 17:04
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 12:30
Test negativ.



3.38.2 Fazit

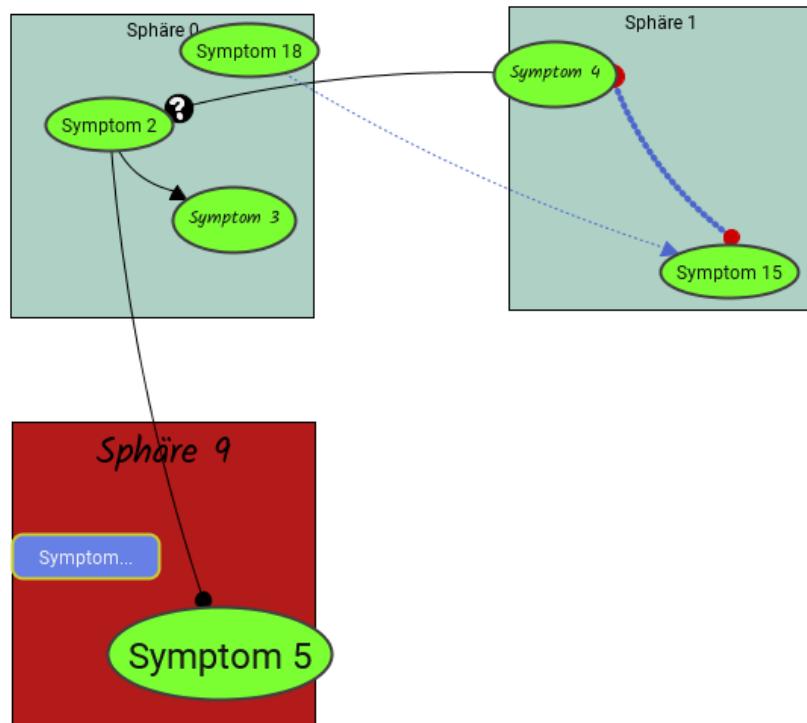
Die Software arbeitet wie erwartet.

3.39 Hinzufügen von Ankerpunkten

Vorerst betätigt der Bearbeiter den Button zum Einblenden/Ausblenden von Ankerpunkten. Anschließend bewegt er von der Relation von **Symptom 4** nach **Symptom 15** die Stelle, an dem die Kante zu **Symptom 4** verläuft, indem er den Rechtsklick (gedrückt halten und bewegen) nahe des genannten Symptoms macht. Selbiges macht er nahe **Symptom 15** um dort selbiges zu bewirken. Bei den Aktionen wird erwartet, dass jeweils ein Ankerpunkt erstellt wird.

3.39.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 24.02.2019 um 18:46
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 12:33
Test negativ.



3.39.2 Fazit

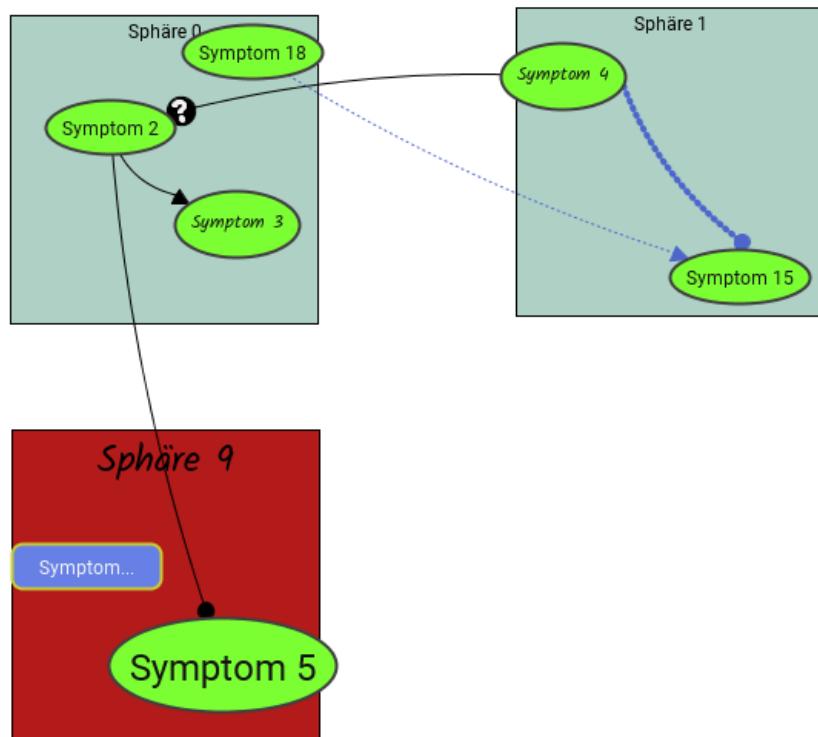
Die Software arbeitet wie erwartet.

3.40 Ankerpunkte ausblenden

Der Bearbeiter betätigt erneut den Button zum Einblenden/Ausblenden von Ankerpunkten. Hierbei wird erwartet, dass die vorerst angezeigten Ankerpunkte ausgeblendet werden.

3.40.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 24.02.2019 um 18:53
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 12:34
Test negativ.



3.40.2 Fazit

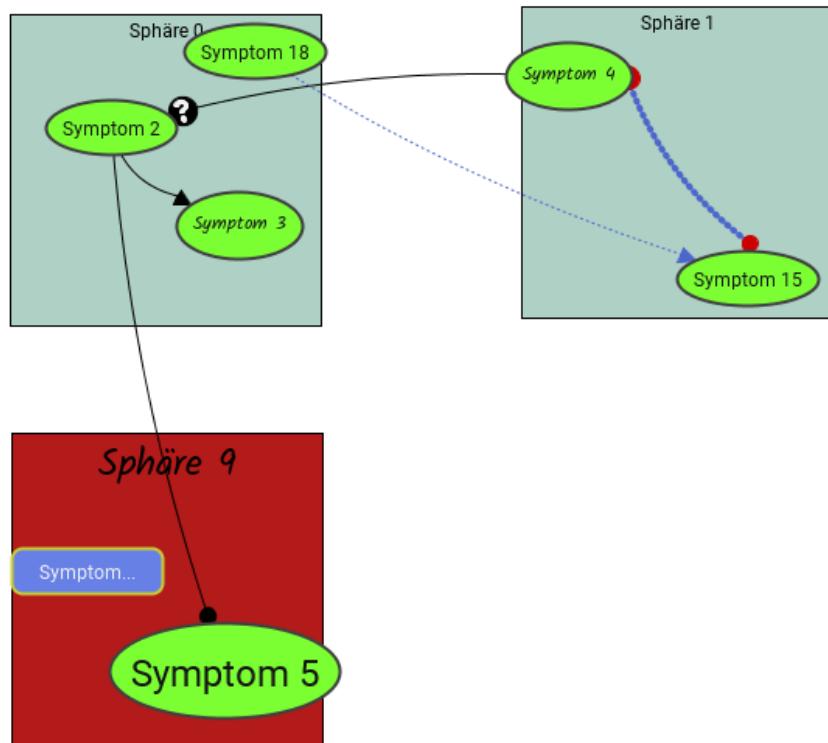
Die Software arbeitet wie erwartet.

3.41 Ankerpunkte einblenden

Der Bearbeiter betätigt erneut den Button zum Einblenden/Ausblenden von Ankerpunkten. Hierbei wird erwartet, dass die vorerst ausgeblendeten Ankerpunkte eingeblendet werden.

3.41.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 24.02.2019 um 19:24
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 12:35
Test negativ.



3.41.2 Fazit

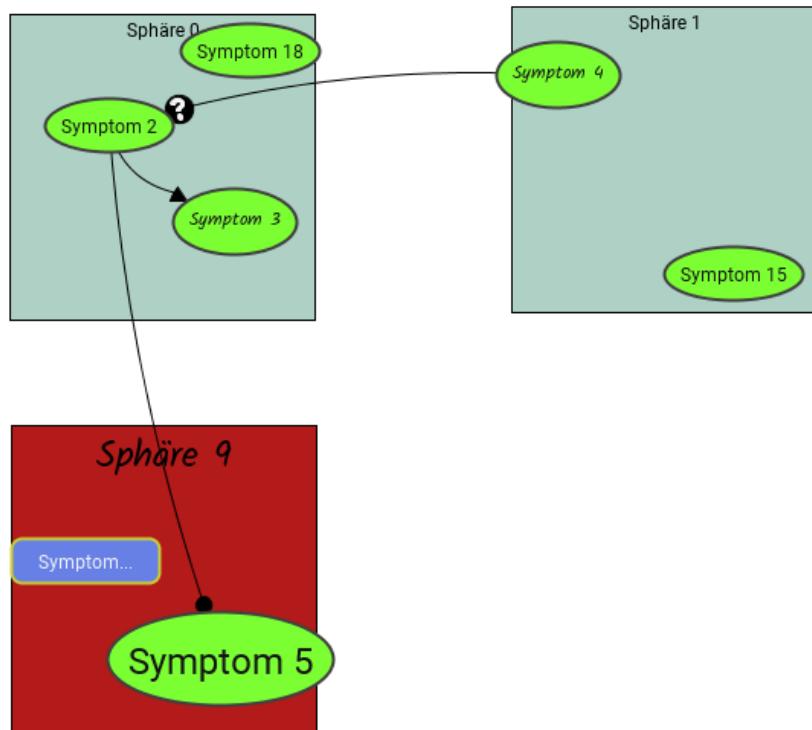
Die Software arbeitet wie erwartet.

3.42 Entfernen mehrerer Relationen

Der Bearbeiter wählt mit Umschalten und Linksklick die Relation von Symptom 18 nach Symptom 15 und die von Symptom 4 nach Symptom 15 aus und betätigt den Button zum Entfernen von Relationen. Hierbei ist zu erwarten, dass beide Relationen inklusive der Ankerpunkte entfernt werden.

3.42.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 24.02.2019 um 19:17
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 12:37
Test negativ.



3.42.2 Fazit

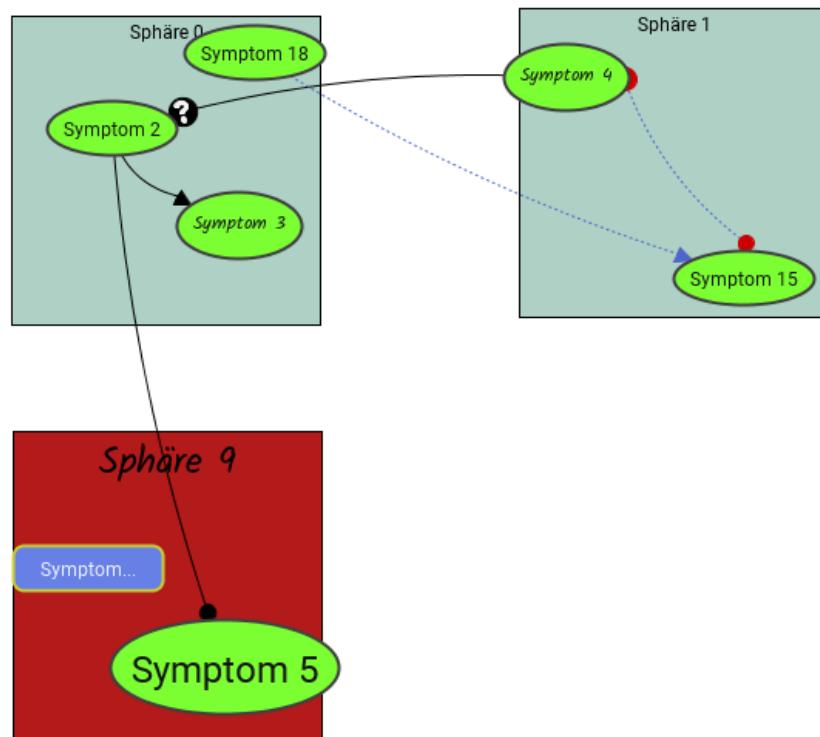
Die Software arbeitet wie erwartet.

3.43 Rückgängig machen des Entfernen mehrerer Relationen mit Ankerpunkten

Der Bearbeiter betätigt den Button zum Rückgängigmachen der letzten Aktion. Hierbei wird erwartet, dass die vorher entfernten Relationen inklusive ihrer Ankerpunkte wieder hinzugefügt werden.

3.43.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 24.02.2019 um 19:35
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 12:39
Test negativ.



3.43.2 Fazit

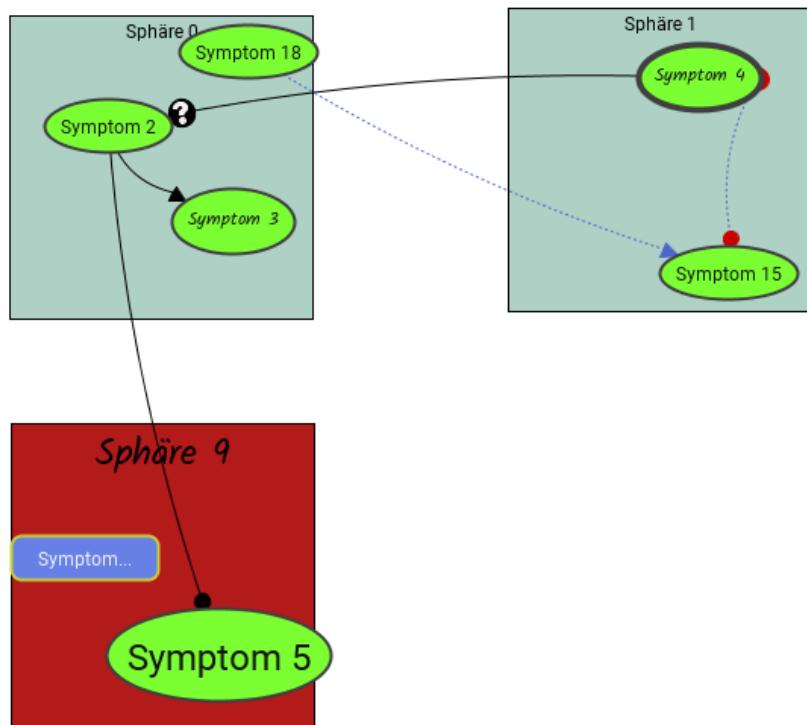
Die Software arbeitet wie erwartet.

3.44 Bewegen eines Symptoms, bei dem eine Relation Ankerpunkte hat

Der Bearbeiter bewegt Symptom 4 innerhalb seiner Sphäre weiter nach rechts. Hierbei wird erwarten, dass die Punkte, an denen die Relation von Symptom 4 nach Symptom 15 an den eben genannten Symptomen andockt nicht verschoben werden, da die Relation Ankerpunkte hat. Ohne Ankerpunkte würden sie sich der Verschiebung anpassen.

3.44.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 25.02.2019 um 12:07
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 12:41
Test negativ.



3.44.2 Fazit

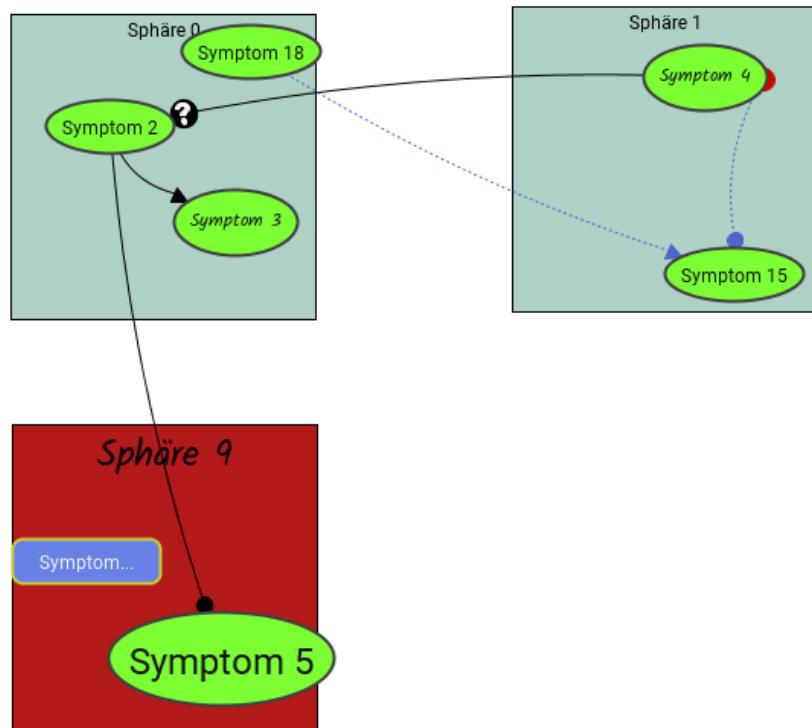
Die Software arbeitet wie erwartet.

3.45 Bei angezeigten Ankerpunkten in den Analyse-Modus wechseln ... ☐□

Der Bearbeiter wechselt in den Analyse-Modus. Hierbei wird erwartet, dass alle im Bearbeiter-Modus angezeigten Ankerpunkte im Analyse-Modus nicht mehr angezeigt werden.

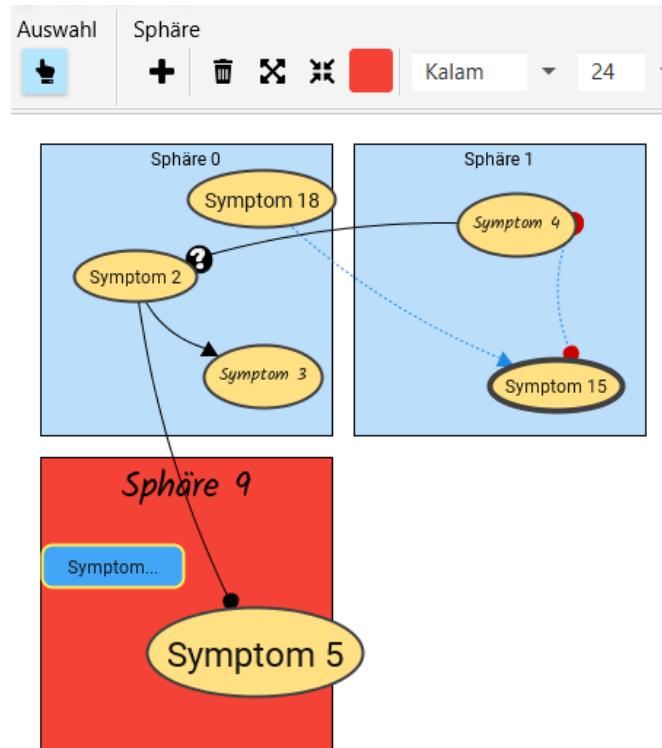
3.45.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 25.02.2019 um 13:01
Test positiv, weil noch ein Ankerpunkt angezeigt wird. Genauer gesagt wird bei der Relation von **Symptom 4** nach **Symptom 15** der Ankerpunkt an **Symptom 4** noch angezeigt.

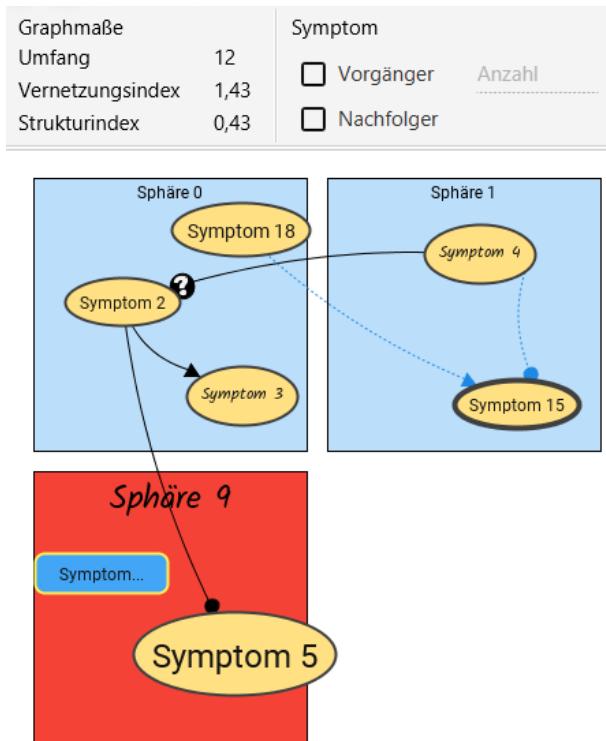


2.Test: Getestet von: Jacky Philipp Mach. Datum: 05.03.2019 um 14:11
Test negativ.

Vor dem Wechsel in den Analyse-Modus:



Nach dem Wechsel in den Analyse-Modus:



3.45.2 Fazit

Nach dem ersten Test wurde ein Wahrheitswert in **Values** hinzugefügt, der dazu führt, dass auch der betroffene Ankerpunkt nicht mehr angezeigt wird, wenn man den Modus wechselt. Beim zweiten Test arbeitet die Software wie erwartet.

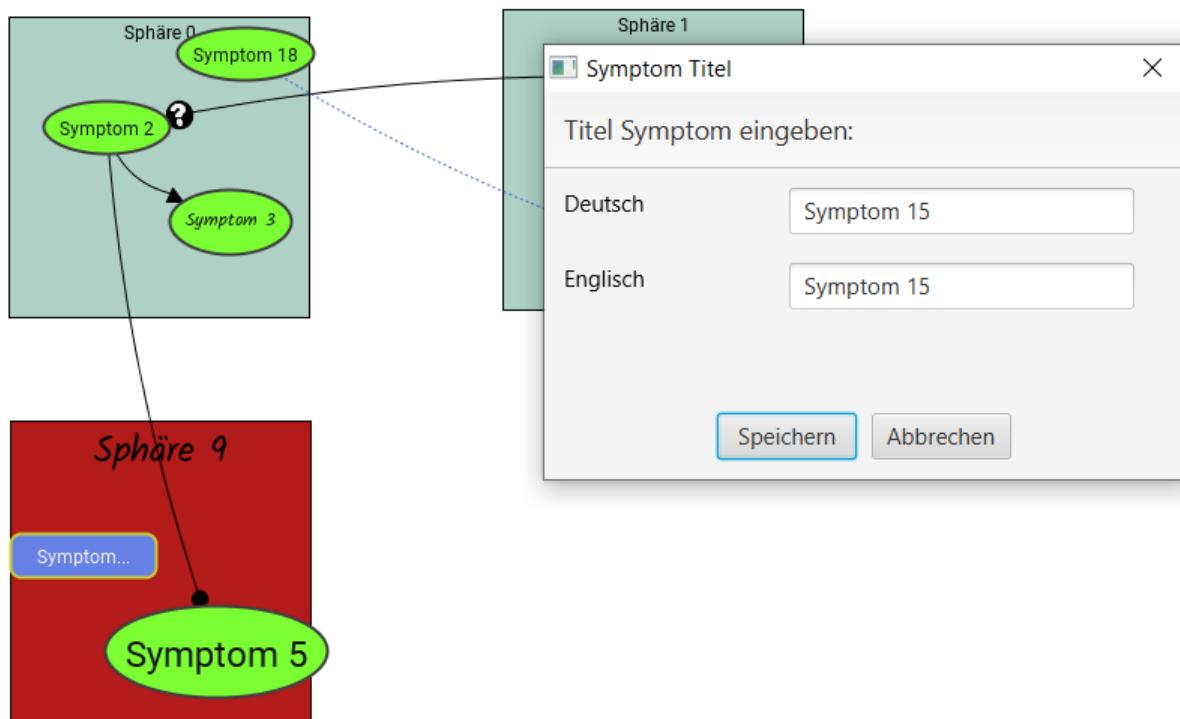
3.46 Änderung der Beschriftung eines Symptoms

Vorerst wird wieder in den Bearbeiter-Modus gewechselt. Mithilfe eines Rechtsklicks auf **Symptom 15** und drücken auf **Titel** öffnet der Bearbeiter ein Fenster zur Eingabe der Beschriftung des Symptoms in Deutsch und Englisch. Dann gibt er in dem Feld für die deutsche Beschriftung **Individualisierung** und in dem für die englische Beschriftung **Individualization** ein und drückt auf **Speichern**. Hierbei wird erwartet, dass in dem Symptom nun **Individualisierung** steht.

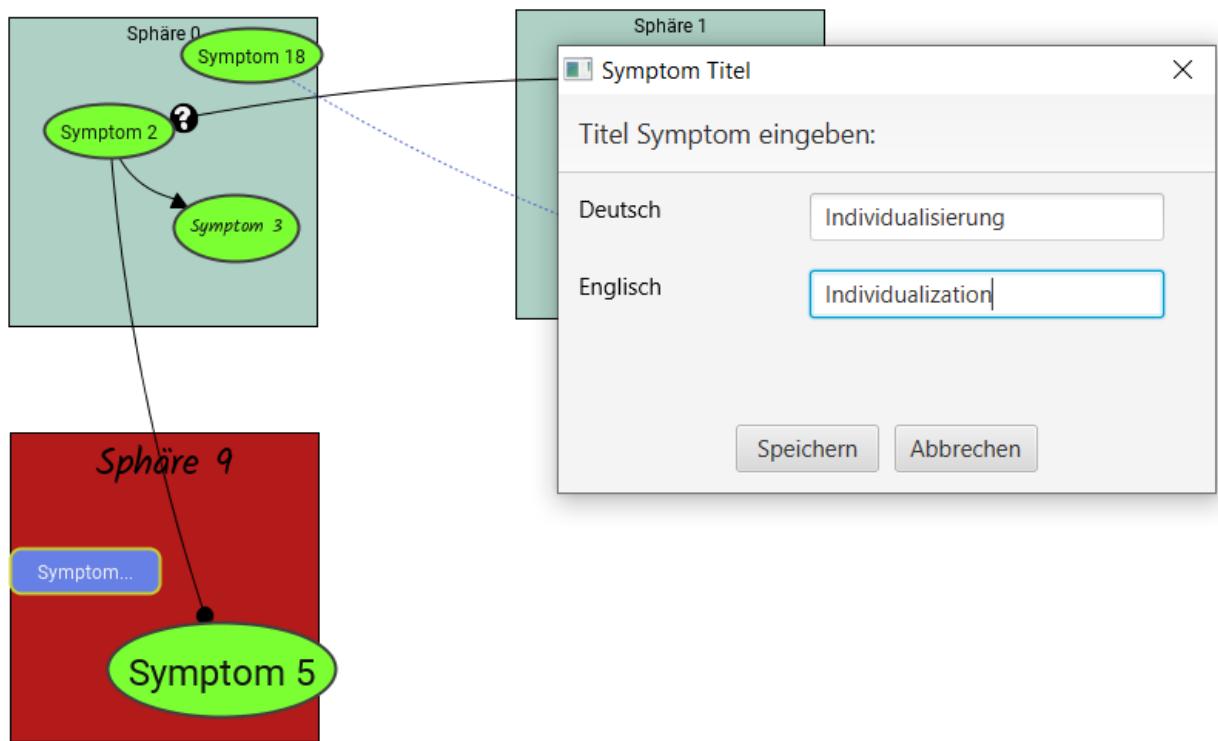
3.46.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 25.02.2019 um 14:59
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 13:01
Test negativ.

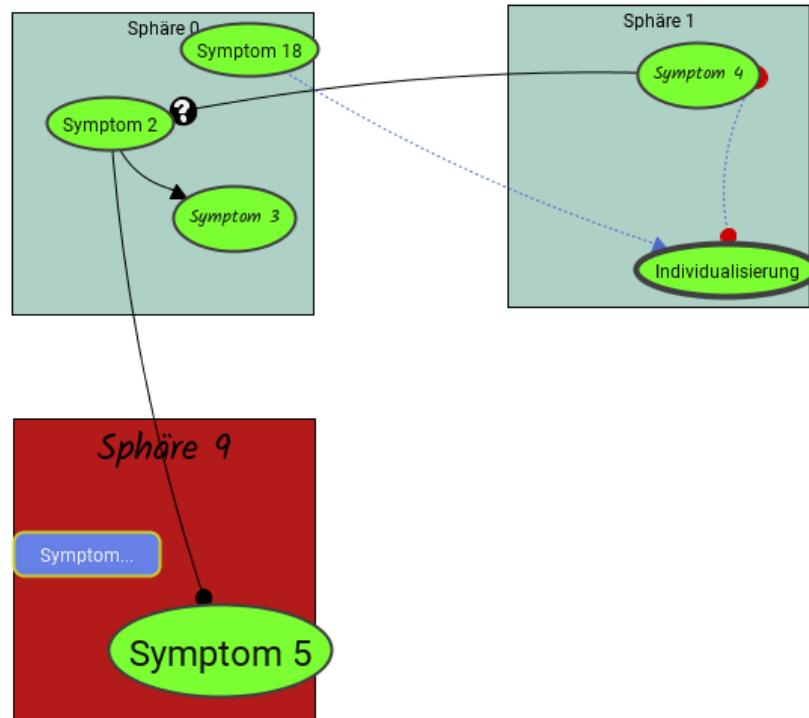
Nach dem Klick auf Titel:



Nach der Eingabe der Beschriftungen:



Nach den Drücken auf Speichern:



3.46.2 Fazit

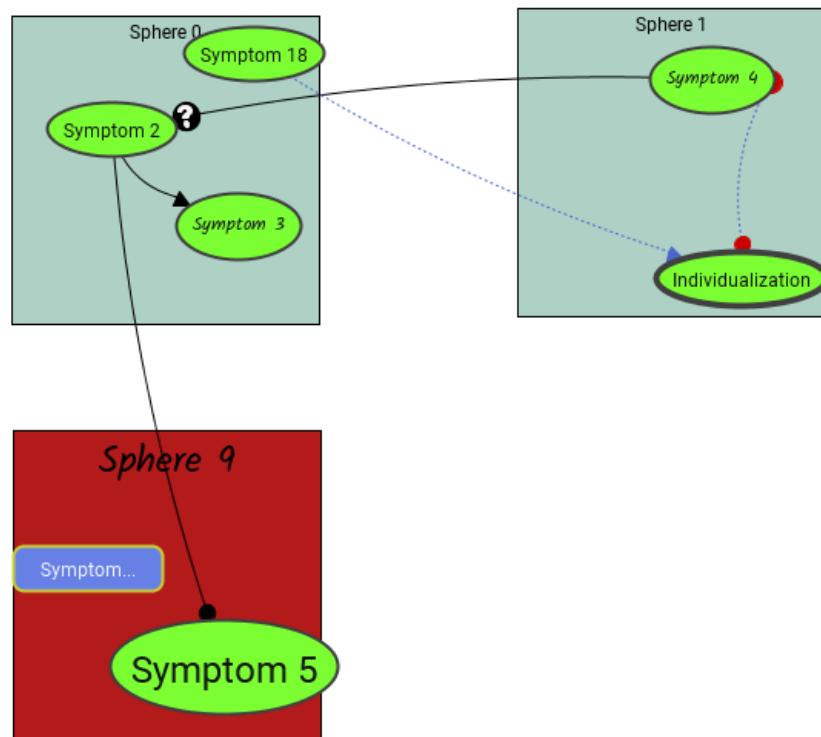
Die Software arbeitet wie erwartet.

3.47 Ändern der Sprache des Graphen ☐☐□

Der Bearbeiter geht auf Optionen, unter erweiterte Spracheinstellungen, Sprache des Graphen und dann auf Englisch. Hierbei wird erwartet, dass die Sprache aller Graph-Elemente auf Englisch gestellt werden.

3.47.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 25.02.2019 um 15:08
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 13:03
Test negativ.



3.47.2 Fazit

Die Software arbeitet wie erwartet.

3.48 Bewegen eines Symptoms auf ein anderes Symptom ☐☐□

Der Bearbeiter bewegt Individualization auf Symptom 4. Hierbei wird erwartet, dass das bewegte Symptom wieder auf seine vorherige Position zurück springt.

3.48.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 25.02.2019 um 15:13
Test negativ. Die Abbildung ist identisch zur vorherigen.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 13:03
Test negativ. Die Abbildung ist identisch zur vorherigen.

3.48.2 Fazit

Die Software arbeitet wie erwartet.

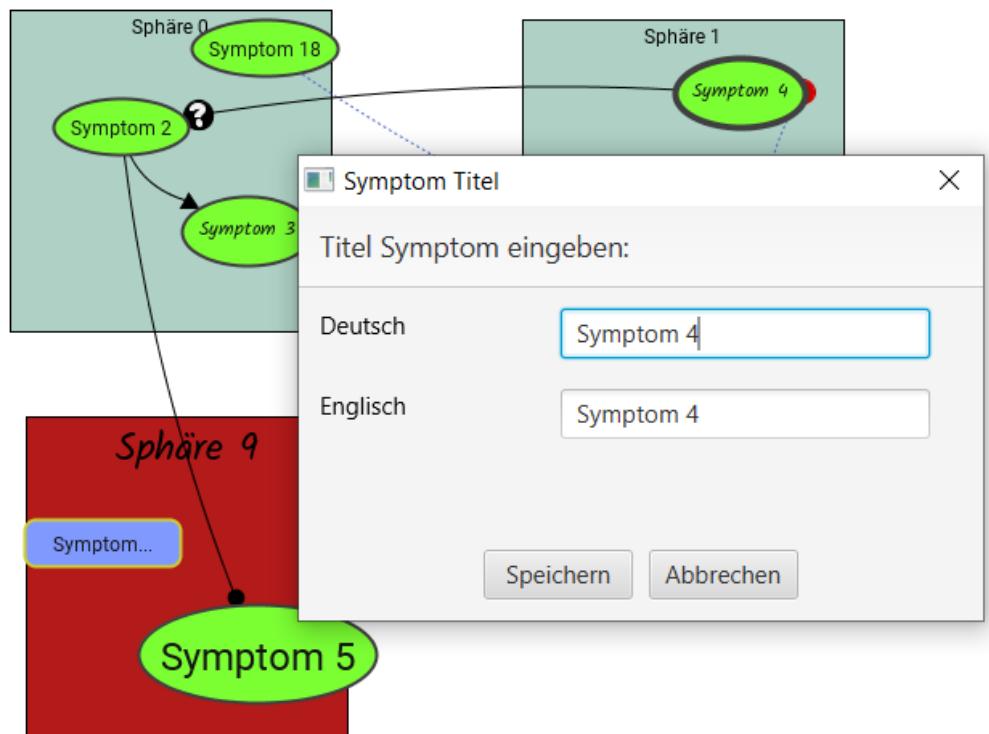
3.49 Änderung der Beschriftung eines Symptoms zu einer bereits existierenden Beschriftung

Vorerst wird die Sprache des Graphen wieder auf Deutsch gestellt. Anschließend macht der Bearbeiter einen Rechtsklick auf **Symptom 4** und drückt auf **Titel**. Dort versucht er die deutsche Beschriftung des Symptoms zu **Individualisierung** zu ändern. Das Leerzeichen nach **Individualisierung** ist wichtig um zu testen, ob die Aktion trotz Leerzeichen verboten wird. Hierbei wird erwartet, dass dies nicht erlaubt wird, da bereits ein anderes Symptom diese Beschriftung hat.

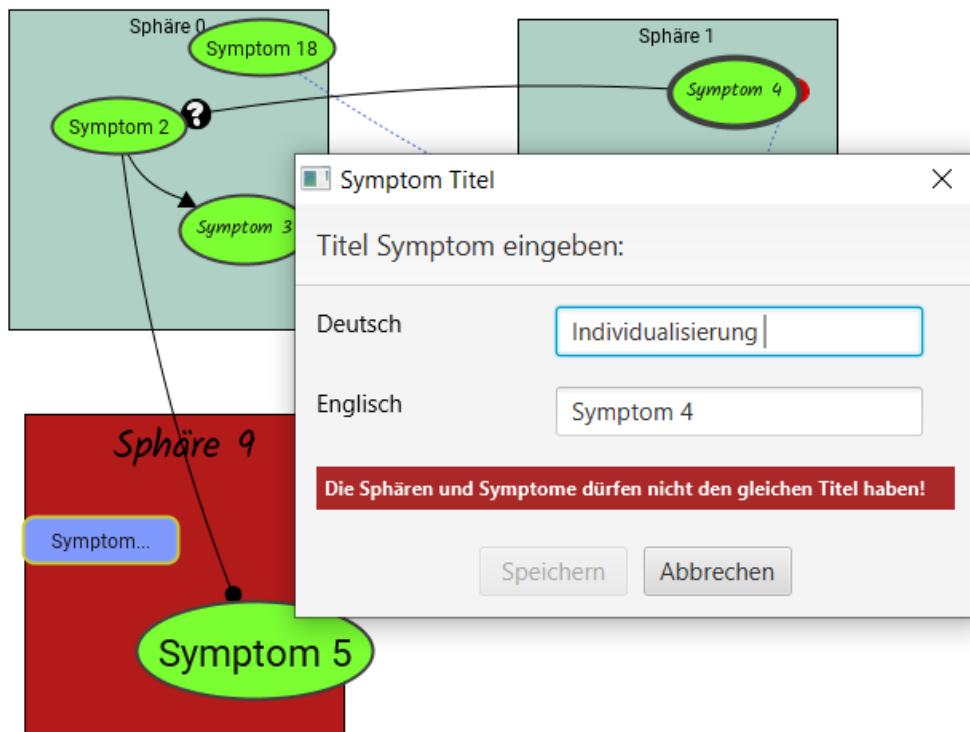
3.49.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 25.02.2019 um 15:20
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 13:05
Test negativ.

Nach den Klick auf Titel:



Nach Eingabe der deutschen Beschriftung:



3.49.2 Fazit

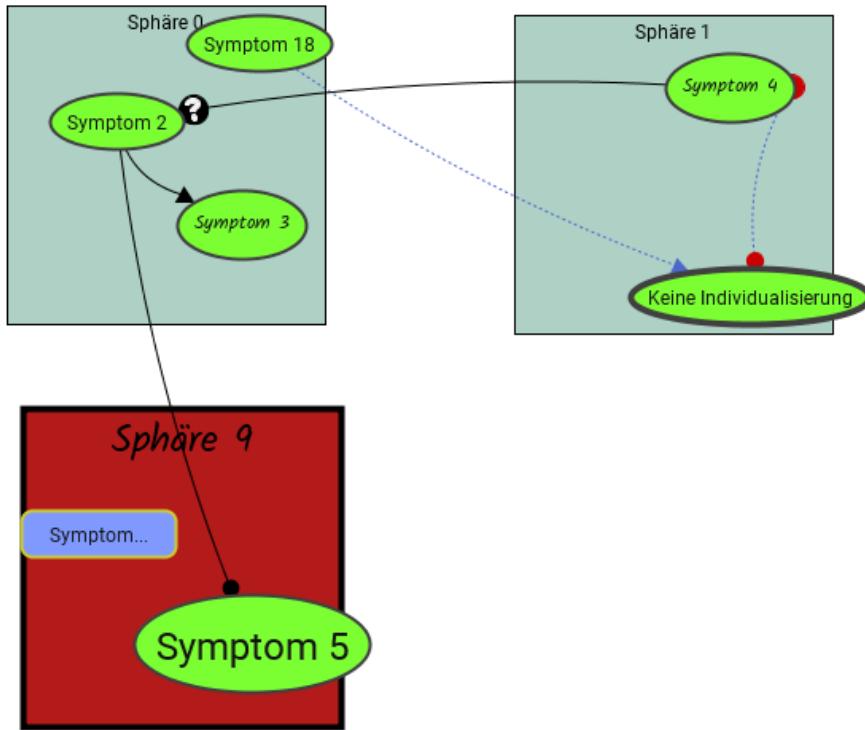
Die Software arbeitet wie erwartet.

3.50 Rückgängig machen der Änderung der Beschriftung von Symptomen ☒☒

Vorerst wird die Beschriftung von **Individualisierung** mit Rechtsklick und Titel zu **Keine Individualisierung** geändert. Anschließend betätigt der Bearbeiter den Button zum Rückgängigmachen der letzten Aktion. Hierbei wird erwartet, dass die Beschriftung des Symptoms wieder zu **Individualisierung** geändert wird. Nach dem Test wird die Beschriftung für die folgenden Tests wieder zu **Keine Individualisierung** geändert.

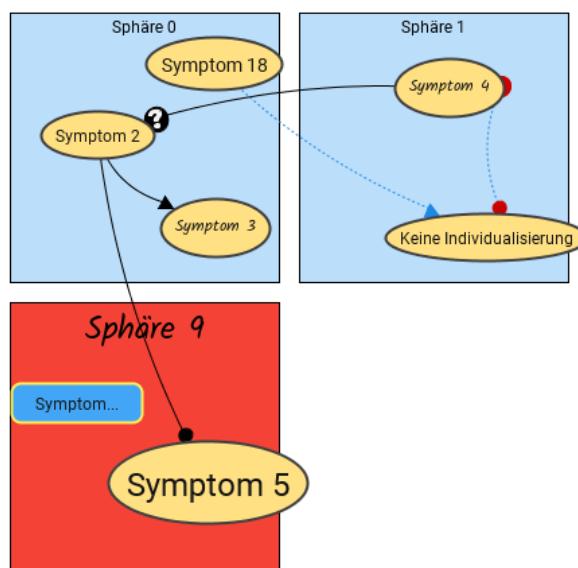
3.50.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 25.02.2019 um 15:44
Test positiv, weil die Aktion nicht rückgängig gemacht wurde.
Nach Änderung der Beschriftung:

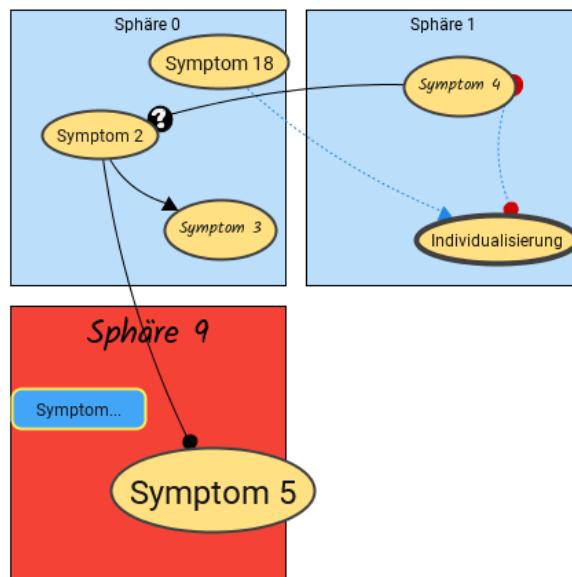


Nach Betätigen des Buttons zum Rückgängigmachen ist der Graph unverändert.
2.Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 13:06
Test negativ.

Nach Änderung der Beschriftung:



Nach Betätigen des Buttons zum Rückgängigmachen der letzten Aktion:



3.50.2 Fazit

Durch die Implementierung der Mehrsprachenunterstützung war die ursprüngliche Implementierung nicht mehr in der Lage die jeweilige Aktion korrekt auszuführen. Diese wurde im Nachhinein angepasst, sodass nun die Software wie erwartet arbeitet.

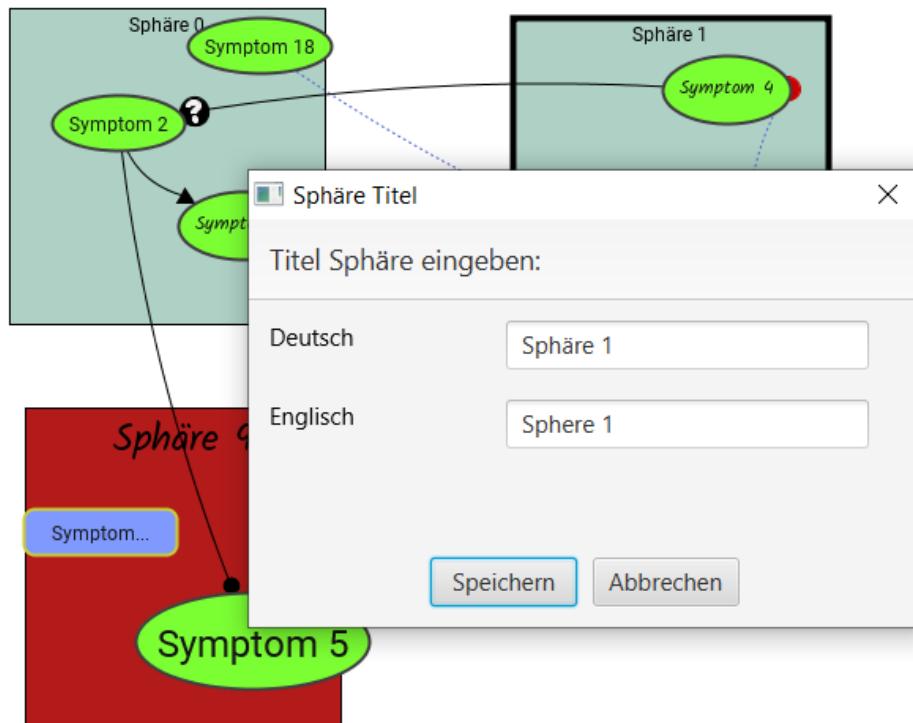
3.51 Änderung der Beschriftung einer Sphäre ☐☐☐

Mithilfe eines Rechtsklicks auf **Sphäre 1** und drücken auf **Titel** öffnet der Bearbeiter ein Fenster zur Eingabe der Beschriftung der Sphäre in Deutsch und Englisch. Dann gibt er in dem Feld für die deutsche Beschriftung **Gesellschaft** und in dem für die englische Beschriftung **Society** ein und drückt auf **Speichern**. Hierbei wird erwartet, dass in der Sphäre nun **Gesellschaft** steht.

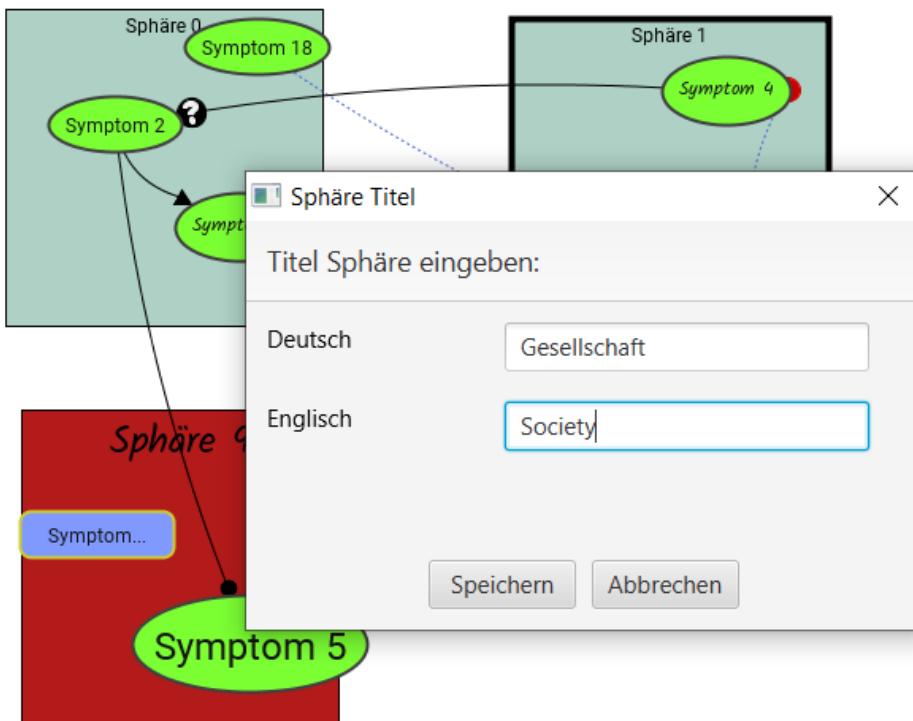
3.51.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 25.02.2019 um 16:16
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 13:09
Test negativ.

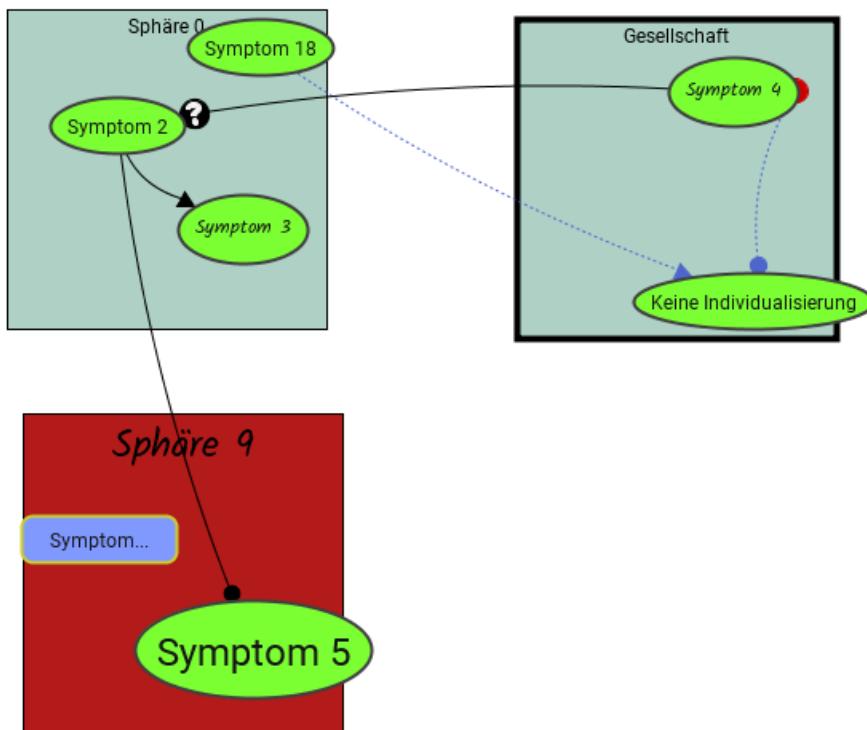
Nach dem Klick auf Titel:



Nach der Eingabe der Beschriftungen:



Nach den Drücken auf Speichern:



3.51.2 Fazit

Die Software arbeitet wie erwartet.

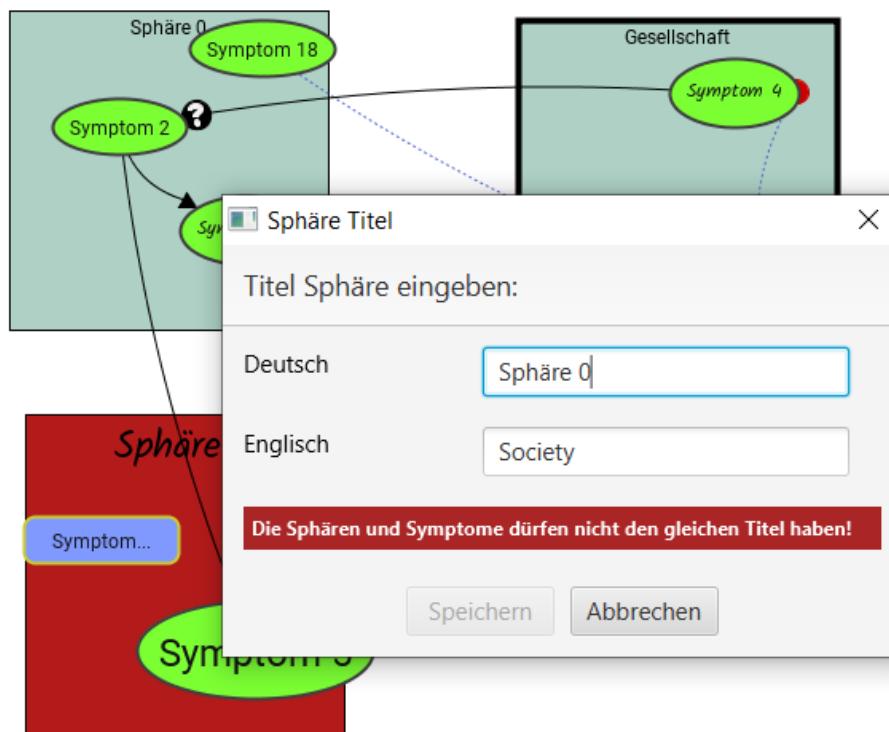
3.52 Änderung der Beschriftung einer Sphäre zu einer bereits existierenden Beschriftung

Der Bearbeiter macht einen Rechtsklick auf die Sphäre **Gesellschaft** und drückt auf **Titel**. Dort versucht er die deutsche Beschriftung der Sphäre zu **Sphäre 0** zu ändern. Hierbei wird erwartet, dass dies nicht erlaubt wird, da bereits eine andere Sphäre diese Beschriftung hat.

3.52.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 25.02.2019 um 16:25
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 13:10
Test negativ.

Nach Eingabe der neuen Beschriftung:



3.52.2 Fazit

Die Software arbeitet wie erwartet.

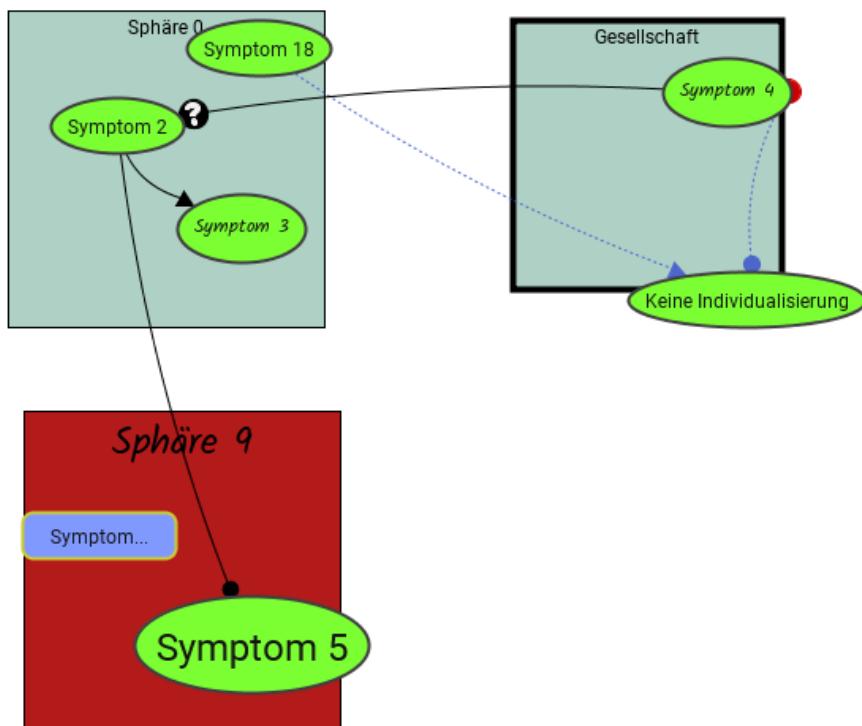
3.53 Eine Sphäre mehr verkleinern als es die Symptome erlauben.....☒☒☒

Der Bearbeiter wählt die Sphäre **Gesellschaft** aus und betätigt den Button zum Verkleinern der Sphäre vier mal. Hierbei wird erwartet, dass dies 3 mal durchgeführt wird, die Sphäre beim vierten mal jedoch nicht mehr verkleinert wird.

3.53.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 25.02.2019 um 16:31
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 13:10
Test negativ.

Die Graphen nach dreimaligem und viermaligem Drücken des Buttons sind identisch.
Nachdem der Button zum Verkleinern vier mal betätigt wurde:



3.53.2 Fazit

Die Software arbeitet wie erwartet.

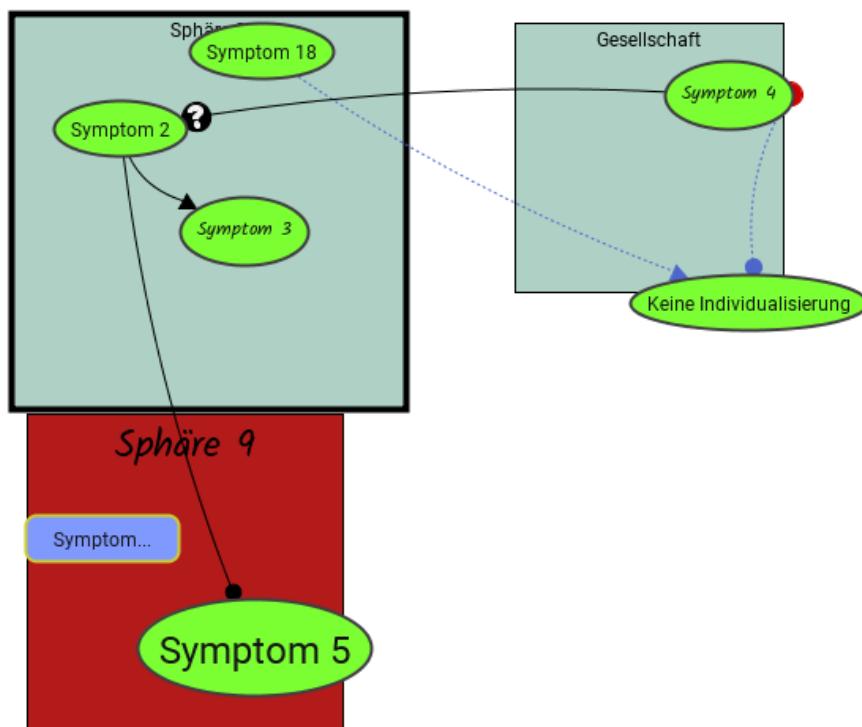
3.54 Eine Sphäre mehr vergrößern als es die anderen Sphären erlauben.....



Vorerst muss in den Ersteller-Modus gewechselt werden um die Änderung des Styles von Sphäre 0 zu erlauben. Anschließend wird wieder in den Bearbeiter-Modus zurück gewechselt. Der Bearbeiter wählt Sphäre 0 aus und betätigt den Button zum Vergrößern der Sphäre sechs mal. Hierbei wird erwartet, dass dies 5 mal durchgeführt wird, die Sphäre beim sechsten mal jedoch nicht mehr vergrößert wird, da sie bereits an Sphäre 9 angrenzt.

3.54.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 25.02.2019 um 16:40
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 13:11
Test negativ.



3.54.2 Fazit

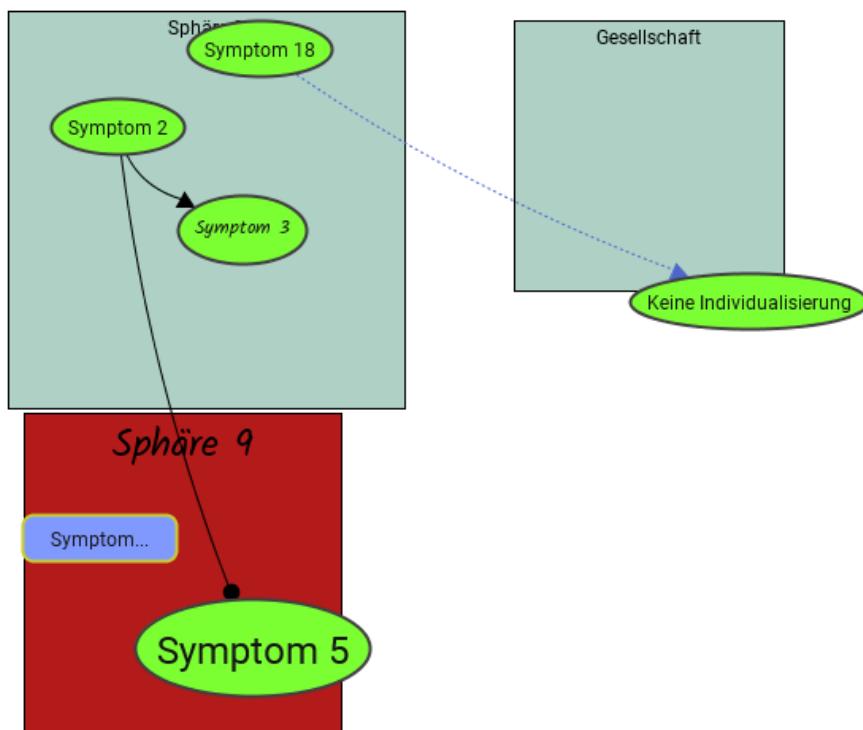
Die Software arbeitet wie erwartet.

3.55 Ausblenden von Elementen

Der Bearbeiter wählt mit Umschalten gleichzeitig Symptom 4, die Relation von Symptom 4 nach Keine Individualisierung und die Relation von Symptom 4 nach Symptom 2 aus und betätigt den Button zum Hinzufügen der ausgewählten Elementen zu den auszublendenden Elementen. Anschließend betätigt er den Button zum Ausblenden der Elemente. Hierbei wird erwartet, dass nur die vorher erwähnten Elemente ausgeblendet werden.

3.55.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 25.02.2019 um 17:22
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 13:16
Test negativ.



3.55.2 Fazit

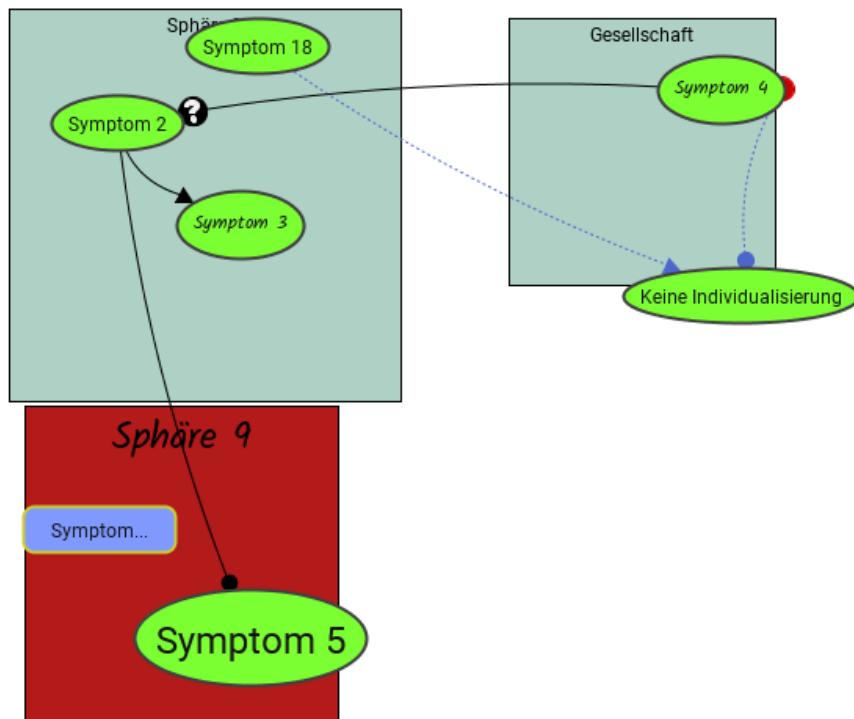
Die Software arbeitet wie erwartet.

3.56 Einblenden von Elementen

Der Bearbeiter betätigt den Button zum Einblenden von Elementen. Hierbei wird erwartet, dass die im letzten Test ausgeblendeten Elemente wieder eingeblendet werden.

3.56.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 25.02.2019 um 17:24
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 13:17
Test negativ.



3.56.2 Fazit

Die Software arbeitet wie erwartet.

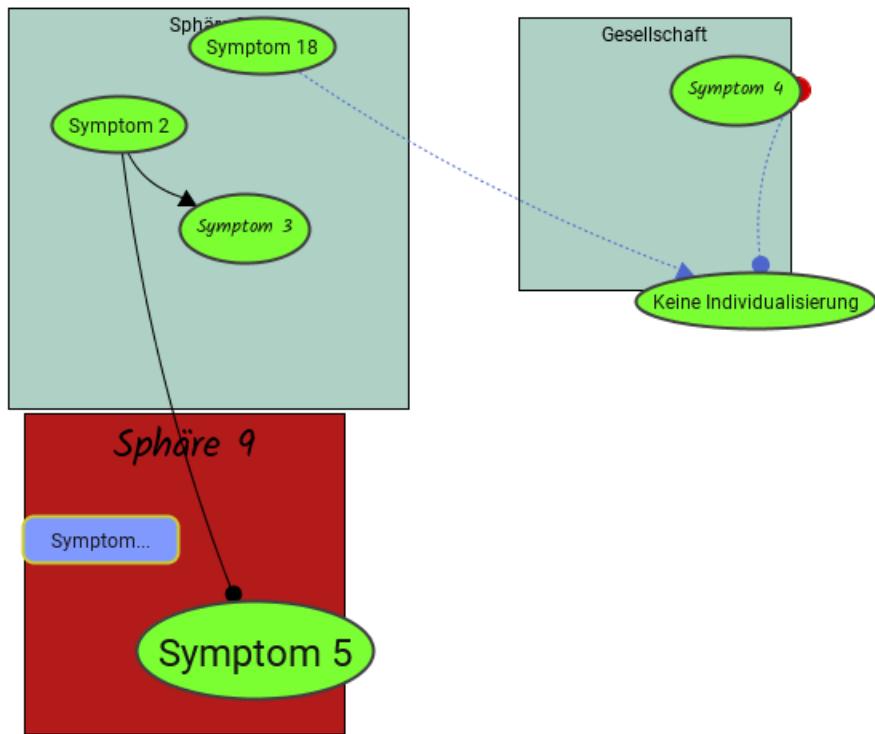
3.57 Elemente aus Menge der auszublendenden Elementen entfernen ... ☐☐☐

Der Bearbeiter wählt mit Umschalten gleichzeitig Symptom 4 und die Relation von Symptom 4 nach Keine Individualisierung aus und betätigt den Button zum Entfernen der ausgewählten Elemente aus der Menge der auszublendenden Elemente. Anschließend betätigt er den Button zum Ausblenden der Elemente. Hierbei wird erwartet, dass nur noch die Relation von Symptom 4 nach Symptom 2 ausgeblendet wird. Danach wird wieder der Button zum Einblenden der Elemente betätigt.

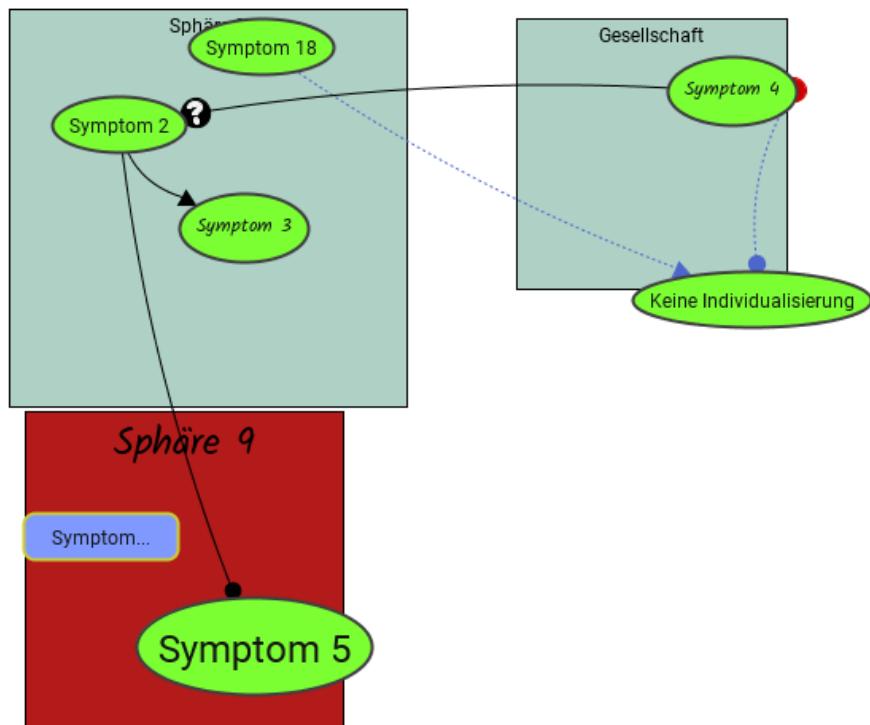
3.57.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 25.02.2019 um 17:28
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 13:18
Test negativ.

Nach dem Ausblenden der Elemente:



Nach Einblenden der Elemente:



3.57.2 Fazit

Die Software arbeitet wie erwartet.

3.58 Hervorheben von Elementen

Der Bearbeiter wählt mit Umschalten gleichzeitig Symptom 2, Symptom 3, die Relation von Symptom 2 nach Symptom 3 und die von Symptom 2 nach Symptom 5 aus und betätigt den Button zum Hinzufügen der ausgewählten Elementen zu den hervorzuhebenden Elementen. Anschließend betätigt er den Button zum Hervorheben der Elemente. Hierbei wird erwartet, dass nur die vorher erwähnten Elemente hervorgehoben werden.

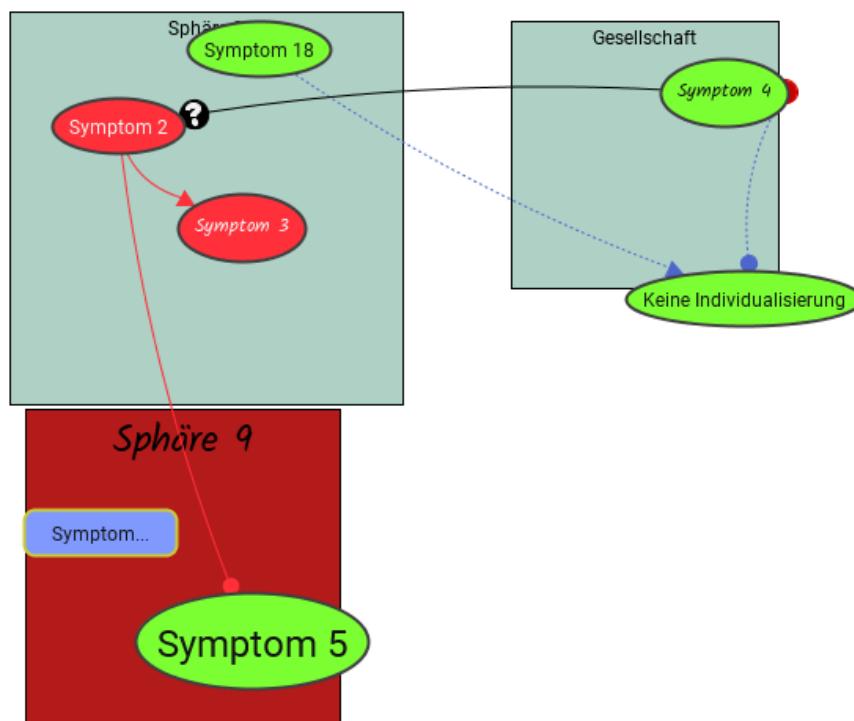
3.58.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 25.02.2019 um 17:50

Test negativ.

2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 13:20

Test negativ.



3.58.2 Fazit

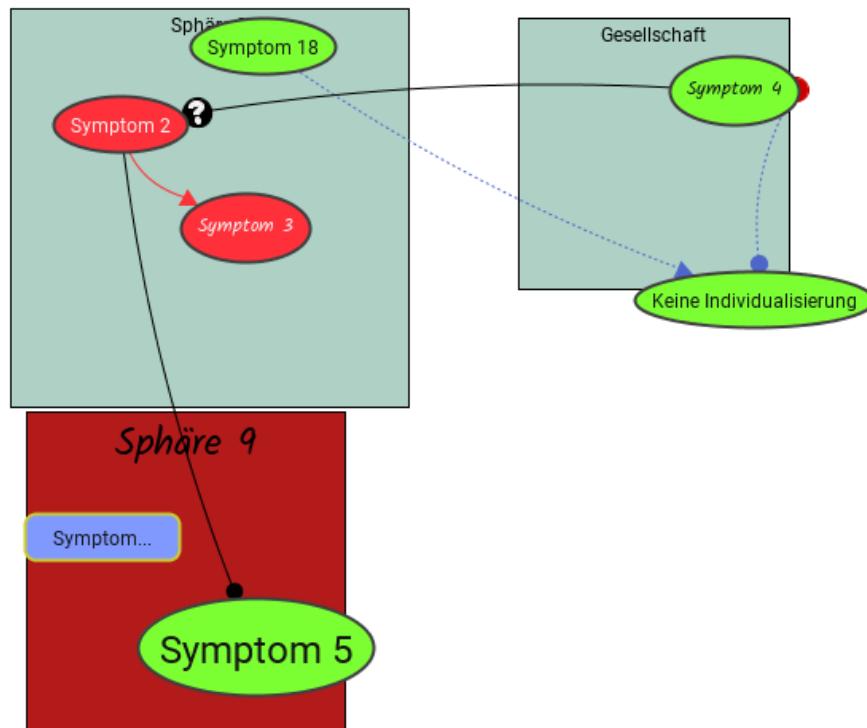
Die Software arbeitet wie erwartet.

3.59 Ein Element aus Menge der hervorzuhebenden Elemente entfernen

Der Bearbeiter wählt die Relation von **Symptom 2** nach **Symptom 5** aus und betätigt den Button zum Entfernen des ausgewählten Elements aus der Menge der hervorzuhebenden Elemente. Hierbei ist zu erwarten, dass die Relation nicht mehr hervorgehoben wird.

3.59.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 25.02.2019 um 18:12
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 13:21
Test negativ.



3.59.2 Fazit

Die Software arbeitet wie erwartet.

Für die Tests für den Verlauf und die Übersicht erstellen wir vorerst unter **Datei** und **Neue Datei** eine neue Datei. Anschließend führen wir folgende Aktionen aus: (Positionen und Verschiebungen, sowie Farben können bei jedem Test variieren)

- Sphäre hinzufügen
- Symptom in die Sphäre hinzufügen (Symptom 1)
- Symptom in die Sphäre hinzufügen (Symptom 2)
- Verstärkende Relation von **Symptom 1** zu **Symptom 2**
- Ändern der Farbe der Relation zu Backstein
- Ändern der Linienart der Relation zu gepunktet
- Ändern des Relationstyps zu abschwächend
- Ändern der Schriftgröße der Sphäre zu 18
- Ändern der Schriftgröße von **Symptom 2** zu 18
- Ändern der Schriftart von **Symptom 2** zu **Kalam**
- Ändern der Schriftart von der Sphäre zu **Mali**
- Ändern der deutschen Beschriftung der Sphäre zu **Gesellschaft** und der englischen zu **Society**
- Ändern der Farbe der Sphäre zu **Kornblumenblau**
- die Sphäre einmal vergrößern
- Ändern der Randfarbe von **Symptom 2** zu **Goldrute**
- Ändern der Füllfarbe von **Symptom 2** zu **Schiefergrau**
- Ändern der Form von **Symptom 2** zu rechteckig
- **Symptom 2** einmal vergrößern
- Sphären automatisch anordnen
- Symptome automatisch anordnen
- die Sphäre bewegen
- **Symptom 2** bewegen
- Ändern der deutschen und englischen Beschriftung von **Symptom 2** zu **Dinkelberry**

- die Relation entfernen
- gleichzeitig beide Symptome entfernen
- die Sphäre entfernen

Diese Aktionen decken alle verschiedenen Log-Einträge ab.

3.60 Exportieren als Protokoll

Der Bearbeiter drückt auf Datei, Exportieren als und dann auf Protokoll. Im sich dann öffnenden Fenster speichert er die Datei im Verzeichnis seiner Wahl. Hierbei wird erwartet, dass in der gespeicherten Textdatei der Verlauf vollständig und korrekt enthalten ist.

3.60.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 27.02.2019 um 22:45

Test negativ.

2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 14:30

Test negativ.

Die Textdatei hat folgenden Inhalt:

```
verlaufesttxt - Editor
Datei Bearbeiten Format Ansicht Hilfe
Nummer des Eintrags: 1
Typ der Aktion: Hinzufügen einer Sphäre
Informationen: Sphäre: "Sphäre 0".
Zeit: 2019-02-26 11:52:43
Nummer des Eintrags: 2
Typ der Aktion: Hinzufügen von Symptomen
Informationen: Symptome: "Symptom 1". In Sphäre: "Sphäre 0";
Zeit: 2019-02-26 11:52:49
Nummer des Eintrags: 3
Typ der Aktion: Hinzufügen von Symptomen
Informationen: Symptome: "Symptom 2". In Sphäre: "Sphäre 0";
Zeit: 2019-02-26 11:52:52
Nummer des Eintrags: 4
Typ der Aktion: Hinzufügen von Realitionen
Informationen: Relationen: Von Symptom: "Symptom 1" zum Symptom: "Symptom 2". Relationsart: Verstärkend;
Zeit: 2019-02-26 11:52:56
Nummer des Eintrags: 5
Typ der Aktion: Änderung der Farbe von Relationen
Informationen: Geänderte Relationen: Von Symptom: "Symptom 1" zum Symptom: "Symptom 2". Alte Farbe: Schwarz, neue Farbe: Backstein. Relationsart: Verstärkend;
Zeit: 2019-02-26 11:54:00
Nummer des Eintrags: 6
Typ der Aktion: Änderung der Linienart von Realitionen
Informationen: Geänderte Relationen: Von Symptom: "Symptom 1" zum Symptom: "Symptom 2". Alte Linienart: Durchgezogen, neue Linienart: Gepunktet. Relationsart: Verstärkend;
Zeit: 2019-02-26 11:54:44
Nummer des Eintrags: 7
Typ der Aktion: Änderung von Relationstypen
Informationen: Geänderte Relationen: Von Symptom: "Symptom 1" zum Symptom: "Symptom 2". Alte Pfeilspitze: Verstärkend, neue Pfeilspitze: Abschwächend;
Zeit: 2019-02-26 11:54:46
Nummer des Eintrags: 8
Typ der Aktion: Änderung der Schriftgröße einer Sphäre
Informationen: Sphäre: "Sphäre 0". Alte Schriftgröße: 12, neue Schriftgröße: 18.
Zeit: 2019-02-26 11:54:53
Nummer des Eintrags: 9
Typ der Aktion: Änderung der Schriftgröße von Symptomen
Informationen: Veränderte Symptome: "Symptom 2". Alte Schriftgröße: 12, neue Schriftgröße: 18;
Zeit: 2019-02-26 12:05:51
Nummer des Eintrags: 10
Typ der Aktion: Änderung der Schriftart von Symptomen
Informationen: Veränderte Symptome: "Symptom 2". Alte Schriftart: Roboto, neue Schriftart: Kalam;
Zeit: 2019-02-26 12:08:13
Nummer des Eintrags: 11
Typ der Aktion: Änderung der Schriftart einer Sphäre
Informationen: Sphäre: "Sphäre 0". Alte Schriftart: Roboto, neue Schriftart: Mali.
Zeit: 2019-02-26 12:08:26
Nummer des Eintrags: 12
Typ der Aktion: Änderung der Beschriftung einer Sphäre
Informationen: Sphäre: "Gesellschaft". Alte Beschriftung(Englisch): Sphere 0, neue Beschriftung(Englisch): Society. Alte Beschriftung(Deutsch): Sphäre 0, neue Beschriftung(Deutsch): Gesellschaft.
Zeit: 2019-02-26 12:08:54
Nummer des Eintrags: 13
Typ der Aktion: Änderung der Farbe einer Sphäre
Informationen: Sphäre: "Gesellschaft". Alte Farbe: Silber, neue Farbe: Kornblumenblau.
Zeit: 2019-02-26 12:09:09
Nummer des Eintrags: 14
Typ der Aktion: Änderung der Größe einer Sphäre
Informationen: Sphäre: "Gesellschaft". Alte Größe: 200.0, neue Größe: 210.0
Zeit: 2019-02-26 12:09:37
```

```
verlaufTest.txt - Editor
Datei Bearbeiten Format Ansicht Hilfe
Nummer des Eintrags: 12
Typ der Aktion: Änderung der Beschriftung einer Sphäre
Informationen: Sphäre: "Gesellschaft". Alte Beschriftung(Englisch): Sphere 0, neue Beschriftung(Englisch): Society. Alte Beschriftung(Deutsch): Sphäre 0, neue Beschriftung(Deutsch): Gesellschaft.
Zeit: 2019-02-26 12:08:54
Nummer des Eintrags: 13
Typ der Aktion: Änderung der Farbe einer Sphäre
Informationen: Sphäre: "Gesellschaft". Alte Farbe: Silber, neue Farbe: Kornblumenblau.
Zeit: 2019-02-26 12:09:09
Nummer des Eintrags: 14
Typ der Aktion: Änderung der Größe einer Sphäre
Informationen: Sphäre: "Gesellschaft". Alte Größe: 200.0, neue Größe: 210.0
Zeit: 2019-02-26 12:09:37
Nummer des Eintrags: 15
Typ der Aktion: Änderung der Umrundungsfarbe von Symptomen
Informationen: Veränderte Symptome: "Symptom 2". Alte Umrundungsfarbe: Dunkles Schiefergrau, neue Umrundungsfarbe: Goldrute;
Zeit: 2019-02-26 12:12:35
Nummer des Eintrags: 16
Typ der Aktion: Änderung der Füllfarbe von Symptomen
Informationen: Veränderte Symptome: "Symptom 2". Alte Füllfarbe: Grüngelb, neue Füllfarbe: Schiefergrau;
Zeit: 2019-02-26 12:13:20
Nummer des Eintrags: 17
Typ der Aktion: Änderung der Form von Symptomen
Informationen: Veränderte Symptome: "Symptom 2". Alte Form: Kreis, neue Form: Rechteck;
Zeit: 2019-02-26 12:13:30
Nummer des Eintrags: 18
Typ der Aktion: Änderung der Größe von Symptomen
Informationen: Veränderte Symptome: "Symptom 2". Alte Größe: 50, neue Größe: 55;
Zeit: 2019-02-26 12:13:34
Nummer des Eintrags: 19
Typ der Aktion: Änderung des Layouts von Spären
Informationen: Die Spären wurden automatisch angeordnet oder die automatische Anordnung wurde rückgängig gemacht.
Zeit: 2019-02-26 12:13:44
Nummer des Eintrags: 20
Typ der Aktion: Änderung des Layouts von Symptomen
Informationen: Die Symptome wurden automatisch angeordnet oder die automatische Anordnung wurde rückgängig gemacht.
Zeit: 2019-02-26 12:13:51
Nummer des Eintrags: 21
Typ der Aktion: Bewegen einer Sphäre
Informationen: Sphäre: "Gesellschaft". Alte Koordinaten: x = 20 y = 20, neue Koordinaten: x = 203 y = 125.
Zeit: 2019-02-26 12:14:05
Nummer des Eintrags: 22
Typ der Aktion: Bewegen von Symptomen
Informationen: Symptome: "Symptom 2". Alte Koordinaten: x = 270 y = 258, neue Koordinaten: x = 262 y = 301;
Zeit: 2019-02-26 12:14:14
Nummer des Eintrags: 23
Typ der Aktion: Änderung der Beschriftung eines Symptoms
Informationen: Symptom: "Dinkelberry". Alte Beschriftung(Englisch): Symptom 2, neue Beschriftung(Englisch): Dinkelberry. Alte Beschriftung(Deutsch): Symptom 2, neue Beschriftung(Deutsch): Dinkelberry.
Zeit: 2019-02-26 12:41:01
Nummer des Eintrags: 24
Typ der Aktion: Entfernen von Relationen
Informationen: Relationen: Von Symptom: "Symptom 1" zum Symptom: "Dinkelberry". Relationsart: Abschwächend;
Zeit: 2019-02-27 16:51:17
Nummer des Eintrags: 25
Typ der Aktion: Entfernen von Symptomen
Informationen: Symptome: "Symptom 1". In Sphäre: "Gesellschaft"; "Dinkelberry". In Sphäre: "Gesellschaft";
Zeit: 2019-02-27 16:51:20
Nummer des Eintrags: 26
Typ der Aktion: Entfernen einer Sphäre
Informationen: Sphäre: "Gesellschaft".
Zeit: 2019-02-27 16:51:22
```

3.60.2 Fazit

Die Software arbeitet wie erwartet.

3.61 Anzeige des Verlaufs innerhalb der Applikation

Der Bearbeiter öffnet in der Seitenleiste den Verlauf und klappt alle noch nicht ausgeklappten Einträge aus. Hierbei wird erwartet, dass der Verlauf vollständig und korrekt enthalten ist.

3.61.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 27.02.2019 um 23:26
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 14:32
Test negativ.

▼ Verlauf

⌚ Alle Logs ▾

- ▼ 1: Hinzufügen einer Sphäre
2019-02-26 11:52:43
Sphäre: "Sphäre 0".
- ▼ 2: Hinzufügen von Symptomen
2019-02-26 11:52:49
- ▼ Symptome
 - "Symptom 1". In Sphäre: "Sphäre 0"
- ▼ 3: Hinzufügen von Symptomen
2019-02-26 11:52:52
- ▼ Symptome
 - "Symptom 2". In Sphäre: "Sphäre 0"
- ▼ 4: Hinzufügen von Relationen
2019-02-26 11:52:56
- ▼ Relationen
 - Von Symptom: "Symptom 1" zum Symptom: "Symptom 2". Relationsart: Verstärkend
- ▼ 5: Änderung der Farbe von Relationen
2019-02-26 11:54:00
- ▼ Geänderte Relationen
 - Von Symptom: "Symptom 1" zum Symptom: "Symptom 2". Alte Farbe: Schwarz, neue Farbe: Backstein. Relationsart: Verstärkend
- ▼ 6: Änderung der Linienart von Relationen
2019-02-26 11:54:44
- ▼ Geänderte Relationen

▼ Verlauf

⌚ Alle Logs ▾

- ▼ 6: Änderung der Linienart von Relationen
2019-02-26 11:54:44
- ▼ Geänderte Relationen
 - Von Symptom: "Symptom 1" zum Symptom: "Symptom 2". Alte Linienart: Durchgezogen, neue Linienart: Gepunktet. Relationsart: Verstärkend
- ▼ 7: Änderung von Relationstypen
2019-02-26 11:54:46
- ▼ Geänderte Relationen
 - Von Symptom: "Symptom 1" zum Symptom: "Symptom 2". Alte Pfeilspitze: Verstärkend, neue Pfeilspitze: Abschwächend
- ▼ 8: Änderung der Schriftgröße einer Sphäre
2019-02-26 11:54:53
Sphäre: "Sphäre 0". Alte Schriftgröße: 12, neue Schriftgröße: 18.
- ▼ 9: Änderung der Schriftgröße von Symptomen
2019-02-26 12:05:51
- ▼ Veränderte Symptome
 - "Symptom 2". Alte Schriftgröße: 12, neue Schriftgröße: 18
- ▼ 10: Änderung der Schriftart von Symptomen
2019-02-26 12:08:13
- ▼ Veränderte Symptome
 - "Symptom 2". Alte Schriftart: Roboto, neue Schriftart: Kalam
- ▼ 11: Änderung der Schriftart einer Sphäre
2019-02-26 12:08:26
Sphäre: "Sphäre 0". Alte Schriftart: Roboto, neue Schriftart: Mali.

Verlauf

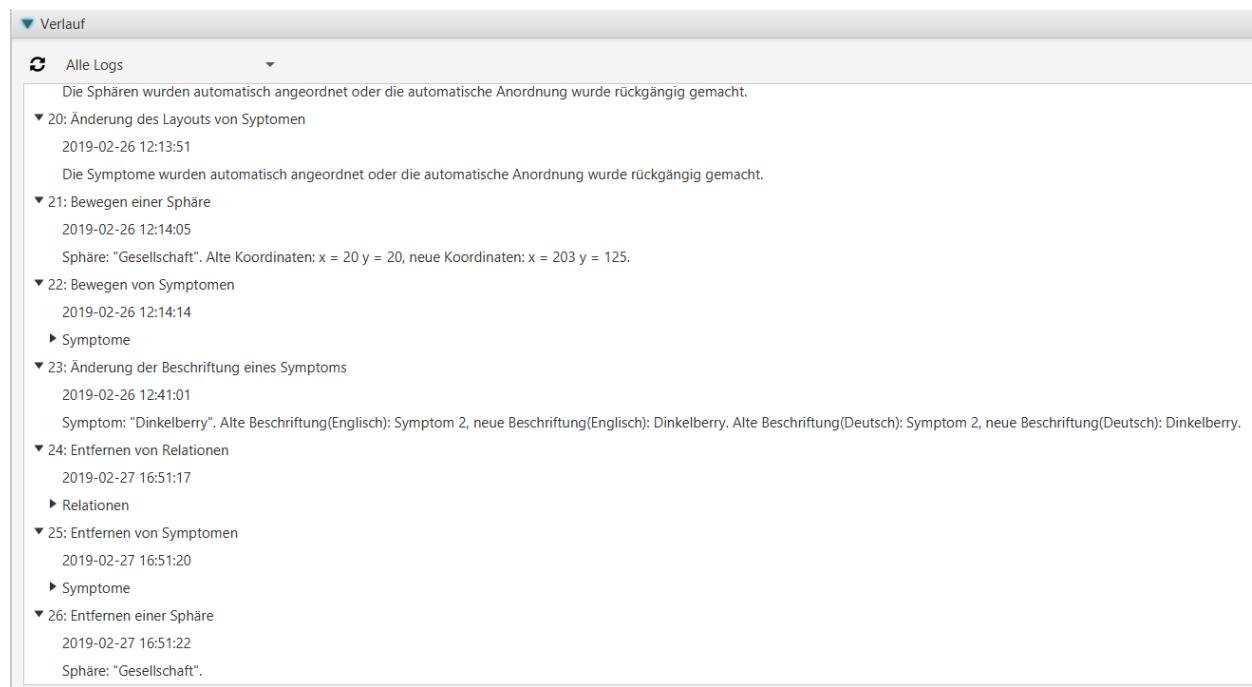
Alle Logs

- ▼ 11: Änderung der Schriftart einer Sphäre
2019-02-26 12:08:26
Sphäre: "Sphäre 0". Alte Schriftart: Roboto, neue Schriftart: Mali.
- ▼ 12: Änderung der Beschriftung einer Sphäre
2019-02-26 12:08:54
Sphäre: "Gesellschaft". Alte Beschriftung(Englisch): Sphere 0, neue Beschriftung(Englisch): Society. Alte Beschriftung(Deutsch): Sphäre 0, neue Beschriftung(Deutsch): Gesellschaft.
- ▼ 13: Änderung der Farbe einer Sphäre
2019-02-26 12:09:09
Sphäre: "Gesellschaft". Alte Farbe: Silber, neue Farbe: Kornblumenblau.
- ▼ 14: Änderung der Größe einer Sphäre
2019-02-26 12:09:37
Sphäre: "Gesellschaft". Alte Größe: 200.0, neue Größe: 210.0
- ▼ 15: Änderung der Umrandungsfarbe von Symptomen
2019-02-26 12:12:35
- ▼ Veränderte Symptome
 - "Symptom 2". Alte Umrandungsfarbe: Dunkles Schiefergrau, neue Umrandungsfarbe: Goldrute
- ▼ 16: Änderung der Füllfarbe von Symptomen
2019-02-26 12:13:20
- ▼ Veränderte Symptome
 - "Symptom 2". Alte Füllfarbe: Grüngelb, neue Füllfarbe: Schiefergrau
- ▼ 17: Änderung der Form von Symptomen
2019-02-26 12:13:30

Verlauf

Alle Logs

- ▼ 17: Änderung der Form von Symptomen
2019-02-26 12:13:30
- ▼ Veränderte Symptome
 - "Symptom 2". Alte Form: Kreis, neue Form: Rechteck
- ▼ 18: Änderung der Größe von Symptomen
2019-02-26 12:13:34
- ▼ Veränderte Symptome
 - "Symptom 2". Alte Größe: 50, neue Größe: 55
- ▼ 19: Änderung des Layouts von Spären
2019-02-26 12:13:44
Die Sphären wurden automatisch angeordnet oder die automatische Anordnung wurde rückgängig gemacht.
- ▼ 20: Änderung des Layouts von Syptomen
2019-02-26 12:13:51
Die Symptome wurden automatisch angeordnet oder die automatische Anordnung wurde rückgängig gemacht.
- ▼ 21: Bewegen einer Sphäre
2019-02-26 12:14:05
Sphäre: "Gesellschaft". Alte Koordinaten: x = 20 y = 20, neue Koordinaten: x = 203 y = 125.
- ▼ 22: Bewegen von Symptomen
2019-02-26 12:14:14
- ▼ Symptome
 - "Symptom 2". Alte Koordinaten: x = 270 y = 258, neue Koordinaten: x = 262 y = 301
- ▼ 23: Änderung der Beschriftung eines Symptoms



3.61.2 Fazit

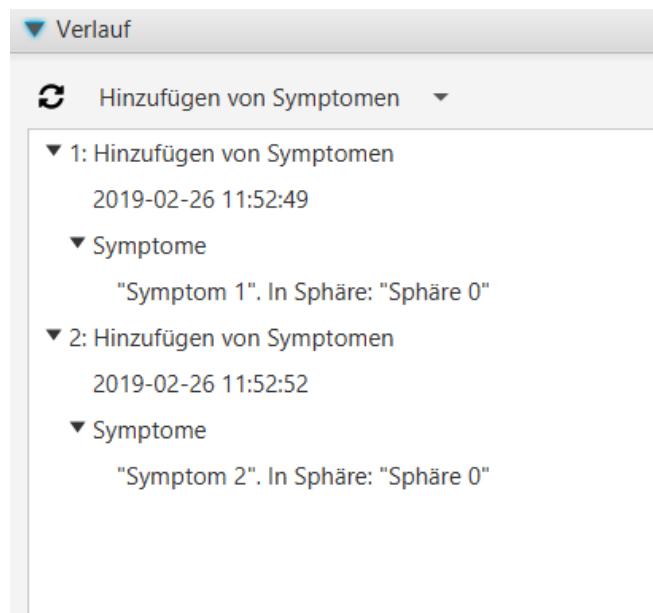
Die Software arbeitet wie erwartet.

3.62 Verlauf filtern

Der Bearbeiter wählt im Menü zum Filtern des Verlaufs **Hinzufügen von Symptomen** aus. Hierbei wird erwartet, dass nur die entsprechenden Einträge angezeigt werden. Ebenso wird erwartet, dass die Nummerierung neu berechnet wird, die Einträge also nicht mehr ihre vorherigen Nummern haben (sofern es davor andere Nummern waren).

3.62.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 27.02.2019 um 23:34
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 14:34
Test negativ.



3.62.2 Fazit

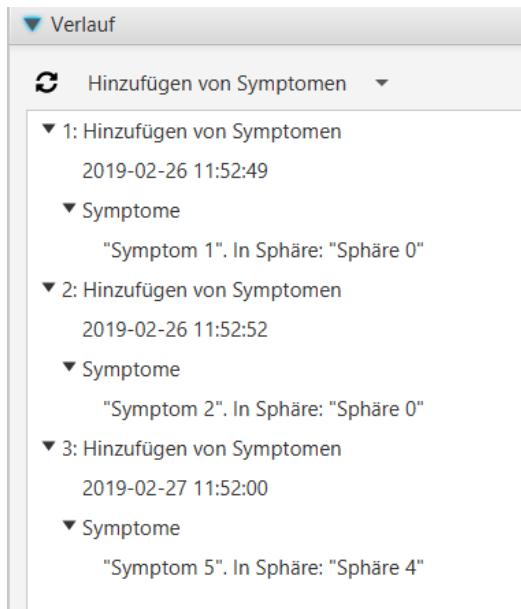
Die Software arbeitet wie erwartet.

3.63 Graph ändern und Verlauf neu laden.....

Der Bearbeiter erstellt eine Sphäre und fügt ihr ein Symptom hinzu. Anschließend betätigt er den Button zum Neuladen des Verlaufs. Hierbei wird erwartet, dass nun drei Einträge bezüglich des Hinzufügens von Symptomen im Verlauf stehen.

3.63.1 Notizen

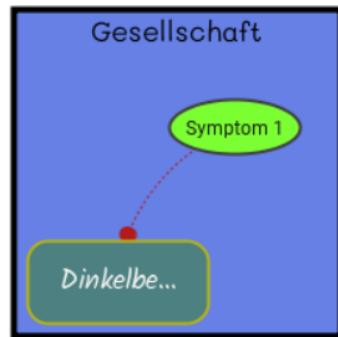
1. Test: Getestet von: Anthony Mendil. Datum: 27.02.2019 um 23:36
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 14:38
Test negativ.



3.63.2 Fazit

Die Software arbeitet wie erwartet.

Nun betätigen wir den Button zum Rückgängigmachen der letzten Aktion drei mal. Dadurch sollte das vorherige Löschen der Elemente wieder rückgängig gemacht werden. Auf diesem Graph soll im folgenden die Übersicht getestet werden.

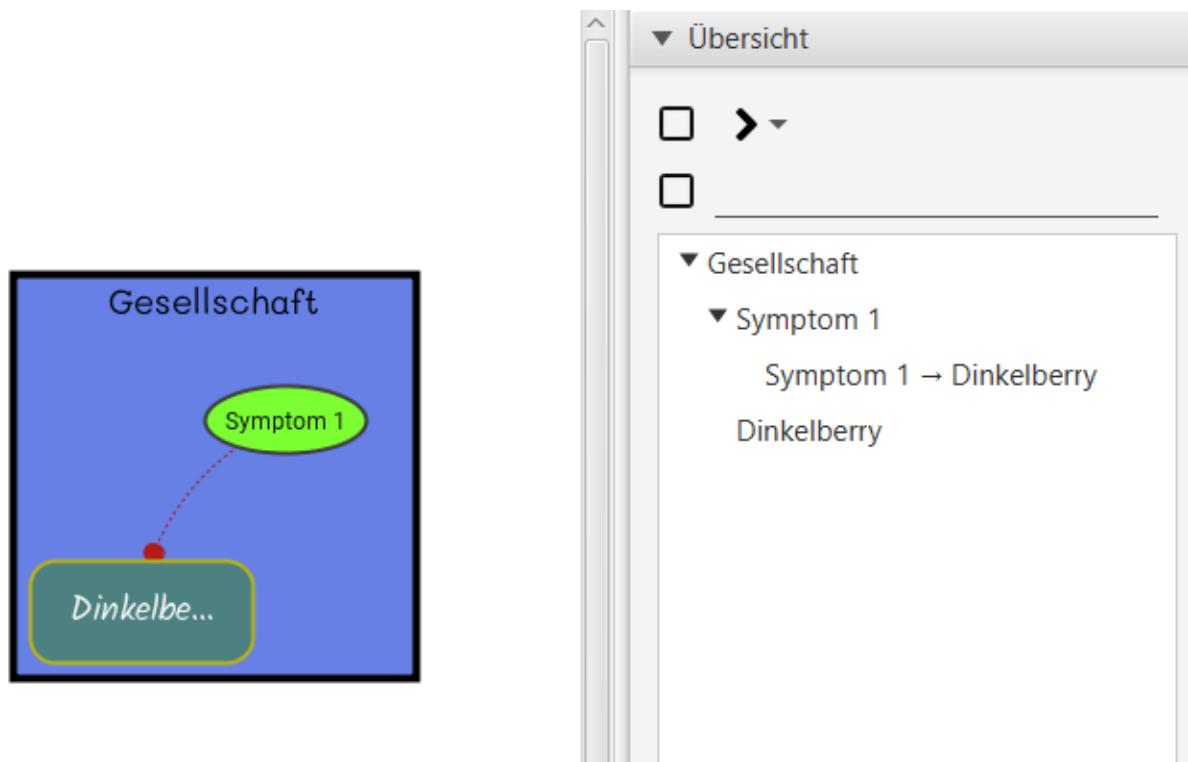


3.64 Anzeige der kompletten Übersicht

Der Bearbeiter öffnet in der Seitenleiste die Übersicht. Es wird erwartet, dass diese vollständig und korrekt ist.

3.64.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 10:16
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 14:40
Test negativ.



3.64.2 Fazit

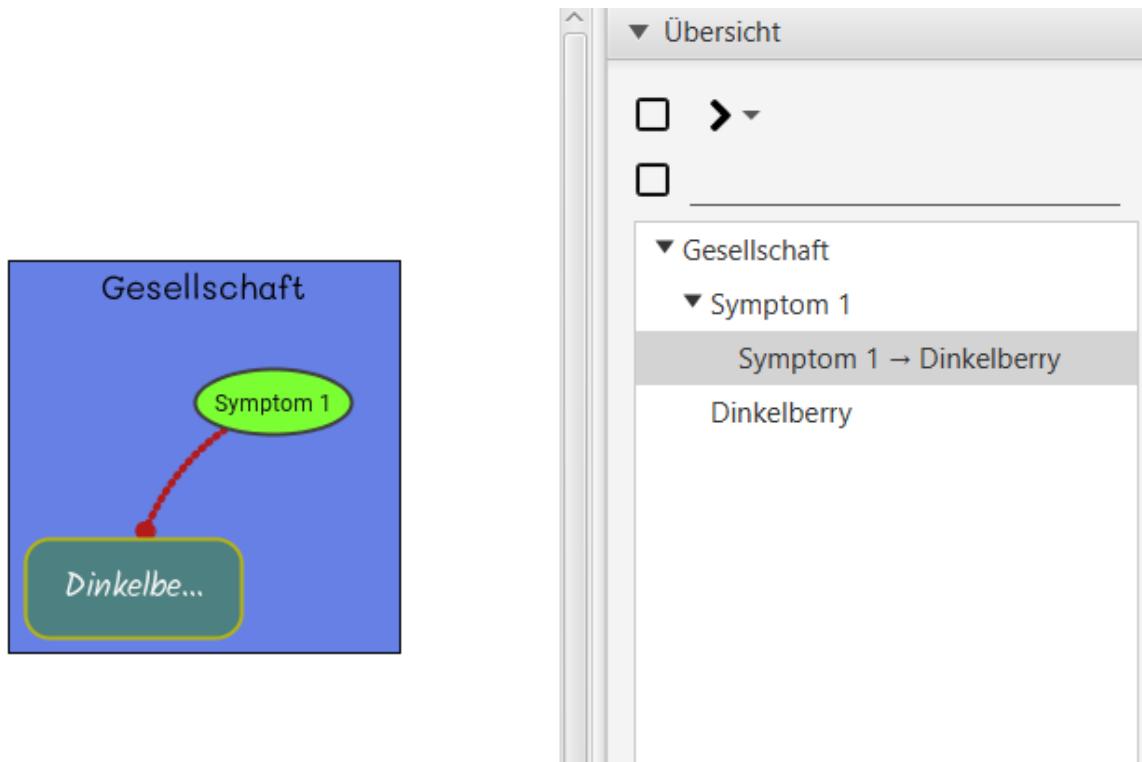
Die Software arbeitet wie erwartet.

3.65 Relation in der Übersicht auswählen

Der Bearbeiter macht einen Linksklick in der Übersicht auf die Relation von **Symptom 1** zu **Dinkelberry**. Es wird erwartet, dass die entsprechende Relation im Graphen ausgewählt wird.

3.65.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 10:49
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 14:41
Test negativ.



3.65.2 Fazit

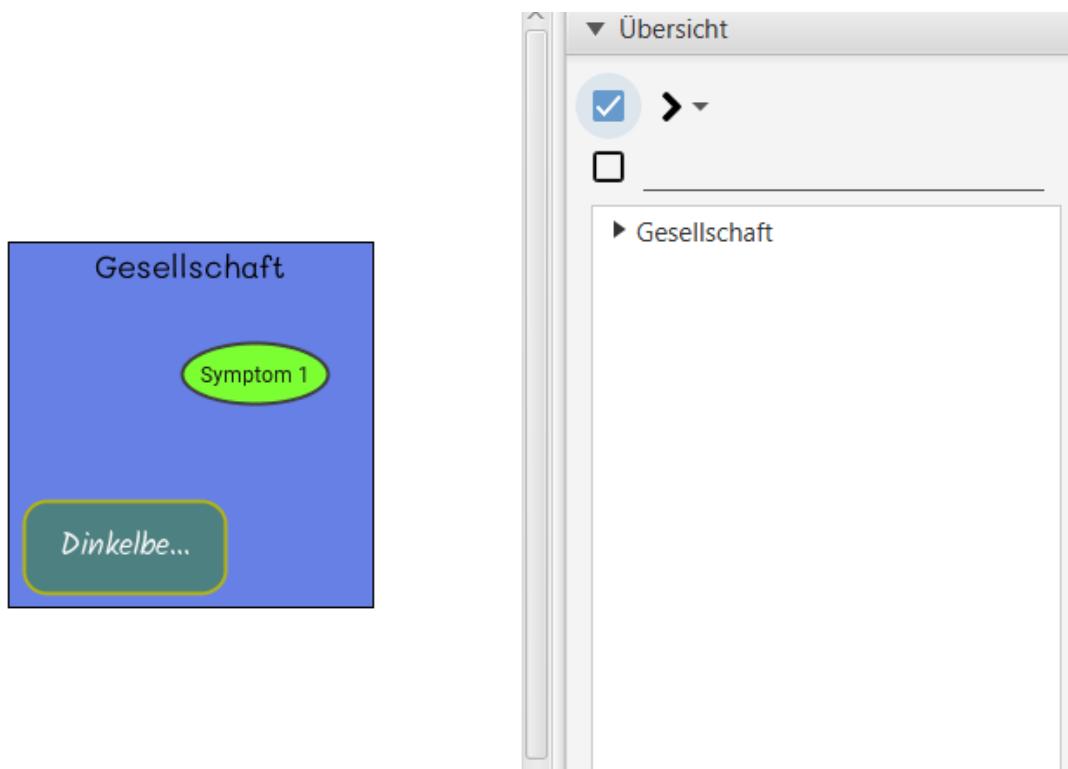
Die Software arbeitet wie erwartet.

3.66 Anzeigen ausschließlich verstärkender Relationen ☐☐☐

Standardmäßig sind in der Seitenleiste im Drop-Down Menü zum Filtern von Relationen bereits verstärkende Relationen ausgewählt. Der Bearbeiter klickt also lediglich auf die daneben liegende Checkbox. Hierbei wird erwartet, dass die abschwächende Relation von Symptom 1 zu Dinkelberry nicht mehr angezeigt wird.

3.66.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 10:58
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 14:42
Test negativ.



3.66.2 Fazit

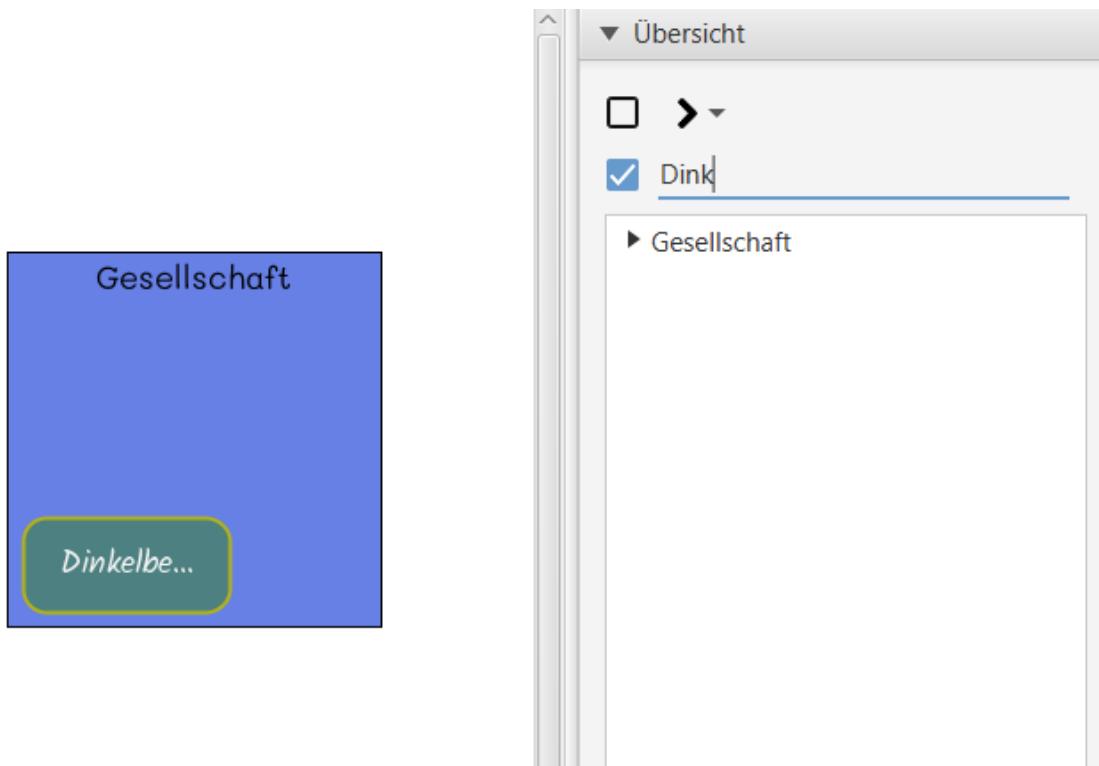
Die Software arbeitet wie erwartet.

3.67 Textsuche

Der Bearbeiter gibt im Textfeld in der Seitenleiste Dink ein. Hierbei wird erwartet, dass nur noch das Symptom Dinkelberry angezeigt wird.

3.67.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 11:03
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 14:45
Test negativ.



3.67.2 Fazit

Die Software arbeitet wie erwartet.

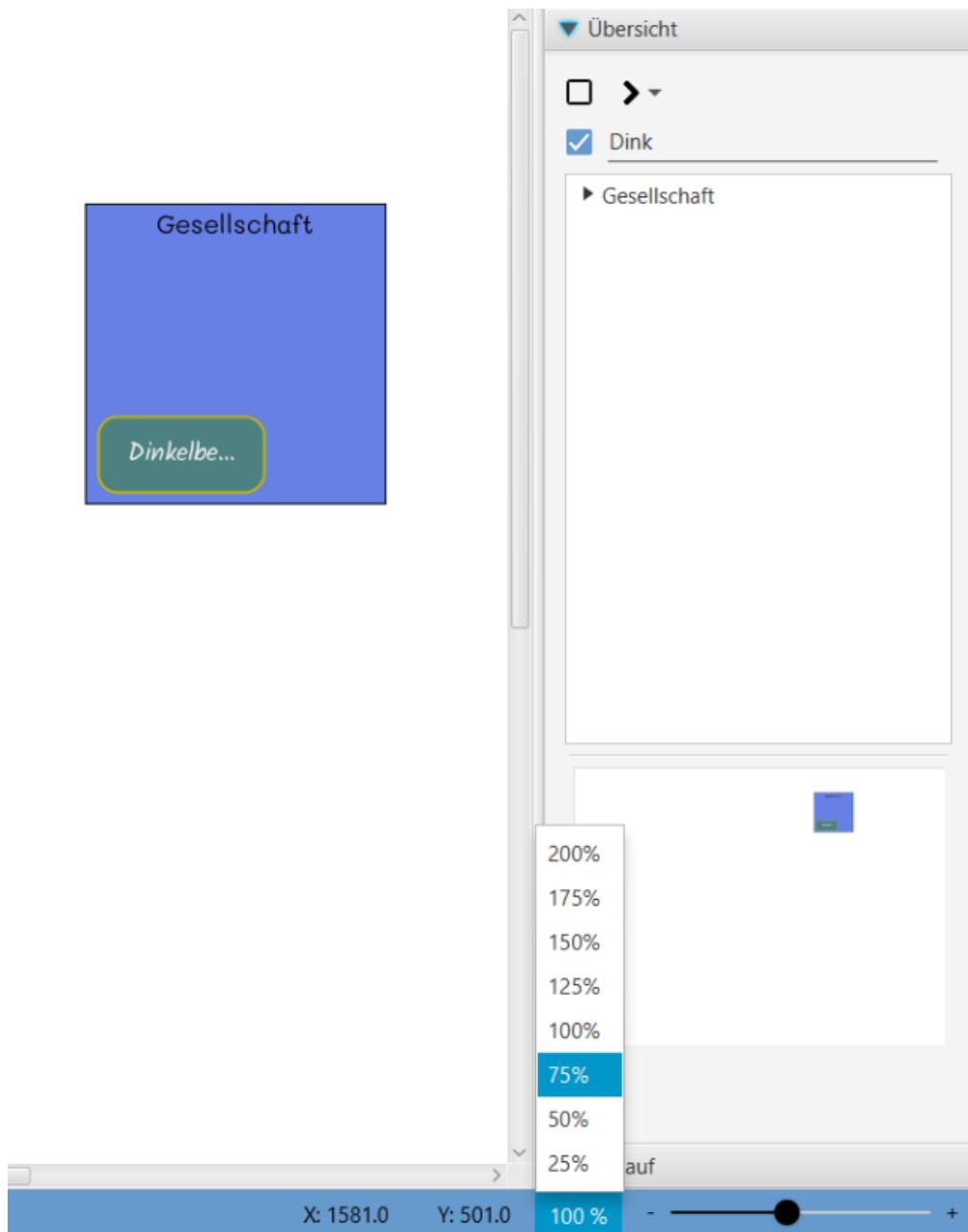
3.68 Zoom Auswahl ☐☐□

Der Bearbeiter klickt auf das Feld für die Auswahl des Zooms und drückt auf 75%. Hierbei wird erwartet, dass die Ansicht des Graphen entsprechend verkleinert wird. Ebenso sollte sich der rechts daneben liegende Balken entsprechend verändern.

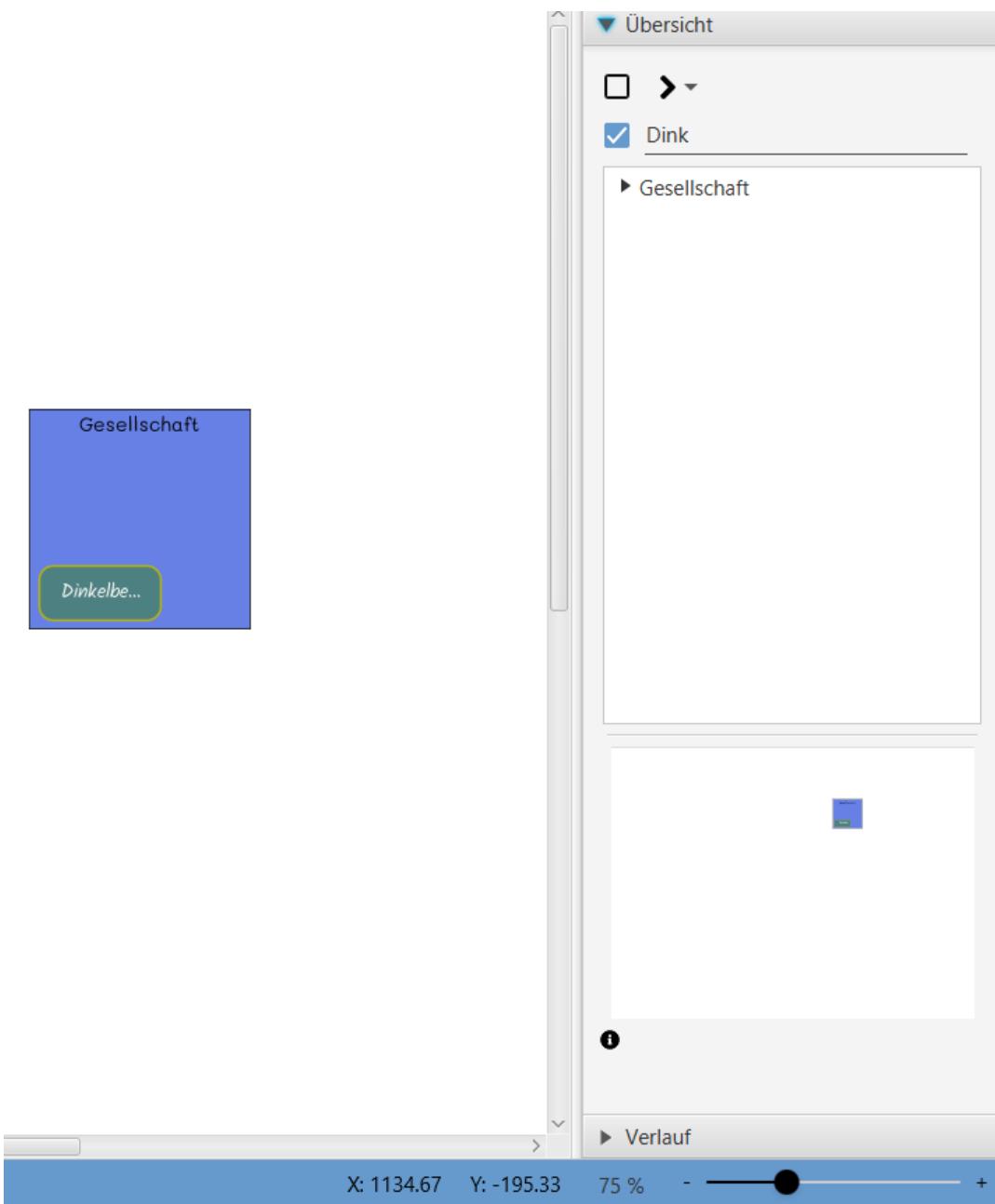
3.68.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 11:10
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 14:46
Test negativ.

Vor dem Klicken:



Nach dem Klicken:



3.68.2 Fazit

Die Software arbeitet wie erwartet.

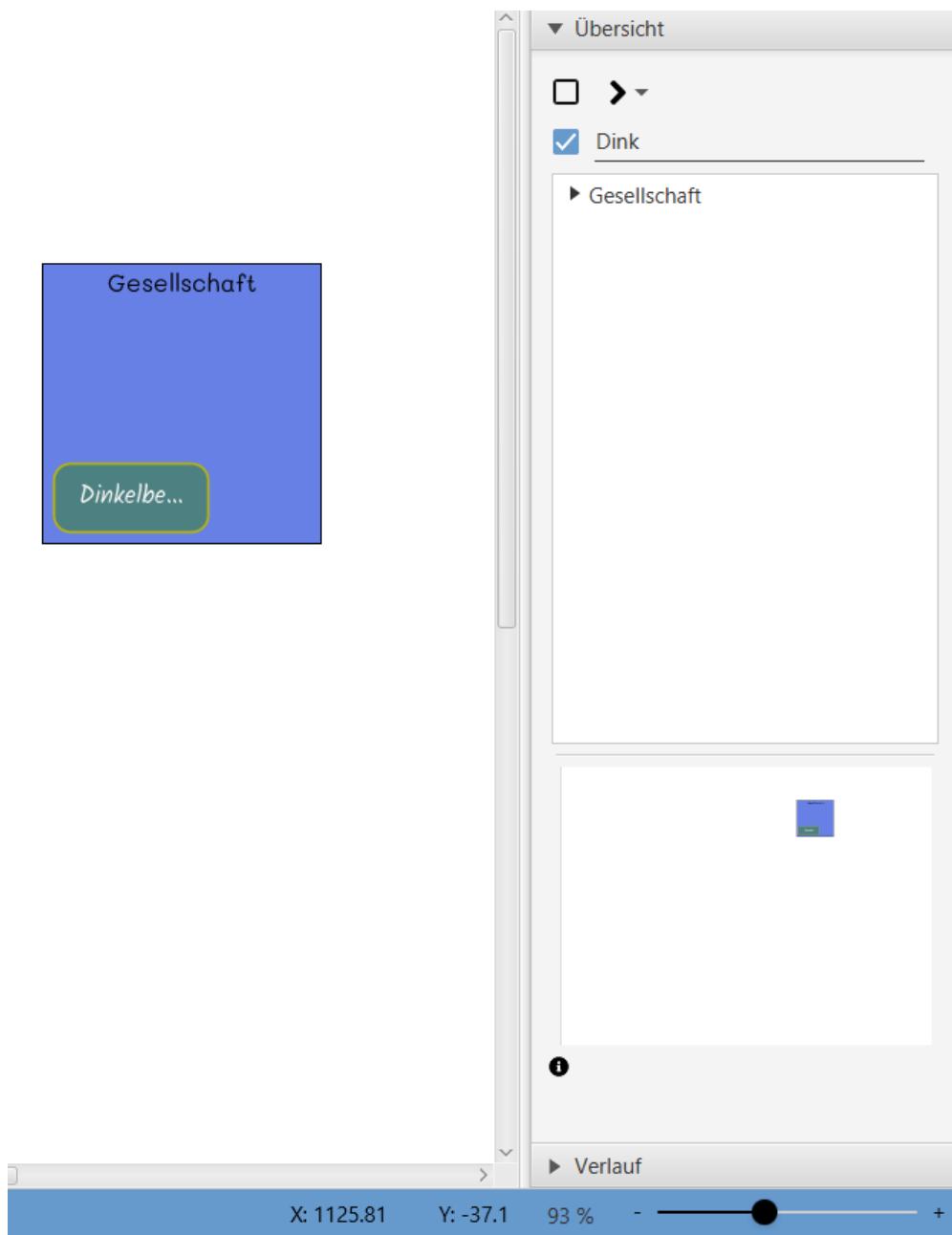
3.69 Zoom Bar

Der Bearbeiter stellt in der Zoom Bar den Zoom auf 93%. Hierbei wird erwartet, dass die Ansicht des Graphen entsprechend verkleinert wird. Ebenso sollte die links daneben

liegende Anzeige aktualisiert werden.

3.69.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 11:25
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 14:48
Test negativ.



3.69.2 Fazit

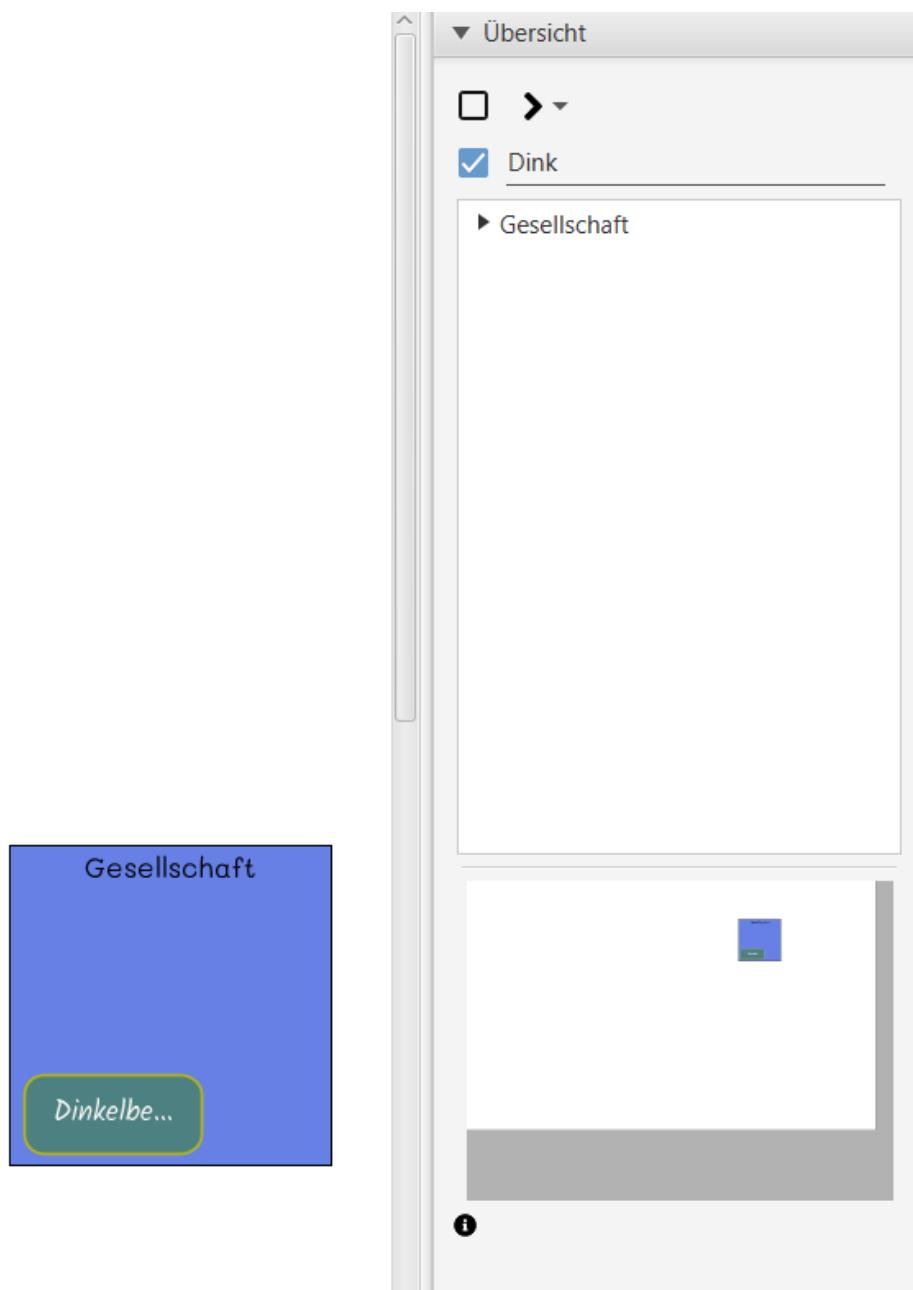
Die Software arbeitet wie erwartet.

3.70 Zoom Kontext

Der Bearbeiter bewegt den Zoom Kontext nach oben und leicht nach links. Hierbei wird erwartet, dass die Ansicht des Graphen entsprechend verschoben wird.

3.70.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 11:34
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 14:52
Test negativ.



3.70.2 Fazit

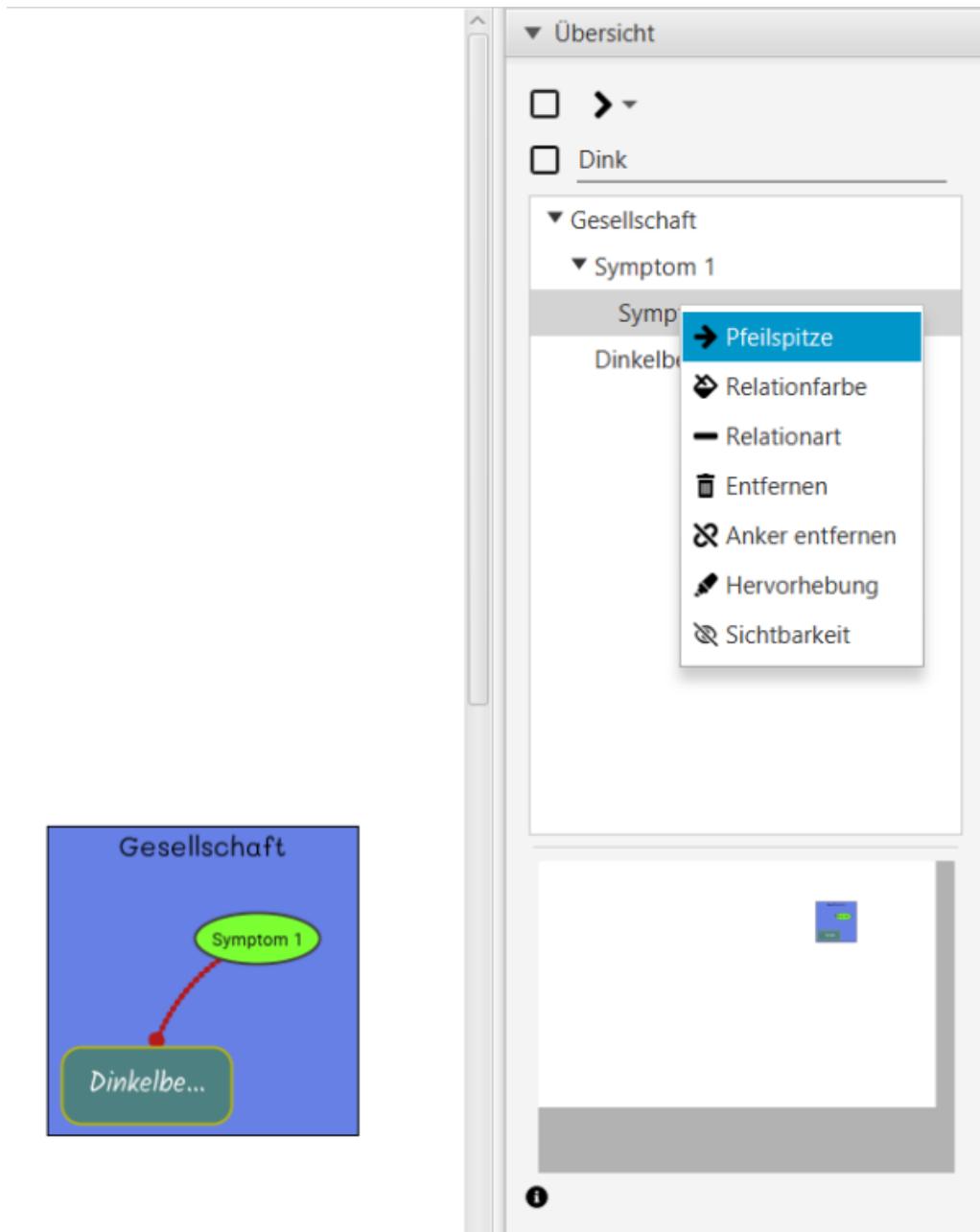
Die Software arbeitet wie erwartet.

3.71 Ändern der Relationsart einer Relation über die Übersicht ☐☐☐

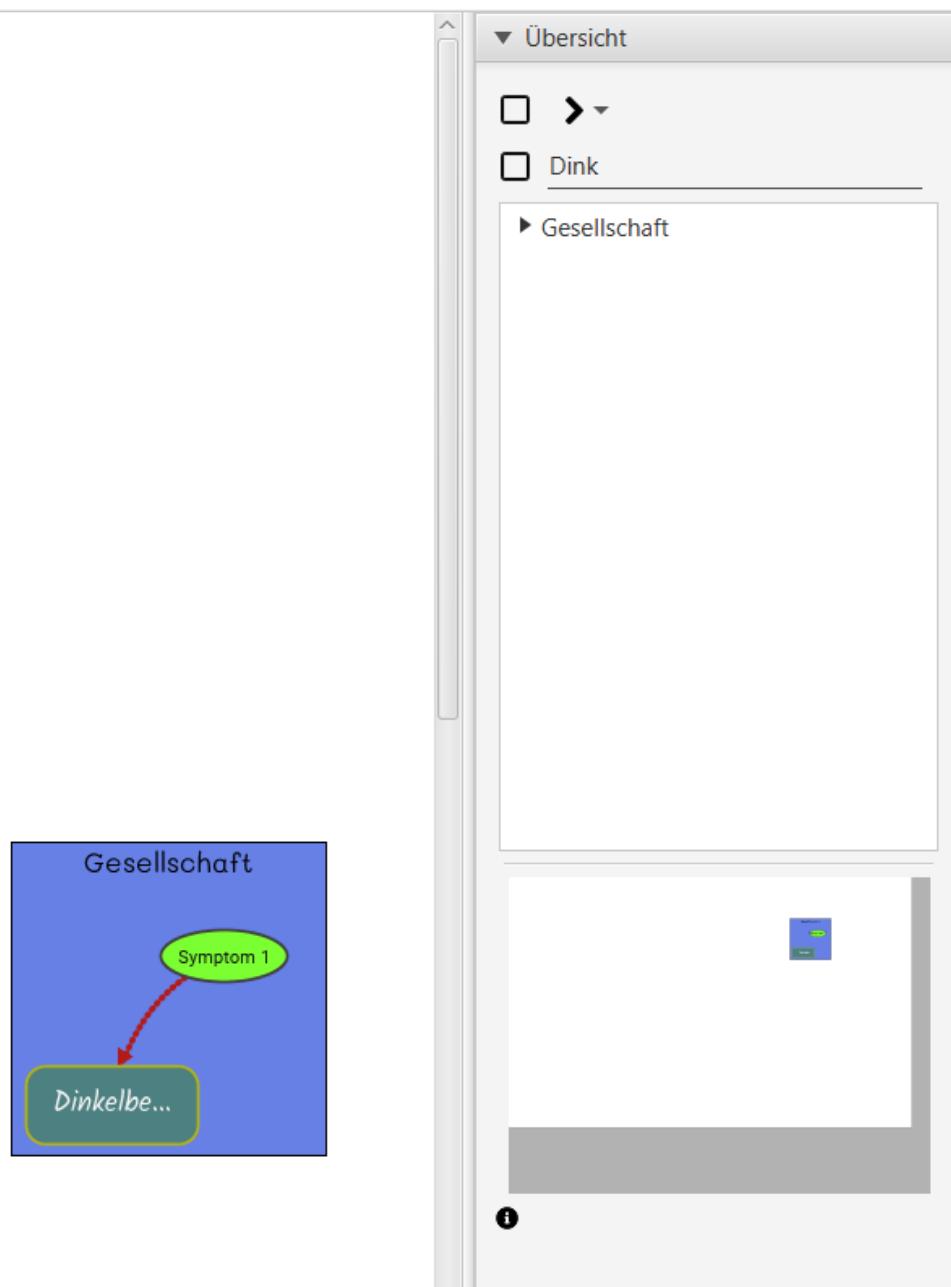
Vorerst wird oben in der Kopfleiste unter **Relation** verstärkende Relationen eingestellt. Anschließend macht der Bearbeiter einen Rechtsklick in der Übersicht auf die Relation von **Symptom 1** nach **Dinkelberry** und drückt auf Pfeilspitze. Hierbei wird erwartet, dass sich die Relationsart der Relation entsprechend ändert.

3.71.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 11:52
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 14:54
Test negativ.
Vor dem Klicken auf **Pfeilspitze**:



Nach dem Klicken auf Pfeilspitze:



3.71.2 Fazit

Die Software arbeitet wie erwartet.

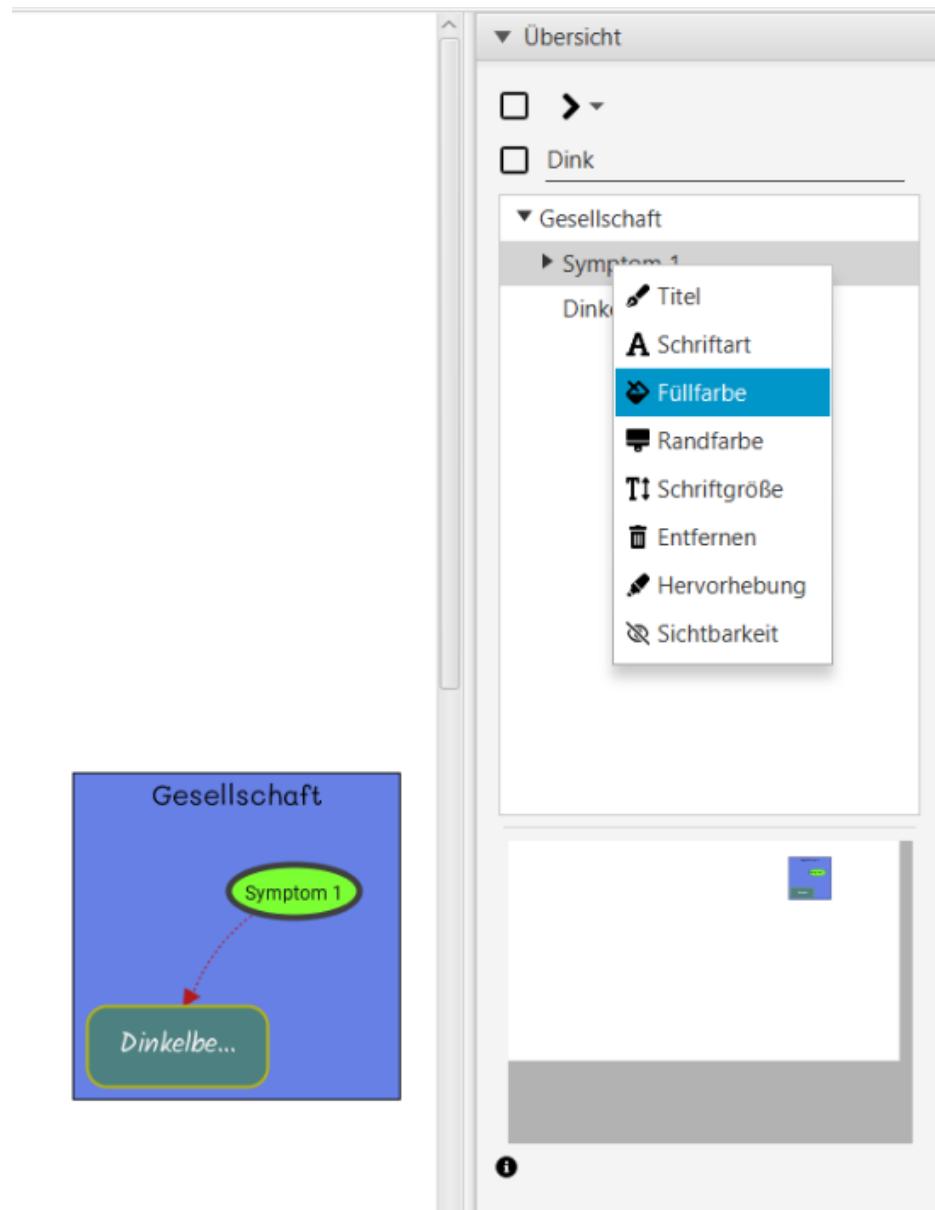
3.72 Änderung der Füllfarbe eines Symptoms über die Übersicht ☐☐☐

Vorerst wird oben in der Kopfleiste unter **Symptom** die Füllfarbe zu einen orange-rot gewechselt. Anschließend macht der Bearbeiter einen Rechtsklick in der Übersicht auf das Symptom **Symptom 1** und drückt auf **Füllfarbe**. Hierbei wird erwartet, dass sich die Füllfarbe des Symptoms entsprechend ändert.

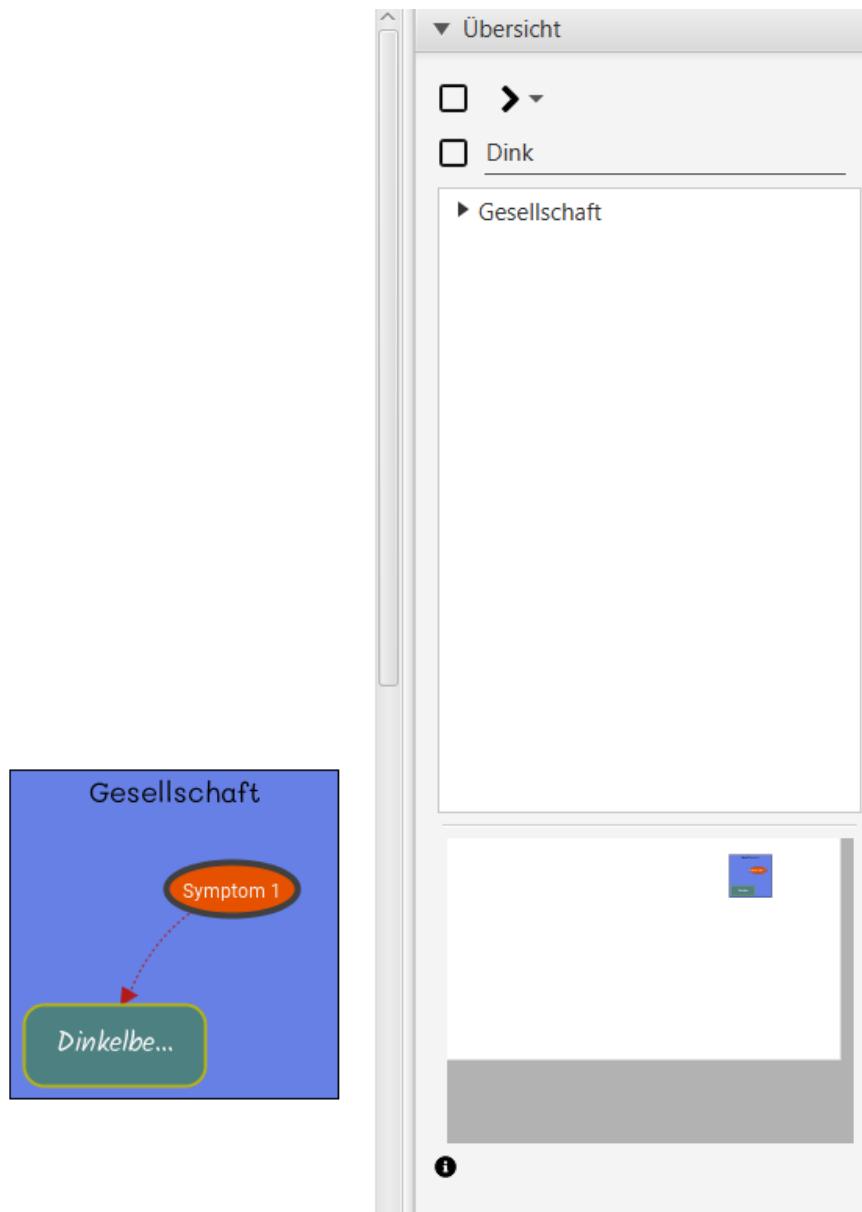
3.72.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 12:05
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 14:56
Test negativ.

Vor dem Klicken auf **Füllfarbe**:



Nach dem Klicken auf Füllfarbe:



3.72.2 Fazit

Die Software arbeitet wie erwartet.

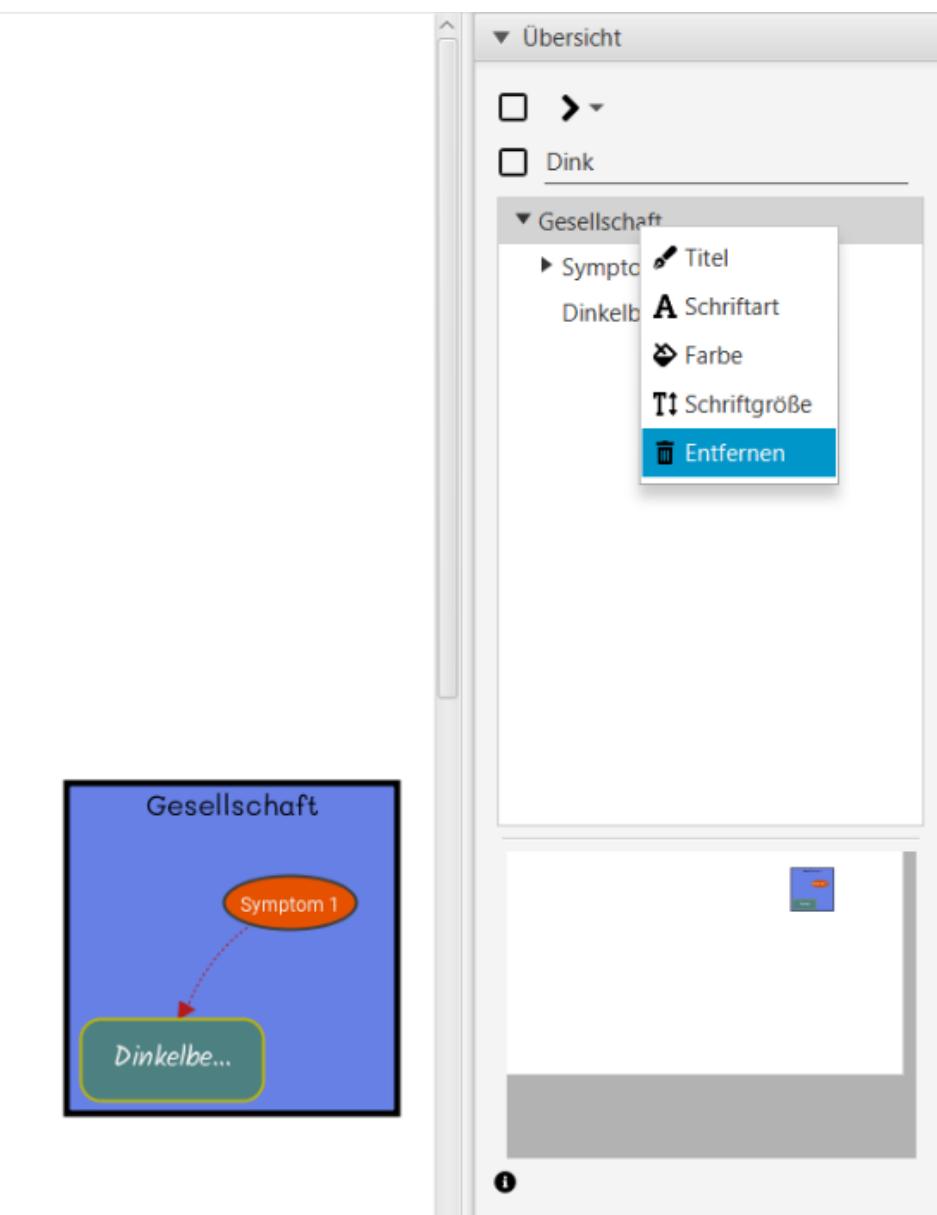
3.73 Löschen einer Sphäre über die Übersicht ☐☐☐

Der Bearbeiter macht einen Rechtsklick in der Übersicht auf **Gesellschaft** und drückt auf **Entfernen**. Hierbei wird erwartet, dass die Sphäre inklusive ihrer Symptome und der Relation entfernt wird. Ebenso darf die Sphäre in der Übersicht nicht mehr angezeigt werden.

3.73.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 12:18
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 14:57
Test negativ.

Vor dem Klicken auf Entfernen:



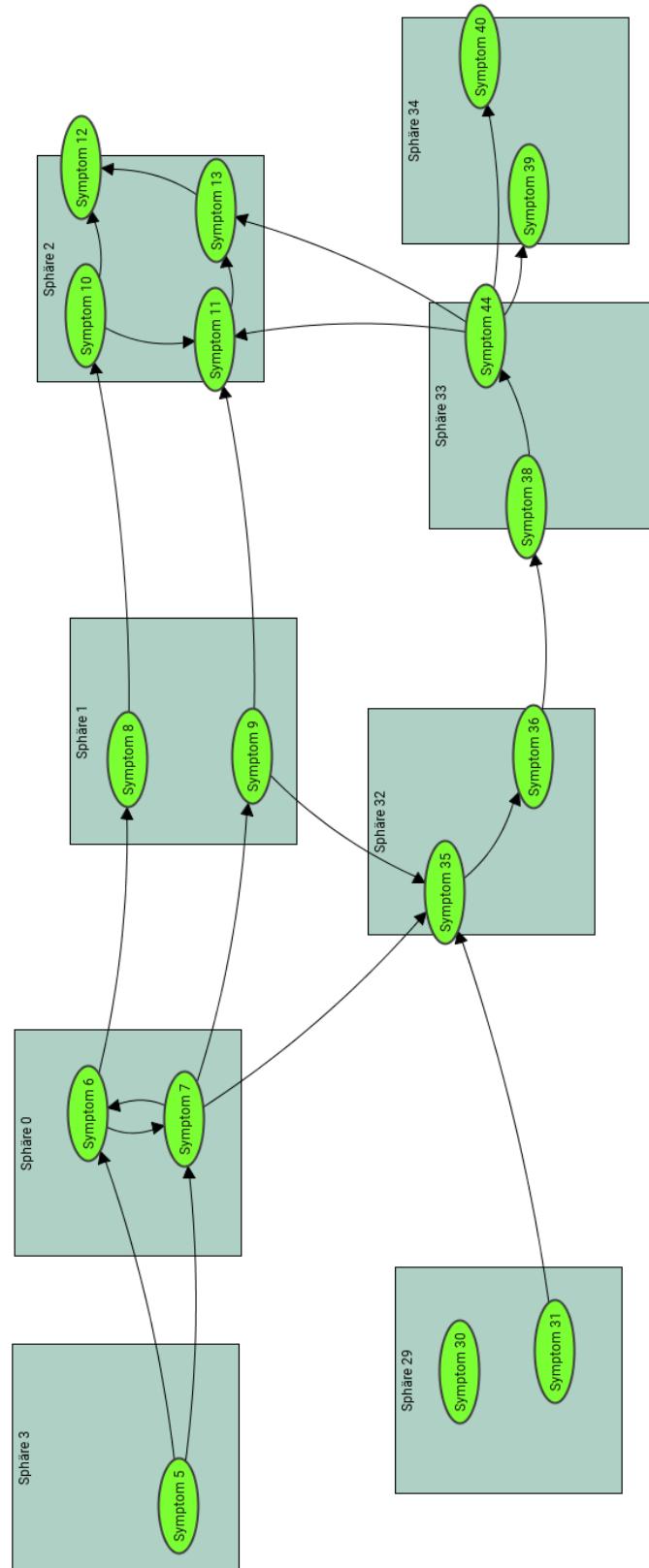
Nach dem Klicken auf Entfernen:



3.73.2 Fazit

Die Software arbeitet wie erwartet.

Zum Testen des Analyse-Modus wird erneut ein neuer Graph erstellt. Die Sphären, Symptome und Relationen die zu erstellen sind, sind aus folgender Abbildung zu entnehmen:



4 Analyse-Modus

Umfang: Anzahl wertbarer Symptome + Anzahl wertbarer Relationen

Vernetzungsindex: $2 \times \text{Anzahl wertbarer Relationen} / \text{Anzahl wertbarer Symptome}$

Strukturindex: Summe aller Pfeilketten, Verzweigungen und Kreisläufe / Anzahl wertbarer Symptome

Vorgänger: Benachbarte Knoten, die über einem Pfad mit dem ausgewählten Knoten verbunden sind, wobei der Pfad ausschließlich aus eingehenden Kanten besteht.

Nachfolger: Benachbarte Knoten, die über einem Pfad mit dem ausgewählten Knoten verbunden sind, wobei der Pfad ausschließlich aus ausgehenden Kanten besteht.

Pfeilkette: Abfolge von mindestens drei Kanten in dieselbe Richtung, ohne dass eine Verzweigung bei den inneren Knoten der Kette integriert ist. Ein Knoten einer Kette ist ein innerer, wenn er nicht der Start- und nicht der Endknoten der Kette ist.

Konvergente Verzweigung: Knoten, zu dem mindestens zwei Kanten hinführen.

Divergente Verzweigung: Knoten, aus dem mindestens zwei Kanten wegführen.

Zyklus (Kreislauf): Geschlossene Kette von Kanten, die alle in die gleiche Richtung laufen, wobei jeder Knoten im Zyklus jeden anderen Knoten im Zyklus über die Kette erreichen kann. Der kleinste Kreislauf ist eine direkte Rückkopplung.

4.1 Korrektes Berechnen und Anzeigen des Umfangs, des Vernetzungsindex und des Strukturindex

Der Bearbeiter muss nichts tun. Es wird erwartet, das links in der Kopfleiste die korrekten Werte angezeigt werden.

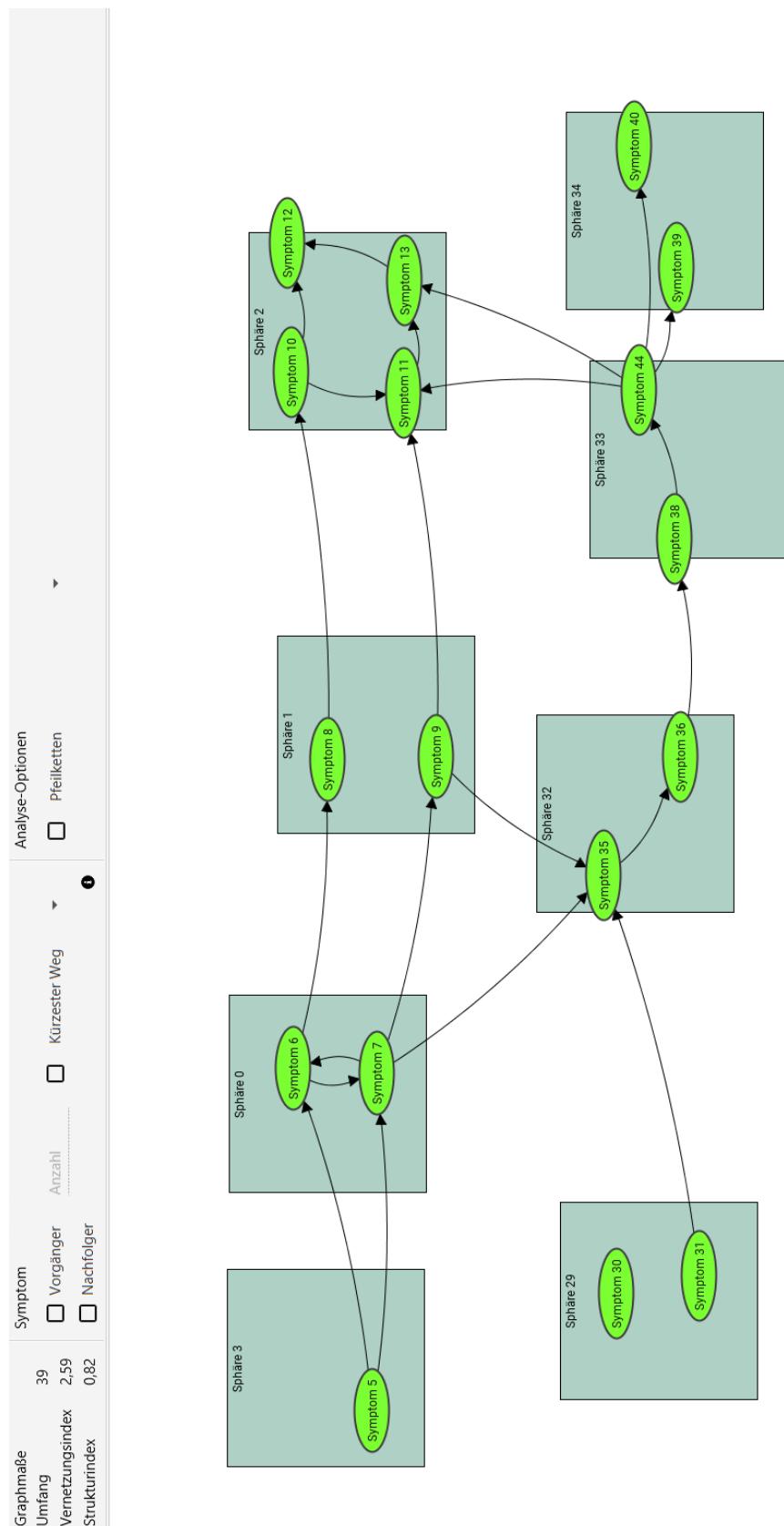
4.1.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 12:53

Test negativ.

2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 15:26

Test negativ.



4.1.2 Fazit

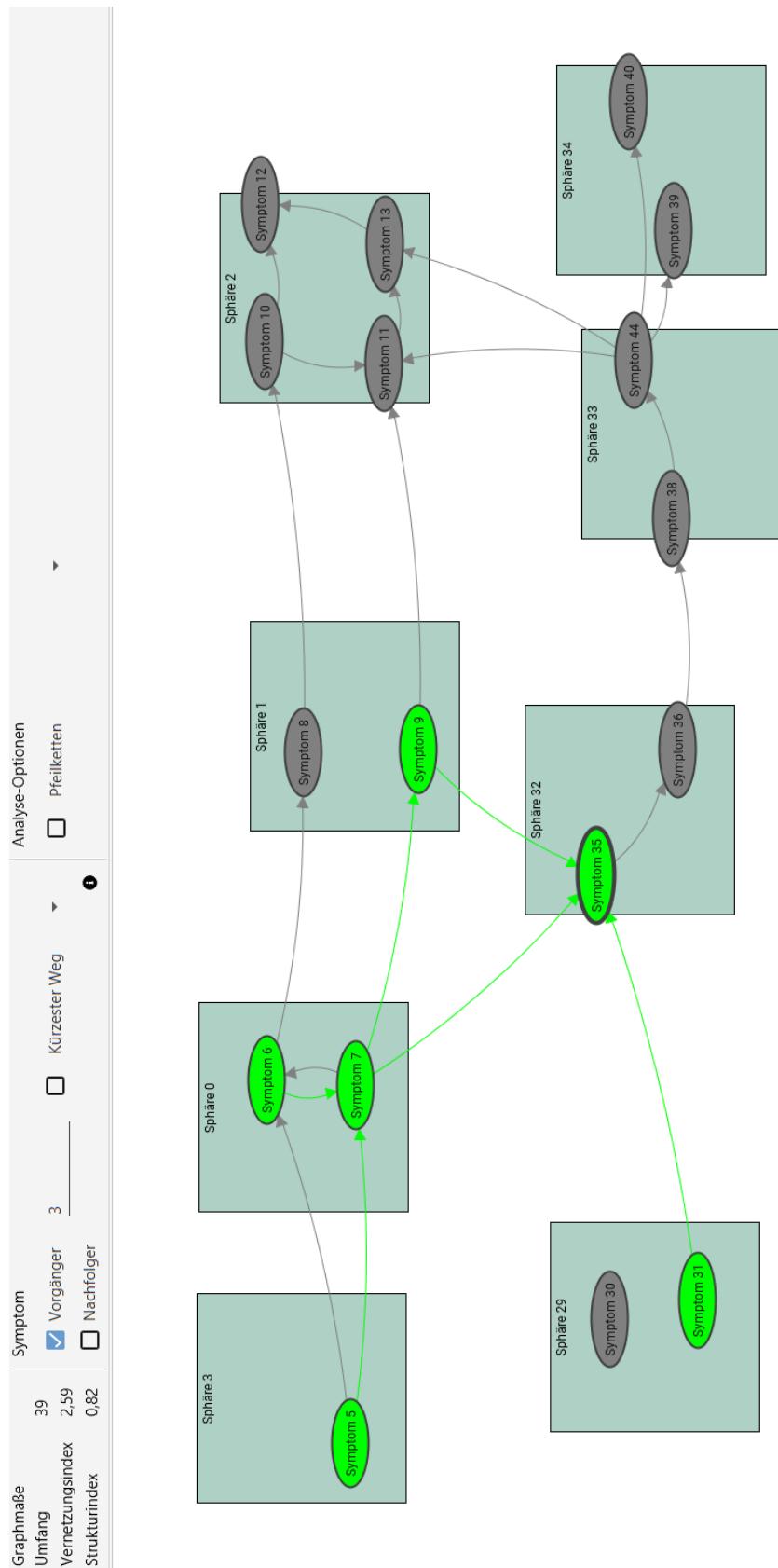
Die Software arbeitet wie erwartet.

4.2 Vorgänger von einem Symptom

Der Bearbeiter wählt **Symptom 35** aus und aktiviert in der Kopfleiste **Vorgänger**. Anschließend gibt er in dem Textfeld daneben 3 ein. Hierbei wird erwartet, dass alle Symptome hervorgehoben werden, von denen man über maximal drei Relationen zu dem ausgewählten Symptom kommt.

4.2.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 13:02
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 15:27
Test negativ.



4.2.2 Fazit

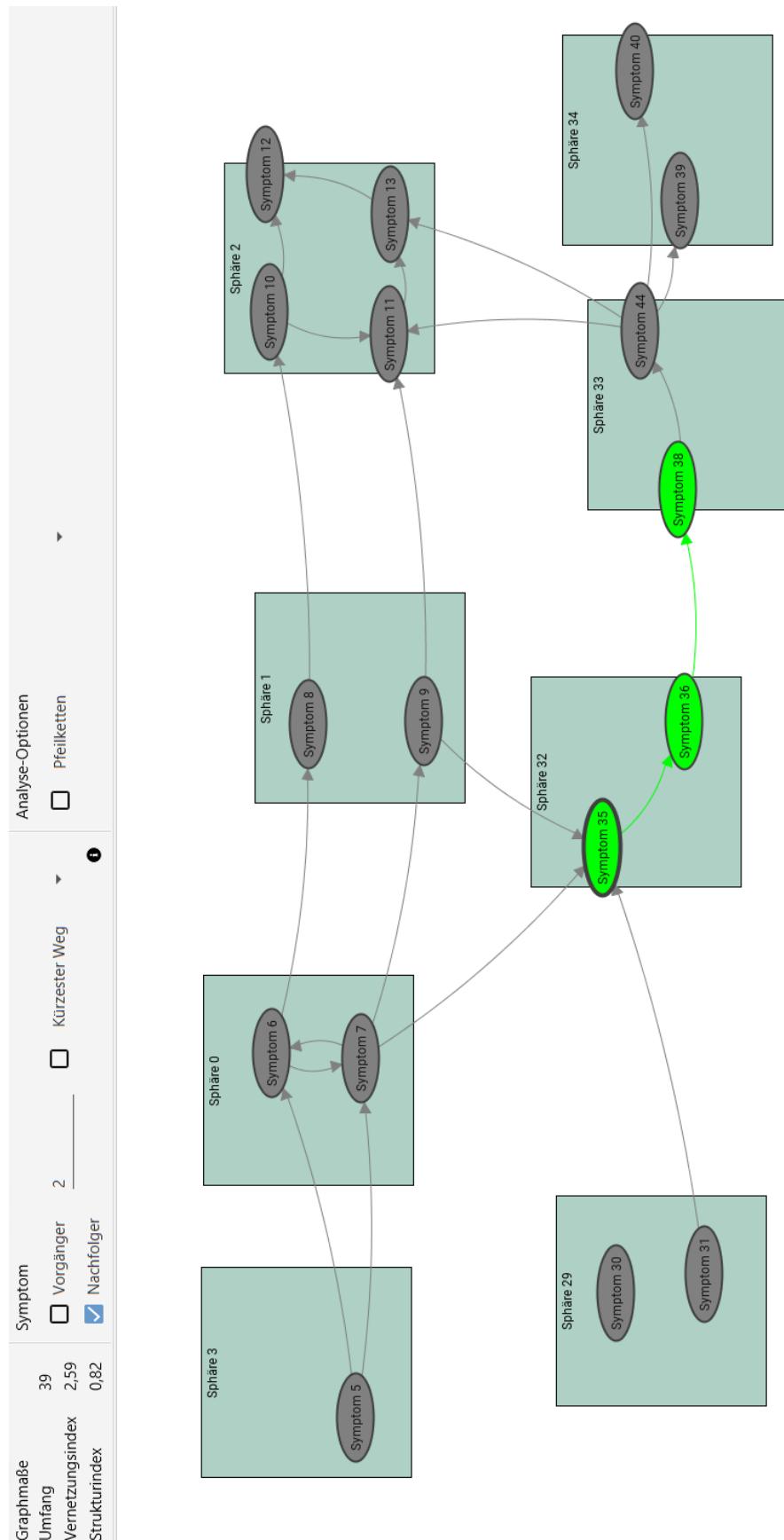
Die Software arbeitet wie erwartet.

4.3 Nachfolger von einem Symptom

Der Bearbeiter deaktiviert in der Kopfleiste **Vorgänger**, aktiviert **Nachfolger** und gibt im Textfeld 2 ein. Hierbei wird erwartet, dass alle Symptome hervorgehoben werden, zu denen man über maximal zwei Relationen von dem ausgewählten Symptom kommt.

4.3.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 13:24
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 15:27
Test negativ.



4.3.2 Fazit

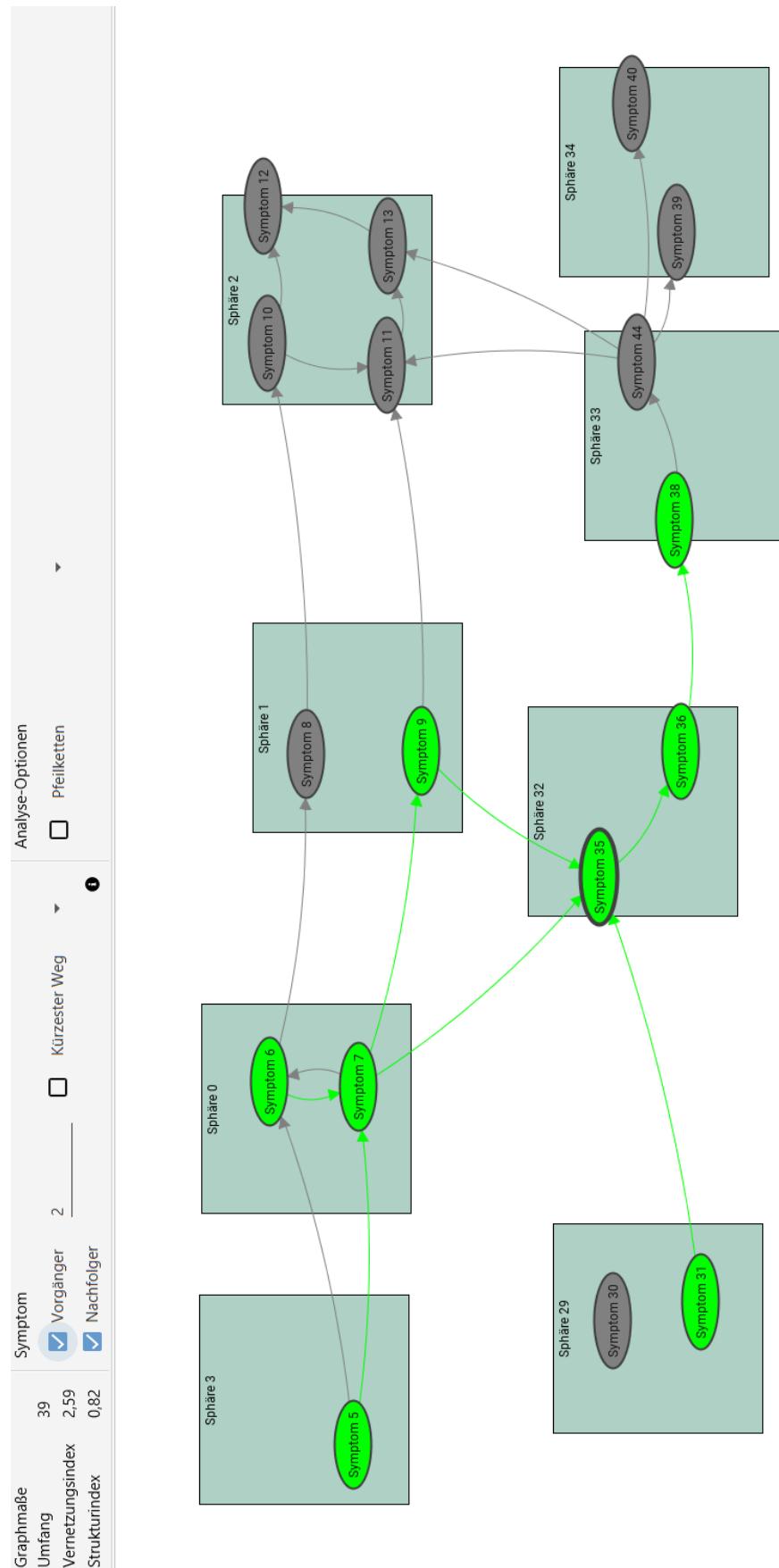
Die Software arbeitet wie erwartet.

4.4 Nachfolger und Vorgänger eines Symptoms ◻◻◻

Der Bearbeiter aktiviert in der Kopfleiste wieder **Vorgänger**. Nun sind beide, **Vorgänger** und **Nachfolger**, aktiviert und im Textfeld steht 2. Hierbei wird erwartet, dass sowohl alle Symptome, zu denen man über maximal zwei Relationen von dem ausgewählten Symptom kommt, als auch alle von denen man über maximal zwei Relationen zu dem ausgewählten Symptom kommt, hervorgehoben werden.

4.4.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 13:33
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 15:29
Test negativ.



4.4.2 Fazit

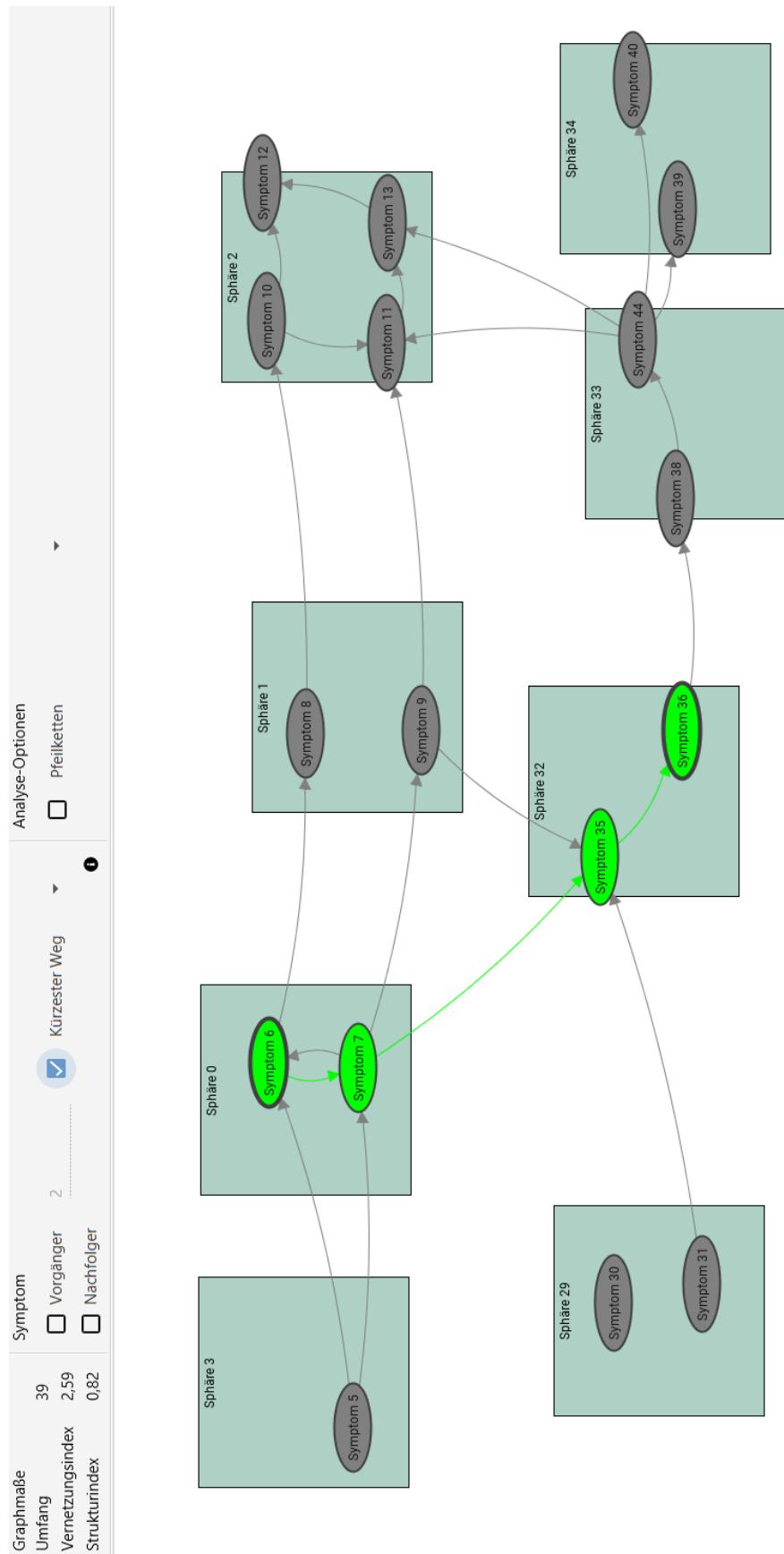
Die Software arbeitet wie erwartet.

4.5 Kürzester Weg von einem Symptom zu einem anderen (Weg existiert)

Der Bearbeiter deselektiert alle momentan ausgewählten Symptome. Anschließend wählt er **Symptom 6** und **Symptom 36** aus und aktiviert **Kürzester Weg**. Hierbei wird erwartet, dass der kürzeste Weg von **Symptom 6** nach **Symptom 36** inklusive aller auf dem Weg liegenden Symptome hervorgehoben wird.

4.5.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 13:54
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 15:30
Test negativ.



4.5.2 Fazit

Die Software arbeitet wie erwartet.

4.6 Kürzester Weg von einem Symptom zu einem anderen (Weg existiert nicht)

Der Bearbeiter wählt Symptom 36 und Symptom 6 aus und aktiviert Kürzester Weg. Hierbei wird erwartet, dass kein Weg angezeigt wird, da kein Weg von Symptom 36 nach Symptom 6 existiert.

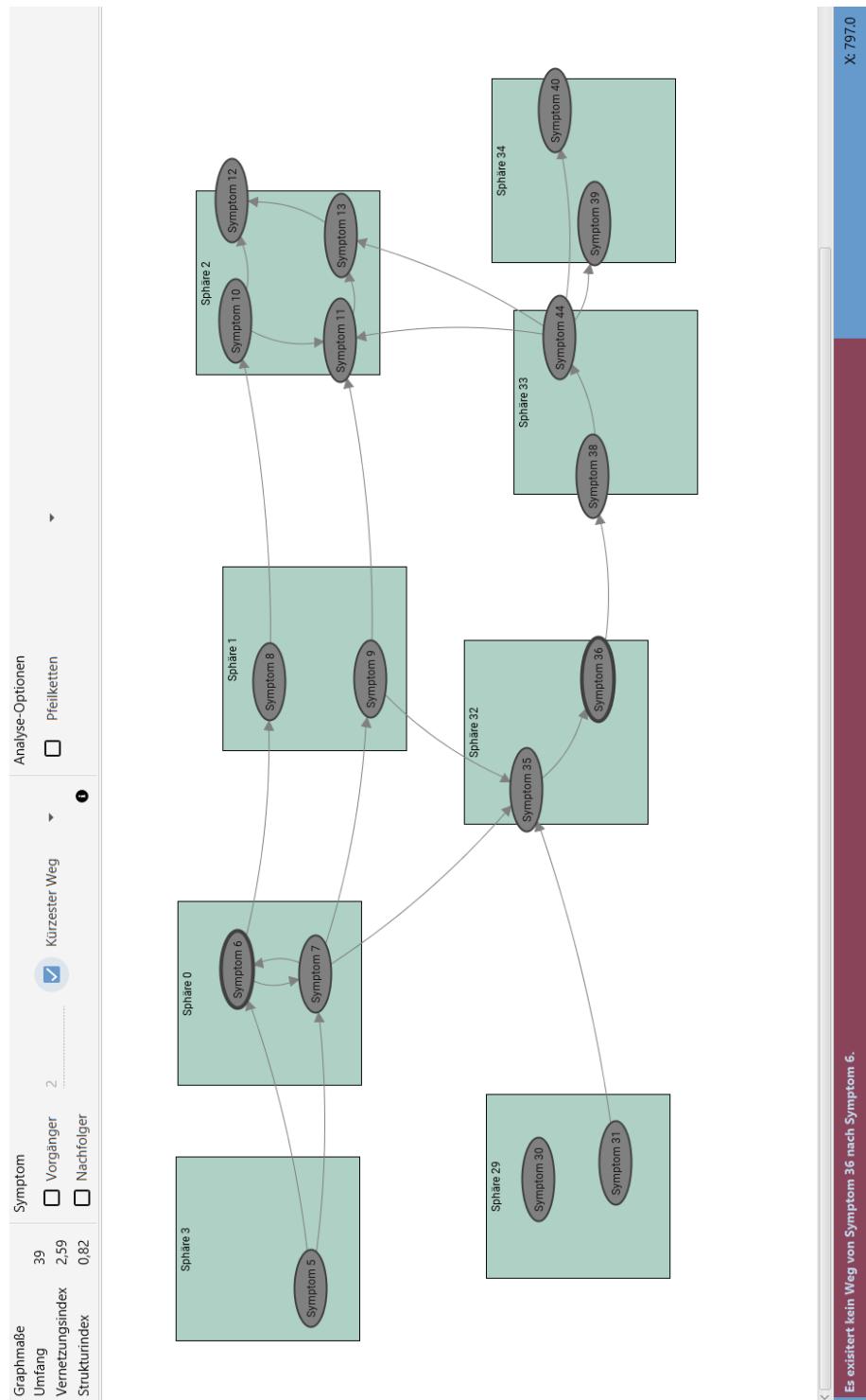
4.6.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 14:11

Test negativ.

2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 15:47

Test negativ.



4.6.2 Fazit

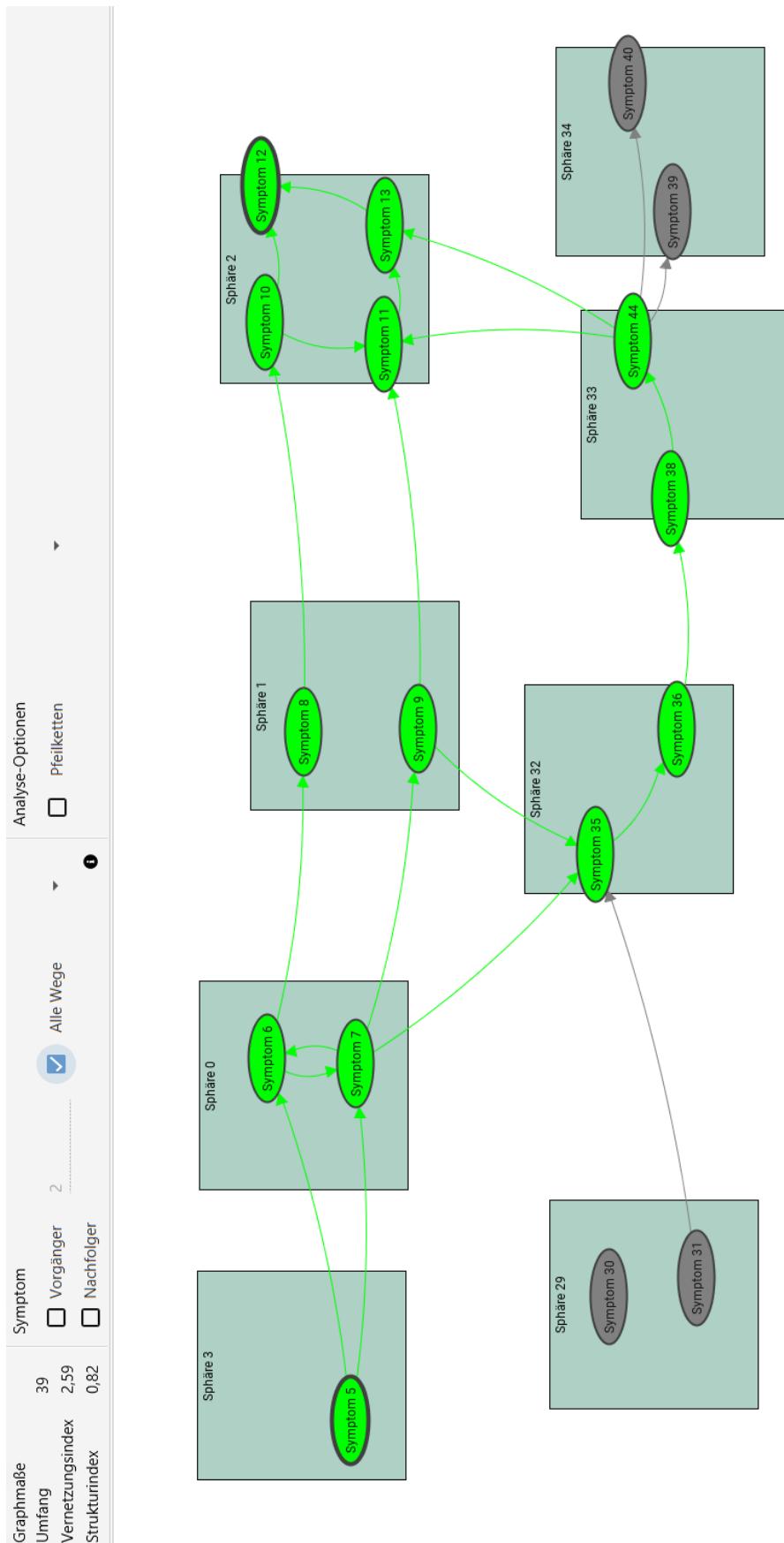
Die Software arbeitet wie erwartet.

4.7 Alle Wege von einem Symptom zu einem anderen □□□

Der Bearbeiter deselektiert alle momentan ausgewählten Symptome. Anschließend wählt er **Symptom 5** und **Symptom 12** aus und aktiviert **Alle Wege**. Hierbei wird erwartet, dass alle Wege von **Symptom 5** nach **Symptom 12** inklusive aller auf den Wegen liegenden Symptome hervorgehoben werden.

4.7.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 14:30
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 15:49
Test negativ.



4.7.2 Fazit

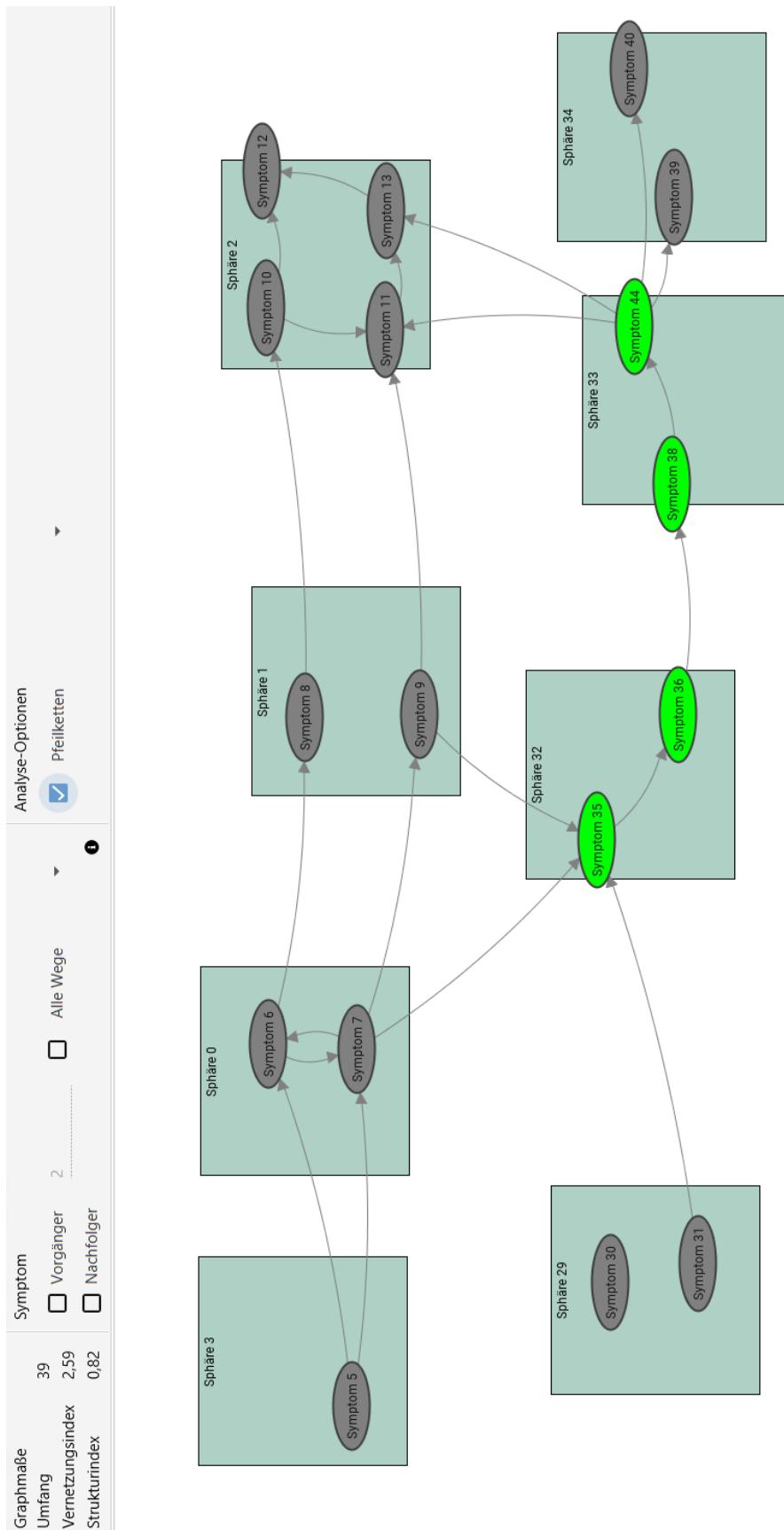
Die Software arbeitet wie erwartet.

4.8 Pfeilketten

Der Bearbeiter deselektiert alle momentan ausgewählten Symptome. Anschließend aktiviert er in der Kopfleiste **Pfeilketten**. Hierbei wird erwartet, dass alle Pfeilketten hervorgehoben werden.

4.8.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 15:29
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 15:50
Test negativ.



4.8.2 Fazit

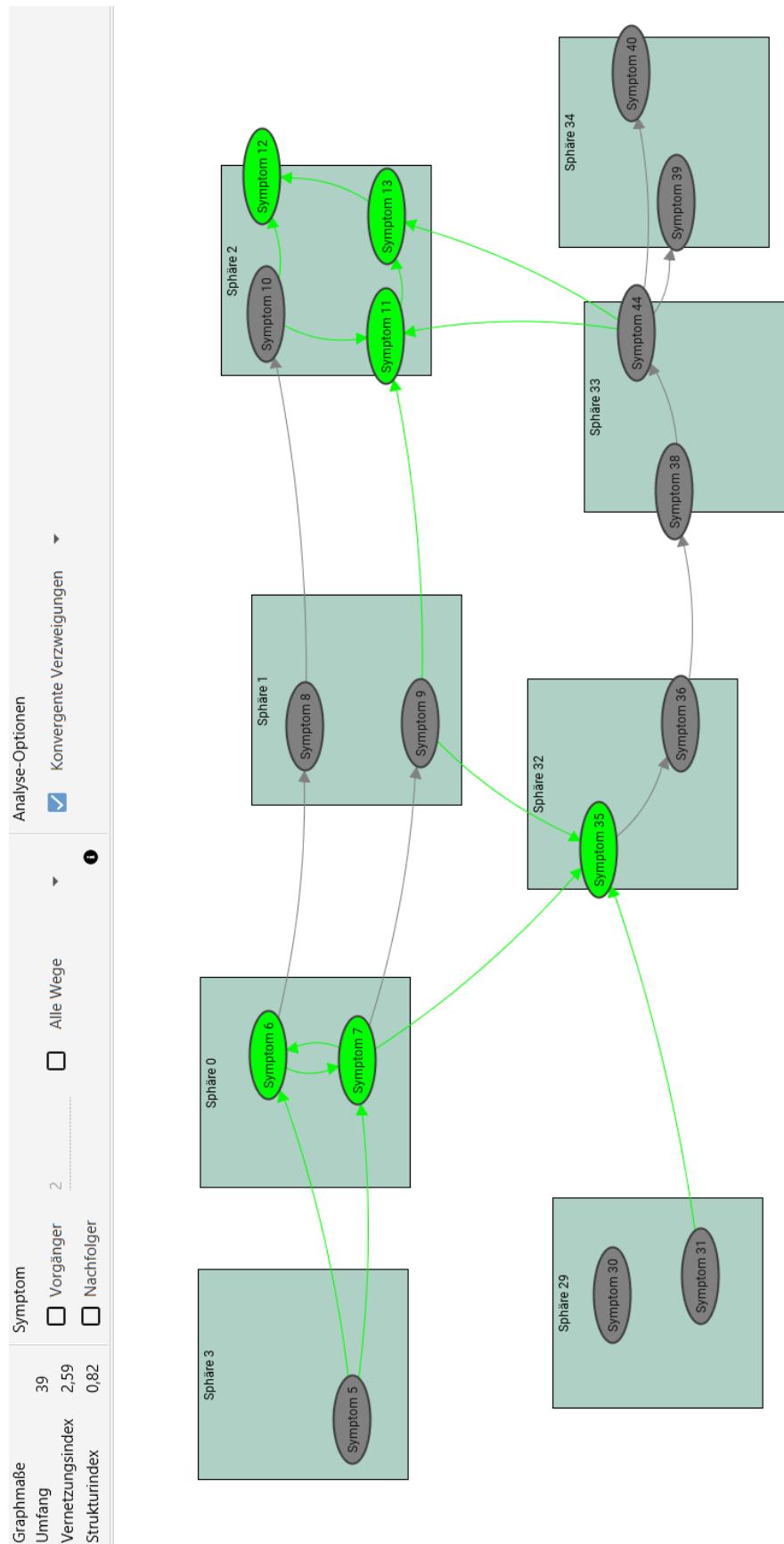
Die Software arbeitet wie erwartet.

4.9 Konvergente Verzweigungen

Der Bearbeiter aktiviert er in der Kopfleiste Konvergente Verzweigungen. Hierbei wird erwartet, dass alle konvergenten Verzweigungen hervorgehoben werden.

4.9.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 15:35
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 15:50
Test negativ.



4.9.2 Fazit

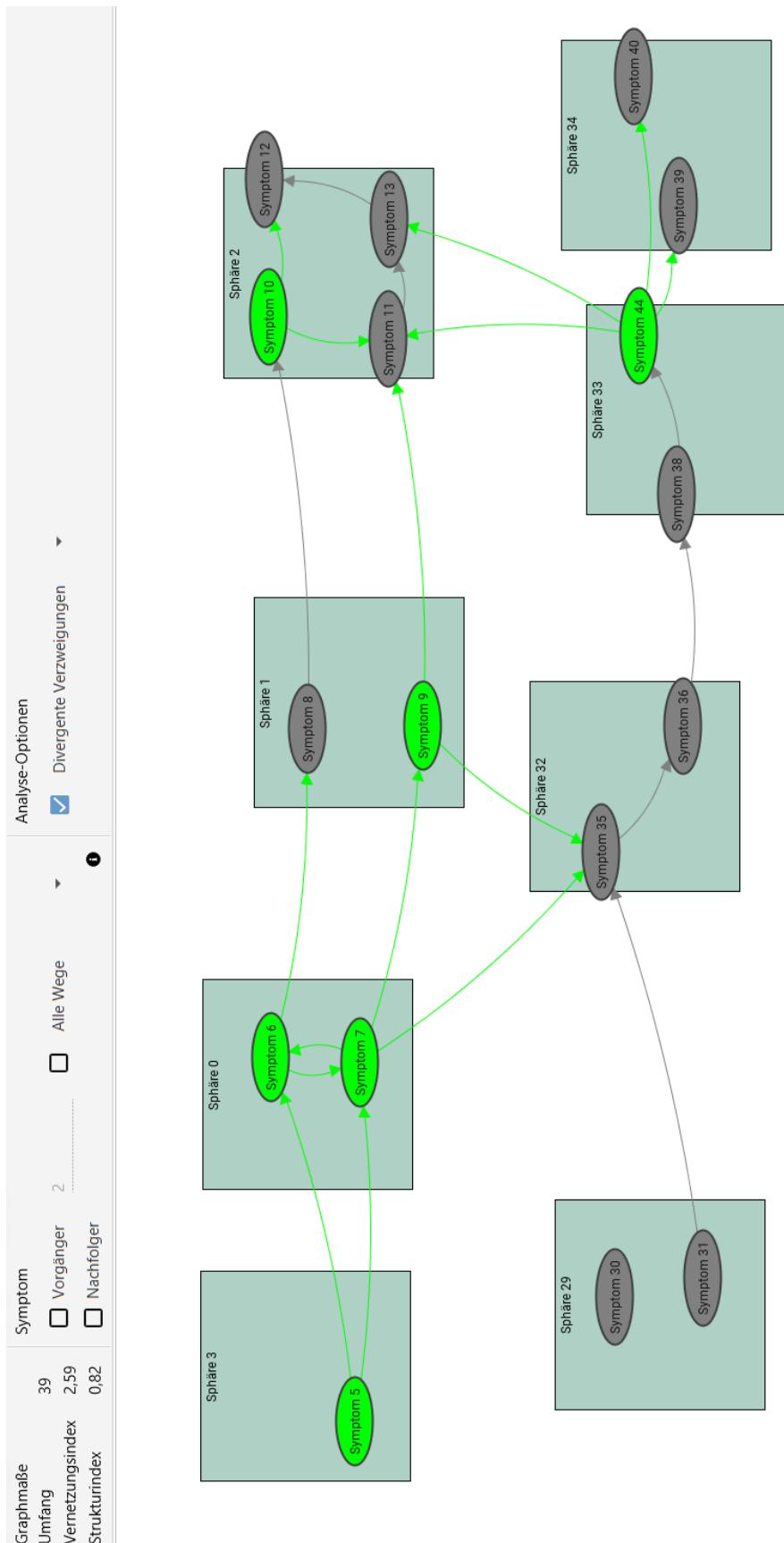
Die Software arbeitet wie erwartet.

4.10 Divergente Verzweigungen

Der Bearbeiter aktiviert er in der Kopfleiste Divergente Verzweigungen. Hierbei wird erwartet, dass alle divergenten Verzweigungen hervorgehoben werden.

4.10.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 15:39
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 15:51
Test negativ.



4.10.2 Fazit

Die Software arbeitet wie erwartet.

4.11 Verzweigungen

Der Bearbeiter aktiviert er in der Kopfleiste **Verzweigungen**. Hierbei wird erwartet, dass alle Verzweigungen hervorgehoben werden.

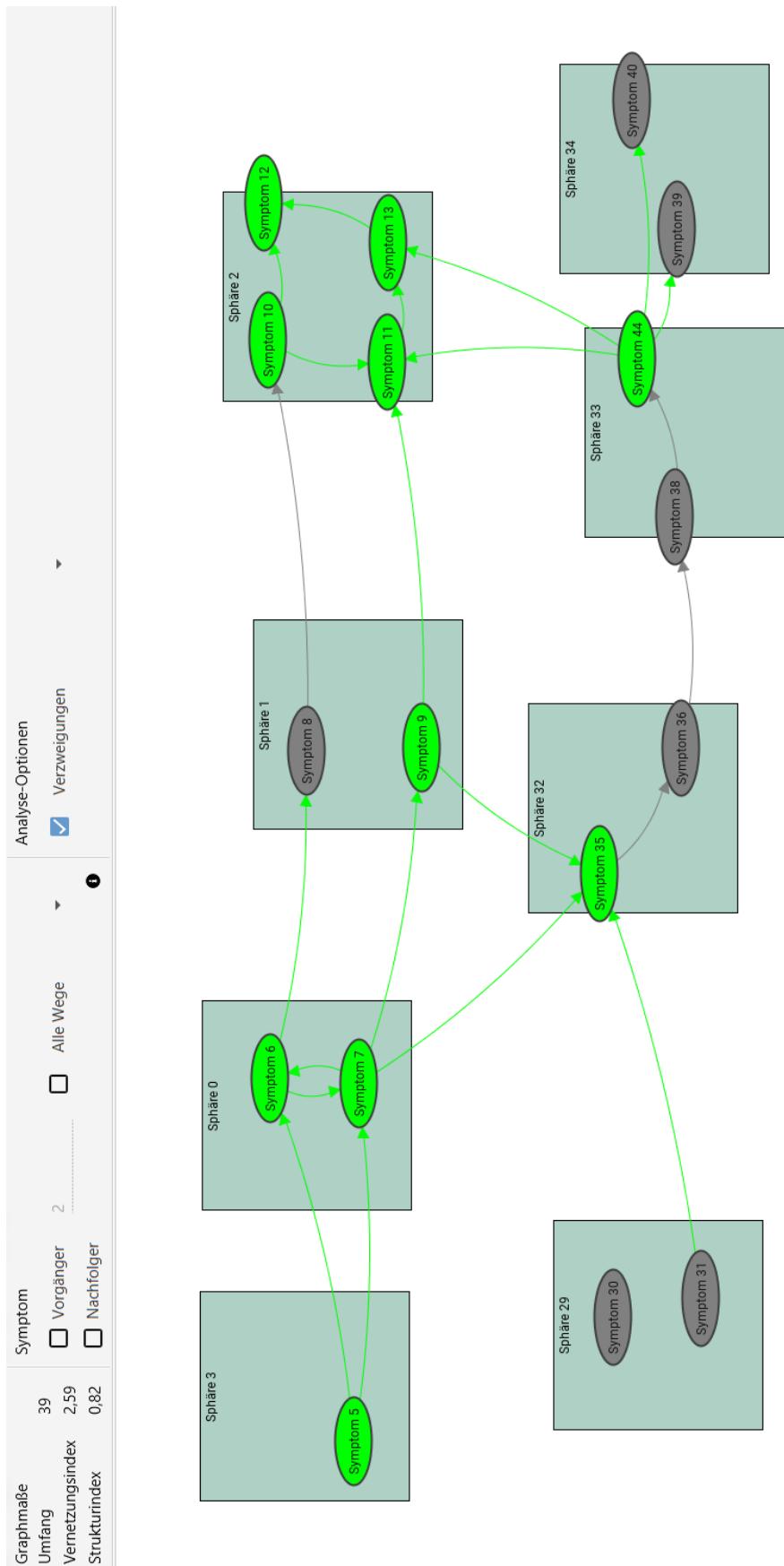
4.11.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 15:41

Test negativ.

2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 15:51

Test negativ.



4.11.2 Fazit

Die Software arbeitet wie erwartet.

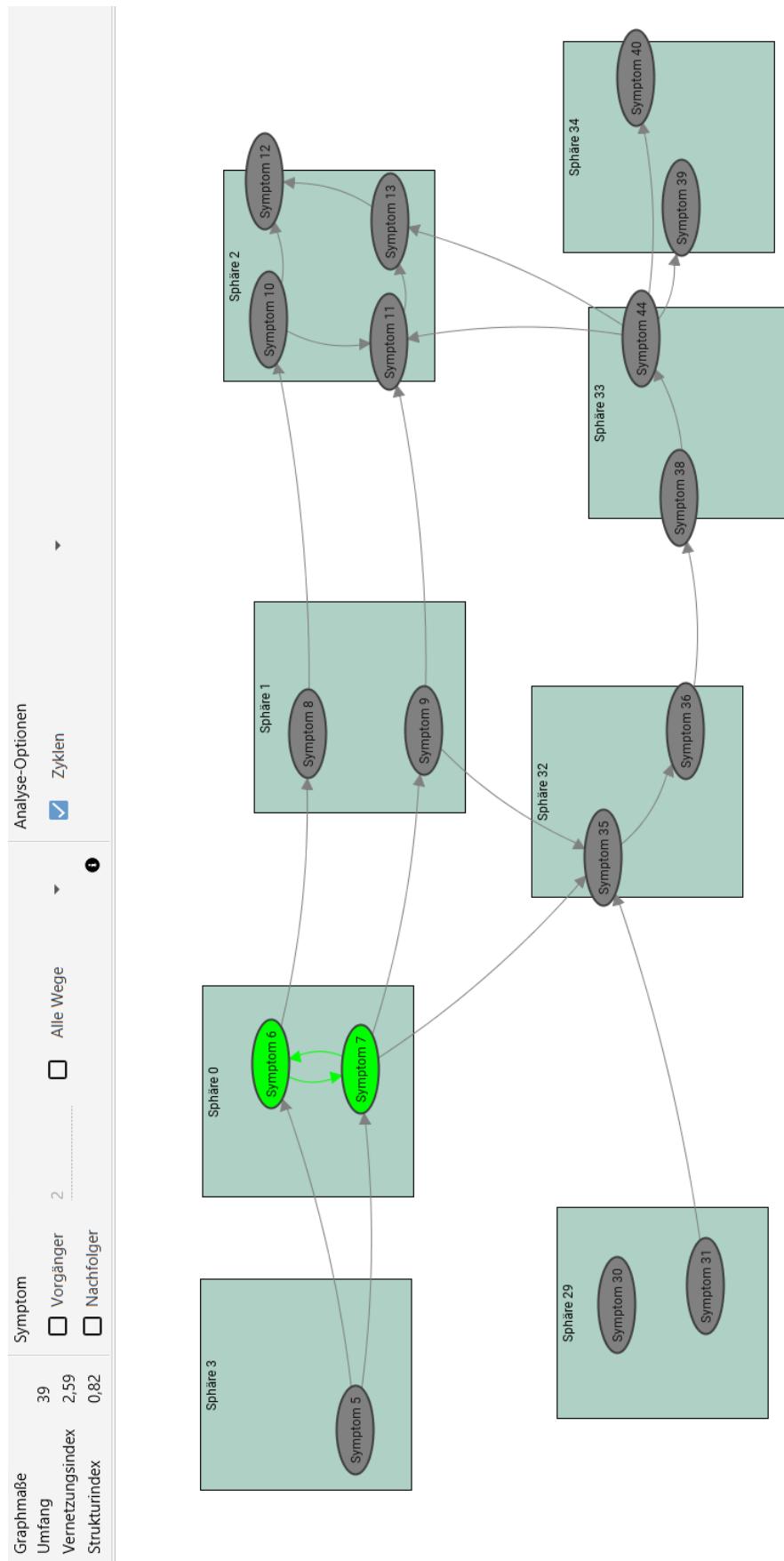
4.12 Zyklen

Der Bearbeiter aktiviert er in der Kopfleiste **Zyklen**. Hierbei wird erwartet, dass alle Zyklen hervorgehoben werden.

4.12.1 Notizen

1.Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 15:43
Test negativ.

2.Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 15:52
Test negativ.

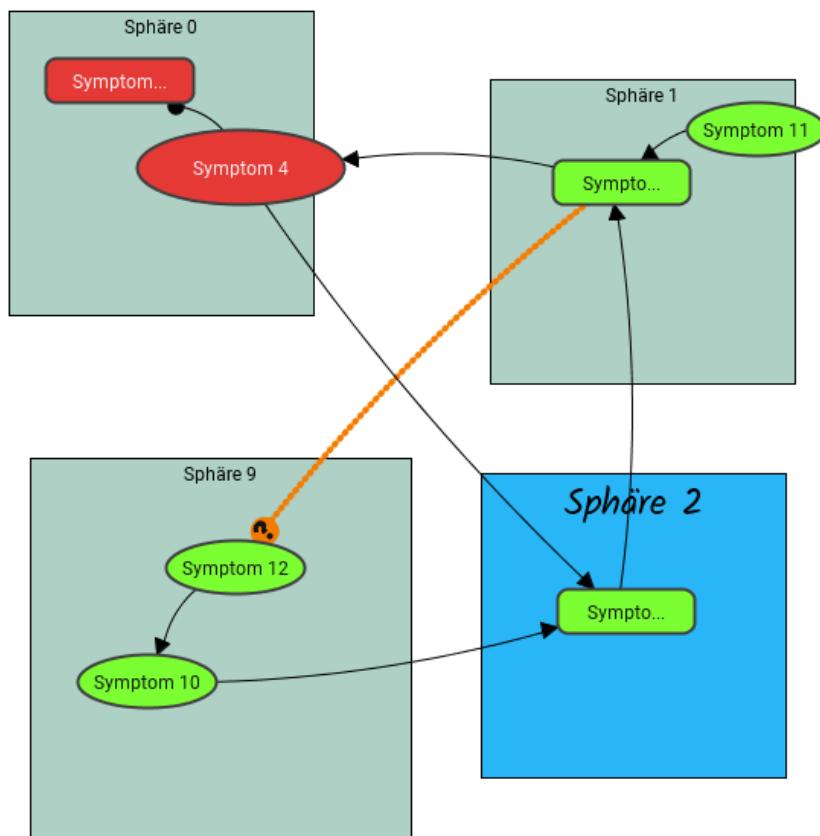


4.12.2 Fazit

Die Software arbeitet wie erwartet.

5 Import, Export, Ändern der Sprache und Hilfe

Die Funktionalitäten `Datei speichern`, `Datei öffnen` und das Ändern der Sprache des Graphen wurden im Laufe des Protokolls bereits getestet. Sie werden an dieser Stelle also nicht erneut getestet. Für die folgenden Tests erstellen wir vorerst einen neuen Graphen:



Anschließend sollen noch Vorlage-Regeln für den Graphen eingestellt werden. Die Eigenschaften des Graphen sowie die genauen Vorlage-Regeln sind an dieser Stelle eher irrelevant. Es ist jedoch wichtig, dass diese Eigenschaften durch die im folgenden getesteten Aktionen unverändert bleiben (bis auf den Verlust der Vorlage-Regeln beim Exportieren der GXL).

5.1 Vorlage exportieren

Der Tester geht auf **Datei, Exportieren als** und **Vorlage exportieren** und wählt dort ein Verzeichnis aus. Hierbei wird erwartet, dass die Vorlage in dem entsprechenden Verzeichnis gespeichert wird.

5.1.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 19:43

Test negativ.

2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 15:57

Test negativ.

5.1.2 Fazit

Die Software arbeitet wie erwartet.

5.2 Vorlage importieren

Der Tester drückt auf **neue Datei** und bestätigt die Abfrage. Anschließend geht er auf **Datei und Vorlage importieren** und wählt dort im entsprechenden Verzeichnis die vor kurzen gespeicherte Vorlage aus. Hierbei wird erwartet, dass der Graph inklusive der Vorlage-Regeln vollständig und korrekt geladen wird.

5.2.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 19:56

Test negativ.

2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 15:58

Test negativ.

5.2.2 Fazit

Die Software arbeitet wie erwartet.

5.3 GXL exportieren

Der Tester geht auf **Datei, Exportieren als** und **GXL** und wählt dort ein Verzeichnis aus. Hierbei wird erwartet, dass die GXL in dem entsprechenden Verzeichnis gespeichert wird.

5.3.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 20:32

Test negativ.

2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 15:58

Test negativ.

5.3.2 Fazit

Die Software arbeitet wie erwartet.

5.4 GXL importieren

Der Tester geht auf Datei und Vorlage importieren. Hierbei wird erwartet, dass der Graph vollständig und korrekt geladen wird. Ebenso müssten die Vorlage-Regeln auf Standartwerte zurückgesetzt wurden sein.

5.4.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 20:41

Test negativ.

2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 15:59

Test negativ.

5.4.2 Fazit

Die Software arbeitet wie erwartet.

5.5 Exportieren als PDF

Der Tester geht auf Datei, Exportieren als und PDF und wählt dort ein Verzeichnis aus. Hierbei wird erwartet, dass eine PDF des Graphen in dem entsprechenden Verzeichnis gespeichert wird.

5.5.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 20:56

Test negativ.

2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 16:00

Test negativ.

5.5.2 Fazit

Die Software arbeitet wie erwartet.

5.6 Drucken

Der Tester geht auf Datei und Drucken und wählt dort einen Drucker aus. Hierbei wird erwartet, dass der Graph von dem entsprechenden Drucker gedruckt wird.

5.6.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 21:02
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 16:02
Test negativ.

5.6.2 Fazit

Die Software arbeitet wie erwartet.

5.7 Sprache der Benutzeroberfläche ändern

Der Tester geht auf Optionen, erweiterte Spracheinstellungen, Sprache der Benutzeroberfläche und wählt Englisch aus. Hierbei wird erwartet, dass die Sprache der Benutzeroberfläche auf Englisch gestellt wird, die des Graphen jedoch Deutsch bleibt.

5.7.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 21:27
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 16:06
Test negativ.

5.7.2 Fazit

Die Software arbeitet wie erwartet.

5.8 Sprache des Graphen und der Benutzeroberfläche ändern

Der Tester geht auf Optionen, Sprache und wählt Englisch aus. Hierbei wird erwartet, dass anschließend die Sprache der Benutzeroberfläche und der des Graphen auf Englisch gestellt sind.

5.8.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 28.02.2019 um 21:45
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 04.03.2019 um 16:07
Test negativ.

5.8.2 Fazit

Die Software arbeitet wie erwartet.

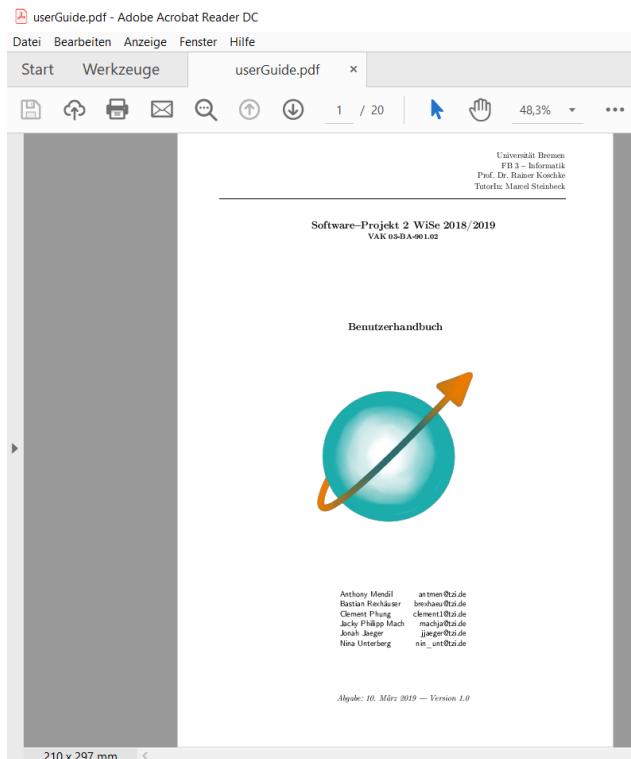
5.9 Anzeigen des Benutzerhandbuchs

Der Tester geht auf Hilfe und Dokumentation. Hierbei wird erwartet, dass das Benutzerhandbuch angezeigt wird.

5.9.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 05.03.2019 um 18:16
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 06.03.2019 um 15:13
Test negativ.

Nach dem Klicken auf Dokumentation im Menü Hilfe.



5.9.2 Fazit

Die Software arbeitet wie erwartet.

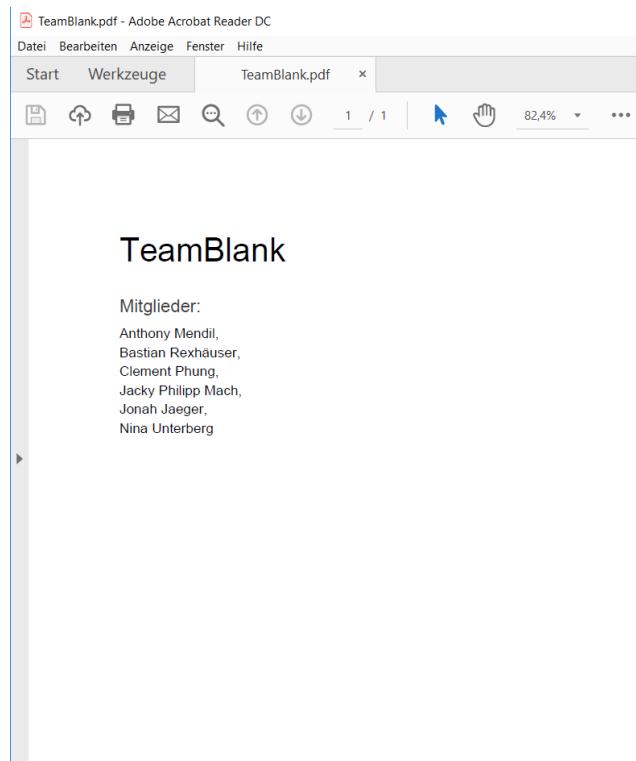
5.10 Über uns

Der Tester geht auf Hilfe und Über uns. Hierbei wird erwartet, dass eine Beschreibung über unser Team angezeigt wird.

5.10.1 Notizen

1. Test: Getestet von: Anthony Mendil. Datum: 05.03.2019 um 18:18
Test negativ.
2. Test: Getestet von: Jacky Philipp Mach. Datum: 06.03.2019 um 15:14
Test negativ.

Nach dem Klicken auf Über uns im Menü Hilfe.



5.10.2 Fazit

Die Software arbeitet wie erwartet.

6 JUnit Tests

6.1 Analysis Test

Im folgenden sieht man automatisierte Tests für die Analyse Funktionen.

```
1 package analysis;
2
3 import actions.analyse.*;
4 import edu.uci.ics.jung.visualization.picking.PickedState;
5 import graph.algorithmen.AnalyseType;
6 import graph.graph.*;
7 import gui.Values;
8 import org.junit.Assert;
9 import org.junit.Test;
10
11 import java.awt.geom.Point2D;
12 import java.util.*;
13
14 public class AnalysisTest {
15     /**
16      * The values to use when creating new elements of the graph.
17      */
18     private Values values = Values.getInstance();
19
20     /**
21      * The factory used to create elements of the graph. These are object of the type
22      * Sphere, Vertex and Edge.
23      */
24     private GraphObjectsFactory factory = new GraphObjectsFactory();
25
26     /**
27      * The list of vertices of the shortest path in the graph.
28      */
29     private ArrayList<Vertex> shortestPathVertices = new ArrayList<>();
30
31     /**
32      * The list of edges of the shortest path in the graph.
33      */
34     private ArrayList<Edge> shortestPathEdges = new ArrayList<>();
35
36     /**
37      * The list of vertices of the shortest path in the graph.
38      */
39     private Set<Vertex> allPathsVertices = new HashSet<>();
40
41     /**
42      * The list of edges of the shortest path in the graph.
43      */
44     private Set<Edge> allPathsEdges = new HashSet<>();
45
46     /**
47      * The start-vertex of the shortest path.
48      */
49     private Vertex vPathStart;
50
51     /**
52      * The sink-vertex of the shortest path.
53     */
54     private Vertex vPathSink;
```

```
54
55     /**
56      * The pivot-vertex for the neighbor-algorithms.
57      */
58     private Vertex neighborVertex;
59
60     /**
61      * The pivot-vertex2 for the neighbor-algorithms.
62      */
63     private Vertex neighborVertex2;
64
65     /**
66      * The predecessor-Set for the neighbor-algorithms.
67      */
68     private Set<Vertex> predecessorVerticesSet = new HashSet<>();
69
70     /**
71      * The predecessor-Set for the neighbor-algorithms.
72      */
73     private Set<Edge> predecessorEdgesSet = new HashSet<>();
74
75     /**
76      * The successor-Set for the neighbor-algorithms.
77      */
78     private Set<Vertex> successorVerticesSet = new HashSet<>();
79
80     /**
81      * The successor-Set for the neighbor-algorithms.
82      */
83     private Set<Edge> successorEdgesSet = new HashSet<>();
84
85     /**
86      * The predecessor/successor vertex set for multiple vertices picked.
87      */
88     private Set<Vertex> predecessorSuccessVertexSet = new HashSet<>();
89
90     /**
91      * The predecessor/successor edges set for multiple vertices picked.
92      */
93     private Set<Edge> predecessorSuccessEdgesSet = new HashSet<>();
94
95     /**
96      * The vertices-set for the cycle.
97      */
98     private Set<Vertex> cycleVerticesSet = new HashSet<>();
99
100    /**
101     * The edges-set for the cycle.
102     */
103    private Set<Edge> cycleEdgesSet = new HashSet<>();
104
105    /**
106     * The vertices-set for the convergent-branches algorithm.
107     */
108    private Set<Vertex> convergentBranchesVertices = new HashSet<>();
109
110    /**
111     * The edges-set for the convergent-branches algorithm.
112     */
113    private Set<Edge> convergentBranchesEdges = new HashSet<>();
114
115    /**
```

```
116     * The vertices-set for the divergent-branches algorithm.  
117     **/  
118     private Set<Vertex> divergentBranchesVertices = new HashSet<>();  
119  
120     /**  
121      * The edges-set for the divergent-branches algorithm.  
122      */  
123     private Set<Edge> divergentBranchesEdges = new HashSet<>();  
124  
125     /**  
126      * The vertices-set for the relation-chain algorithm.  
127      */  
128     private Set<Vertex> relationChainVertices = new HashSet<>();  
129  
130     /**  
131      * The edges-set for the relation-chain algorithm.  
132      */  
133     private Set<Edge> relationChainEdges = new HashSet<>();  
134  
135     /**  
136      * The syndrom instance of the application containing the graph.  
137      */  
138     private Syndrom syndrom = Syndrom.getInstance();  
139  
140     /**  
141      * The graph which is set.  
142      */  
143     private SyndromGraph<Vertex, Edge> graph = new SyndromGraph<>();  
144  
145  
146     /**  
147      * Sets the test graph, This graph has one of each construct, which can be filtered  
148      *  
149      * @param withElements Indicator whether the graph has elements.  
150      */  
151  
152     private void setupSyndrom(boolean withElements) {  
153         syndrom.generateNew();  
154         syndrom.setTemplate(new Template(25, 50, 75, true, false, true));  
155         if (withElements) {  
156             generateGraphElements();  
157         }  
158         syndrom.getVv().getGraphLayout().setGraph(graph);  
159     }  
160  
161     /**  
162      * Generates the graph with the elements. The graph contains: 4 spheres, 10  
163      * vertices and 11 edges.  
164      * All attributes for the test are gonna be set in this method.  
165      */  
166     private void generateGraphElements() {  
167         values.setFontSizeSphere(10);  
168         values.setFillPaintSphere(new java.awt.Color(86, 151, 31, 183));  
169  
170         Sphere s1 = factory.createSphere(new Point2D.Double(196, 116));  
171         Sphere s2 = factory.createSphere(new Point2D.Double(660, 116));  
172         Sphere s3 = factory.createSphere(new Point2D.Double(1101, 116));  
173         Sphere s4 = factory.createSphere(new Point2D.Double(660, 470));  
174  
175         Vertex v1 = factory.createVertex(new Point2D.Double(260, 241));  
176         Vertex v2 = factory.createVertex(new Point2D.Double(334, 159));
```

```
176     Vertex v3 = factory.createVertex(new Point2D.Double(721, 157));  
177     Vertex v4 = factory.createVertex(new Point2D.Double(810, 243));  
178     Vertex v5 = factory.createVertex(new Point2D.Double(1141, 248));  
179     Vertex v6 = factory.createVertex(new Point2D.Double(1267, 247));  
180     Vertex v7 = factory.createVertex(new Point2D.Double(1187, 154));  
181     Vertex v8 = factory.createVertex(new Point2D.Double(774, 522));  
182     Vertex v9 = factory.createVertex(new Point2D.Double(692, 630));  
183     Vertex v10 = factory.createVertex(new Point2D.Double(827, 625));  
184  
185     Edge e1 = factory.createEdge();  
186     Edge e2 = factory.createEdge();  
187     Edge e3 = factory.createEdge();  
188     Edge e4 = factory.createEdge();  
189     Edge e5 = factory.createEdge();  
190     Edge e6 = factory.createEdge();  
191     Edge e7 = factory.createEdge();  
192     Edge e8 = factory.createEdge();  
193     //e9 was removed.  
194     Edge e10 = factory.createEdge();  
195     Edge e11 = factory.createEdge();  
196     Edge e12 = factory.createEdge();  
197  
198     graph.getSpheres().add(s1);  
199     graph.getSpheres().add(s2);  
200     graph.getSpheres().add(s3);  
201     graph.getSpheres().add(s4);  
202  
203     s1.getVertices().add(v1);  
204     s1.getVertices().add(v2);  
205     s2.getVertices().add(v3);  
206     s2.getVertices().add(v4);  
207     s3.getVertices().add(v5);  
208     s3.getVertices().add(v6);  
209     s3.getVertices().add(v7);  
210     s4.getVertices().add(v8);  
211     s4.getVertices().add(v9);  
212     s4.getVertices().add(v10);  
213  
214     //relationChain  
215     graph.addEdgeExisting(e1, v1, v2);  
216     graph.addEdgeExisting(e2, v2, v3);  
217     graph.addEdgeExisting(e3, v3, v4);  
218     graph.addEdgeExisting(e4, v4, v5);  
219     //cycle  
220     graph.addEdgeExisting(e5, v5, v6);  
221     graph.addEdgeExisting(e6, v6, v7);  
222     graph.addEdgeExisting(e7, v7, v5);  
223  
224     graph.addEdgeExisting(e8, v1, v8);  
225     graph.addEdgeExisting(e10, v8, v9);  
226     graph.addEdgeExisting(e11, v8, v10);  
227     graph.addEdgeExisting(e12, v9, v10);  
228  
229     //initializing shortest path  
230     shortestPathVertices.add(v1);  
231     shortestPathVertices.add(v8);  
232     shortestPathVertices.add(v10);  
233  
234     shortestPathEdges.add(e8);  
235     shortestPathEdges.add(e11);  
236  
237     vPathStart = v1;
```

```
238     vPathSink = v10;
239
240     //initializing all paths
241     allPathsVertices.add(v1);
242     allPathsVertices.add(v8);
243     allPathsVertices.add(v9);
244     allPathsVertices.add(v10);
245
246     allPathsEdges.add(e8);
247     allPathsEdges.add(e10);
248     allPathsEdges.add(e11);
249     allPathsEdges.add(e12);
250
251     //initializing for the neighbors-algorithm
252
253     neighborVertex = v3;
254     neighborVertex2 = v4;
255
256     //(predecessor)
257
258     predecessorVerticesSet.add(v3);
259     predecessorVerticesSet.add(v2);
260     predecessorVerticesSet.add(v1);
261
262     predecessorEdgesSet.add(e1);
263     predecessorEdgesSet.add(e2);
264
265     predecessorSuccessEdgesSet.add(e2);
266     predecessorSuccessEdgesSet.add(e3);
267     predecessorSuccessEdgesSet.add(e4);
268
269     predecessorSuccessVertexSet.add(v2);
270     predecessorSuccessVertexSet.add(v3);
271     predecessorSuccessVertexSet.add(v4);
272     predecessorSuccessVertexSet.add(v5);
273
274     //(successor)
275
276     successorVerticesSet.add(v3);
277     successorVerticesSet.add(v4);
278     successorVerticesSet.add(v5);
279
280     successorEdgesSet.add(e3);
281     successorEdgesSet.add(e4);
282
283     //initializing cycles
284     cycleVerticesSet.add(v5);
285     cycleVerticesSet.add(v6);
286     cycleVerticesSet.add(v7);
287
288     cycleEdgesSet.add(e5);
289     cycleEdgesSet.add(e6);
290     cycleEdgesSet.add(e7);
291
292     //initializing convergent branches
293     convergentBranchesVertices.add(v5);
294     convergentBranchesVertices.add(v10);
295
296     convergentBranchesEdges.add(e4);
297     convergentBranchesEdges.add(e7);
298     convergentBranchesEdges.add(e11);
```

```
300     convergentBranchesEdges.add(e12);
301
302     //initializing divergent branches
303     divergentBranchesVertices.add(v1);
304     divergentBranchesVertices.add(v8);
305
306     divergentBranchesEdges.add(e1);
307     divergentBranchesEdges.add(e8);
308     divergentBranchesEdges.add(e10);
309     divergentBranchesEdges.add(e11);
310
311     //initializing relationChains
312     relationChainVertices.add(v1);
313     relationChainVertices.add(v2);
314     relationChainVertices.add(v3);
315     relationChainVertices.add(v4);
316     relationChainVertices.add(v5);
317
318     relationChainEdges.add(e1);
319     relationChainEdges.add(e2);
320     relationChainEdges.add(e3);
321     relationChainEdges.add(e4);
322
323 }
324
325 /**
326 * Tests the setup, if it differs from the implemented graph.
327 */
328
329 @Test
330 public void testSetup() {
331     setupSyndrom(true);
332     Assert.assertEquals(4, graph.getSpheres().size());
333     Assert.assertEquals(10, graph.getVertices().size());
334     Assert.assertEquals(11, graph.getEdges().size());
335 }
336
337 /**
338 * Tests the calculation of the scop index. According to the site
339 * http://www.informatik.uni-bremen.de/st/Lehre/swp/anforderungen/
340 * the scope index is the sum of vertices and the relations.
341 */
342
343 @Test
344 public void testScopeIndex() {
345     setupSyndrom(true);
346     GraphDimensionAction graphDimensionAction = new GraphDimensionAction();
347     graphDimensionAction.action();
348     Assert.assertEquals(21, graphDimensionAction.getScope());
349 }
350
351 /**
352 * Tests the calculation of the scope index, if the graph doesn't contain any
353 * elements.
354 */
355
356 @Test
357 public void testScopeIndexNoElements() {
358     setupSyndrom(false);
359     GraphDimensionAction graphDimensionAction = new GraphDimensionAction();
360     graphDimensionAction.action();
361     Assert.assertEquals(0, graphDimensionAction.getScope());
362 }
```

```
362     /**
363      * Tests the calculation of the network index. According to the site
364      * http://www.informatik.uni-bremen.de/st/Lehre/swp/anforderungen/
365      * the network index is the number of relations multiplied by 2 divided by
366      * the number of vertices.
367      */
368     @Test
369     public void testNetworkIndex() {
370         setupSyndrom(true);
371         GraphDimensionAction graphDimensionAction = new GraphDimensionAction();
372         graphDimensionAction.action();
373         Assert.assertEquals(2.2, graphDimensionAction.getNetworkIndex(), 0);
374     }
375
376     /**
377      * Tests the calculation of the network index, if the graph doesn't contain any
378      * elements. Therefore the amount of vertices is zero and the division by zero is
379      * undefined.
380      */
381     @Test
382     public void testNetworkIndexNoVertices() {
383         setupSyndrom(false);
384         GraphDimensionAction graphDimensionAction = new GraphDimensionAction();
385         graphDimensionAction.action();
386         Assert.assertEquals(-1, graphDimensionAction.getNetworkIndex(), 0);
387     }
388
389     /**
390      * Tests the calculation of the structure index. According to the site
391      * http://www.informatik.uni-bremen.de/st/Lehre/swp/anforderungen/
392      * the structure index is the sum of relation-chains, branches and cycles
393      * divided by symptoms.
394      */
395     @Test
396     public void testStructureIndex() {
397         setupSyndrom(true);
398         GraphDimensionAction graphDimensionAction = new GraphDimensionAction();
399         graphDimensionAction.action();
400         Assert.assertEquals(0.6, graphDimensionAction.getStructureIndex(), 0);
401     }
402
403     /**
404      * Tests the calculation of the structure index, if the graph doesn't contain any
405      * elements. Therefore the amount of vertices is zero and the division by zero is
406      * undefined.
407      */
408     @Test
409     public void testStructureIndexNoVertices() {
410         setupSyndrom(false);
411         GraphDimensionAction graphDimensionAction = new GraphDimensionAction();
412         graphDimensionAction.action();
413         Assert.assertEquals(-1, graphDimensionAction.getStructureIndex(), 0);
414     }
415
416     /**
417      * Tests the shortest path algorithm. It returns graph path between both vertices.
418      */
419     @Test
420     public void testShortestPath() {
421         setupSyndrom(true);
422         PickedState<Vertex> pickedState = syndrom.getVv().getPickedVertexState();
423         //Picking the vertices.
```

```
424     pickedState.pick(vPathStart, true);
425     pickedState.pick(vPathSink, true);
426     AnalysisGraphShortestPathAction analysisGraphShortestPathAction = new
427         AnalysisGraphShortestPathAction();
428     analysisGraphShortestPathAction.action();
429     List<Vertex> verticesList = analysisGraphShortestPathAction.getVerticesAnalyse
430         ();
431     verticesList.sort(Comparator.comparingInt(Vertex::getId));
432     List<Edge> edgesList = analysisGraphShortestPathAction.getEdgesAnalyse();
433     edgesList.sort(Comparator.comparingInt(Edge::getId));
434     Assert.assertEquals(shortestPathVertices, verticesList);
435     Assert.assertEquals(shortestPathEdges, edgesList);
436 }
437 /**
438 * Tests the shortest path algorithm, if no path between two vertices exists.
439 * It should throw an IllegalStateException, because the context of the alert-text
440 * is not set.
441 */
442 @Test(expected = IllegalStateException.class)
443 public void testShortestPathNoPath() {
444     setupSyndrom(true);
445     PickedState<Vertex> pickedState = syndrom.getVv().getPickedVertexState();
446     //Picking the vertices.
447     pickedState.pick(vPathSink, true);
448     pickedState.pick(vPathStart, true);
449     AnalysisGraphShortestPathAction analysisGraphShortestPathAction = new
450         AnalysisGraphShortestPathAction();
451     analysisGraphShortestPathAction.action();
452     List<Vertex> verticesList = analysisGraphShortestPathAction.getVerticesAnalyse
453         ();
454     List<Edge> edgesList = analysisGraphShortestPathAction.getEdgesAnalyse();
455     Assert.assertEquals(new ArrayList<>(), verticesList);
456     Assert.assertEquals(new ArrayList<>(), edgesList);
457 }
458 /**
459 * Tests the shortest path algorithm, if the set of picked vertices is empty.
460 * It should throw an IllegalStateException, because the context of the alert-text
461 * is not set.
462 */
463 @Test(expected = IllegalStateException.class)
464 public void testShortestPathNoVertexSelected() {
465     setupSyndrom(true);
466     PickedState<Vertex> pickedState = syndrom.getVv().getPickedVertexState();
467     //No vertex picked.
468     AnalysisGraphShortestPathAction analysisGraphShortestPathAction = new
469         AnalysisGraphShortestPathAction();
470     analysisGraphShortestPathAction.action();
471 /**
472 * Tests the shortest path algorithm, if the set of picked vertices has three
473 * elements.
474 * It should throw an IllegalStateException, because the context of the alert-text
475 * is not set.
476 */
477 @Test(expected = IllegalStateException.class)
478 public void testShortestPathThreeVertexSelected() {
479     setupSyndrom(true);
500     PickedState<Vertex> pickedState = syndrom.getVv().getPickedVertexState();
501     //Three vertices picked.
```

```
480     pickedState.pick(vPathStart, true);
481     pickedState.pick(vPathSink, true);
482     pickedState.pick(neighborVertex, true);
483     AnalysisGraphShortestPathAction analysisGraphShortestPathAction = new
484         AnalysisGraphShortestPathAction();
485     analysisGraphShortestPathAction.action();
486 }
487 /**
488 * Tests the all paths algorithm. It returns the set of all vertices and edges used
489 * in the graph path.
490 */
491 @Test
492 public void testAllPaths() {
493     setupSyndrom(true);
494     PickedState<Vertex> pickedState = syndrom.getVv().getPickedVertexState();
495     //Picking the vertices.
496     pickedState.pick(vPathStart, true);
497     pickedState.pick(vPathSink, true);
498     AnalysisGraphAllPathsAction analysisGraphAllPathsAction = new
499         AnalysisGraphAllPathsAction();
500     analysisGraphAllPathsAction.action();
501     List<Vertex> verticesList = analysisGraphAllPathsAction.getVerticesAnalyse();
502     Set<Vertex> verticesSet = new HashSet<>();
503     verticesSet.addAll(verticesList);
504     List<Edge> edgesList = analysisGraphAllPathsAction.getEdgesAnalyse();
505     Set<Edge> edgesSet = new HashSet<>();
506     edgesSet.addAll(edgesList);
507     edgesList.sort(Comparator.comparingInt(Edge::getId));
508     Assert.assertEquals(allPathsVertices, verticesSet);
509     Assert.assertEquals(allPathsEdges, edgesSet);
510 }
511 /**
512 * Tests the all paths algorithm, if no path between two vertices exists.
513 * It should throw an IllegalStateException, because the context of the alert-text
514 * is not set.
515 */
516 @Test(expected = IllegalStateException.class)
517 public void testAllPathsNoPath() {
518     setupSyndrom(true);
519     PickedState<Vertex> pickedState = syndrom.getVv().getPickedVertexState();
520     //Picking the vertices.
521     pickedState.pick(vPathSink, true);
522     pickedState.pick(vPathStart, true);
523     AnalysisGraphAllPathsAction analysisGraphAllPathsAction = new
524         AnalysisGraphAllPathsAction();
525     analysisGraphAllPathsAction.action();
526     List<Vertex> verticesList = analysisGraphAllPathsAction.getVerticesAnalyse();
527     Set<Vertex> verticesSet = new HashSet<>();
528     verticesSet.addAll(verticesList);
529     List<Edge> edgesList = analysisGraphAllPathsAction.getEdgesAnalyse();
530     Set<Edge> edgesSet = new HashSet<>();
531     edgesSet.addAll(edgesList);
532     edgesList.sort(Comparator.comparingInt(Edge::getId));
533     Assert.assertEquals(allPathsVertices, verticesSet);
534     Assert.assertEquals(allPathsEdges, edgesSet);
535 }
536 /**
537 * Tests the all paths algorithm, if the set of picked vertices is empty.
538 * It should throw an IllegalStateException, because the context of the alert-text
```

```
539     * is not set.  
540     */  
541     @Test(expected = IllegalStateException.class)  
542     public void testAllPathsNoVertexSelected() {  
543         setupSyndrom(true);  
544         PickedState<Vertex> pickedState = syndrom.getVv().getPickedVertexState();  
545         //No vertices picked.  
546         AnalysisGraphAllPathsAction analysisGraphAllPathsAction = new  
547             AnalysisGraphAllPathsAction();  
548         analysisGraphAllPathsAction.action();  
549     }  
550     /**  
551      * Tests the all paths algorithm, if the set of picked vertices has three elements.  
552      * It should throw an IllegalStateException, because the context of the alert-text  
553      * is not set.  
554      */  
555     @Test(expected = IllegalStateException.class)  
556     public void testAllPathsThreeVertexSelected() {  
557         setupSyndrom(true);  
558         PickedState<Vertex> pickedState = syndrom.getVv().getPickedVertexState();  
559         //No vertices picked.  
560         pickedState.pick(vPathStart, true);  
561         pickedState.pick(vPathSink, true);  
562         pickedState.pick(neighborVertex, true);  
563         AnalysisGraphAllPathsAction analysisGraphAllPathsAction = new  
564             AnalysisGraphAllPathsAction();  
565         analysisGraphAllPathsAction.action();  
566     }  
567     /**  
568      * Tests the predecessor algorithm. It returns all vertices and edges gotten within  
569      * zero iterations. Basically only the selected graph should be returned.  
570      */  
571     @Test  
572     public void testPredecessorZeroIterations() {  
573         setupSyndrom(true);  
574         PickedState<Vertex> pickedState = syndrom.getVv().getPickedVertexState();  
575         //Picking the vertex  
576         pickedState.pick(neighborVertex, true);  
577         AnalysisGraphNeighborsAction analysisGraphNeighborsAction = new  
578             AnalysisGraphNeighborsAction(AnalyseType.NEIGHBOUR_PREDECESSOR, 0);  
579         analysisGraphNeighborsAction.action();  
580         Set<Vertex> vertices = new HashSet<>();  
581         vertices.add(neighborVertex);  
582         Set<Vertex> vertexSet = new HashSet<>(analysisGraphNeighborsAction.  
583             getVerticesAnalyse());  
584         Assert.assertEquals(vertices, vertexSet);  
585         Assert.assertEquals(new ArrayList<Edge>(), analysisGraphNeighborsAction.  
586             getEdgesAnalyse());  
587     }  
588     /**  
589      * Tests the successor algorithm. It returns all vertices and edges gotten within  
590      * zero iterations. Basically only the selected graph should be returned.  
591      */  
592     @Test  
593     public void testSuccessorZeroIterations() {  
594         setupSyndrom(true);  
595         PickedState<Vertex> pickedState = syndrom.getVv().getPickedVertexState();  
596         //Picking the vertex
```

```
596     pickedState.pick(neighborVertex, true);
597     AnalysisGraphNeighborsAction analysisGraphNeighborsAction = new
598         AnalysisGraphNeighborsAction(AnalyseType.NEIGHBOUR_SUCCESSOR, 0);
599     analysisGraphNeighborsAction.action();
600     Set<Vertex> vertices = new HashSet<>();
601     vertices.add(neighborVertex);
602     Set<Vertex> vertexSet = new HashSet<>(analysisGraphNeighborsAction.
603         getVerticesAnalyse());
604     Assert.assertEquals(vertices, vertexSet);
605     Assert.assertEquals(new ArrayList<Edge>(), analysisGraphNeighborsAction.
606         getEdgesAnalyse());
607 }
608 /**
609 * Tests the predecessor algorithm. It returns all vertices and edges gotten within
610 * two iterations.
611 */
612 @Test
613 public void testPredecessorTwoIterations() {
614     setupSyndrom(true);
615     PickedState<Vertex> pickedState = syndrom.getVv().getPickedVertexState();
616     //Picking the vertex
617     pickedState.pick(neighborVertex, true);
618     AnalysisGraphNeighborsAction analysisGraphNeighborsAction = new
619         AnalysisGraphNeighborsAction(AnalyseType.NEIGHBOUR_PREDECESSOR, 2);
620     analysisGraphNeighborsAction.action();
621     Set<Vertex> vertexSet = new HashSet<>(analysisGraphNeighborsAction.
622         getVerticesAnalyse());
623     Set<Edge> edgesSet = new HashSet<>(analysisGraphNeighborsAction.getEdgesAnalyse
624         ());
625     Assert.assertEquals(predecessorVerticesSet, vertexSet);
626     Assert.assertEquals(predecessorEdgesSet, edgesSet);
627 }
628 /**
629 * Tests the successor algorithm. It returns all vertices and edges gotten within
630 * two iterations.
631 */
632 @Test
633 public void testSuccessorTwoIterations() {
634     setupSyndrom(true);
635     PickedState<Vertex> pickedState = syndrom.getVv().getPickedVertexState();
636     //Picking the vertex
637     pickedState.pick(neighborVertex, true);
638     AnalysisGraphNeighborsAction analysisGraphNeighborsAction = new
639         AnalysisGraphNeighborsAction(AnalyseType.NEIGHBOUR_SUCCESSOR, 2);
640     analysisGraphNeighborsAction.action();
641     Set<Vertex> vertexSet = new HashSet<>(analysisGraphNeighborsAction.
642         getVerticesAnalyse());
643     Set<Edge> edgesSet = new HashSet<>(analysisGraphNeighborsAction.getEdgesAnalyse
644         ());
645     Assert.assertEquals(successorVerticesSet, vertexSet);
646     Assert.assertEquals(successorEdgesSet, edgesSet);
647 }
```

```
648     */
649     @Test
650     public void testPredecessorSuccessorTwoIterations() {
651         setupSyndrom(true);
652         PickedState<Vertex> pickedState = syndrom.getVv().getPickedVertexState();
653         //Picking the vertex
654         pickedState.pick(neighborVertex, true);
655         AnalysisGraphNeighborsAction analysisGraphNeighborsAction = new
656             AnalysisGraphNeighborsAction(AnalyseType.NEIGHBOUR_PREDECESSOR_SUCCESSOR, 2);
657         analysisGraphNeighborsAction.action();
658         Set<Vertex> vertexSet = new HashSet<>(analysisGraphNeighborsAction.
659             getVerticesAnalyse());
660         Set<Edge> edgesSet = new HashSet<>(analysisGraphNeighborsAction.getEdgesAnalyse
661             ());
662         Set<Vertex> bothVertices = successorVerticesSet;
663         bothVertices.addAll(predecessorVerticesSet);
664         Set<Edge> bothEdges = successorEdgesSet;
665         bothEdges.addAll(predecessorEdgesSet);
666         Assert.assertEquals(bothVertices, vertexSet);
667         Assert.assertEquals(bothEdges, edgesSet);
668     }
669     /**
670      * Tests the predecessor-successor algorithm, if no vertex is picked.
671      * It should throw an IllegalStateException, because the context of the alert-text
672      * is not set.
673     */
674     @Test(expected = IllegalStateException.class)
675     public void testPredecessorSuccessorZeroIterationsNoVertexPicked() {
676         setupSyndrom(true);
677         PickedState<Vertex> pickedState = syndrom.getVv().getPickedVertexState();
678         //No vertex picked
679         AnalysisGraphNeighborsAction analysisGraphNeighborsAction = new
680             AnalysisGraphNeighborsAction(AnalyseType.NEIGHBOUR_PREDECESSOR_SUCCESSOR, 2);
681         analysisGraphNeighborsAction.action();
682     }
683     /**
684      * Tests the predecessor-successor algorithm, if two vertices are picked.
685     */
686     @Test
687     public void testPredecessorSuccessorZeroIterationsTwoVertexPicked() {
688         setupSyndrom(true);
689         PickedState<Vertex> pickedState = syndrom.getVv().getPickedVertexState();
690         //two vertices picked.
691         pickedState.pick(neighborVertex, true);
692         pickedState.pick(neighborVertex2, true);
693         AnalysisGraphNeighborsAction analysisGraphNeighborsAction = new
694             AnalysisGraphNeighborsAction(AnalyseType.NEIGHBOUR_PREDECESSOR_SUCCESSOR, 1);
695         analysisGraphNeighborsAction.action();
696         Set<Vertex> vertexSet = new HashSet<>(analysisGraphNeighborsAction.
697             getVerticesAnalyse());
698         Set<Edge> edgeSet = new HashSet<>(analysisGraphNeighborsAction.getEdgesAnalyse
699             ());
700         Assert.assertEquals(predecessorSuccessorVertexSet, vertexSet);
701         Assert.assertEquals(predecessorSuccessorEdgesSet, edgeSet);
702     }
703     /**
704      * Tests the cycle detection algorithm. It returns all vertices and edges inside
705      * that cycle.
706  }
```

```
702     */
703     @Test
704     public void testCycle() {
705         setupSyndrom(true);
706         AnalysisGraphCyclesAction analysisGraphCyclesAction = new
707             AnalysisGraphCyclesAction();
708         analysisGraphCyclesAction.action();
709         Set<Vertex> vertexSet = new HashSet<>(analysisGraphCyclesAction.
710             getVerticesAnalyse());
711         Set<Edge> edgeSet = new HashSet<>(analysisGraphCyclesAction.getEdgesAnalyse());
712         Assert.assertEquals(cycleVerticesSet, vertexSet);
713         Assert.assertEquals(cycleEdgesSet, edgeSet);
714     }
715     /**
716      * Test the cycle detection algorithm, if no cycle is inside the graph.
717      * Therefore it should return an empty set for both, the vertices and the edges.
718      */
719     @Test
720     public void testCycleNoCycles() {
721         setupSyndrom(false);
722         AnalysisGraphCyclesAction analysisGraphCyclesAction = new
723             AnalysisGraphCyclesAction();
724         analysisGraphCyclesAction.action();
725         Assert.assertEquals(new ArrayList<>(), analysisGraphCyclesAction.
726             getVerticesAnalyse());
727         Assert.assertEquals(new ArrayList<>(), analysisGraphCyclesAction.
728             getEdgesAnalyse());
729     }
730     /**
731      * Tests the detection of all convergent branches. The set of vertices should be
732      * returned. An convergent branch has an incoming degree of at least 2.
733      */
734     @Test
735     public void testConvergentBranch() {
736         setupSyndrom(true);
737         AnalysisGraphConvergentBranchesAction analysisGraphConvergentBranchesAction =
738             new AnalysisGraphConvergentBranchesAction();
739         analysisGraphConvergentBranchesAction.action();
740         Set<Vertex> vertexSet = new HashSet<>(analysisGraphConvergentBranchesAction.
741             getVerticesAnalyse());
742         Set<Edge> edgeSet = new HashSet<>(analysisGraphConvergentBranchesAction.
743             getEdgesAnalyse());
744         Assert.assertEquals(convergentBranchesVertices, vertexSet);
745         Assert.assertEquals(convergentBranchesEdges, edgeSet);
746     }
747     /**
748      * Tests the detection of all divergent branches. The set of vertices should be
749      * returned. An divergent branch has an outgoing degree of at least 2.
750      */
751     @Test
752     public void testDivergentBranch() {
753         setupSyndrom(true);
754         AnalysisGraphDivergentBranchesAction analysisGraphDivergentBranchesAction = new
755             AnalysisGraphDivergentBranchesAction();
756         analysisGraphDivergentBranchesAction.action();
757         Set<Vertex> vertexSet = new HashSet<>(analysisGraphDivergentBranchesAction.
758             getVerticesAnalyse());
759         Set<Edge> edgeSet = new HashSet<>(analysisGraphDivergentBranchesAction.
760             getEdgesAnalyse());
```

```
753     Assert.assertEquals(divergentBranchesVertices, vertexSet);
754     Assert.assertEquals(divergentBranchesEdges, edgeSet);
755 }
756 /**
757 * Tests the detection of all branches. The set of vertices should be
758 * returned. An branch has an incoming or outgoing degree of at least 2.
759 */
760 @Test
761 public void testDivergentConvergentBranch() {
762     setupSyndrom(true);
763     AnalysisGraphBranchesAction analysisGraphBranchesAction = new
764         AnalysisGraphBranchesAction();
765     analysisGraphBranchesAction.action();
766     Set<Vertex> vertexSet = new HashSet<>(analysisGraphBranchesAction.
767         getVerticesAnalyse());
768     Set<Edge> edgeSet = new HashSet<>(analysisGraphBranchesAction.getEdgesAnalyse()
769         );
770     convergentBranchesVertices.addAll(divergentBranchesVertices);
771     convergentBranchesEdges.addAll(divergentBranchesEdges);
772     Assert.assertEquals(convergentBranchesVertices, vertexSet);
773     Assert.assertEquals(convergentBranchesEdges, edgeSet);
774 }
775 /**
776 * Tests the detection of relation chains. The set of vertices and relations
777 * should be returned.
778 */
779 @Test
780 public void testRelationChain() {
781     setupSyndrom(true);
782     AnalysisGraphEdgeChainsAction analysisGraphEdgeChainsAction = new
783         AnalysisGraphEdgeChainsAction();
784     analysisGraphEdgeChainsAction.action();
785     Set<Vertex> vertexSet = new HashSet<>(analysisGraphEdgeChainsAction.
786         getVerticesAnalyse());
787     Set<Edge> edgeSet = new HashSet<>(analysisGraphEdgeChainsAction.getEdgesAnalyse
788         ());
789     Assert.assertEquals(relationChainVertices, vertexSet);
790     Assert.assertEquals(relationChainEdges, edgeSet);
791 }
792 }
```

6.2 GXLio Test

Im folgenden sieht man automatisierte Tests für den GXL Import und Export.

```
1 package io;
2
3 import graph.graph.*;
4 import gui.Values;
5 import gui.properties.Language;
6 import net.sourceforge.gxl.*;
7 import org.apache.log4j.BasicConfigurator;
8 import org.apache.log4j.Logger;
9 import org.freehep.graphicsbase.util.Assert;
10 import org.junit.AfterClass;
11 import org.junit.Test;
12 import org.xml.sax.SAXException;
```

```
13
14 import java.awt.*;
15 import java.awt.geom.Point2D;
16 import java.io.File;
17 import java.io.IOException;
18 import java.util.*;
19 import java.util.List;
20
21
22 public class GXLioTest {
23
24     /**
25      * The values to use when creating new elements of the graph.
26      */
27     private Values values = Values.getInstance();
28
29     /**
30      * The logger sed to document the behaviour of this testclass.
31      */
32     private static Logger logger = Logger.getLogger(GXLioTest.class);
33
34     /**
35      * The factory used to create elements of the graph. These are object of the type
36      * Sphere, Vertex and Edge.
37      */
38     private GraphObjectsFactory factory = new GraphObjectsFactory();
39
40     /**
41      * The GXLDocument that is created from the file with the name below.
42      */
43     private GXLDocument doc = null;
44
45     /**
46      * The name of the gxl file.
47      */
48     private String nameTestGraph = "testGraph.gxl";
49
50     /**
51      * The syndrom instance of the application containing the graph that gets exported.
52      */
53     private Syndrom syndrom = Syndrom.getInstance();
54
55     /**
56      * The graph that gets exported.
57      */
58     private SyndromGraph<Vertex, Edge> graph = new SyndromGraph<>();
59
60     /**
61      * The list of spheres that belong to the graph those data gets exported.
62      */
63     private List<Sphere> spheresList = new ArrayList<>();
64
65     /**
66      * The list of vertices that belong to the graph those data gets exported.
67      */
68     private List<Vertex> verticesList = new ArrayList<>();
69
70     /**
71      * The list of edges that belong to the graph those data gets exported.
72      */
73     private List<Edge> edgesList = new ArrayList<>();
```

```
74     /**
75      * The name of the GXLGraph element that describes the data of the exported
76      * syndromgraph.
77     */
78     private String syndromName = "syndrom";
79
80     /**
81      * The instance of the class GXLio that is used to tests the functionality of the
82      * class by calling its methods.
83     */
84     private GXLio gxlio = new GXLio();
85
86     /**
87      * This method is called before each test method.
88      * It generates a new syndrom, sets a template and some graph elements to this
89      * syndrom / the graph of the syndrom,
90      * exports the graph data, generates a new graph again to make sure the old graph
91      * data gets deleted from the system
92      * and then initialises the GXLDocument attribute in this class. The last step also
93      * loads the graph data from the gxl into the system,
94      *
95      * @param pWithTemplate
96      * @return the attribute of this test that is of type GXLio.
97      * @throws IOException if the File can't be created or the file that is specified
98      * for the import can't be found.
99      * @throws SAXException if their occurs any problem parsing the document
100     */
101    private GXLio prepareSyndrom(boolean pWithTemplate) throws IOException,
102    SAXException {
103        BasicConfigurator.configure();
104        // generate new graph and set as actual graph
105        syndrom.generateNew();
106        syndrom.setTemplate(new Template(25, 50, 75, true, false, true));
107        generateGraphElements();
108        syndrom.getVv().getGraphLayout().setGraph(graph);
109        // export graph
110
111        gxlio.exportGXL(new File(nameTestGraph), pWithTemplate);
112        // generate new clean syndrom
113        syndrom.generateNew();
114        doc = new GXLDocument(new File(nameTestGraph));
115        return gxlio;
116    }
117
118    /**
119     * javadocTODO
120     */
121    @AfterClass
122    public static void endAll() throws IOException {
123        new File("testGraph.gxl").deleteOnExit();
124    }
125
126    /**
127     * This method creates some spheres, vertices and edges with different values for
128     * some of their attributes.
129     */
130    private void generateGraphElements() {
131        values.setFontSizeSphere(10);
132        values.setFillPaintSphere(new java.awt.Color(86, 151, 31, 183));
133        Sphere s1 = factory.createSphere(new Point2D.Double(20, 20));
134        s1.setLockedAnnotation(true);
135        Sphere s2 = factory.createSphere(new Point2D.Double(270, 50));
```

```
128     s2.setHeight(250.0);
129     s2.setWidth(300.0);
130     s2.setLockedAnnotation(true);
131     s2.setLockedVertices(true);
132     Map<String, String> annotation2 = s2.getAnnotation();
133     annotation2.put(Language.GERMAN.name(), "Sphäre Nummer 1");
134     s2.setAnnotation(annotation2);
135     Sphere s3 = factory.createSphere(new Point2D.Double(620, 20));
136     Map<String, String> annotation3 = s3.getAnnotation();
137     annotation3.put(Language.ENGLISH.name(), "Sphere number 2");
138     s3.setAnnotation(annotation3);
139     values.setFillPaintSphere(new java.awt.Color(24, 54, 11, 178));
140     values.setFontSizeSphere(24);
141     Sphere s4 = factory.createSphere(new Point2D.Double(200, 310));
142     s4.setLockedMaxAmountVertices("17");
143     s4.setLockedPosition(true);
144     Map<String, String> annotation4 = s4.getAnnotation();
145     annotation4.put(Language.GERMAN.name(), "Sphäre Nummer 3");
146     annotation4.put(Language.ENGLISH.name(), "Sphere number 3");
147     s4.setAnnotation(annotation4);
148     s4.setFont("Kalam");
149     Sphere s5 = factory.createSphere(new Point2D.Double(445, 310));
150     s5.setLockedMaxAmountVertices("5");
151     s5.setLockedPosition(true);
152     s5.setLockedStyle(true);
153     s5.setFont("Kalam");

154
155     values.setFontSizeVertex(12);
156     values.setShapeVertex(VertexShapeType.CIRCLE);
157     values.setFillPaintVertex(new java.awt.Color(88, 88, 219, 220));
158     values.setFontVertex("Roboto");
159     Vertex v1 = factory.createVertex(new Point2D.Double(80, 80));
160     v1.setLockedStyle(true);
161     Vertex v2 = factory.createVertex(new Point2D.Double(340, 140));
162     Vertex v3 = factory.createVertex(new Point2D.Double(415, 190));
163     v3.setFillColor(new java.awt.Color(0xC80070));
164     v3.setSize(120);
165     v3.setDrawColor(new java.awt.Color(200, 10, 10, 203));
166     v3.setLockedStyle(true);
167     v3.setLockedAnnotation(true);
168     v3.setLockedPosition(true);
169     Map<String, String> annotationV3 = v3.getAnnotation();
170     annotationV3.put(Language.GERMAN.name(), "Symptom Nummer 7");
171     annotationV3.put(Language.ENGLISH.name(), "Symptom number 7");
172     v3.setAnnotation(annotationV3);
173     Vertex v4 = factory.createVertex(new Point2D.Double(485, 140));
174     v4.setLockedAnnotation(true);
175     Vertex v5 = factory.createVertex(new Point2D.Double(740, 80));
176     v5.setLockedAnnotation(true);
177     values.setShapeVertex(VertexShapeType.RECTANGLE);
178     values.setFillPaintVertex(new java.awt.Color(11, 19, 90, 220));
179     values.setFontSizeVertex(7);
180     Vertex v6 = factory.createVertex(new Point2D.Double(220, 360));
181     v6.setLockedStyle(true);
182     Map<String, String> annotationV6 = v6.getAnnotation();
183     annotationV6.put(Language.GERMAN.name(), "Symptom Nummer 10");
184     v6.setAnnotation(annotationV6);
185     Vertex v7 = factory.createVertex(new Point2D.Double(280, 420));
186     v7.setFont("Mali");
187     v7.setFontSize(14);
188     v7.setSize(80);
189     v7.setLockedPosition(true);
```

```
190     Map<String, String> annotationV7 = v7.getAnnotation();
191     annotationV7.put(Language.ENGLISH.name(), "Symptom number 11");
192     v7.setAnnotation(annotationV7);
193     Vertex v8 = factory.createVertex(new Point2D.Double(555, 420));
194     v8.setFont("Mali");
195     v8.setFontSize(14);
196     v8.setSize(80);
197     v8.setLockedPosition(true);
198     Vertex v9 = factory.createVertex(new Point2D.Double(620, 360));
199
200     values.setEdgeArrowType(EdgeArrowType.EXTENUATING);
201     values.setStrokeEdge(StrokeType.BASIC);
202     Edge e1 = factory.createEdge();
203     e1.setLockedEdgeType(true);
204     Edge e2 = factory.createEdge();
205     e2.setStroke(StrokeType.BASIC_WEIGHT);
206     e2.setLockedEdgeType(true);
207     values.setEdgeArrowType(EdgeArrowType.NEUTRAL);
208     Edge e3 = factory.createEdge();
209     e3.setStroke(StrokeType.DOTTED_WEIGHT);
210     e3.setLockedStyle(true);
211     Edge e4 = factory.createEdge();
212     e4.setStroke(StrokeType.DOTTED);
213     values.setEdgeArrowType(EdgeArrowType.REINFORCED);
214     Edge e5 = factory.createEdge();
215     Edge e6 = factory.createEdge();
216     e6.setStroke(StrokeType.DASHED);
217     e6.setColor(new java.awt.Color(28, 56, 249, 130));
218     e6.setLockedStyle(true);
219     Edge e7 = factory.createEdge();
220     e7.setStroke(StrokeType.DASHED);
221     e7.setColor(new java.awt.Color(28, 56, 249, 130));
222     e7.setLockedStyle(true);
223     Edge e8 = factory.createEdge();
224     e8.setStroke(StrokeType.DASHED_WEIGHT);
225     e8.setLockedEdgeType(true);
226
227     s1.getVertices().add(v1);
228     s2.getVertices().add(v2);
229     s2.getVertices().add(v3);
230     s2.getVertices().add(v4);
231     s3.getVertices().add(v5);
232     s4.getVertices().add(v6);
233     s4.getVertices().add(v7);
234     s5.getVertices().add(v8);
235     s5.getVertices().add(v9);
236
237     graph.getSpheres().add(s1);
238     graph.getSpheres().add(s2);
239     graph.getSpheres().add(s3);
240     graph.getSpheres().add(s4);
241     graph.getSpheres().add(s5);
242
243     graph.addEdge(e1, v1, v2);
244     graph.addEdge(e2, v5, v4);
245     graph.addEdge(e3, v7, v6);
246     graph.addEdge(e4, v8, v9);
247     graph.addEdge(e5, v6, v3);
248     graph.addEdge(e6, v7, v3);
249     graph.addEdge(e7, v8, v3);
250     graph.addEdge(e8, v9, v3);
251     spheresList = graph.getSpheres();
```

```
252     verticesList.addAll(graph.getVertices());
253     verticesList.sort(Comparator.comparingInt(Vertex::getId));
254     edgesList.addAll(graph.getEdges());
255     edgesList.sort(Comparator.comparingInt(Edge::getId));
256 }
257
258 // <----- name of graph ----->
259
260 /**
261 * This method tests if the name of a graph is exported and imported correctly.
262 *
263 * @throws IOException if the File can't be created or the file that is specified
264 * for the import can't be found.
265 * @throws SAXException if there occurs any problem parsing the document
266 */
267 @Test
268 public void testGraphName() throws IOException, SAXException {
269     prepareSyndrom(false).importGXL(new File(nameTestGraph), false);
270     Assert.assertEquals("UntitledGraph", syndrom.getGraphName());
271 }
272
273 // <----- holistic comparision ----->
274
275 /**
276 * This method tests, if the list of spheres contained in the graph belonging to
277 * the syndrom after importing a gxl-File
278 * is equal to the list of spheres of the graph that gets imported.
279 *
280 * @throws IOException if the File can't be created or the file that is specified
281 * for the import can't be found.
282 * @throws SAXException if there occurs any problem parsing the document
283 */
284 @Test
285 public void testSphereList() throws IOException, SAXException {
286     prepareSyndrom(false).importGXL(new File(nameTestGraph), false);
287     SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().
288         getGraphLayout().getGraph();
289     Assert.assertEquals(spheresList, g.getSpheres());
290 }
291
292 /**
293 * This method tests, if the list of vertices contained in the graph belonging to
294 * the syndrom after importing a gxl-File
295 * is equal to the list of vertices of the graph that is imported.
296 *
297 * @throws IOException if the File can't be created or the file that is specified
298 * for the import can't be found.
299 * @throws SAXException if there occurs any problem parsing the document
300 */
301 @Test
302 public void testVertexList() throws IOException, SAXException {
303     prepareSyndrom(false).importGXL(new File(nameTestGraph), false);
304     SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().
305         getGraphLayout().getGraph();
306     List<Vertex> importedVertices = new ArrayList<>(g.getVertices());
307     importedVertices.sort(Comparator.comparingInt(Vertex::getId));
308     Assert.assertEquals(verticesList, importedVertices);
309 }
310
311 /**
312 *
```

```
306     * This method tests, if the list of edges contained in the graph belonging to the
307     * syndrom after importing a gxl-File
308     * is equal to the list of edges of the graph that is imported.
309     *
310     * @throws IOException if the File can't be created or the file that is specified
311     * for the import can't be found.
312     * @throws SAXException if their occurs any problem parsing the document
313     */
314     @Test
315     public void testEdgeList() throws IOException, SAXException {
316         prepareSyndrom(false).importGXL(new File(nameTestGraph), false);
317         SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv();
318         getGraphLayout().getGraph();
319         List<Edge> importedEdges = new ArrayList<>(g.getEdges());
320         importedEdges.sort(Comparator.comparingInt(Edge::getId));
321         Assert.assertEquals(edgesList, importedEdges);
322     }
323
324     /**
325     * This method tests, if the number of spheres, vertices and edges is correct after
326     * the import of a gxl File.
327     * Correct in this sense means that it is identical to the number of spheres,
328     * vertices and edges of the graph that gets exported.
329     *
330     * @throws IOException if the File can't be created or the file that is specified
331     * for the import can't be found.
332     * @throws SAXException if their occurs any problem parsing the document
333     */
334     @Test
335     public void testElementNumberOfSyndromGraph() throws IOException, SAXException {
336         prepareSyndrom(false).importGXL(new File(nameTestGraph), false);
337         SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv();
338         getGraphLayout().getGraph();
339         Assert.assertEquals(8, g.getEdgeCount());
340         Assert.assertEquals(9, g.getVertexCount());
341         Assert.assertEquals(5, g.getSpheres().size());
342     }
343
344     /**
345     * This method tests if the GXLGraph-Object created from the File that is created by
346     * the emport of a graph
347     * contains only one GXLGraph (the exported syndrom) and if this graph contains the
348     * right amount of childs.
349     * Childs in this sense are spheres, vertices and edges.
350     *
351     * @throws IOException if the File can't be created or the file that is specified
352     * for the import can't be found.
353     * @throws SAXException if their occurs any problem parsing the document
354     */
355     @Test
356     public void testElementNumberOfGXLGraph() throws IOException, SAXException {
357         prepareSyndrom(false);
358         Assert.assertEquals(1, doc.getDocumentElement().getGraphCount());
359         Assert.assertEquals(23, doc.getElement(syndromName).getChildCount());
360     }
361
362     /**
363     * This method tests if the GXLGraph-Object created from the File that is created by
364     * the emport of a graph
```

```
356     * contains two GXLGraphs. One of them represents the syndromgraph and the other
357     * GXLGraph contains the rules of the template.
358     *
359     * @throws IOException if the File can't be created or the file that is specified
360     * for the import can't be found.
361     * @throws SAXException if their occurs any problem parsing the document
362     */
363     @Test
364     public void testElementNumberWithTemplateOfGXLGraph() throws IOException,
365     SAXException {
366         prepareSyndrom(true);
367         Assert.assertEquals(2, doc.getDocumentElement().getGraphCount());
368     }
369
370     /**
371      * This method tests if a GXLValidationException is thrown when a method call
372      * occurs on a GXLGraph that would lead
373      * to an invalid GXLDocument. The document would not be valid any longer after the
374      * method call, because two GXLElements would have the same id.
375      * This is not allowed in GXL notation.
376      *
377      * @throws IOException if the File can't be created or the file that is specified
378      * for the import can't be found.
379      * @throws SAXException if their occurs any problem parsing the document
380      */
381     @Test(expected = GXLValidationException.class)
382     public void testExceptionAddElementToGXLGraphThatAlreadyContainsThisID() throws
383     IOException, SAXException {
384         prepareSyndrom(true);
385         GXLElement duplicateIDEElement = new GXLNode("1");
386         doc.getElement("syndrom").add(duplicateIDEElement);
387     }
388
389     // <----- template ----->
390
391     /**
392      * This method tests if the template belonging to the syndrom that is created from
393      * the File that is created by the emport of a graph
394      * contains the correct number vulues for the attributes that are numbers.
395      *
396      * @throws IOException if the File can't be created or the file that is specified
397      * for the import can't be found.
398      * @throws SAXException if their occurs any problem parsing the document
399      */
400     @Test
401     public void testNumberAttributesInTemplateWithTemplateOfGXLGraph() throws
402     IOException, SAXException {
403         prepareSyndrom(true).importGXL(new File(nameTestGraph), true);
404
405         Assert.assertEquals(25, syndrom.getTemplate().getMaxSpheres());
406         Assert.assertEquals(50, syndrom.getTemplate().getMaxVertices());
407         Assert.assertEquals(75, syndrom.getTemplate().getMaxEdges());
408     }
409
410     /**
411      * This method tests if the template graph contains the right number of attributes.
412      * These attributes are the attributes that can be seen in the two tests below this
413      * test method.
414      *
415      * @throws IOException if the File can't be created or the file that is specified
416      * for the import can't be found.
```

```
406     * @throws SAXException if their occurs any problem parsing the document
407     */
408     @Test
409     public void testAttributeNumberOfGXLTemplate() throws IOException, SAXException {
410         prepareSyndrom(true);
411         GXLDocument document = new GXLDocument(new File(nameTestGraph));
412         GXLGraph templateGraph = (GXLGraph) document.getElement("template");
413         Assert.assertEquals(6, templateGraph.getAttributeCount());
414     }
415
416 /**
417     * This method tests if the template belonging to the syndrom that is created from
418     * the File that is created by the import of a graph
419     * contains the correct boolean values for the attributes that are booleans.
420     *
421     * @throws IOException if the File can't be created or the file that is specified
422     * for the import can't be found.
423     * @throws SAXException if their occurs any problem parsing the document
424     */
425     @Test
426     public void testBooleanAttributesInTemplateWithTemplateOfGXLGraph() throws
427         IOException, SAXException {
428         prepareSyndrom(true).importGXL(new File(nameTestGraph), true);
429         Assert.assertEquals(true, syndrom.getTemplate().isReinforcedEdgesAllowed());
430         Assert.assertEquals(false, syndrom.getTemplate().isExtenuatingEdgesAllowed());
431         Assert.assertEquals(true, syndrom.getTemplate().isNeutralEdgesAllowed());
432     }
433
434 /**
435     * This method tests if a NullPointerException is thrown when a gxl File without
436     * any template data is tried to be imported as a gxl with template.
437     *
438     * @throws IOException if the File can't be created or the file that is specified
439     * for the import can't be found.
440     * @throws SAXException if their occurs any problem parsing the document
441     *
442     @Test(expected = NullPointerException.class)
443     public void
444         testExceptionImportWithTemplateAfterExportWithoutTemplateInTemplateWithTemplate()
445             throws IOException, SAXException {
446             prepareSyndrom(false).importGXL(new File(nameTestGraph), true);
447         }*/
448
449 /**
450     * This method tests if a NullPointerException is thrown when a method call occurs
451     * on a empty GXLGraph element.
452     * The element below is empty because the export creates a gxl File without a graph
453     * with the name "template".
454     *
455     * @throws IOException if the File can't be created or the file that is specified
456     * for the import can't be found.
457     * @throws SAXException if their occurs any problem parsing the document
458     */
459     @Test(expected = NullPointerException.class)
460     public void
461         testExceptionSearchForGXLTemplateAfterExportWithoutTemplateInTemplateWithTemplate
462             () throws IOException, SAXException {
463             prepareSyndrom(false);
464             GXLDocument gxlDocument = new GXLDocument(new File(nameTestGraph));
465             GXLGraph templateGraph = (GXLGraph) gxlDocument.getElement("template");
466             templateGraph.getGraphElementCount();
467         }
```

```
456 // <————— relations of graph elements —————>
457 /**
458 * This method tests if the spheres that are created by importing the specified gxl
459 * -File contain the right vertices.
460 *
461 * @throws IOException if the File can't be created or the file that is specified
462 * for the import can't be found.
463 * @throws SAXException if their occurs any problem parsing the document
464 */
465 @Test
466 public void testVerticesInSpheresOfGraph() throws IOException, SAXException {
467     prepareSyndrom(false).importGXL(new File(nameTestGraph), false);
468     SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().
469         getGraphLayout().getGraph();
470     ArrayList<Sphere> spheres = (ArrayList<Sphere>) g.getSpheres();
471     Assert.assertEquals(new ArrayList<>(Collections.singletonList(5)), getNodeIDs(
472         spheres.get(0).getVertices()));
473     Assert.assertEquals(new ArrayList<>(Arrays.asList(6, 7, 8)), getNodeIDs(spheres.
474         get(1).getVertices()));
475     Assert.assertEquals(new ArrayList<>(Collections.singletonList(9)), getNodeIDs(
476         spheres.get(2).getVertices()));
477     Assert.assertEquals(new ArrayList<>(Arrays.asList(10, 11)), getNodeIDs(spheres.
478         get(3).getVertices()));
479     Assert.assertEquals(new ArrayList<>(Arrays.asList(12, 13)), getNodeIDs(spheres.
480         get(4).getVertices()));
481 }
482 /**
483 * This is a helper method that returns the IDs of the vertices that are passed to
484 * this method as list.
485 *
486 * @param pVertices a list of vertices those IDs are of interest
487 * @return the IDs of vertices contained in the passed list a list of Integers.
488 */
489 private ArrayList<Integer> getNodeIDs(LinkedList<Vertex> pVertices) {
490     ArrayList<Integer> verticesIDs = new ArrayList<>();
491     for (Vertex v : pVertices) {
492         verticesIDs.add(v.getId());
493     }
494     return verticesIDs;
495 }
496 /**
497 * This method tests if the edges that are created by importing the specified gxl-
498 * File connect the right vertices.
499 *
500 * @throws IOException if the File can't be created or the file that is specified
501 * for the import can't be found.
502 * @throws SAXException if their occurs any problem parsing the document
503 */
504 @Test
505 public void testEdgesConnectingRightVerticesOfGraph() throws IOException,
506     SAXException {
507     prepareSyndrom(false).importGXL(new File(nameTestGraph), false);
508     SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().
509         getGraphLayout().getGraph();
510     ArrayList<Edge> edges = new ArrayList<>(g.getEdges());
511     edges.sort(Comparator.comparingInt(Edge::getId));
512     Assert.assertEquals(5, g.getEndpoints(edges.get(0)).getFirst().getId());
513     Assert.assertEquals(6, g.getEndpoints(edges.get(0)).getSecond().getId());
```

```
505     Assert.assertEquals(9, g.getEndpoints(edges.get(1)).getFirst().getId());
506     Assert.assertEquals(8, g.getEndpoints(edges.get(1)).getSecond().getId());
507     Assert.assertEquals(11, g.getEndpoints(edges.get(2)).getFirst().getId());
508     Assert.assertEquals(10, g.getEndpoints(edges.get(2)).getSecond().getId());
509     Assert.assertEquals(12, g.getEndpoints(edges.get(3)).getFirst().getId());
510     Assert.assertEquals(13, g.getEndpoints(edges.get(3)).getSecond().getId());
511     Assert.assertEquals(10, g.getEndpoints(edges.get(4)).getFirst().getId());
512     Assert.assertEquals(7, g.getEndpoints(edges.get(4)).getSecond().getId());
513     Assert.assertEquals(11, g.getEndpoints(edges.get(5)).getFirst().getId());
514     Assert.assertEquals(7, g.getEndpoints(edges.get(5)).getSecond().getId());
515     Assert.assertEquals(5, g.getEndpoints(edges.get(0)).getFirst().getId());
516     Assert.assertEquals(12, g.getEndpoints(edges.get(6)).getFirst().getId());
517     Assert.assertEquals(5, g.getEndpoints(edges.get(0)).getSecond().getId());
518     Assert.assertEquals(7, g.getEndpoints(edges.get(6)).getSecond().getId());
519     Assert.assertEquals(13, g.getEndpoints(edges.get(7)).getFirst().getId());
520     Assert.assertEquals(7, g.getEndpoints(edges.get(7)).getSecond().getId());
521 }
522
523 // <----- coordinates ----->
524
525 /**
526 * This method tests if the spheres of the graph that is created by importing the
527 * specified gxl file have the right value for the coordinates-attribute.
528 *
529 * @throws IOException if the File can't be created or the file that is specified
530 * for the import can't be found.
531 * @throws SAXException if their occurs any problem parsing the document
532 */
533 @Test
534 public void testSpheresCoordinatesOfGraph() throws IOException, SAXException {
535     prepareSyndrom(false).importGXL(new File(nameTestGraph), false);
536     SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv();
537     getGraphLayout().getGraph();
538     ArrayList<Sphere> spheres = (ArrayList<Sphere>) g.getSpheres();
539     Assert.assertEquals(new Point2D.Double(20, 20), spheres.get(0).getCoordinates());
540     Assert.assertEquals(new Point2D.Double(270, 50), spheres.get(1).getCoordinates());
541     Assert.assertEquals(new Point2D.Double(620, 20), spheres.get(2).getCoordinates());
542     Assert.assertEquals(new Point2D.Double(200, 310), spheres.get(3).getCoordinates());
543     Assert.assertEquals(new Point2D.Double(445, 310), spheres.get(4).getCoordinates());
544 }
545
546 /**
547 * This method tests if the vertices of the graph that is created by importing the
548 * specified gxl file have the right value for the coordinates-attribute.
549 *
550 * @throws IOException if the File can't be created or the file that is specified
551 * for the import can't be found.
552 * @throws SAXException if their occurs any problem parsing the document
553 */
554 @Test
555 public void testVerticesCoordinatesOfGraph() throws IOException, SAXException {
556     prepareSyndrom(false).importGXL(new File(nameTestGraph), false);
557     SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv();
558     getGraphLayout().getGraph();
559     ArrayList<Vertex> vertices = new ArrayList<>(g.getVertices());
560     vertices.sort(Comparator.comparingInt(Vertex::getId));
```

```
555     Assert.assertEquals(new Point2D.Double(80, 80), vertices.get(0).getCoordinates
556         ());
557     Assert.assertEquals(new Point2D.Double(340, 140), vertices.get(1).
558         getCoordinates());
559     Assert.assertEquals(new Point2D.Double(415, 190), vertices.get(2).
560         getCoordinates());
561     Assert.assertEquals(new Point2D.Double(485, 140), vertices.get(3).
562         getCoordinates());
563     Assert.assertEquals(new Point2D.Double(740, 80), vertices.get(4).getCoordinates
564         ());
565     Assert.assertEquals(new Point2D.Double(220, 360), vertices.get(5).
566         getCoordinates());
567     Assert.assertEquals(new Point2D.Double(280, 420), vertices.get(6).
568         getCoordinates());
569     Assert.assertEquals(new Point2D.Double(555, 420), vertices.get(7).
570         getCoordinates());
571     Assert.assertEquals(new Point2D.Double(620, 360), vertices.get(8).
572         getCoordinates());
573 }
574
575 // <----- annotation ----->
576
577 /**
578 * This method tests if the spheres of the graph that is created by importing the
579 * specified gxl file have the right value for the annotation-attribute.
580 *
581 * @throws IOException if the File can't be created or the file that is specified
582 * for the import can't be found.
583 * @throws SAXException if their occurs any problem parsing the document
584 */
585 @Test
586 public void testSpheresAnnotationOfGraph() throws IOException, SAXException {
587     prepareSyndrom(false).importGXL(new File(nameTestGraph), false);
588     SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().
589         getGraphLayout().getGraph();
590     ArrayList<Sphere> spheres = (ArrayList<Sphere>) g.getSpheres();
591     Assert.assertEquals("Sphere 0", convertMapToList(spheres.get(0).getAnnotation()
592         .get(1)));
593     Assert.assertEquals("Sphäre 0", convertMapToList(spheres.get(0).getAnnotation()
594         .get(3)));
595     Assert.assertEquals("Sphere 1", convertMapToList(spheres.get(1).getAnnotation()
596         .get(1)));
597     Assert.assertEquals("Sphäre Nummer 1", convertMapToList(spheres.get(1).
598         getAnnotation()).get(3));
599     Assert.assertEquals("Sphere number 2", convertMapToList(spheres.get(2).
600         getAnnotation()).get(1));
601     Assert.assertEquals("Sphäre 2", convertMapToList(spheres.get(2).getAnnotation()
602         .get(3)));
603     Assert.assertEquals("Sphere number 3", convertMapToList(spheres.get(3).
604         getAnnotation()).get(1));
605     Assert.assertEquals("Sphäre Nummer 3", convertMapToList(spheres.get(3).
606         getAnnotation()).get(3));
607     Assert.assertEquals("Sphere 4", convertMapToList(spheres.get(4).getAnnotation()
608         .get(1)));
609     Assert.assertEquals("Sphäre 4", convertMapToList(spheres.get(4).getAnnotation()
610         .get(3)));
611 }
612
613 /**
614 * This method tests if the vertices of the graph that is created by importing the
615 * specified gxl file have the right value for the annotation-attribute.
616 *
```

```
594     * @throws IOException if the File can't be created or the file that is specified
595     * for the import can't be found.
596     * @throws SAXException if their occurs any problem parsing the document
597     */
598     @Test
599     public void testVerticesAnnotaionOfGraph() throws IOException, SAXException {
600         prepareSyndrom(false).importGXL(new File(nameTestGraph), false);
601         SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().
602             getGraphLayout().getGraph();
603         ArrayList<Vertex> vertices = new ArrayList<>(g.getVertices());
604         vertices.sort(Comparator.comparingInt(Vertex::getId));
605         Assert.assertEquals("Symptom 5", convertMapToList(vertices.get(0).getAnnotation
606             ().get(1)));
607         Assert.assertEquals("Symptom 5", convertMapToList(vertices.get(0).getAnnotation
608             ().get(3)));
609         Assert.assertEquals("Symptom 6", convertMapToList(vertices.get(1).getAnnotation
610             ().get(1)));
611         Assert.assertEquals("Symptom 6", convertMapToList(vertices.get(1).getAnnotation
612             ().get(3)));
613         Assert.assertEquals("Symptom number 7", convertMapToList(vertices.get(2).
614             getAnnotation()).get(1));
615         Assert.assertEquals("Symptom Nummer 7", convertMapToList(vertices.get(2).
616             getAnnotation()).get(3));
617         Assert.assertEquals("Symptom 8", convertMapToList(vertices.get(3).getAnnotation
618             ().get(1)));
619         Assert.assertEquals("Symptom 8", convertMapToList(vertices.get(3).getAnnotation
620             ().get(3)));
621         Assert.assertEquals("Symptom 9", convertMapToList(vertices.get(4).getAnnotation
622             ().get(1)));
623         Assert.assertEquals("Symptom 9", convertMapToList(vertices.get(4).getAnnotation
624             ().get(3)));
625         Assert.assertEquals("Symptom 10", convertMapToList(vertices.get(5).
626             getAnnotation()).get(1));
627         Assert.assertEquals("Symptom Nummer 10", convertMapToList(vertices.get(5).
628             getAnnotation()).get(3));
629         Assert.assertEquals("Symptom number 11", convertMapToList(vertices.get(6).
630             getAnnotation()).get(1));
631         Assert.assertEquals("Symptom 11", convertMapToList(vertices.get(6).
632             getAnnotation()).get(3));
633         Assert.assertEquals("Symptom 12", convertMapToList(vertices.get(7).
634             getAnnotation()).get(1));
635         Assert.assertEquals("Symptom 12", convertMapToList(vertices.get(7).
636             getAnnotation()).get(3));
637     }
638
639     /**
640      * This is a helper Method that returns the values of the passed map as list.
641      */
642     private ArrayList<String> convertMapToList(Map<String, String> pHashMap) {
643         ArrayList<String> list = new ArrayList<>();
644         for (Map.Entry map : pHashMap.entrySet()) {
645             list.add((String) map.getKey());
646             list.add((String) map.getValue());
647         }
648         return list;
649     }
650
651     // <----- size ----->
652
653     /**
654
```

```
637     * This method tests if the spheres of the graph that is created by importing the
638     * specified gxl file have the right value for the width-attribute.
639     *
640     * @throws IOException if the File can't be created or the file that is specified
641     * for the import can't be found.
642     * @throws SAXException if their occurs any problem parsing the document
643     */
644     @Test
645     public void testSpheresWidthOfGraph() throws IOException, SAXException {
646         prepareSyndrom(false).importGXL(new File(nameTestGraph), false);
647         SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().getGraphLayout().getGraph();
648         ArrayList<Sphere> spheres = (ArrayList<Sphere>) g.getSpheres();
649         Assert.assertEquals(200.0, spheres.get(0).getWidth());
650         Assert.assertEquals(300.0, spheres.get(1).getWidth());
651         Assert.assertEquals(200.0, spheres.get(2).getWidth());
652         Assert.assertEquals(200.0, spheres.get(3).getWidth());
653         Assert.assertEquals(200.0, spheres.get(4).getWidth());
654     }
655     /**
656     * This method tests if the spheres of the graph that is created by importing the
657     * specified gxl file have the right value for the height-attribute.
658     *
659     * @throws IOException if the File can't be created or the file that is specified
660     * for the import can't be found.
661     * @throws SAXException if their occurs any problem parsing the document
662     */
663     @Test
664     public void testSpheresHeightOfGraph() throws IOException, SAXException {
665         prepareSyndrom(false).importGXL(new File(nameTestGraph), false);
666         SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().getGraphLayout().getGraph();
667         ArrayList<Sphere> spheres = (ArrayList<Sphere>) g.getSpheres();
668         Assert.assertEquals(200.0, spheres.get(0).getHeight());
669         Assert.assertEquals(250.0, spheres.get(1).getHeight());
670         Assert.assertEquals(200.0, spheres.get(2).getHeight());
671         Assert.assertEquals(200.0, spheres.get(3).getHeight());
672         Assert.assertEquals(200.0, spheres.get(4).getHeight());
673     }
674     /**
675     * This method tests if the vertices of the graph that is created by importing the
676     * specified gxl file have the right value for the size-attribute.
677     *
678     * @throws IOException if the File can't be created or the file that is specified
679     * for the import can't be found.
680     * @throws SAXException if their occurs any problem parsing the document
681     */
682     @Test
683     public void testVerticesSizeOfGraph() throws IOException, SAXException {
684         prepareSyndrom(false).importGXL(new File(nameTestGraph), false);
685         SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().getGraphLayout().getGraph();
686         ArrayList<Vertex> vertices = new ArrayList<>(g.getVertices());
687         vertices.sort(Comparator.comparingInt(Vertex::getId));
688         Assert.assertEquals(50, vertices.get(0).getSize());
689         Assert.assertEquals(50, vertices.get(1).getSize());
```

```
690     Assert.assertEquals(80, vertices.get(6).getSize());
691     Assert.assertEquals(80, vertices.get(7).getSize());
692     Assert.assertEquals(50, vertices.get(8).getSize());
693 }
694
695 // <———— Color —————>
696
697 /**
698 * This method tests if the spheres of the graph that is created by importing the
699 * specified gxl file have the right value for the fillColor-attribute.
700 *
701 * @throws IOException if the File can't be created or the file that is specified
702 * for the import can't be found.
703 * @throws SAXException if their occurs any problem parsing the document
704 */
705 @Test
706 public void testSpheresFillColorOfGraph() throws IOException, SAXException {
707     prepareSyndrom(false).importGXL(new File(nameTestGraph), false);
708     SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().
709         getGraphLayout().getGraph();
710     ArrayList<Sphere> spheres = (ArrayList<Sphere>) g.getSpheres();
711     Assert.assertEquals(new java.awt.Color(86, 151, 31, 183), spheres.get(0).
712         getColor());
713     Assert.assertEquals(new java.awt.Color(86, 151, 31, 183), spheres.get(1).
714         getColor());
715     Assert.assertEquals(new java.awt.Color(86, 151, 31, 183), spheres.get(2).
716         getColor());
717     Assert.assertEquals(new java.awt.Color(24, 54, 11, 178), spheres.get(3).
718         getColor());
719     Assert.assertEquals(new java.awt.Color(24, 54, 11, 178), spheres.get(4).
720         getColor());
721 }
722 /**
723 * This method tests if the vertices of the graph that is created by importing the
724 * specified gxl file have the right value for the fillColor-attribute.
725 *
726 * @throws IOException if the File can't be created or the file that is specified
727 * for the import can't be found.
728 * @throws SAXException if their occurs any problem parsing the document
729 */
730 @Test
731 public void testVerticesFillColorOfGraph() throws IOException, SAXException {
732     prepareSyndrom(false).importGXL(new File(nameTestGraph), false);
733     SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().
734         getGraphLayout().getGraph();
735     ArrayList<Vertex> vertices = new ArrayList<>(g.getVertices());
736     vertices.sort(Comparator.comparingInt(Vertex::getId));
737     Assert.assertEquals(new java.awt.Color(88, 88, 219, 220), vertices.get(0).
738         getFillColor());
739     Assert.assertEquals(new java.awt.Color(88, 88, 219, 220), vertices.get(1).
740         getFillColor());
741     Assert.assertEquals(new java.awt.Color(0xC80070), vertices.get(2).getFillColor
742         ());
743     Assert.assertEquals(new java.awt.Color(88, 88, 219, 220), vertices.get(3).
744         getFillColor());
745     Assert.assertEquals(new java.awt.Color(88, 88, 219, 220), vertices.get(4).
746         getFillColor());
747     Assert.assertEquals(new java.awt.Color(11, 19, 90, 220), vertices.get(5).
748         getFillColor());
749     Assert.assertEquals(new java.awt.Color(11, 19, 90, 220), vertices.get(6).
750         getFillColor());
```

```
734     Assert.assertEquals(new java.awt.Color(11, 19, 90, 220), vertices.get(7).  
735         getFillColor());  
736     Assert.assertEquals(new java.awt.Color(11, 19, 90, 220), vertices.get(8).  
737         getFillColor());  
738     }  
739     /**  
740      * This method tests if the vertices of the graph that is created by importing the  
741      * specified gxl file have the right value for the drawColor-attribute.  
742      *  
743      * @throws IOException if the File can't be created or the file that is specified  
744      * for the import can't be found.  
745      * @throws SAXException if their occurs any problem parsing the document  
746      */  
747     @Test  
748     public void testVerticesDrawColorOfGraph() throws IOException, SAXException {  
749         prepareSyndrom(false).importGXL(new File(nameTestGraph), false);  
750         SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().  
751             getGraphLayout().getGraph();  
752         ArrayList<Vertex> vertices = new ArrayList<>(g.getVertices());  
753         vertices.sort(Comparator.comparingInt(Vertex::getId));  
754         Assert.assertEquals(Values.getInstance().getDrawPaintVertex(), vertices.get(0).  
755             getDrawColor());  
756         Assert.assertEquals(Values.getInstance().getDrawPaintVertex(), vertices.get(1).  
757             getDrawColor());  
758         Assert.assertEquals(new java.awt.Color(200, 10, 10, 203), vertices.get(2).  
759             getDrawColor());  
760         Assert.assertEquals(Values.getInstance().getDrawPaintVertex(), vertices.get(3).  
761             getDrawColor());  
762         Assert.assertEquals(Values.getInstance().getDrawPaintVertex(), vertices.get(4).  
763             getDrawColor());  
764         Assert.assertEquals(Values.getInstance().getDrawPaintVertex(), vertices.get(5).  
765             getDrawColor());  
766         Assert.assertEquals(Values.getInstance().getDrawPaintVertex(), vertices.get(6).  
767             getDrawColor());  
768         Assert.assertEquals(Values.getInstance().getDrawPaintVertex(), vertices.get(7).  
769             getDrawColor());  
770         Assert.assertEquals(Values.getInstance().getDrawPaintVertex(), vertices.get(8).  
771             getDrawColor());  
772     }  
773     /**  
774      * This method tests if the edges of the graph that is created by importing the  
775      * specified gxl file have the right value for the color-attribute.  
776      *  
777      * @throws IOException if the File can't be created or the file that is specified  
778      * for the import can't be found.  
779      * @throws SAXException if their occurs any problem parsing the document  
780      */  
781     @Test  
782     public void testEdgesColorOfGraph() throws IOException, SAXException {  
783         prepareSyndrom(false).importGXL(new File(nameTestGraph), false);  
784         SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().  
785             getGraphLayout().getGraph();  
786         ArrayList<Edge> edges = new ArrayList<>(g.getEdges());  
787         edges.sort(Comparator.comparingInt(Edge::getId));  
788         Assert.assertEquals(Values.getInstance().getEdgePaint(), edges.get(0).getColor  
789             ());  
790         Assert.assertEquals(Values.getInstance().getEdgePaint(), edges.get(1).getColor  
791             ());
```

```
776     Assert.assertEquals(Values.getInstance().getEdgePaint(), edges.get(2).getColor
    ());
777     Assert.assertEquals(Values.getInstance().getEdgePaint(), edges.get(3).getColor
    ());
778     Assert.assertEquals(Values.getInstance().getEdgePaint(), edges.get(4).getColor
    ());
779     Assert.assertEquals(new java.awt.Color(28, 56, 249, 130), edges.get(5).getColor
    ());
780     Assert.assertEquals(new java.awt.Color(28, 56, 249, 130), edges.get(6).getColor
    ());
781     Assert.assertEquals(Values.getInstance().getEdgePaint(), edges.get(7).getColor
    ());
782 }
783
784 // <———— shape ——————>
785
786 /**
787 * This method tests if the vertices of the graph that is created by importing the
788 * specified gxl file have the right value for the shape-attribute.
789 *
790 * @throws IOException if the File can't be created or the file that is specified
791 * for the import can't be found.
792 * @throws SAXException if their occurs any problem parsing the document
793 */
794 @Test
795 public void testVerticesShapeOfGraph() throws IOException, SAXException {
796     prepareSyndrom(true).importGXL(new File(nameTestGraph), true);
797     SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().
798         getGraphLayout().getGraph();
799     ArrayList<Vertex> vertices = new ArrayList<>(g.getVertices());
800     vertices.sort(Comparator.comparingInt(Vertex::getId));
801     Assert.assertEquals(VertexShapeType.CIRCLE, vertices.get(0).getShape());
802     Assert.assertEquals(VertexShapeType.CIRCLE, vertices.get(1).getShape());
803     Assert.assertEquals(VertexShapeType.CIRCLE, vertices.get(2).getShape());
804     Assert.assertEquals(VertexShapeType.CIRCLE, vertices.get(3).getShape());
805     Assert.assertEquals(VertexShapeType.CIRCLE, vertices.get(4).getShape());
806     Assert.assertEquals(VertexShapeType.RECTANGLE, vertices.get(5).getShape());
807     Assert.assertEquals(VertexShapeType.RECTANGLE, vertices.get(6).getShape());
808     Assert.assertEquals(VertexShapeType.RECTANGLE, vertices.get(7).getShape());
809     Assert.assertEquals(VertexShapeType.RECTANGLE, vertices.get(8).getShape());
810 }
811
812 /**
813 * This method tests if the spheres of the graph that is created by importing the
814 * specified gxl file have the right value for the font-attribute.
815 *
816 * @throws IOException if the File can't be created or the file that is specified
817 * for the import can't be found.
818 * @throws SAXException if their occurs any problem parsing the document
819 */
820 @Test
821 public void testSpheresFontOfGraph() throws IOException, SAXException {
822     prepareSyndrom(true).importGXL(new File(nameTestGraph), true);
823     SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().
824         getGraphLayout().getGraph();
825     ArrayList<Sphere> spheres = (ArrayList<Sphere>) g.getSpheres();
826     Assert.assertEquals(values.getFontSphere(), spheres.get(0).getFont());
827     Assert.assertEquals(values.getFontSphere(), spheres.get(1).getFont());
828     Assert.assertEquals(values.getFontSphere(), spheres.get(2).getFont());
```

```
826     Assert.assertEquals("Kalam", spheres.get(3).getFont());
827     Assert.assertEquals("Kalam", spheres.get(4).getFont());
828 }
829 /**
830 * This method tests if the spheres of the graph that is created by importing the
831 * specified gxl file have the right value for the fontSize-attribute.
832 *
833 * @throws IOException if the File can't be created or the file that is specified
834 * for the import can't be found.
835 * @throws SAXException if their occurs any problem parsing the document
836 */
837 @Test
838 public void testSpheresFontSizeOfGraph() throws IOException, SAXException {
839     prepareSyndrom(true).importGXL(new File(nameTestGraph), true);
840     SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().
841         getGraphLayout().getGraph();
842     ArrayList<Sphere> spheres = (ArrayList<Sphere>) g.getSpheres();
843     Assert.assertEquals(10, spheres.get(0).getFontSize());
844     Assert.assertEquals(10, spheres.get(1).getFontSize());
845     Assert.assertEquals(10, spheres.get(2).getFontSize());
846     Assert.assertEquals(24, spheres.get(3).getFontSize());
847     Assert.assertEquals(24, spheres.get(4).getFontSize());
848 }
849 /**
850 * This method tests if the vertices of the graph that is created by importing the
851 * specified gxl file have the right value for the font-attribute.
852 *
853 * @throws IOException if the File can't be created or the file that is specified
854 * for the import can't be found.
855 * @throws SAXException if their occurs any problem parsing the document
856 */
857 @Test
858 public void testVerticesFontOfGraph() throws IOException, SAXException {
859     prepareSyndrom(true).importGXL(new File(nameTestGraph), true);
860     SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().
861         getGraphLayout().getGraph();
862     ArrayList<Vertex> vertices = new ArrayList<>(g.getVertices());
863     vertices.sort(Comparator.comparingInt(Vertex::getId));
864     Assert.assertEquals("Roboto", vertices.get(0).getFont());
865     Assert.assertEquals("Roboto", vertices.get(1).getFont());
866     Assert.assertEquals("Roboto", vertices.get(2).getFont());
867     Assert.assertEquals("Roboto", vertices.get(3).getFont());
868     Assert.assertEquals("Roboto", vertices.get(4).getFont());
869     Assert.assertEquals("Roboto", vertices.get(5).getFont());
870     Assert.assertEquals("Mali", vertices.get(6).getFont());
871     Assert.assertEquals("Mali", vertices.get(7).getFont());
872     Assert.assertEquals("Roboto", vertices.get(8).getFont());
873 }
874 /**
875 * This method tests if the vertices of the graph that is created by importing the
876 * specified gxl file have the right value for the fontSize-attribute.
877 *
878 * @throws IOException if the File can't be created or the file that is specified
879 * for the import can't be found.
880 * @throws SAXException if their occurs any problem parsing the document
881 */
882 @Test
883 public void testVerticesFontSizeOfGraph() throws IOException, SAXException {
884     prepareSyndrom(true).importGXL(new File(nameTestGraph), true);
```

```
880     SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().  
881         getGraphLayout().getGraph();  
882     ArrayList<Vertex> vertices = new ArrayList<>(g.getVertices());  
883     vertices.sort(Comparator.comparingInt(Vertex::getId));  
884     Assert.assertEquals(12, vertices.get(0).getFontSize());  
885     Assert.assertEquals(12, vertices.get(1).getFontSize());  
886     Assert.assertEquals(12, vertices.get(2).getFontSize());  
887     Assert.assertEquals(12, vertices.get(3).getFontSize());  
888     Assert.assertEquals(12, vertices.get(4).getFontSize());  
889     Assert.assertEquals(7, vertices.get(5).getFontSize());  
890     Assert.assertEquals(14, vertices.get(6).getFontSize());  
891     Assert.assertEquals(14, vertices.get(7).getFontSize());  
892     Assert.assertEquals(7, vertices.get(8).getFontSize());  
893 }  
894 // <———— sphere locking —————>  
895 /**  
 * This method tests if the spheres of the graph that is created by importing the  
 * specified gxl file have the right value for the isLockedAnnotation-attribute.  
 *  
 * @throws IOException if the File can't be created or the file that is specified  
 * for the import can't be found.  
 * @throws SAXException if their occurs any problem parsing the document  
 */  
902 @Test  
903 public void testSpheresLockedAnnotationOfGraph() throws IOException, SAXException {  
904     prepareSyndrom(true).importGXL(new File(nameTestGraph), true);  
905     SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().  
906         getGraphLayout().getGraph();  
907     ArrayList<Sphere> spheres = (ArrayList<Sphere>) g.getSpheres();  
908     Assert.assertTrue(spheres.get(0).isLockedAnnotation());  
909     Assert.assertTrue(spheres.get(1).isLockedAnnotation());  
910     Assert.assertFalse(spheres.get(2).isLockedAnnotation());  
911     Assert.assertFalse(spheres.get(3).isLockedAnnotation());  
912     Assert.assertFalse(spheres.get(4).isLockedAnnotation());  
913 }  
914 /**  
 * This method tests if the spheres of the graph that is created by importing the  
 * specified gxl file have the right value for the isLockedVertices-attribute.  
 *  
 * @throws IOException if the File can't be created or the file that is specified  
 * for the import can't be found.  
 * @throws SAXException if their occurs any problem parsing the document  
 */  
920 @Test  
921 public void testSpheresLockedVerticesOfGraph() throws IOException, SAXException {  
922     prepareSyndrom(true).importGXL(new File(nameTestGraph), true);  
923     SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().  
924         getGraphLayout().getGraph();  
925     ArrayList<Sphere> spheres = (ArrayList<Sphere>) g.getSpheres();  
926     Assert.assertFalse(spheres.get(0).isLockedVertices());  
927     Assert.assertTrue(spheres.get(1).isLockedVertices());  
928     Assert.assertFalse(spheres.get(2).isLockedVertices());  
929     Assert.assertFalse(spheres.get(3).isLockedVertices());  
930     Assert.assertFalse(spheres.get(4).isLockedVertices());  
931 }  
932 /**
```

```
934     * This method tests if the spheres of the graph that is created by importing the
935     * specified gxl file have the right value for the isLockedPosition-attribute.
936     *
937     * @throws IOException if the File can't be created or the file that is specified
938     * for the import can't be found.
939     * @throws SAXException if their occurs any problem parsing the document
940     */
941     @Test
942     public void testSpheresLockedPositionOfGraph() throws IOException, SAXException {
943         prepareSyndrom(true).importGXL(new File(nameTestGraph), true);
944         SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().getGraphLayout().getGraph();
945         ArrayList<Sphere> spheres = (ArrayList<Sphere>) g.getSpheres();
946         Assert.assertFalse(spheres.get(0).isLockedPosition());
947         Assert.assertFalse(spheres.get(1).isLockedPosition());
948         Assert.assertFalse(spheres.get(2).isLockedPosition());
949         Assert.assertTrue(spheres.get(3).isLockedPosition());
950         Assert.assertTrue(spheres.get(4).isLockedPosition());
951     }
952     /**
953     * This method tests if the spheres of the graph that is created by importing the
954     * specified gxl file have the right value for the isLockedStyle-attribute.
955     *
956     * @throws IOException if the File can't be created or the file that is specified
957     * for the import can't be found.
958     * @throws SAXException if their occurs any problem parsing the document
959     */
960     @Test
961     public void testSpheresLockedStyleOfGraph() throws IOException, SAXException {
962         prepareSyndrom(true).importGXL(new File(nameTestGraph), true);
963         SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().getGraphLayout().getGraph();
964         ArrayList<Sphere> spheres = (ArrayList<Sphere>) g.getSpheres();
965         Assert.assertFalse(spheres.get(0).isLockedStyle());
966         Assert.assertFalse(spheres.get(1).isLockedStyle());
967         Assert.assertFalse(spheres.get(2).isLockedStyle());
968         Assert.assertFalse(spheres.get(3).isLockedStyle());
969         Assert.assertTrue(spheres.get(4).isLockedStyle());
970     }
971     /**
972     * This method tests if the spheres of the graph that is created by importing the
973     * specified gxl file have the right value for the maxAmountVertices-attribute.
974     *
975     * @throws IOException if the File can't be created or the file that is specified
976     * for the import can't be found.
977     * @throws SAXException if their occurs any problem parsing the document
978     */
979     @Test
980     public void testSpheresLockedMaxAmountVerticesOfGraph() throws IOException,
981     SAXException {
982         prepareSyndrom(true).importGXL(new File(nameTestGraph), true);
983         SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().getGraphLayout().getGraph();
984         ArrayList<Sphere> spheres = (ArrayList<Sphere>) g.getSpheres();
985         Assert.assertEquals("", spheres.get(0).getLockedMaxAmountVertices());
986         Assert.assertEquals("", spheres.get(1).getLockedMaxAmountVertices());
987         Assert.assertEquals("", spheres.get(2).getLockedMaxAmountVertices());
988         Assert.assertEquals("17", spheres.get(3).getLockedMaxAmountVertices());
989         Assert.assertEquals("5", spheres.get(4).getLockedMaxAmountVertices());
990     }
```

```
986
987
988 // <———— vertex locking —————>
989 /**
990 * This method tests if the vertices of the graph that is created by importing the
991 * specified gxl file have the right value for the isLockedStyle-attribute.
992 *
993 * @throws IOException if the File can't be created or the file that is specified
994 * for the import can't be found.
995 * @throws SAXException if there occurs any problem parsing the document
996 */
997 @Test
998 public void testVerticesLockedStyleOfGraph() throws IOException, SAXException {
999     prepareSyndrom(true).importGXL(new File(nameTestGraph), true);
1000    SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().getGraphLayout().getGraph();
1001    ArrayList<Vertex> vertices = new ArrayList<>(g.getVertices());
1002    vertices.sort(Comparator.comparingInt(Vertex::getId));
1003    Assert.assertTrue(vertices.get(0).isLockedStyle());
1004    Assert.assertFalse(vertices.get(1).isLockedStyle());
1005    Assert.assertTrue(vertices.get(2).isLockedStyle());
1006    Assert.assertFalse(vertices.get(3).isLockedStyle());
1007    Assert.assertFalse(vertices.get(4).isLockedStyle());
1008    Assert.assertTrue(vertices.get(5).isLockedStyle());
1009    Assert.assertFalse(vertices.get(6).isLockedStyle());
1010    Assert.assertFalse(vertices.get(7).isLockedStyle());
1011    Assert.assertFalse(vertices.get(8).isLockedStyle());
1012 }
1013 /**
1014 * This method tests if the vertices of the graph that is created by importing the
1015 * specified gxl file have the right value for the isLockedAnnotation-attribute.
1016 *
1017 * @throws IOException if the File can't be created or the file that is specified
1018 * for the import can't be found.
1019 * @throws SAXException if there occurs any problem parsing the document
1020 */
1021 @Test
1022 public void testVerticesLockedAnnotationOfGraph() throws IOException, SAXException {
1023     prepareSyndrom(true).importGXL(new File(nameTestGraph), true);
1024    SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().getGraphLayout().getGraph();
1025    ArrayList<Vertex> vertices = new ArrayList<>(g.getVertices());
1026    vertices.sort(Comparator.comparingInt(Vertex::getId));
1027    Assert.assertFalse(vertices.get(0).isLockedAnnotation());
1028    Assert.assertFalse(vertices.get(1).isLockedAnnotation());
1029    Assert.assertTrue(vertices.get(2).isLockedAnnotation());
1030    Assert.assertTrue(vertices.get(3).isLockedAnnotation());
1031    Assert.assertTrue(vertices.get(4).isLockedAnnotation());
1032    Assert.assertFalse(vertices.get(5).isLockedAnnotation());
1033    Assert.assertFalse(vertices.get(6).isLockedAnnotation());
1034 }
1035 /**
1036 * This method tests if the vertices of the graph that is created by importing the
1037 * specified gxl file have the right value for the isLockedPosition-attribute.
1038 *
```

```
1039     * @throws IOException if the File can't be created or the file that is specified
1040     * for the import can't be found.
1041     * @throws SAXException if their occurs any problem parsing the document
1042     */
1043     @Test
1044     public void testVerticesLockedPositionOfGraph() throws IOException, SAXException {
1045         prepareSyndrom(true).importGXL(new File(nameTestGraph), true);
1046         SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().
1047             getGraphLayout().getGraph();
1048         ArrayList<Vertex> vertices = new ArrayList<>(g.getVertices());
1049         vertices.sort(Comparator.comparingInt(Vertex::getId));
1050         Assert.assertFalse(vertices.get(0).isLockedPosition());
1051         Assert.assertFalse(vertices.get(1).isLockedPosition());
1052         Assert.assertTrue(vertices.get(2).isLockedPosition());
1053         Assert.assertFalse(vertices.get(3).isLockedPosition());
1054         Assert.assertFalse(vertices.get(4).isLockedPosition());
1055         Assert.assertTrue(vertices.get(5).isLockedPosition());
1056         Assert.assertTrue(vertices.get(6).isLockedPosition());
1057         Assert.assertTrue(vertices.get(7).isLockedPosition());
1058         Assert.assertFalse(vertices.get(8).isLockedPosition());
1059     }
1060
1061     /**
1062      * This method tests if the edges of the graph that is created by importing the
1063      * specified gxl file have the right value for the isLockedStyle-attribute.
1064      *
1065      * @throws IOException if the File can't be created or the file that is specified
1066      * for the import can't be found.
1067      * @throws SAXException if their occurs any problem parsing the document
1068      */
1069     @Test
1070     public void testEdgesLockedStyleOfGraph() throws IOException, SAXException {
1071         prepareSyndrom(true).importGXL(new File(nameTestGraph), true);
1072         SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().
1073             getGraphLayout().getGraph();
1074         ArrayList<Edge> edges = new ArrayList<>(g.getEdges());
1075         edges.sort(Comparator.comparingInt(Edge::getId));
1076         Assert.assertFalse(edges.get(0).isLockedStyle());
1077         Assert.assertFalse(edges.get(1).isLockedStyle());
1078         Assert.assertTrue(edges.get(2).isLockedStyle());
1079         Assert.assertFalse(edges.get(3).isLockedStyle());
1080         Assert.assertFalse(edges.get(4).isLockedStyle());
1081         Assert.assertTrue(edges.get(5).isLockedStyle());
1082         Assert.assertTrue(edges.get(6).isLockedStyle());
1083     /**
1084      * This method tests if the edges of the graph that is created by importing the
1085      * specified gxl file have the right value for the isLockedEdgeType-attribute.
1086      *
1087      * @throws IOException if the File can't be created or the file that is specified
1088      * for the import can't be found.
1089      * @throws SAXException if their occurs any problem parsing the document
1090      */
1091     @Test
1092     public void testEdgesLockedEdgeTypeOfGraph() throws IOException, SAXException {
1093         prepareSyndrom(true).importGXL(new File(nameTestGraph), true);
1094         SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().
1095             getGraphLayout().getGraph();
```

```
1093     ArrayList<Edge> edges = new ArrayList<>(g.getEdges());
1094     edges.sort(Comparator.comparingInt(Edge::getId));
1095     Assert.assertTrue(edges.get(0).isLockedEdgeType());
1096     Assert.assertTrue(edges.get(1).isLockedEdgeType());
1097     Assert.assertFalse(edges.get(2).isLockedEdgeType());
1098     Assert.assertFalse(edges.get(3).isLockedEdgeType());
1099     Assert.assertFalse(edges.get(4).isLockedEdgeType());
1100     Assert.assertFalse(edges.get(5).isLockedEdgeType());
1101     Assert.assertFalse(edges.get(6).isLockedEdgeType());
1102     Assert.assertTrue(edges.get(7).isLockedEdgeType());
1103 }
1104
1105 // <———— other edge style —————>
1106
1107 /**
1108 * This method tests if the edges of the graph that is created by importing the
1109 * specified gxl file have the right value for the arrowType-attribute.
1110 *
1111 * @throws IOException if the File can't be created or the file that is specified
1112 * for the import can't be found.
1113 * @throws SAXException if their occurs any problem parsing the document
1114 */
1115 @Test
1116 public void testEdgesArrowTypeOfGraph() throws IOException, SAXException {
1117     prepareSyndrom(false).importGXL(new File(nameTestGraph), false);
1118     SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().
1119         getGraphLayout().getGraph();
1120     ArrayList<Edge> edges = new ArrayList<>(g.getEdges());
1121     edges.sort(Comparator.comparingInt(Edge::getId));
1122     Assert.assertEquals(EdgeArrowType.EXTENUATING, edges.get(0).getArrowType());
1123     Assert.assertEquals(EdgeArrowType.EXTENUATING, edges.get(1).getArrowType());
1124     Assert.assertEquals(EdgeArrowType.NEUTRAL, edges.get(2).getArrowType());
1125     Assert.assertEquals(EdgeArrowType.NEUTRAL, edges.get(3).getArrowType());
1126     Assert.assertEquals(EdgeArrowType.REINFORCED, edges.get(4).getArrowType());
1127     Assert.assertEquals(EdgeArrowType.REINFORCED, edges.get(5).getArrowType());
1128     Assert.assertEquals(EdgeArrowType.REINFORCED, edges.get(6).getArrowType());
1129     Assert.assertEquals(EdgeArrowType.REINFORCED, edges.get(7).getArrowType());
1130 }
1131 /**
1132 * This method tests if the edges of the graph that is created by importing the
1133 * specified gxl file have the right value for the stroke-attribute.
1134 *
1135 * @throws IOException if the File can't be created or the file that is specified
1136 * for the import can't be found.
1137 */
1138 @Test
1139 public void testEdgesStrokeTypeOfGraph() throws IOException, SAXException {
1140     prepareSyndrom(false).importGXL(new File(nameTestGraph), false);
1141     SyndromGraph<Vertex, Edge> g = (SyndromGraph<Vertex, Edge>) syndrom.getVv().
1142         getGraphLayout().getGraph();
1143     ArrayList<Edge> edges = new ArrayList<>(g.getEdges());
1144     edges.sort(Comparator.comparingInt(Edge::getId));
1145     Assert.assertEquals(StrokeType.BASIC, edges.get(0).getStroke());
1146     Assert.assertEquals(StrokeType.BASIC_WEIGHT, edges.get(1).getStroke());
1147     Assert.assertEquals(StrokeType.DOTTED_WEIGHT, edges.get(2).getStroke());
1148     Assert.assertEquals(StrokeType.DOTTED, edges.get(3).getStroke());
1149     Assert.assertEquals(StrokeType.BASIC, edges.get(4).getStroke());
1150     Assert.assertEquals(StrokeType.DASHED, edges.get(5).getStroke());
1151     Assert.assertEquals(StrokeType.DASHED, edges.get(6).getStroke());
```

```
1149     Assert.assertEquals(StrokeType.DASHED_WEIGHT, edges.get(7).getStroke());
1150 }
1151
1152 public void testColorDescription(){
1153     java.awt.Color color = new Color(11, 22, 33, 100);
1154     Assert.assertEquals("java.awt.Color[r=11,g=22,b=33,a=100]", gxlio.
1155         getPaintDescription(color));
1156 /*
1157     @Before
1158     public void prepare() {
1159         doc = new GXLDocument();
1160         GXLGraph gx1Graph = new GXLGraph(syndromName);
1161         GXLNNode sphere0 = new GXLNNode("0");
1162         java.awt.Color fillPaint = new java.awt.Color(255, 0, 0, 255);
1163         String paintDescription = getPaintDescription(fillPaint);
1164         sphere0.setAttr("color", new GXLString(paintDescription));
1165         GXLNNode sphere1 = new GXLNNode("1");
1166         sphere1.setAttr("width", new GXLString("25.0"));
1167         sphere1.setAttr("height", new GXLString("30.0"));
1168         GXLNNode sphere2 = new GXLNNode("2");
1169
1170         Point2D coordinates = new java.awt.geom.Point2D.Double(250.0, 500.0);
1171         GXLNNode vertex0 = new GXLNNode("3");
1172         vertex0.setAttr("coordinates", new GXLString(coordinates.toString()));
1173         GXLNNode vertex1 = new GXLNNode("4");
1174         vertex1.setAttr("size", new GXLInt(15));
1175         GXLNNode vertex2 = new GXLNNode("5");
1176         GXLNNode vertex3 = new GXLNNode("6");
1177         GXLNNode vertex4 = new GXLNNode("7");
1178         GXLNNode vertex5 = new GXLNNode("8");
1179         GXLEdge edge0 = new GXLEdge("2", "3");
1180         GXLEdge edge1 = new GXLEdge("3", "5");
1181         GXLEdge edge2 = new GXLEdge("6", "8");
1182         GXLEdge edge3 = new GXLEdge("8", "6");
1183         GXLEdge edge4 = new GXLEdge("2", "7");
1184         GXLEdge edge5 = new GXLEdge("5", "4");
1185         gx1Graph.add(sphere0);
1186         gx1Graph.add(sphere1);
1187         gx1Graph.add(sphere2);
1188         gx1Graph.add(vertex0);
1189         gx1Graph.add(vertex1);
1190         gx1Graph.add(vertex2);
1191         gx1Graph.add(vertex3);
1192         gx1Graph.add(vertex4);
1193         gx1Graph.add(vertex5);
1194         gx1Graph.add(edge0);
1195         gx1Graph.add(edge1);
1196         gx1Graph.add(edge2);
1197         gx1Graph.add(edge3);
1198         gx1Graph.add(edge4);
1199         gx1Graph.add(edge5);
1200         doc.getDocumentElement().add(gx1Graph);
1201         try {
1202             doc.write(new File("GXLTest"));
1203         } catch (IOException e) {
1204             logger.error(e.toString());
1205         }
1206     }
1207
1208     @Test
1209     public void testElementNumber() {
```

```
1210     logger.info("Ich bin das GXLDokument: " + doc);
1211     int numberOfGraphs = doc.getDocumentElement().getGraphCount();
1212     Assert.assertEquals(1, numberOfGraphs);
1213     int numberOfElementsInTheGraph = doc.getElement(syndromName).getChildCount();
1214     Assert.assertEquals(15, numberOfElementsInTheGraph);
1215 }
1216
1217 @Test
1218 public void testColor() {
1219     int numberOfGraphs = doc.getDocumentElement().getGraphCount();
1220     Assert.assertEquals(1, numberOfGraphs);
1221     GXLNNode sphere0 = (GXLNNode) doc.getElement("0");
1222     java.awt.Color expectedColor = new Color(255, 0, 0, 255);
1223     String sphereColorDescription = ((GXLString) sphere0.getAttr("color")).getValue()
1224         .getValue();
1225     String[] colorArray = gxlio.getNumberArrayFromString(sphereColorDescription);
1226     java.awt.Color sphereColor = new Color(Integer.parseInt(colorArray[0]), Integer
1227         .parseInt(colorArray[1]),
1228         Integer.parseInt(colorArray[2]), Integer.parseInt(colorArray[3]));
1229     logger.info("Beschreibung der erwarteten Farbe: " + getPaintDescription(
1230         expectedColor));
1231     logger.info("Beschreibung der Farbe der Sphäre: " + getPaintDescription(
1232         sphereColor));
1233     Assert.assertEquals(expectedColor, sphereColor);
1234 }
1235
1236 @Test
1237 public void testCoordinates() {
1238     GXLNNode vertex0 = (GXLNNode) doc.getElement("3");
1239     String coordinatesDescription = ((GXLString) vertex0.getAttr("coordinates"))
1240         .getValue().getValue();
1241     String[] coordinatesArray = gxlio.getNumberArrayFromString(
1242         coordinatesDescription);
1243     Assert.assertEquals(250.0, Double.parseDouble(coordinatesArray[0]), 0.0);
1244     Assert.assertEquals(500.0, Double.parseDouble(coordinatesArray[1]), 0.0);
1245 }
1246
1247 @Test
1248 public void testSize() {
1249     GXLNNode sphere1 = (GXLNNode) doc.getElement("1");
1250     GXLNNode vertex1 = (GXLNNode) doc.getElement("4");
1251     double sphereWidth = Double.parseDouble(((GXLString) sphere1.getAttr("width"))
1252         .getValue().getValue());
1253     double sphereheight = Double.parseDouble(((GXLString) sphere1.getAttr("height")
1254         .getValue().getValue());
1255     Assert.assertEquals(25.0, sphereWidth, 0.0);
1256     Assert.assertEquals(30.0, sphereheight, 0.0);
1257     int vertexSize = (((GXLint) vertex1.getAttr("size")).getValue()).getIntValue();
1258     Assert.assertEquals(15, vertexSize);
1259 }
1260
1261 /**
1262  * Forms a description of a color.
1263  *
1264  * @param color the color that need to be described
1265  * @return the description of the color as a String
1266  */
1267 private String getPaintDescription(Color color) {
1268     return ("java.awt.Color[r=" + color.getRed() + ",g=" + color.getGreen()
1269         + ",b=" + color.getBlue() + ",a=" + color.getAlpha() + "]");
1270 }
```

```
1264      }
1265      */
1266 }
```

6.3 OOFio Test

Im folgenden sieht man automatisierte Tests für den OOF Import und Export.

```
1 package io;
2
3 import log_management.DatabaseManager;
4 import log_management.dao.PersonalEntityManagerFactory;
5 import org.apache.commons.io.FileUtils;
6 import org.apache.commons.io.IOUtils;
7 import org.apache.log4j.BasicConfigurator;
8 import org.junit.*;
9
10 import javax.persistence.EntityManagerFactory;
11 import javax.persistence.Persistence;
12 import java.io.File;
13 import java.io.IOException;
14 import java.nio.charset.StandardCharsets;
15 import java.nio.file.Paths;
16
17 public class OOFioTest {
18
19     private static OOFio oofio;
20     private static File simpleGraph;
21     private static File exportedSimpleGraph;
22     private static String anyOOF;
23     private static String simpleGraphGXL="";
24     private static String simpleGraphJSON ="";
25     private static DatabaseManager databaseManager;
26
27     /**
28      * javadocTODO
29      * @throws IOException javadocTODO
30      */
31     @BeforeClass
32     public static void prepareOnce() throws IOException {
33         BasicConfigurator.configure();
34         EntityManagerFactory entityManagerFactory = Persistence.
35             createEntityManagerFactory("TestPersistenceUnit");
36         PersonalEntityManagerFactory.setEntityManagerFactory(entityManagerFactory);
37         databaseManager = DatabaseManager.getInstance();
38         oofio = new OOFio();
39         simpleGraph = Paths.get(".graphitTest", "simpleGraph.oof").toFile();
40         FileUtils.copyInputStreamToFile(OOFioTest.class.getResourceAsStream("/
41             simpleGraph.oof"), simpleGraph);
42         exportedSimpleGraph=Paths.get(".graphitTest", "ExportSimpleGraph.oof").toFile()
43             ;
44         simpleGraphGXL = IOUtils.toString(OOFioTest.class.getResourceAsStream("/
45             simpleGraphGXL.gxl"), StandardCharsets.UTF_8);
46         simpleGraphJSON = IOUtils.toString(OOFioTest.class.getResourceAsStream("/
47             simpleGraphJSON.json"), StandardCharsets.UTF_8);
48         anyOOF=IOUtils.toString(OOFioTest.class.getResourceAsStream("/simpleGraph.oof")
49             , StandardCharsets.UTF_8);
50     }
51 }
```

```
46     /**
47      * javadocTODO
48      */
49     @AfterClass
50     public static void endAll() throws IOException {
51         PersonalEntityManagerFactory.getInstance().close();
52         FileUtils.deleteDirectory(new File(".graphitTest"));
53     }
54
55     /**
56      * javadocTODO
57      */
58     @After
59     public void end(){
60         databaseManager.getGraphDao().delete(-1);
61     }
62
63     private void resetDB() throws IOException {
64         databaseManager.getGraphDao().delete(-1);
65         PersonalEntityManagerFactory.getInstance().close();
66         FileUtils.deleteDirectory(new File(".graphitTest"));
67         prepareOnce();
68     }
69
70
71     /**
72      * javadocTODO
73      */
74     @Test
75     public void testCreateOOF() throws IOException {
76         Assert.assertEquals(anyOOF, oofio.createOOF(oofio.gxlFromOOF(anyOOF),oofio.
77             jsonFromOOF(anyOOF)));
78     }
79
80     /**
81      * javadocTODO
82      */
83     @Test
84     public void testEmptyDatabase() {
85         Assert.assertTrue(databaseManager.databaseEmpty());
86     }
87
88     /**
89      * javadocTODO
90      */
91     @Test
92     public void testImportOOFGXL() {
93         oofio.importOOF(simpleGraph);
94         Assert.assertEquals(
95             simpleGraphGXL.trim().replaceAll("\n|\s","");
96             databaseManager.getGraphDao().gxlFromDatabase().trim().replaceAll("\n
97             |\s","");
98         }
99
100    /**
101     * javadocTODO
102     */
103    @Test
104    public void testImportOOFJSON() {
105        oofio.importOOF(simpleGraph);
106        Assert.assertEquals(
```

```
106             simpleGraphJSON.trim().replaceAll("\\\\n|\\\\r|\\\\d","");
107             databaseManager.getLogDao().getAllString().replaceAll("\\d","");
108         }
109
110     /**
111      * javadocTODO
112      */
113     @Test
114     public void testExportOOFGXL() {
115         oofio.importOOF(simpleGraph);
116         oofio.exportAsOOF(exportedSimpleGraph);
117         Assert.assertEquals(
118             simpleGraphGXL.trim().replaceAll("\\n|\\s","");
119             oofio.gxlFromOOF(FileHandler.fileToString(exportedSimpleGraph)).trim().
120             replaceAll("\\n|\\s"));
121     }
122
123     /**
124      * javadocTODO
125      */
126     @Test
127     public void testExportOOFJSON() throws IOException {
128         resetDB();
129         oofio.importOOF(simpleGraph);
130         oofio.exportAsOOF(exportedSimpleGraph);
131         Assert.assertEquals(
132             simpleGraphJSON.trim().replaceAll("\\\\n|\\\\r|\\\\d").replaceAll("\\d",
133             ""),
134             oofio.jsonFromOOF(FileHandler.fileToString(exportedSimpleGraph)).
135             replaceAll("\\d")));
136     }
137 }
```