ПРИЛОЖЕНИЕ А

# ОТЧЕТ ПО НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

**Моделирование структуры и динамики «онлайн»-дискуссии средствами многоагентных систем**

**по проекту РФФИ № 16-06-00184 А «Разработка и исследование моделей «online» дискуссии на материале обсуждения политических новостей»**

**2018 г.**

## Аннотация

В отчете представлено описание модели, имитирующей процесс создания дискуссии вокруг статьи. В модели участвуют агенты-авторы различных типов, которые, пользуясь определенным набором стохастических правил, генерируют комментарии к статье, создавая тем самым дерево дискуссии. Основной особенностью построенной распределенной (многоагентной) модели является максимальная приближенность к реальному процессу организации дискуссии вокруг статьи: поведение авторов является индивидуализированным и реализуется посредством многоуровневой иерархической марковской цепи. Помимо вышеуказанных моделей в отчете присутствует описание программного модуля, реализующего визуализацию построенных марковских цепей и расчет их характеристик.

Отчет состоит из следующих разделов: в первом разделе приводится описание структуры модели, описываются основные агенты и переменные, необходимые для их успешного функционирования; во втором разделе приводится более детальное и формальное описание блока стохастических правил, отвечающих за поведение агентов модели; в третьем разделе приводится описание блока вывода стохастических правил поведения агентов во внешние файлы;в четвертом разделе описывается созданная программная реализация и проводится ее валидация посредством сравнительного анализа характеристик графов дискуссии и марковских цепей поведения агентов. Также отчет содержит 2 приложения с исходным кодом построенной модели и программного модуля визуализации марковских цепей поведения агентов.

## 1. Описание структуры модели (агенты, переменные)

В модели присутствуют три вида агентов: агенты активные (интеллектуальные, обладающие собственным поведением и принимающие решения), агенты пассивные (являющиеся продуктом решений активных агентов и не имеющие собственного поведения) и агенты служебные (обладают заранее заданным поведением, предназначены для ведения различного рода статистик по ходу развития модели).

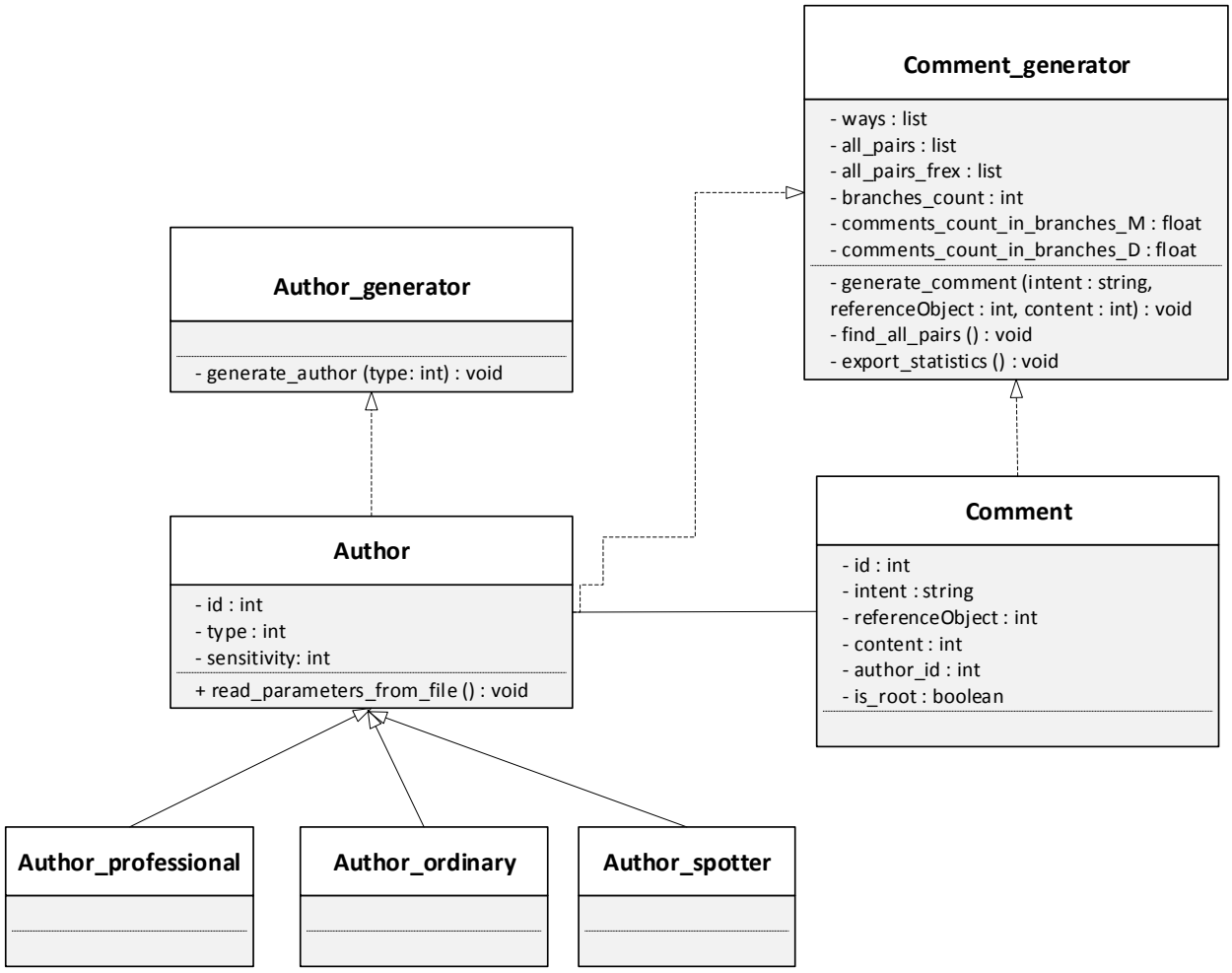



*Рис. 1: UML-представление агентов модели*

Поскольку программная среда Jason является агентно-ориентированной, дальнейшее описание каждого типа агента мы будем проводить в терминах BDI-описания (убеждения (Beliefs); желания (Desires); намерения (Intention)).

К активным агентам относятся агенты-авторы, которые, следуя определенным закономерностям, создают комментарии, являющиеся представителями второго вида агентов. Комментарии (рис. 1), не обладая собственным поведением, характеризуются только своими атрибутами: ***intent***-интентом (определяющим смысловую нагрузку

комментария), ***referenceObject***-референциальным объектом (определяющим, на что или кого направлен создаваемый комментарий), ***content***-контентом, ***author_id***-идентификатором принадлежности конкретному автору (который данный комментарий сгенерировал), а также вспомогательным ***is_root***-атрибутом-флагом, позволяющим отличить оригинальную статью (корневую вершину дискуссии) от комментариев к ней (вершины ветвей дискуссии).Данный перечень атрибутов является достаточным для определения поведения агентов-авторов на каждом шаге, а также для того, чтобы максимально полно охарактеризовать каждый комментарий с точки зрения дальнейшего интент-контентного анализа. Таким образом, с точки зрения BDI-модели, комментарии действительно являются пассивными агентами, поскольку содержат только блок убеждений (B), описываемый перечисленными выше атрибутами, не имея собственных намерений (I) и желаний (D).

Также важно заметить, что мы предполагаем, что в зависимости от контента комментария, они могут быть распределены на содержательные (имеют отношение к главной теме дискуссии) и несодержательные (не имеют отношение к главной теме дискуссии, чаще содержат личностные замечания и пр.).

В свою очередь, каждый автор (рис. 1) характеризуется своим уникальным ***id***-номером, а также ***type***-типом, который определяет значение ***sensitivity***-чувствительности к содержательным/несодержательным комментариям, возникающим в процессе организации дискуссии. С поведенческой точки зрения каждый автор обладает способностью генерировать комментарии, основываясь на нескольких глобальных переменных окружения, а также собственном типе и модели поведения.

При выделении типов агентов-авторов мы взяли классификацию людей относительно их следования устоявшимся собственным паттернам поведения. Согласно данной классификации, люди разделяются на «фанатиков» (строго следуют паттерну поведения, либо не следуют ему вовсе), «приспособленцев» (адаптируют собственные паттерны поведения в зависимости от требований внешней среды) и «оригиналы» (не обладают ярко выраженными паттернами поведения, а только вырабатывают их в процессе социальных коммуникаций).

Исходя из этого, а также принимая во внимание тот факт, что авторы могут генерироватькак содержательные, так и несодержательные комментарии в процессе организации дискуссии, в нашей модели есть следующие агенты-авторы:

- ***Профессионалы*** – данные авторы имеют устойчивую модель поведения и с заданной, высокой, долей вероятности реагируют на содержательные комментарии, оставляя без внимания несодержательные. Порог «чувствительности» к несодержательным комментариям задается пороговым правилом.

- ***Обычные*** – данные авторы не имеют ярко выраженных паттернов поведения, а приобретают их в процессе участия в дискуссии. При этом начальная «чувствительность» к содержательным и несодержательным комментариям также задается пороговым правилом, но в дальнейшем его численные характеристики могут меняться.

- ***Корректировщики*** – данные авторы больше выполняют роль мотиваторов для других авторов к участию в дискуссии. В некотором смысле можно утверждать, что их функция похожа на роль модератора дискуссии – они могут корректировать ее таким образом, чтобы сбалансировать распределение в дискуссии содержательных и несодержательных комментариев, или интенсивность комментариев от каждого отдельного автора.

Каждый тип авторов обладает, таким образом, явно выраженным набором BDI-характеристик. В качестве убеждений (B) у автора выступает его тип, а также значение пороговой «чувствительности» к содержательным/несодержательным комментариям. В качестве желаний (D) – максимальная вовлеченность в дискуссию. С формальной точки зрения это значит, что автор продолжает участвовать в дискуссии до тех пор, пока существует хотя бы один комментарий, на который он может ответить. В тот момент, когда количество таких комментариев становится нулевым и не меняется с течением времени, можно считать, что автор перестает участвовать в дискуссии и исключается из нее. Очевидно, когда данная ситуация возникает у всех участников дискуссии, сама дискуссия считается оконченной и завершается. В качестве же намерений (I) у каждого автора выступает набор поведенческих правил, аналогичных тем, которые были рассмотрены и реализованы в моделях NetLogo и которые более подробно представлены в Разделе 2. Однако, в отличие от ранее реализованных моделей NetLogo, в настоящей модели Jason поведение агентов-авторов является полностью индивидуализированным, поскольку теперь автор принимает решение о генерации комментария не на основании общей структуры дискуссии и своего прошлого поведения, а основываясь лишь на том наборе комментариев, которые ему интересны и согласуются с его моделью «чувствительности».

Помимо представленных двух типов агентов в модели также существуют служебные агенты, к которым относятся агенты-генераторы авторов и комментариев соответственно. Их основная роль – ведение общей статистики по модели в целом (количество авторов каждого типа, количество комментариев и статистика по ним для каждого агента-автора, общее количество комментариев в дискуссии, статистика по комментариям и их отдельным характеристикам и пр.). Для этого в них выделены следующие атрибуты: ***ways***-является переменной-списком, хранящей в себе все возможные пути от корневой вершины к листовым вершинам ветвей дискуссии, пара переменных ***all_pairs*** и ***all_pairs_frex***, отвечающих за хранение парных правил на каждом шаге работы модели (см. раздел 3), ***branches_count***-переменная, хранящая количество ветвей дискуссии, ***comments_count_in_branches_M***-переменная, хранящая среднее количество комментариев во всех ветвях дискуссии, ***comments_count_in_branches_D***-переменная, хранящая среднеквадратическое отклонение количества комментариев во всех ветвях дискуссии (см. раздел 3).

## 2. Описание стохастического блока правил поведения агентов

Поведение агентов-авторов в модели не является фиксированным – оно определяется набором стохастических правил, применение которых зависит от значения переменных окружения, а также состояния дерева дискуссии к текущему шагу работы модели.

На начальном этапе работы создаются все участвующие в дискуссии агенты-авторы. Случайным образом из них выбирается один автор, который генерирует оригинальный комментарий (статью), к которой все остальные авторы могут генерировать свои комментарии. После того, как автор сформулировал комментарий, он отправляет сообщение о нем всем остальным участникам дискуссии. Получив сообщение о сгенерированном комментарии, агент-автор анализирует его соответствие собственному типу и принимает решение о том, отвечать на данный комментарий или же нет. Происходит это следующим образом: если контент комментария удовлетворяет типу агента, то данный входной комментарий добавляется в список комментариев, на которые агент-автор может отметить в дальнейшем, в противном случае комментарий игнорируется и ответ на него не генерируется. В дальнейшем из числа комментариев, оставшихся «к ответу» (удовлетворяют его типу), агент-автор выбирает один и принимает решение о том, отвечать на него сейчас или же позже. При этом он руководствуется собственным пороговым правилом генерации комментария. Данная процедура повторяется для всех комментариев, содержащихся в списке «к ответу». В случае, если автор решает ответить на комментарий, он генерирует собственный комментарий, и также уведомляет о нем всех остальных участников дискуссии. Таким образом, каждый раз, при получении уведомления о сгенерированном комментарии схема повторяется вновь.

Как видим, данная процедура организации дискуссии максимально приближена к процессу организации реальной онлайн-дискуссии и позволяет сделать гипотезу о том, что предлагаемый набор агентов-авторов, а также модель их поведения достаточны для того, чтобы создать компьютерную модель онлайн-дискуссии, позволяющей создавать дискуссии, соответствующие реальным.

Важным является тот факт, что длякаждого автора используется собственная многоуровневая индивидуальная модель поведения, содержащая внутри себя многоуровневую иерархическую марковскую модель (рис. 2).

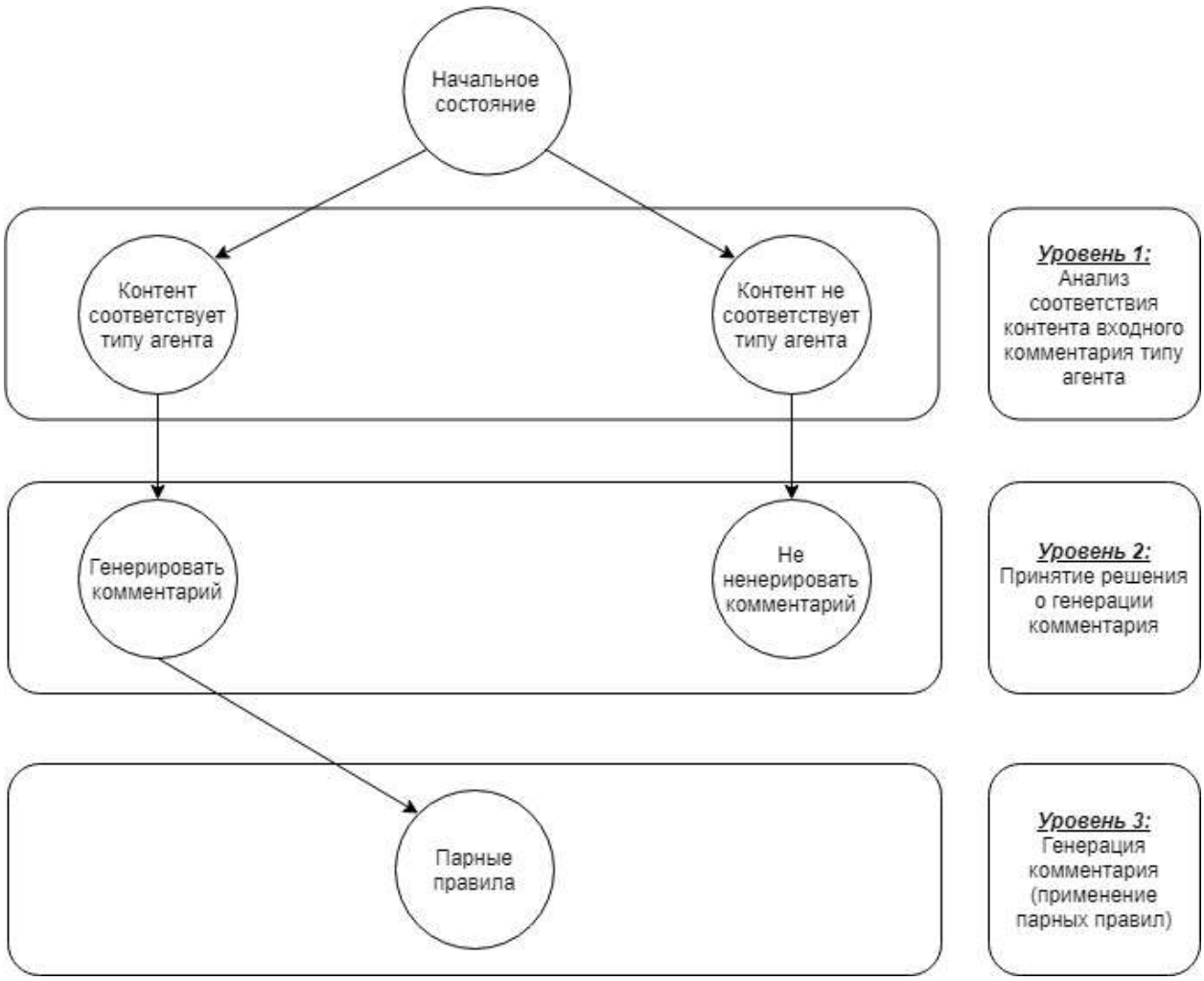



*Рис. 2: Концептуальная схема марковской модели поведения агента*

Формально данная модель может быть описана следующим набором стохастических правил:

- **<u>Пороговое правило анализа соответствия контентавходного комментария типу агента:</u>**

$$\begin{cases} \text{если } \left|content_{root} - content_{comment\ i}^{\ t}\right| < sensitivity_i^t, & \text{поместить комментарий в список "к ответу"} \\ \text{иначе,} & \text{оставить комментарий без ответа} \end{cases}$$

где $content_{root}$ – значение контента оригинальной статьи (комментарий с флагом is_root = true), $content_{comment\ i}^{\ t}$ – значение контента входного комментария для автора *i*автора в момент времени *t*, а пороговое значение чувствительности$sensitivity_i^t$ определяется файлом, содержащим сведения о модели поведения автора *i* в момент времени *t*.

- **<u>Пороговое правило генерации комментария:</u>**

$$\begin{cases} \text{если } x < {C_{gc}}_i^t, & \text{комментарий генерируется} \\ \text{иначе,} & \text{комментарий не генерируется} \end{cases}$$

где $x$ – значение равномерно распределенной случайной величины *X(0,1)*, а пороговое значение ${C_{gc}}_i^t$ определяется файлом, содержащим сведения о модели поведения автора $i$ в момент времени *t*.

- **Стохастическое правило определения характеристик генерируемого комментария:**

Можно сказать, что каждый автор содержит в данном случае в основе своего поведения иерархическую марковскую цепь, которая задаетсядля каждого автора индивидуально, на основании файла, описывающего модель его поведения в момент времени *t*.Каждое парное правило имеет следующий вид: "$i_1 r_1 \rightarrow i_2 r_2\ N$", где $i_1 r_1$ – интент и референциальный объект комментария-родителя, $i_2 r_2$ – интент и референциальный объект комментария-потомка и $N$ – число, характеризующее количество, которое данная пара комментариев родителя-потомка встретилась в дереве дискуссии. Группируя семейство таких правил по комментарию-родителю, мы получаем для каждого из них следующий набор парных правил: $i_p r_p \rightarrow \{i_j r_j N_j\}$, где $\{i_j r_j N_j\}$ – семейство всех комментариев-потомков, связанных с текущим комментарием-родителем $i_p r_p$. На основании чего можно сгенерировать следующий набор пороговых правил для каждого из комментариев-родителей:

$$\begin{cases} \text{если } 0 \le x < N_1, & \text{генерируется комментарий с характеристиками } i_1 r_1 \\ \text{если } N_1 \le x < N_2, & \text{генерируется комментарий с характеристиками } i_2 r_2 \\ & \dots \\ \text{если } N_{j-1} \le x < N_j, & \text{генерируется комментарий с характеристиками } i_j r_j \end{cases}$$

где $x$ – значение равномерно распределенной случайной величины $X(0, \sum_j N_j)$.

Исходя из вышеперечисленных правил, в самом общем виде поведение агента-автора может быть описано следующим образом. На каждом шаге проверяется соответствие входного комментария типу агента с помощью ***порогового правила анализа соответствия контента входного комментария типу агента***. Осуществляется такая проверка путем сравнения величины модуля разности контентов оригинальной статьи и входного комментария с величиной, описывающей чувствительность автора к отклонению контента комментария от контента статьи. В случае, если модуль меньше порога чувствительности (комментарий относится непосредственно или близок к содержанию статьи), комментарий включается в список комментариев «к ответу». После того, как проанализированы все входные комментарии, автор начинает генерировать ответные

комментарии к комментариям из списка «к ответу». В этом случае автор применяет ***пороговое правило генерации комментария***: если согласно данному правилу решено комментарий не оставлять, то комментарий не генерируется; в противном случае применяется ***стохастическое правило определения характеристик генерируемого комментария***. Данная логика применяется к каждому из агентов-авторов, тем самым используется иерархическая марковская цепь для определения индивидуального поведения авторов.

## 3. Описание блока вывода статистики (стохастических правил) в файл

Помимо основного функционала создания дискуссии, модель поддерживает разного рода статистические показатели и возможность вывода информации по модели в файл. Для этого используется функционал, реализуемый двумя имеющимися в модели служебными агентами-генераторами комментариев и авторов соответственно.

К основным статистическим показателям модели, значение которых может быть интересно пользователю, относятся количество ветвей дискуссии, среднее количество комментариев в них, а также среднеквадратическое отклонение количества комментариев в ветвях дискуссии (атрибуты***branches_count***, ***comments_count_in_branches_M***, ***comments_count_in_branches_D***соответственно). Данные показатели интересны с той точки зрения, что позволяют не только проследить интенсивность дискуссии и ее развитие с течением времени, но также сравнить между собой степень активности авторов по каждому из направлений беседы.

Кроме того, модель поддерживает вывод в файл стохастических парных правил, необходимых для более детального последующего анализа модели и сравнения результатов ее работы с показателями реальных деревьев дискуссии. Для этого используются функция ***find-all-pairs***, которая формирует переменные-списки ***all_pairs*** и ***all_pairs_frex***, содержащих все пары комментариев-родителей и комментариев-потомков и количество, которое данная пара комментариев родителя-потомка встретилась в дереве дискуссии соответственно, и функция ***export_statistics***, которая выводит парные правила в виде троек "$i_1r_1;\ i_2r_2;\ N$", где $i_1r_1$ – интент и референциальный объект комментария-родителя, $i_2r_2$ – интент и референциальный объект комментария-потомка и $N$ – число, характеризующее количество, которое данная пара комментариев родителя-потомка встретилась в дереве дискуссии.На выходе получается *.csv файл Excel, который может быть преобразован в табличную форму, а также использоваться в качестве входного файла при проведении интент-контентного анализа.

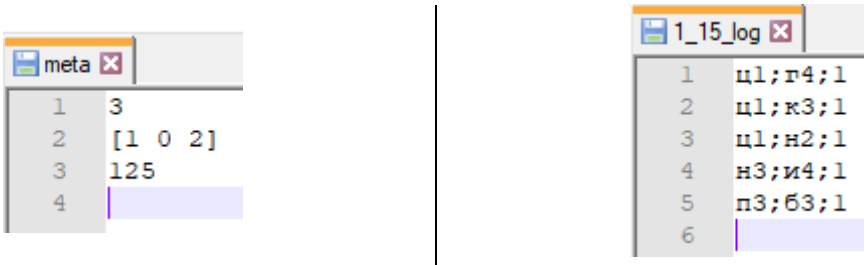
## 4. Программная реализация

Для программной реализации были использованы следующие технологии и программные продукты:

- Для реализации модели дискуссии (централизованной и распределенной) – среда многоагентного моделирования Jason (версия 2.3+):
    - Распределенная модель– 647 строк кода;
- Для визуализации и анализа графов марковских моделей поведения агентов модели – программный модуль, реализованный на языке программирования Java (версия 8+) с использованием библиотеки с открытым исходным кодом для визуализации графов JUNG(версия 2.0.1+) – 58 строк кода.

Таким образом, суммарно реализовано 705 строк кода.

На вход модели подается набор файлов, описывающих начальные параметры, необходимые для инициализации работы модели: количество авторов, распределение авторов по типам, продолжительность работы модели (в тактах, является опциональным – если не задано, среда работает до тех пор, пока есть возможность генерации комментариев). На выходе модель может генерировать файлы для каждого их авторов, описывающие пары комментариев, сгенерированные в ходе дискуссии (более подробное описание структуры таких файлов – см. Раздел 3). Примеры таких файлов приведены на рис. 3.




| Входной файл с мета-данными | Выходной файл с парными правилами |
|---|---|

*Рис. 3: Примеры входных и выходных файлов модели*

Такая структура файлов делает возможным последующее их сравнение с файлами моделей среды NetLogo, реализованными ранее, а также с показателями реальных дискуссии, используемыми в процессе работы над моделью.

Существенным отличием модели Jason от построенных ранее моделей NetLogo является отсутствие у нее графического интерфейса для управления моделью. Данная особенность вызвана тем, что граф дискуссии построенной модели Jason является

результатом взаимодействия и коммуникации агентов-авторов различных типов, поведение которых индивидуализировано и не нуждается во внешнем управлении. Кроме того, поведение авторов в данном случае максимально приближено к реальному, является более общим: при решении о генерации комментария агенты-авторы опираются на то, какие комментарии получают на вход и как они согласуются с их типом и моделью чувствительности, а не на то, какие комментарии они генерировали ранее и в какую часть дискуссии попадал новый комментарий. Тем самым, получается полноценная многоагентная структура агентов-авторов и их коммуникаций, позволяющая сопоставить результаты работы модели с показателями реальных дискуссий и сделать вывод о корректности структуры агентов-авторов и достаточности построенной модели для имитации дискуссий, приближенных к действительности.

Программный модуль Javaпринимает на вход группу файлов с парными правилами (рис. 3), на выходе же строит графическое отображение заданных парных правил. В последующем среда поддерживает возможность объединения различных графических изображений в одно с последующим экспортом в файлы различных расширений (например, pdf, рис. 4).

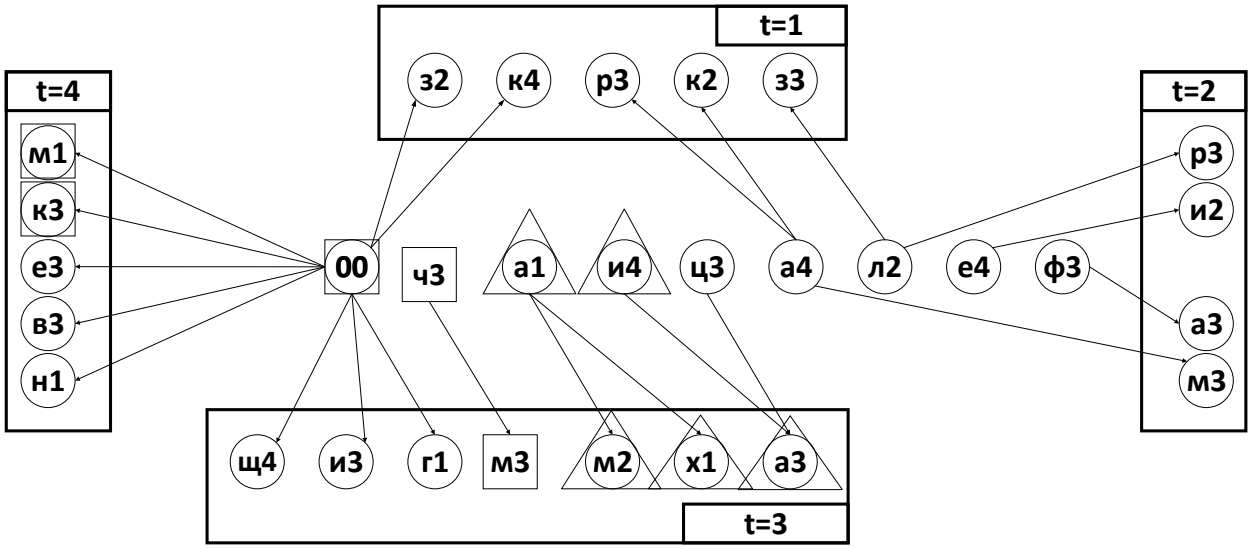



*Рис. 4: Фрагментвизуализации выходного графа марковской модели агентов-авторов*

Как видим, в данном случае (рис. 4) при визуализации сведения, относящиеся к разным файлам, отображаются в разных секторах схемы, тем самым подчеркивая изменения в марковской модели поведения авторов с течением времени.Различными фигурами (окружность, квадрат, треугольник) выделяются интенции комментариев, сгенерированных авторами различных типов: окружностями – для обычных агентов, треугольниками – для профессионалов, квадратами – для корректировщиков

## 5. Валидация модели

Для валидации модели Jason мы использовали данные, полученные ранее при работе с моделью NetLogo, а также с графами реальных дискуссий. Проводились эксперименты как на малых, так и на больших размерностях.

Изначально, для модели NetLogo были определены следующие параметры: 2 автора, которые на каждом шаге генерируют комментарий с вероятность 0,9; вероятность отнесения созданного комментария к корневой вершине 0,25 и 0,75 к листовой вершине соответственно. Величина периода создания лога для распределенной модели – 5 тактов. Длительность модели – 25 тактов. На выходе была получена следующая марковская цепь поведения агента-автора (рис. 5). В центре отражены общие вершины-родители, которые возникали на протяжении всей работы модели, в разных же кластерах сгруппированы вершины-комментарии, являющиеся потомками, возникавшими на разных этапах работы модели. Каждый кластер – отдельный этап работы модели, длительность которого совпадает с величиной периода логирования модели (в данном случае – 5 тактов).

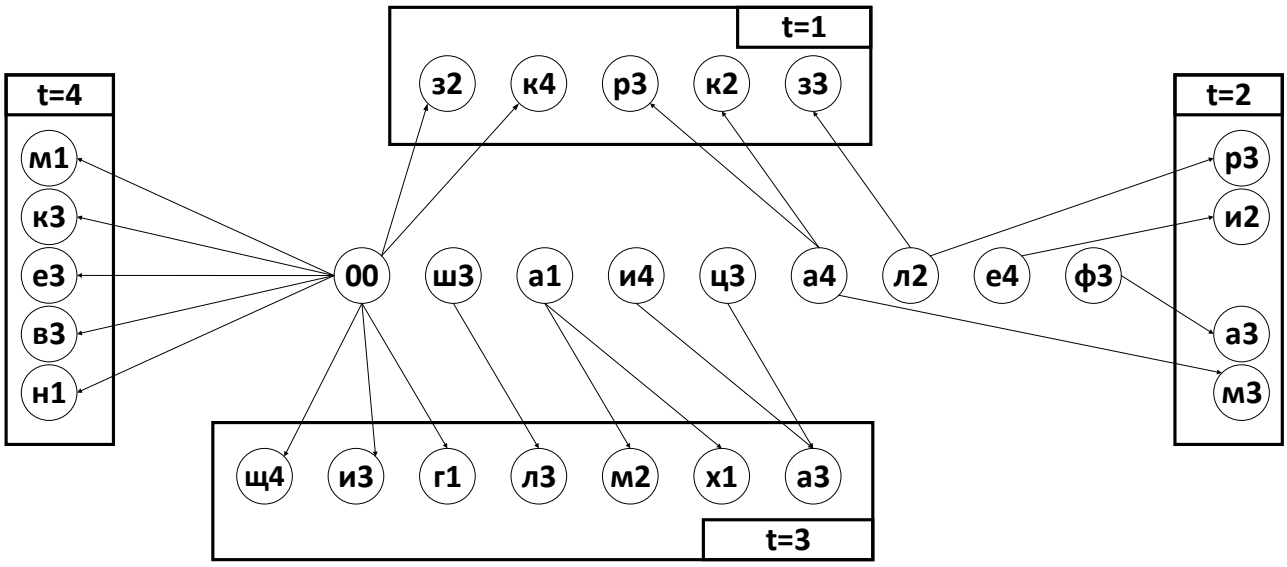



Рис. 5. Визуализация выходного графа марковской цепи агента-автора (окружности – комментарии (их интент и референциальный объект), в прямоугольники объединены комментарии, сгенерированные автором на каждом периоде работы модели)

После этого проведем эксперимент на больших размерностях, когда количество тактов работы модели значительно (~200). В этом случае можно получить следующие характеристики авторов (рассмотрим на примере первого – рис. 6).

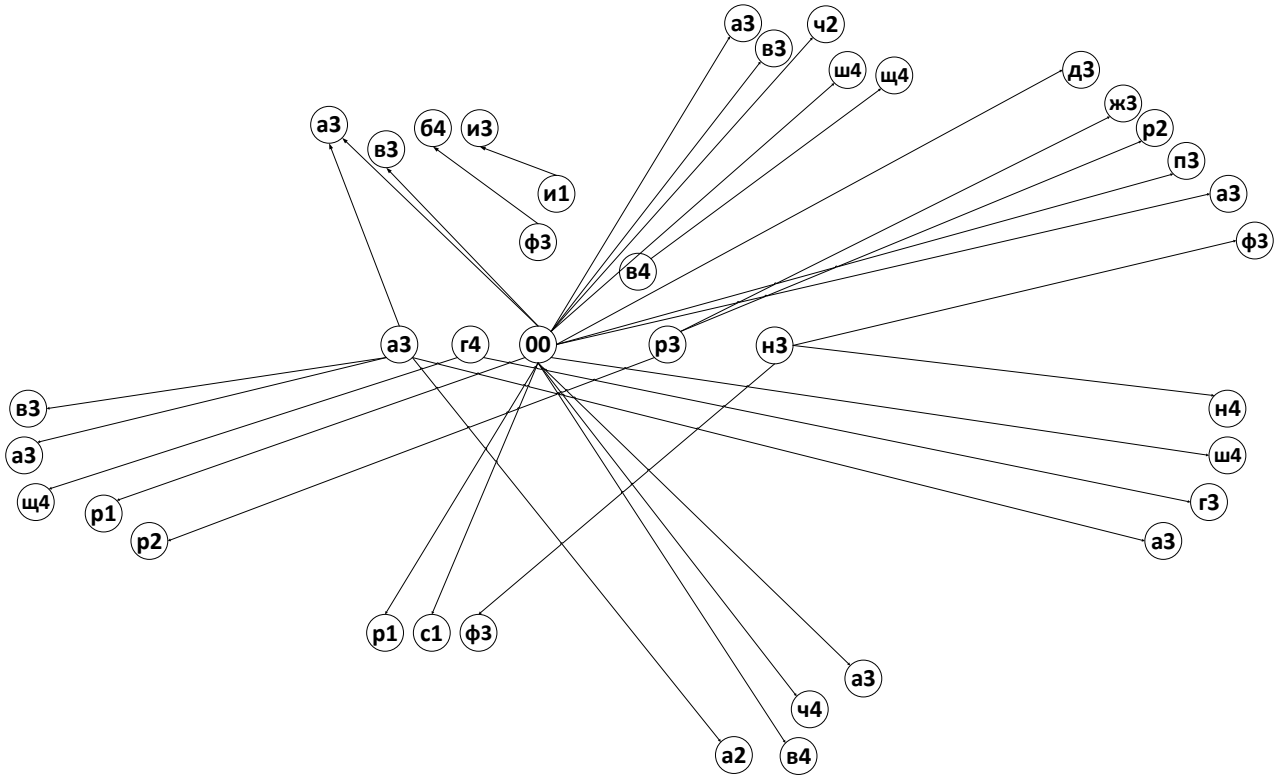



Рис. 6. Фрагмент марковской цепи первого автора распределенной модели (230 тактов)

После того, как у нас есть исходные марковские модели поведения авторов, мы можем сопоставить их с результатами работы модели Jason, чтобы убедиться в ее корректности и эффективности. Такое сравнение позволит убедиться в том, что модель Jason в поведенческом аспекте соответствует распределенной модели NetLogo и может быть использована для описания поведения агентов-авторов различных типов. Для этого сначала повторим эксперимент с малыми размерностями: создадим по 1 агенту-автору каждого типа и запустим модель на 25 тактов, с шагом сохранения результатов работы модели и характеристик дерева дискуссии в 5 тактов. Таким образом, получается следующая структура агентов-авторов модели малой размерности (рис. 4).

В данном случае все 3 марковские цепи для авторов наложены друг на друга. Анализируя их по отдельности, можно утверждать только то, что поведение авторов с течением времени зависит от их модели поведения и является устойчивым в интенциональном смысле. Однако такого сравнения недостаточно для того, чтобы утверждать эквивалентность результатов модели Jason с результатами модели NetLogo и реальными дискуссиями. Для этого необходимо объединить марковские цепи различных авторов в единую и сопоставить полученную кумулятивную марковскую цепь с цепями авторов из модели NetLogo и реальных дискуссий.

В данном случае (рис. 4, рис. 5), на дискуссии малых размерностей можно утверждать очевидное сходство между полученными марковскими цепями с рассмотренными ранее в модели NetLogo. Так, например, обычный автор генерировал на первых этапах работы модели интенции, идентичные полученным в модели NetLogo: з3, к4, р3, к2 и з3 на первом этапе и т.д. Данное сходство может быть объяснено тем, что в начальный момент времени в дискуссии присутствует малое количество комментариев и автор в основном придерживается своей стандартной модели поведения. Однако, в отличие от модели NetLogo, в нашей присутствуют также агенты-авторы двух отличных от обычного типов. И если в первые два этапа модели они в ней не участвуют, поскольку дискуссия развивается активно и без их вмешательства, то начиная с третьего этапа, возникает потребность скорректировать поведение обычных авторов. В данном случае (рис. 4) авторы-профессионалы сохранили устойчивость дискуссии, поскольку из нее начали пропадать комментарии с интенциями м и а, хотя ранее они в ней присутствовали. В то же время, авторы-корректировщики добавили новый комментарий с интенцией м3, который до этого фигурировал в дискуссии в наименьшем количестве и не получал дальнейшего развития. Аналогичные интенции данных типов авторов наблюдаются на заключительном, 4-ом этапе работы модели. Как результат, полученное дерево дискуссии по ряду характеристик приближается к тем, что были получены при работе с моделью NetLogo, в частности, центральная клика вершин в обоих случаях (рис. 4 и рис. 5) идентична.

Таким образом, можно утверждать, что полученная многоагентная модель дискуссииJason на малых размерностях позволяет получить дерево дискуссии, схожее по своей структуре и интенциональным связям с деревьями дискуссии моделиNetLogo и, поскольку ранее было установлено их соответствие реальным деревьям дискуссий, с реальными дискуссиями.

Аналогичные эквивалентные результаты можно получить, анализируя кумулятивную марковскую цепь авторов модели Jason на больших размерностях (230 тактов работы модели). В данном случае мы определили модель следующим образом: в модели присутствовали 2 обычных автора, 1 автор-эксперт и 1 автор-корректировщик. Время работы модели – 230 тактов, периодичность записи результатов работы модели – 5 тактов.

Таким образом, структура модели максимально приближена к модели NetLogo, отличие состоит только в наличии корректирующих элементов в виде авторов-экспертов и корректировщиков.

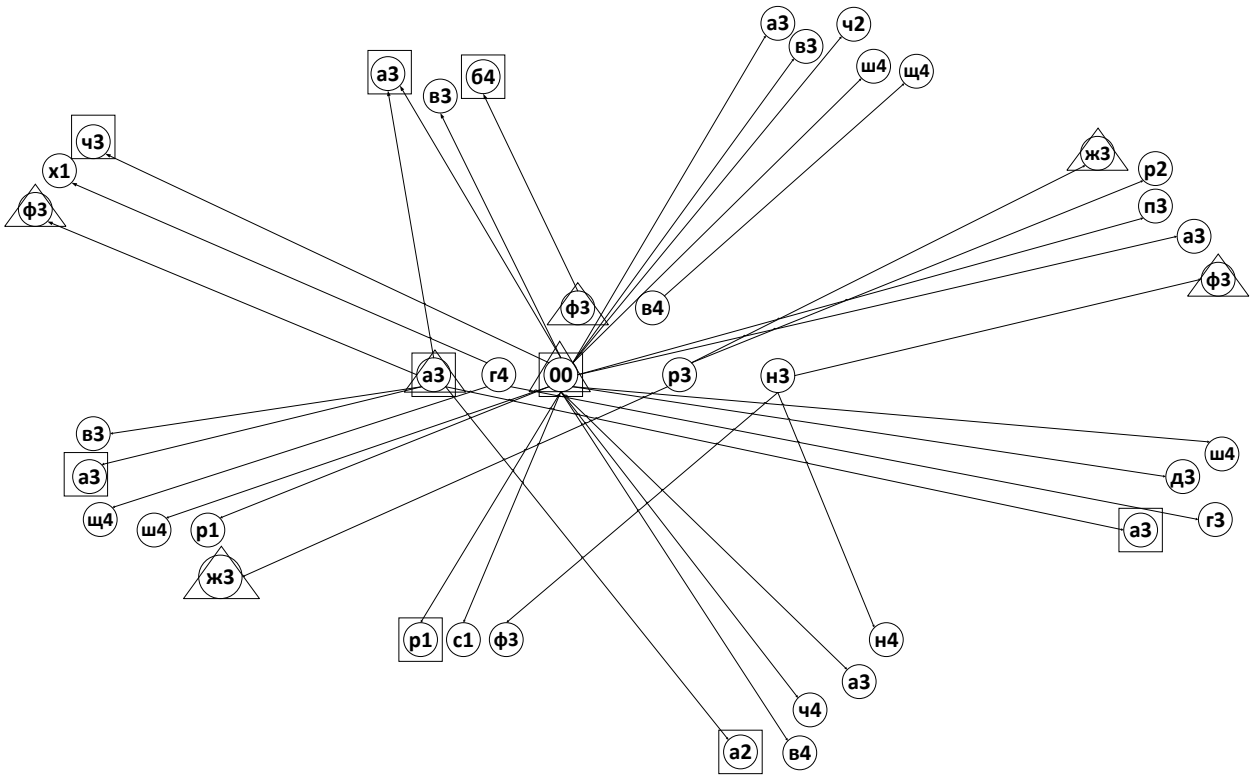



Рис. 7. Фрагмент марковской цепи авторов для модели Jason

Различными фигурами в данном случае (рис. 7) выделены комментарии и их интенции, относящиеся к различным типам авторов: кругами – комментарии обычных авторов, треугольниками – авторов-профессионалов, квадратами – авторов-корректировщиков.

Аналогично марковской цепи, визуализирующей результаты работы модели NetLogo (рис. 6), в центральной части графа собраны все интенции комментариев-родителей, которые генерировались в процессе дискуссии. Проводя анализ полученного кумулятивного графа марковской цепи всех типов авторов можно получить следующие результаты. Так, видно, что между авторами различных типов есть как сходства, так и различия. Сходством является тот факт, что на различных этапах дискуссии все авторы оставляли комментарии к комментариям с интенциями а3, а также к корневой вершине (исходная статья). В то же время есть и различия в поведении авторов, например, обычные авторы оставляли комментарии к комментариям с интенциями г4, р3, н3, в4, ф3, а авторы-корректировщики – к комментариям с интенциями ф3.

Кроме того, как и в модели NetLogo, анализируя различные сегменты марковской цепи (по часовой стрелке), можно выделить особенности поведения авторов различных типов в различные этапы работы модели.

Так, например, в первые два этапы работы (рис. 7) модели в основном в обсуждении участвуют обычные авторы и авторы-профессионалы, а агенты-корректировщики присоединяются к дискуссии только в более поздние этапы работы (начиная с третьего), причем если на третьем этапе работы модели их участие в дискуссии минимально (генерируются комментарии лишь с одной интенцией), то на заключительных этапах работы модели их активность повышается (8 этап работы – две из трех интенций комментариев сгенерированы авторами-корректировщиками). Данное наблюдение может трактоваться таким образом, что дискуссия носит динамический характер: если в своем начале она является активной и в ней участвует большое количество авторов, генерирующих большой объем комментариев, то с течением времени количество комментариев снижается, поскольку авторы удовлетворяют свои потребности в ходе дискуссии, получая ответы от других участников на свои комментарии, и тем самым теряя необходимость в генерации новых комментариев. Таким образом, можно утверждать, что существует обратная линейная зависимость между количеством комментариев (интенсивность дискуссии) и ее длительностью: чем дольше длится дискуссии, тем меньшее количество комментариев в ней будет сгенерировано обычными авторами и авторами-профессионалами.

Помимо данных наблюдений мы можем увидеть соответствие полученной дискуссии основным моделям, заложенным в поведение авторов различных типов. Так, например, обычные авторы генерировали на первом этапе работы модели комментарии с интенциями а3, в3, ч2, ш4, щ4; на втором этапе – с интенциями ж3, р2, п3, а3, ф3 и т.д.; авторы-профессионалы на первом и третьем этапе не генерировали комментариев, на втором и последующих этапах – комментарии с интенциями ж3, ф3. Исходя из этого можно сделать вывод, что поведение обычных агента-автора в настоящий момент времени не зависит от того, какие именно комментарии он оставлял на предыдущих этапах. Это полностью соответствует определению неоднородной марковской цепи, и подтверждает эффективность использования данного инструмента для описания поведения автора в ходе дискуссии. С другой стороны, поведение автора-профессионала является устойчивым и не меняется с течением дискуссии, тем самым соответствуя определению однородной марковской цепи, используемой при описании данного типа авторов.

Если же анализировать поведение авторов-корректировщиков, то они не имеют ярко выраженной модели поведения и строят ее, основываясь на конкретном состоянии дерева дискуссии, добавляя комментарии либо к комментариям с наиболее редкими интенциями (например, на пятом этапе модели были созданы комментарии с интенцией

р1, которая до этого не встречалась в дискуссии, а на седьмом этапе – с интенцией ч3, которая в течение двух предшествующих этапов модели не встречалась в ней, хотя до этого была сгенерирована), либо вовлекая в беседу авторов, количество комментариев в каждом такте работы модели которых уменьшается. Тем самым можно сказать, что их поведение зависит только от текущего состояния дерева дискуссии и не описывается строго в виде марковской цепи. Именно в этом смысле они не являются обычными авторами, поскольку не имеют ярко выраженной модели поведения, а служат лишь корректирующим элементов дискуссии.

Наконец, анализируя устойчивость поведения авторов, можно проанализировать наборы пар-интенций, образующихся на различных этапах работы модели. Так, например, видно, что в случае обычных авторов на различных этапах работы модели интенция а3 порождала интенции а3 (на 3-ем, 6-ом и 8-ом этапах работы модели) а также интенции а2 и в3. Такая относительная устойчивость в поведении авторов, как и в случае с моделью NetLogo, говорит о том, что, несмотря на изменение набора интенций, порождаемых на различных этапах работы, сами пары интенций являются устойчивыми и определяются моделью поведения каждого агента-автора. В этом смысле поведение автора является устойчивым в интенциональном аспекте. Причем, наиболее устойчивыми в этом смысле являются авторы-профессионалы, модель парных правил которых практически не меняется с течением времени, а наименее – авторы-корректировщики, модель парных правил которых определяется состоянием дерева дискуссии в каждый конкретный момент времени.

Сравнивая полученную кумулятивную марковскую цепь авторов (рис. 7) с марковской цепью одного автора из модели NetLogo (рис. 6) можно заметить следующие сходства: так, в центральную клику входят комментарии с интенциями а3, г4, ф3, в4, р3, н3 и оригинальная статья.

Кроме того, в обоих случаях сохраняется тенденция, что в начальных этапах модели комментарии генерируются в большинстве к оригинальной статье, в то время как с течением времени возрастает количество комментариев к комментариям других авторов.

Похожие сходства можно заметить и при проведении интенционального сравнения различных этапов работы модели: так, на 6-ом этапе работы модели в обоих случаях были сгенерированы комментарии с интенциями в3, а3, щ4, ш4, р1. Отличие заключается только в интенции ж3 в случае модели Jason против интенции р2 в модели NetLogo. Однако данное различие возникло вследствие того, что интенция ж3 была выбрана

автором-корректировщиком как одна из наименее активных интенций в процессе дискуссии, в то время как интенция р2 уже присутствовала в ней до этого. Как результат, вследствие логики работы автора-корректировщика, была сгенерирована интенция с меньшей частотой встреч.

С другой стороны, можно утверждать, что поведение авторов несколько отличается между двумя моделями: так, обычные агенты в модели Jason не генерировали комментарии с интенциями ч3 и ф1 на заключительных этапах работы модели, это делали агенты-корректировщики. Но отсутствие комментариев с данными характеристиками связано с тем фактом, что поведение обычного автора в модели является не статическим, как в случае моделиNetLogo, но динамическим: то, какой комментарий автор генерирует в настоящий момент времени, зависит от того, какие комментарии он генерировал ранее и какие комментарии в настоящее время имеются в дискуссии и остаются без ответа других авторов. Таким образом, поведение автора является более социальным, в отличие от моделиNetLogo, и позволяет достичь большего разнообразия как в поведении отдельных авторов, так и в структуре дискуссии в целом.

В то же время по итогам сравнения моделей между собой можно утверждать, что и на дискуссиях больших размерностей модели показывают схожее поведение на уровне тенденций, различаясь только в отдельных интенциональных наборах различных этапов работы модели, вызванных тем, что поведение авторов индивидуализировано и не может быть абсолютно идентичным даже при одинаковых изначальных параметрах моделей.

Таким образом, созданная модель Jason подтверждает ранее выдвинутую гипотезу, что внедрение трех различных классов агентов: обычных, профессионалов и корректирующих – является достаточным для получения структуры дискуссии, идентичной реальной.

Если говорить о дальнейшем развитии построенной модели, то видится целесообразным сделать их еще более адаптивными. Так, мы увидели, что модель Jason дает более разнообразные в сравнении с моделью NetLogo, результаты в том случае, если оставить в модели только обычных авторов и авторов-профессионалов, исключив из нее корректировщиков, позволяющих корректировать поведение отдельных авторов. Однако, присутствие последних позволяет получить более индивидуализированные модели поведения, которые не просто следуют заранее заданным моделям поведения, но изменяются с течением времени в зависимости как от окружения авторов (другие комментарии, их интенциональные характеристики и пр.), так и от их прошлого

поведения (ранее созданные комментарии). В этом смысле видится целесообразным объединение обеих моделей в одну на основании моделей конечных автоматов с памятью, что позволит в полной мере определить поведение, основываясь на его предыдущих действий и состоянии графа дискуссии к настоящему моменту времени.

## Приложение 1. Исходный код модели Jason

```
/* This model was created for RFFI-project and describe the discussion creation*/

Authors.mas2j

/* Jason Project */

MAS authors {

	infrastructure: Centralised

	environment: TestEnv

	agents:

		authors_generator;

}

Authors_generator.asl

// Agent Authors_generator in project Authors.mas2j

/* Beliefs */

agentCounter(0).

allAuthors([]).

!start.

/* Intentions */

+!start :

	agentCounter(AC) & AC < 2 <-

		-+agentCounter(AC+1);

		?agentCounter(C);

		.concat("author_",C,Name);

		?allAuthors(AA);

		.concat(AA,[Name],AA1);

		-+allAuthors(AA1);

		.wait(100);

		.create_agent(Name, "author.asl");

		.wait(100);

		.send(Name, achieve, initiateAuthor(math.round(math.random(3))));

		!!start.
```

```
+!start :
	agentCounter(AC) & AC == 2 <-
		.wait(10);
		-+agentCounter(AC+1);
		.send(author_1, achieve, startDiscussion);
		!!start.
+!start<-true.
+!informAboutNewComment(CN, AN) :
	true <-
		?allAuthors(AA);
		for (.member(Y,AA))
		{
			.send(Y, achieve, continueDiscussion(CN, AN));
		}.
```

Author.asl

```
/* Intentions */
+!start :
	true <-
		.my_name(MyName);
		.print("Created: ", MyName).
+!initiateAuthor(AT)[source(authors_generator)] :
	true <-
		-+type(AT);
		.my_name(MyName);
		?type(CAT);
		.print(MyName, ": type = ", CAT).
+!startDiscussion :
	true <-
		.print("Discussion initiated");
		?commentCount(CC);
		-+commentCount(CC+1);
```

```
                .my_name(MyName);
                .concat(MyName,"_",CC,Name);
                .create_agent(Name, "comment.asl");
                .wait(100);
                .send(Name, achieve, createComment(0, 0, 0, MyName, null, null))
                .wait(100);
                .send(authors_generator, achieve, informAboutNewComment(Name, MyName)).
+!continueDiscussion(CN, AN) :
        .my_name(MyName) &MyName \== AN &commentCount(CC) & CC < 5 <-
                ?commentCount(CC);
                -+commentCount(CC+1);
                .my_name(MyName);
                .concat(MyName,"_",CC,Name);
                .create_agent(Name, "comment.asl");
                .wait(100);
                .send(Name, achieve, createComment(1, 1, 1, MyName, AN, CN))
                .wait(100);
                .send(authors_generator, achieve, informAboutNewComment(Name, MyName)).
+!continueDiscussion(CN, AN) :
        .my_name(MyName) &MyName == AN <- true.
```

Comment.asl

```
// Agent Comment in project Authors.mas2j
/* Beliefs */
!start.
/* Intentions */
+!createComment (I, C, RO, A, TA, OC) :
        true <-
                +intent(I);
                +content(C);
                +refObject(RO);
                +commentAuthor(A);
```

```
              +targetAuthor(TA);
              +originalComment(OC);
              .print("seted params: ",I,"_",C,"_",RO,"_",A,"_",TA,"_",OC).
+!start :
       true <-
              .my_name(MyName);
              .print("Comment ", MyName, " generated").
TestEnv.java
// Environment code for project testenv.mas2j
import jason.asSyntax.*;
import jason.environment.*;
import java.util.logging.*;
public class TestEnv extends Environment {
       public static String readFile(String file) throws IOException {
              BufferedReader reader = new BufferedReader(new FileReader(file));
              String line = null;
              String ls = System.getProperty("line.separator");
              try {
                     while ((line = reader.readLine()) != null) {
                            this.rules.add(line);
                     }
                     return stringBuilder.toString();
              } finally {
                     reader.close();
              }
       }
              public static void writeStatistics() {
              try (FileWriter writer = new FileWriter(this.id + "_" + this.type +
"log.txt", false)) {
                     for (int i = 0; i<this.rules.length(); i++)
                     {
```

```
                        writer.write(this.rules.get(i));
                        writer.append('\n');
                    }
                    writer.flush();
                } catch (IOException ex) {
                    System.out.println(ex.getMessage());
                }
            }
private Logger logger = Logger.getLogger("testenv.mas2j."+TestEnv.class.getName());
    /** Called before the MAS execution with the args informed in .mas2j */
    @Override
    public void init(String[] args) {
super.init(args);
addPercept(ASSyntax.parseLiteral("percept(demo)"));
    }
    @Override
    public booleanexecuteAction(String agName, Structure action) {
logger.info("executing: "+action+", but not implemented!");
        if (true) { // you may improve this condition
             switch (action.name) {
                        case "init" :
                               informAgsEnvironmentChanged();
                               break;
                        case "initiateAuthor" :
                               readFile(agName);
                               informAgsEnvironmentChanged();
                               break;
                        case "createComment" :
                               this.createComment<- true;
                               informAgsEnvironmentChanged();
                               break;
```

```
                        case "writeStatistics" :
                                writeStatistics();
                                break;
                    }
         }
        return true; // the action was executed with success
    }
    /** Called before the end of MAS execution */
    @Override
    public void stop() {
super.stop();
}
}
```

CommentEnv.java

```
// Environment code for project CommentEnv.mas2j
import jason.asSyntax.*;
import jason.environment.*;
import java.util.logging.*;
public class CommentEnv extends Environment {
       public static String readFile(String file) throws IOException {
              BufferedReader reader = new BufferedReader(new FileReader(file));
              String line = null;
              String ls = System.getProperty("line.separator");
              try {
                     while ((line = reader.readLine()) != null) {
                            this.params.add(line);
                     }
                     return stringBuilder.toString();
              } finally {
                     reader.close();
              }
```

```
	}
		public static void updateStatistics() {
		try (FileWriter writer = new FileWriter(this.id + "_" + this.type +
"log.txt", false)) {
			for (int i = 0; i<this.ways.length(); i++)
			{
				writer.write(this.ways.get(i));
				writer.append('\n');
			}
			writer.flush();
		} catch (IOException ex) {
			System.out.println(ex.getMessage());
		}
	}
	private Logger logger =
Logger.getLogger("testenv.mas2j."+TestEnv.class.getName());
    /** Called before the MAS execution with the args informed in .mas2j */
    @Override
    public void init(String[] args) {
super.init(args);
addPercept(ASSyntax.parseLiteral("percept(demo)"));
    }
    @Override
    public booleanexecuteAction(String agName, Structure action) {
logger.info("executing: "+action+", but not implemented!");
	if (true) { // you may improve this condition
			switch (action.name) {
				case "init" :
					informAgsEnvironmentChanged();
					break;
				case "initiateComment" :
					readFile(agName);
```

```
                                informAgsEnvironmentChanged();
                                break;
                        case "createComment" :
                                this.createComment<- true;
                                informAgsEnvironmentChanged();
                                break;
                        case "updateStatistics" :
                                updateStatistics();
                                break;
                }
        }
        return true; // the action was executed with success
    }
    /** Called before the end of MAS execution */
    @Override
    public void stop() {
super.stop();
}
}
```

## Приложение 2. Исходный код программного модуля визуализации марковской модели поведения автора

```
//This model was created for RFFI-project and describe the discussion creation process
import java.awt.Dimension;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.util.ArrayList;
import javax.swing.JFrame;
import edu.uci.ics.jung.algorithms.layout.CircleLayout;
import edu.uci.ics.jung.graph.DirectedSparseGraph;
import edu.uci.ics.jung.visualization.VisualizationViewer;
import edu.uci.ics.jung.visualization.control.DefaultModalGraphMouse;
import edu.uci.ics.jung.visualization.decorators.ToStringLabeller;
import edu.uci.ics.jung.visualization.renderers.Renderer;
import edu.uci.ics.jung.visualization.control.ModalGraphMouse;
public class MCVisualization {
public static void main(String[] args) throws Exception {
DirectedSparseGraph<String, String> g = new DirectedSparseGraph<String, String>();
for (int auth_i = 0; auth_i<Integer.valueOf(args[0]); auth_i++)
for    (int    i    =    Integer.valueOf(args[1]);    i<=    Integer.valueOf(args[2]);
i+=Integer.valueOf(args[1])) {
File file = new File("C:/Users/Boris/Documents/Работа/2016-2017/РФФИ/tmp1/" + auth_i +
"_" + i + "_log");
FileReaderfr = new FileReader(file);
BufferedReader reader = new BufferedReader(fr);
ArrayList<String> vertexes1 = new ArrayList<String>();
ArrayList<String> vertexes2 = new ArrayList<String>();
ArrayList<String> edges = new ArrayList<String>();
String line = reader.readLine();
while (line != null) {
System.out.println(line);
String[] parts = line.split(";");
vertexes1.add(parts[0]);
vertexes2.add(parts[1] + "_" + (i/3));
edges.add(parts[2]);
line = reader.readLine();
}
for (String str : vertexes1) {
g.addVertex(str);
}
for (String str : vertexes2) {
g.addVertex(str);
}
for (int j = 0; j <edges.size(); j++) {
g.addEdge(edges.get(j)  +  "("  +  vertexes1.get(j)  +  ","  +  vertexes2.get(j)  +  ")",
vertexes1.get(j), vertexes2.get(j));
}
}
VisualizationViewer<String, String>vv =
new  VisualizationViewer<String,  String>(new  CircleLayout<String,  String>(g),  new
Dimension(400, 400));
vv.getRenderer().getVertexLabelRenderer().setPosition(Renderer.VertexLabel.Position.CN
TR);
vv.getRenderContext().setVertexLabelTransformer(new ToStringLabeller());
vv.getRenderContext().setEdgeLabelTransformer(new ToStringLabeller());
final       DefaultModalGraphMouse<String,       Number>graphMouse       =       new
DefaultModalGraphMouse<String, Number>();
graphMouse.setMode(ModalGraphMouse.Mode.PICKING);
vv.setGraphMouse(graphMouse);
JFrame frame = new JFrame();
frame.getContentPane().add(vv);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.pack();
frame.setVisible(true);
}
}
```