ОТ ИМИТАЦИОННОЙ МОДЕЛИ К ЦИФРОВОМУ ДВОЙНИКУ: АНАЛИЗ ОПЫТА ВЫПОЛНЕНИЯ КОММЕРЧЕСКИХ ПРОЕКТОВ

FROM SIMULATION MODELS TO DIGITAL TWINS: ANALYSIS OF INDUSTRIAL PROJECTS EXPERIENCE

Малыханов Андрей Анатольевич, ООО «Амальгама», Ульяновск;
Черненко Виталий Евгеньевич, ООО «Амальгама», Ульяновск

Malykhanov Andrey, "Amalgama" LLC, Ulyanovsk;
Chernenko Vitaliy, "Amalgama" LLC, Ulyanovsk

Ключевые слова: имитационное моделирование, цифровой двойник, платформа для моделирования, системы поддержки принятия решений.

Keywords: simulation, digital twin, simulation modeling platform, decision support systems.

Введение

Современные программные и аппаратные средства позволяют создавать детальные имитационные модели производственных и логистических систем. Такие модели часто требуют большого объема усилий по тестированию, верификации и валидации [1]. Успешное создание детальной имитационной модели позволяет не только ответить на поставленные в проекте задачи, но и начать внедрять модель в контуры среднесрочного и оперативного планирования. Именно модели, построенные на низком уровне абстракции, часто становятся модулями так называемых «цифровых двойников». Цифровой двойник – это цифровая копия физического объекта или процесса, помогающая повышать эффективность функционирования моделируемого объекта [2, 3, 4]. Необходимо отметить, что имитационная модель – лишь один из модулей цифрового двойника. В то же время, большинство определений цифровых двойников так или иначе подразумевают наличие в их составе имитационных моделей [5, 6].

Большинство имитационных моделей, входящих в состав «цифровых двойников», являются детальными, построенными на низком уровне абстракции [5]. Наш опыт показывает, что процесс создания имитационной модели для цифрового двойника намного больше похож на процесс разработки программного обеспечения, чем на традиционный процесс моделирования.

Для подтверждения этой точки зрения будут рассмотрены два проекта, в которых имитационная модель создавалась не только для ответа на конкретные вопросы, но и для регулярного использования в качестве средства поддержки принятия решений. Обе имитационные модели разрабатывались в среде AnyLogic, но потребовали значительных усилий по разработке функциональности, не свойственной имитационным моделям.

В примерах показывается, как объем трудозатрат смещается с использования стандартных средств среды AnyLogic на программирование, разработку алгоритмов, создание пользовательского интерфейса и тестирование.

Пример 1. Моделирование медного производства

Медное производство – система с большим количеством зависимостей между компонентами

В проекте для одного из российских металлургических предприятий требовалось создать имитационную модель для выполнения обоснованной оценки производительности цеха внепечной обработки медного производства, а также проверки эффекта от внедрения предлагаемых операционных улучшений. Упрощенная схема транспортных потоков на медном производстве показана на рис. 1.

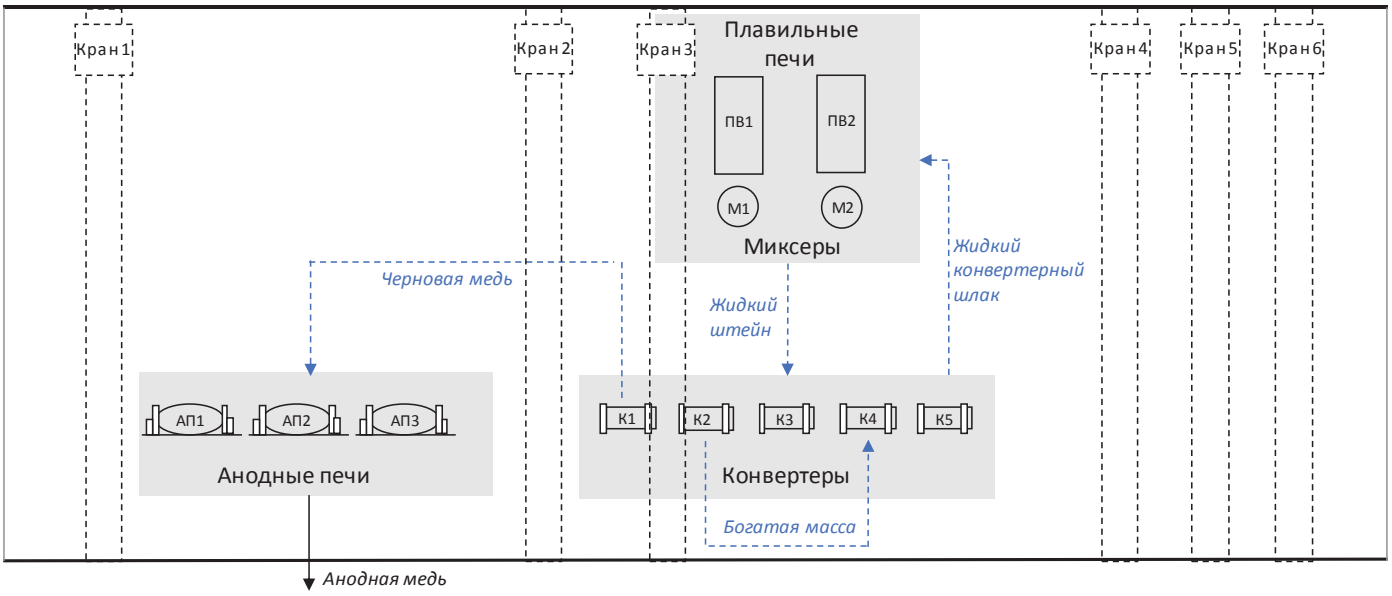


Рис. 1. Схема транспортных потоков медного производства

Жидкий штейн постоянно генерируется в плавильных печах, сливается в ковши и транспортируется в ковшах кранами к конвертерам. В конвертерах медный полупродукт проходит три периода конвертирования. После первого периода конвертирования расплав может быть перемещен в ковше в другой конвертер для продолжения цикла. В конце второго периода конвертирования образуется черновая медь, которая перевозится кранами в ковшах к анодным печам. В анодных печах черновая медь обрабатывается партиями по 6 ковшей и разливается в медные аноды.

Создание библиотеки как результат решения задачи моделирования кранов

Моделирование работы кранов – первая очевидная сложность при построении ИМ медного производства, ввиду, среди прочих, следующих факторов:

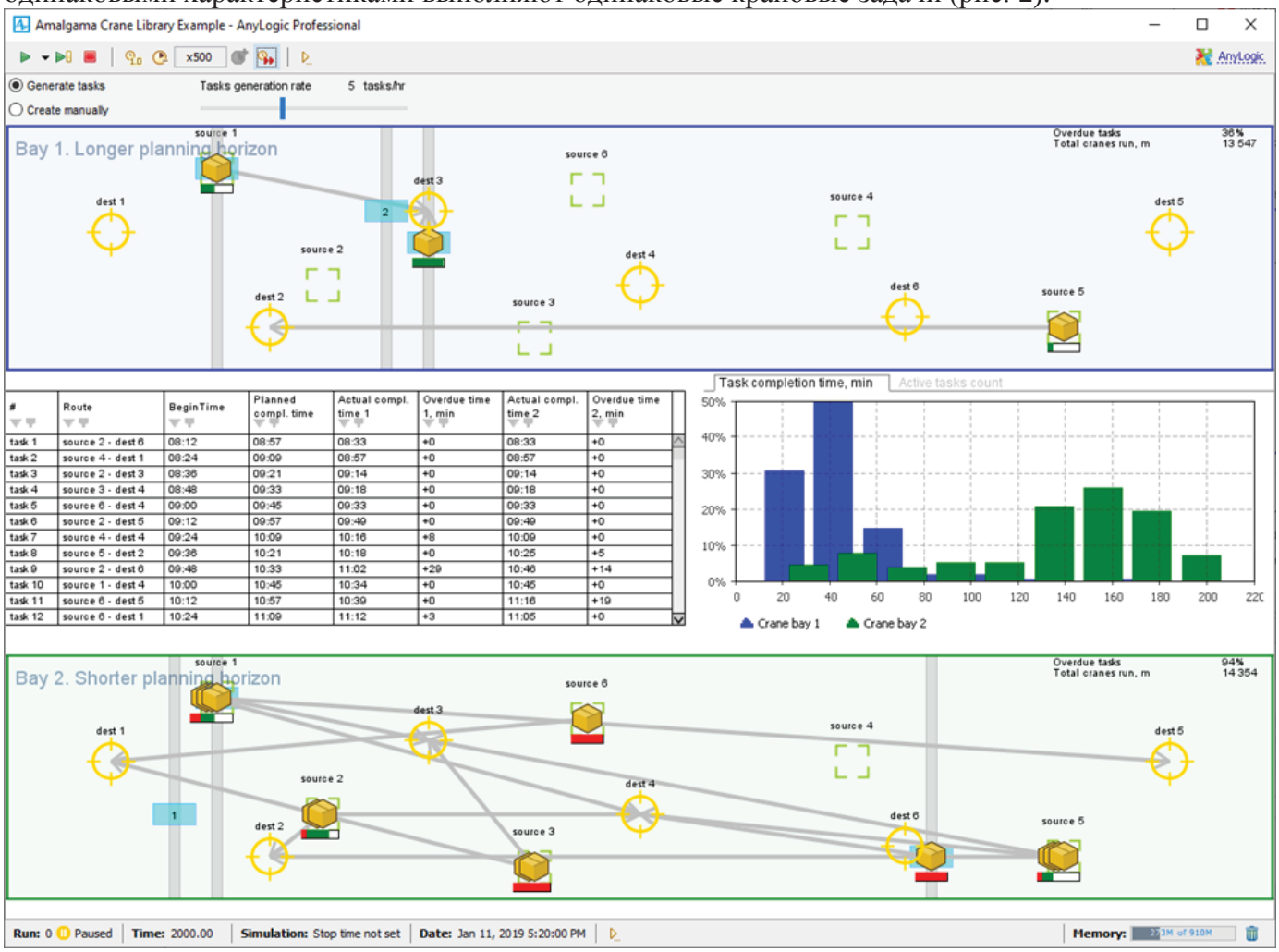
Крановые операции почти всегда включают более 2 точек транспортировки. Например, взять пустой ковш, слить штейн из миксера, перелить в конвертер, поставить на свободную позицию в цехе

Некоторые крановые операции невозможно начинать, не завершив другие. Например, нельзя слить богатую массу из конвертера, не слив из него конвертерный шлак

Операции имеют динамически меняющиеся приоритеты. Например, приоритет слива штейна из миксера возрастает по мере его наполнения.

Готовые библиотеки в составе сред моделирования общего назначения (Simio, ProModel, AnyLogic) не предоставляют возможности имитировать работу кранов со всеми необходимыми ограничениями. Очевидная реализация логики работы кранов, рассматривающая несколько ближайших крановых задач и не учитывающая при планировании будущие операции, не позволяет адекватно моделировать работу цеха. При такой реализации краны в модели работают намного менее эффективно, чем даже в реальном цехе. Для реализации реалистичной работы кранов потребовалось реализовать алгоритм составления расписания работы кранов на заданный горизонт времени на основе метода ветвей и границ с различными критериями отсечения нереализуемых или неэффективных вариантов.

Для иллюстрации значимости кранового алгоритма рассмотрим пример, сравнивающий долю вовремя выполненных крановых задач системой из трех кранов при различном горизонте планирования. Сравнение производится с помощью модели, в которой две крановые системы с одинаковыми характеристиками выполняют одинаковые крановые задачи (рис. 2).



#	Route	BeginTime	Planned compl. time	Actual compl. time 1	Overdue time 1, min	Actual compl. time 2	Overdue time 2, min
task 1	source 2 - dest 6	08:12	08:57	08:33	+0	08:33	+0
task 2	source 4 - dest 1	08:24	09:09	08:57	+0	08:57	+0
task 3	source 2 - dest 3	08:36	09:21	09:14	+0	09:14	+0
task 4	source 3 - dest 4	08:48	09:33	09:18	+0	09:18	+0
task 5	source 6 - dest 4	09:00	09:45	09:33	+0	09:33	+0
task 6	source 2 - dest 5	09:12	09:57	09:49	+0	09:49	+0
task 7	source 4 - dest 4	09:24	10:09	10:16	+8	10:09	+0
task 8	source 5 - dest 2	09:36	10:21	10:18	+0	10:25	+5
task 9	source 2 - dest 6	09:48	10:33	11:02	+29	10:46	+14
task 10	source 1 - dest 4	10:00	10:45	10:34	+0	10:45	+0
task 11	source 6 - dest 5	10:12	10:57	10:39	+0	11:16	+19
task 12	source 6 - dest 1	10:24	11:09	11:12	+3	11:05	+0

Рис. 2. Демонстрационная модель, сравнивающая алгоритмы работы кранов

В таблице 1 показаны результаты моделирования на протяжении 2000 часов модельного времени. Алгоритм планирования крановых операций, рассматривающий до 10 задач на горизонте 6 часов, позволяет повысить среднюю долю выполненных вовремя задач с 6% до 64%, сократив также и суммарный пробег кранов на 5%, по сравнению с алгоритмом, рассматривающим не более 3 задач на горизонте 2 часов.

Таблица 1. Сравнение алгоритмов работы кранов

№	Горизонт планирования, часов	Максимальное количество рассматриваемых задач	Доля задач, выполненных вовремя, %	Суммарный пробег кранов, м
1	2	3	6 %	14 354
2	6	10	64 %	13 547

Таким образом, сложность разработки модели медного производства переместилась из чисто «имитационной» области в область разработки алгоритма итерационной оптимизации, или, более общо, в область теории и технологии программирования.

Удачная декомпозиция задачи позволила создать библиотеку моделирования кранов, которая может применяться для моделирования других металлургических производств или, например, контейнерных терминалов.

Моделирование перемещения ковшей с металлом требует разработки сложного алгоритма планирования

Второй, менее очевидной сложностью моделирования являлась диспетчеризация ковшей с металлом. На первый взгляд, модель может использовать следующую логику: по окончании каждого цикла конвертирования меди определить места транспортировки ковшей с образовавшимся полупродуктом на основании состояния соседних конвертеров. Именно такой подход чаще всего используется при разработке дискретно-событийных моделей. Однако использование этого подхода неизбежно приводит модель в состояние блокировки – например, в цехе образуется слишком много ковшей богатой массы, а конвертеров недостаточно для начала новых циклов работы. В этом случае продолжение моделирования невозможно, и модель вынуждена завершать исполнение с ошибкой.

Решение этой проблемы состояло в создании отдельного модуля – планировщика работы цеха. Планировщик составляет связный непротиворечивый план обработки меди на некоторое время вперед, а имитационная модель лишь следует составленному плану, инициируя перепланирования при наступлении критических отставаний. Даже простая генерация какого-нибудь непротиворечивого плана выплавки меди – алгоритмически нетривиальная задача. Еще более непростой задачей является составление плана, позволяющего максимизировать выпуск продукции цеха – черновой меди. Для решения этой задачи был разработан итерационный пошаговый алгоритм планирования с постепенным ослаблением ограничений на каждом шаге.

Таблица 2 показывает, что логика планирования занимает 18,7% всего объема исходного программного кода модели, и потребовала 19,1% всех трудозатрат на реализацию проекта. Таким образом, сложность разработки модели снова переносится с моделирования самого процесса (конкретно – перемещение ковшей с металлом) на его планирование.

Таблица 2. Сравнение объемов и трудозатрат на разработку модулей ИМ медного производства

Модуль	Размер файла модели, КБ	Доля в суммарном размере файлов, %	Объем трудозатрат, человеко-часов	Доля в общем объеме трудозатрат, %
Планировщик	537	18,7 %	290	19,1 %
Визуализация плана	332	11,6 %	211	13,8 %
Логика и визуализация имитационной модели (включая крановую библиотеку)	1990	69,7 %	1071	67,1 %

Верификация плана выплавки меди требует гибких средств визуализации

Один из эффективных способов верификации построенного плана – его визуализация и анализ вместе с сотрудниками производства. Визуализация должна быть интерактивной, должна показывать связи между запланированными процессами, а также позволять просматривать работу планировщика по шагам. Важно отметить, что сама визуализация – это не просто стандартный график, а сложный объект пользовательского интерфейса (рис. 3), состоящий из:

1) диаграмм Гантта работы конвертеров и анодных печей;
2) графиков уровня штейна в миксерах плавильных печей;
3) связей – переливов ковшей с полупродуктом между агрегатами.

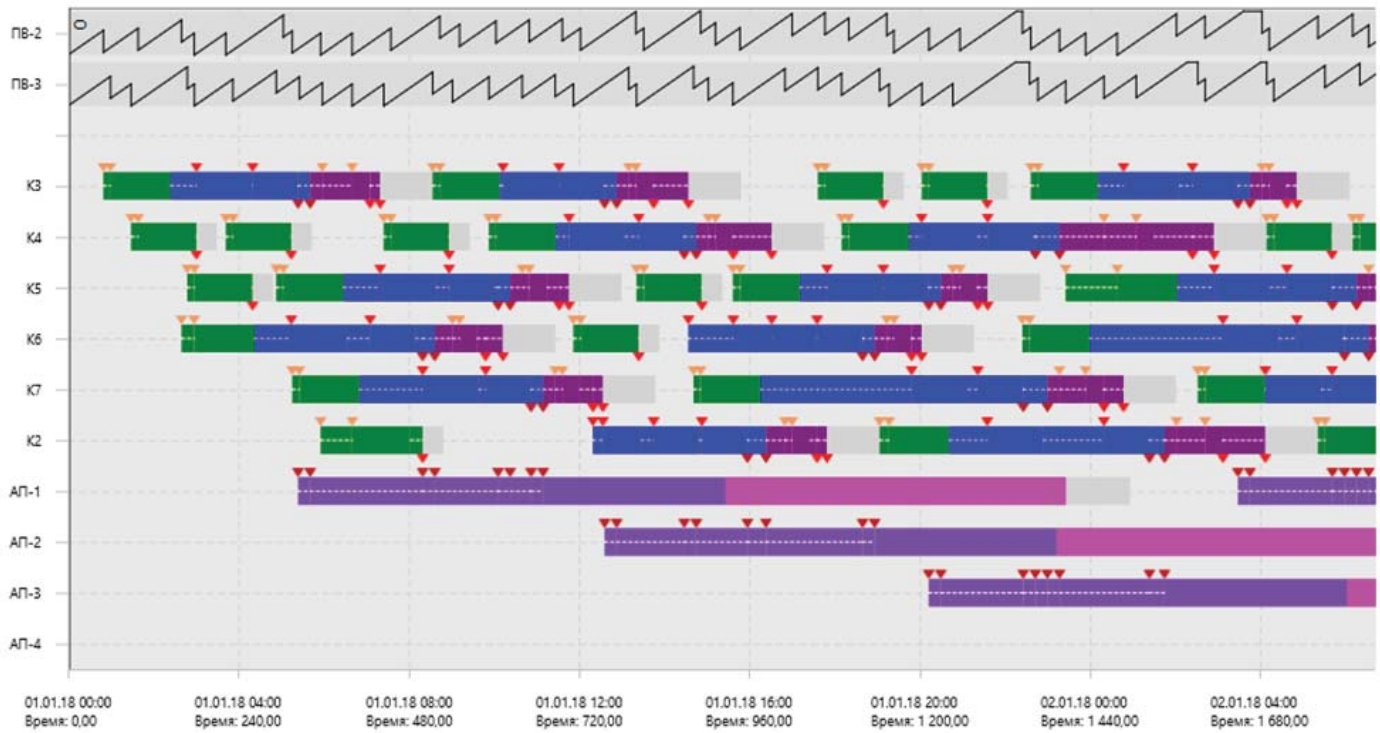


Рис. 3. Пример визуализации плана производства меди

Очевидно, что задача разработки такой визуализации выходит далеко за рамки имитационного моделирования и является, скорее, стандартной задачей разработки индустриального ПО. Существующие инструменты разработки подобных визуализаций не всегда легко совместимы с имитационными моделями, разработанными в средах моделирования. Одним из вариантов реализации данного пользовательского интерфейса могло быть создание внешнего по отношению к модели модуля и интеграция модели на AnyLogic с этим модулем. Однако такое решение усложняет разработку, так как разработчик должен вести разработку одновременно в двух различных средах программирования. Поэтому было принято решение реализовать компонент для отображения плана работы цеха в самой модели AnyLogic на языке Java с помощью технологии JavaFX.

Развитие и поддержка модели требует регрессионного тестирования

В ходе разработки и внедрения модели алгоритм планирования постоянно дорабатывался. После каждой доработки необходимо было убедиться, что качество планирования не ухудшилось ни на одном из тестовых сценариев. Этот процесс проверки называется регрессионным тестированием. В рассматриваемом примере регрессионное тестирование проводилось вручную, без использования средств автоматизации. Автоматизация регрессионного тестирования позволила бы значительно сократить трудоемкость тестирования и время выпуска обновлений.

Пример 2. Моделирование подземных горных работ

Крупной российской горно-металлургической компании требовалось создание имитационной модели для поддержки принятия решений при годовом планировании работы подземных рудников. Особенностью подземных рудников является большое число параметров, нелинейно влияющих на производительность всей системы. Так, значительное влияние на объем выработки может оказывать концентрация техники на отдельных участках рудника, дефицит рудоспусков, порядок выработки очистных камер, выполнение вспомогательных операций, а также проведение горно-подготовительных работ параллельно с очистными.

Единая среда моделирования для 6 рудников создана на платформе Eclipse

Компании-заказчику необходимо поддерживать принятие решений с помощью ИМ на 6 рудниках. Одним из способов достижения этой цели была разработка 6 разных имитационных моделей. Но такой подход значительно затруднил бы поддержку решения. Кроме того, подготовка сценариев для моделирования оказалась бы невозможной в первую очередь из-за высокой

трудоемкости задания геометрии рудника на нескольких горизонтах. Поэтому было принято решение создать и внедрить единую систему моделирования подземных горных работ для 6 рудников.

Для решения данной задачи была разработана предметно-ориентированная среда редактирования данных для имитационного моделирования подземных горных работ [7], интерфейс которой показан на рис. 4.

Все данные в системе группируются по сценариям и экспериментам. Мастер-данные являются общими для всех экспериментов. Расписания и параметры диспетчеризации оборудования могут задаваться индивидуально для каждого эксперимента. Система поддерживает импорт данных из ГГИС, EAM-систем или файлов MS Excel. Разработанная система является индустриальным ПО, обеспечивающим функционирование имитационной модели. Разработка системы велась в среде Eclipse – универсальном средстве разработки программ на Java.

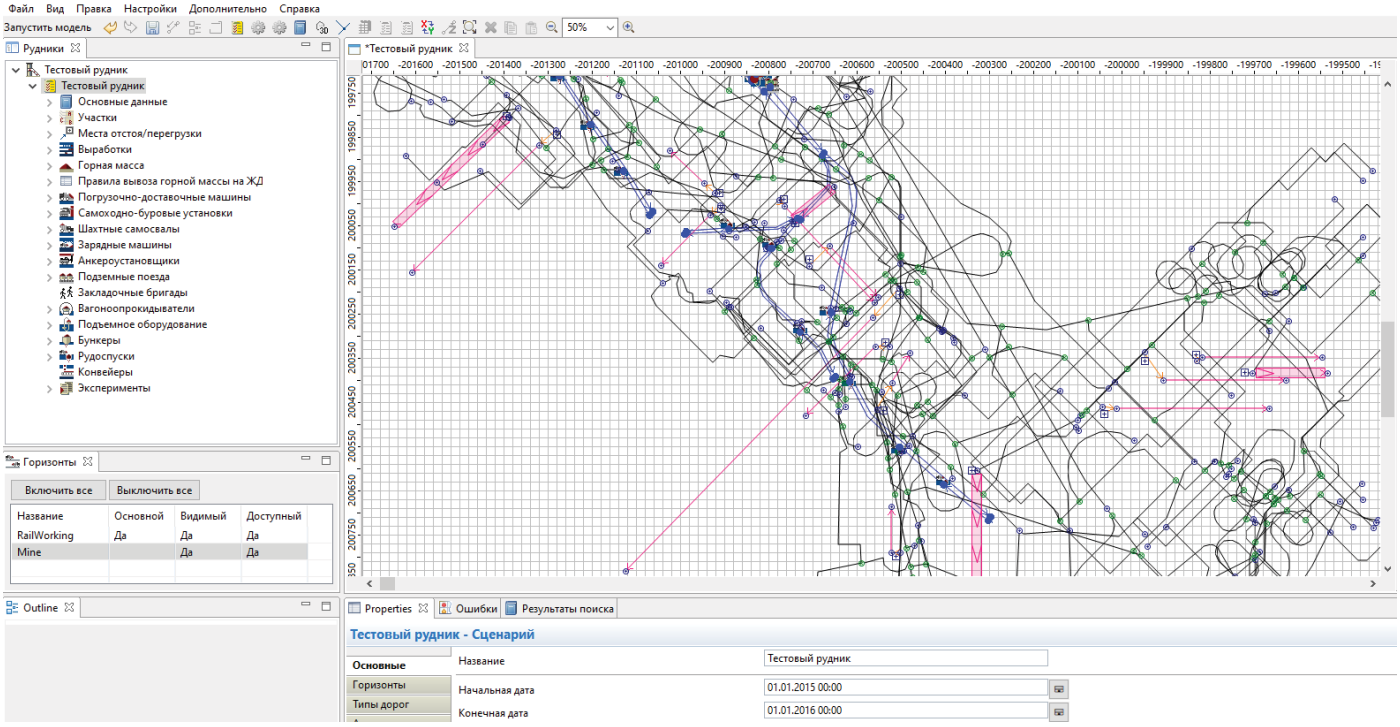


Рис. 4 Пользовательский интерфейс среды моделирования горных работ

Эффективная реализация логики перемещения агентов требует знания теории графов и владения средствами профилирования программного кода

Основной задачей при построении модели подземных горных работ стало моделирование передвижения подвижного оборудования по транспортной сети рудника. Подвижное оборудование включает в себя погрузочно-доставочные машины, шахтные автосамосвалы, самоходные буровые установки и вспомогательную технику.

Очевидной реализацией передвижения могло бы стать простое движение между точками графа транспортной сети по геометрически кратчайшему пути. Такую реализацию предоставляют все инструменты, поддерживающие агентный подход к имитационному моделированию, однако она не учитывает несколько существенных факторов:

1) по некоторым выработкам могут проехать не все единицы оборудования;
2) в течение работы модели выполняется проходка и, соответственно, образуются новые участки транспортной сети рудника;
3) для моделирования некоторых операций может быть недостаточно поиска кратчайшего пути между двумя узлами графа. Например, «везти руду к ближайшему свободному рудоспуску»;

4) одни узлы транспортной сети участвуют в маршрутах намного чаще, чем другие. Пример – рудоспуски, являющиеся конечными точками транспортировки руды на добычном горизонте.

Изменчивость весов ребер графа транспортной сети делает неэффективным использование алгоритма Флойда [8], рассчитывающего кратчайшие пути между всеми вершинами графа с алгоритмической сложностью $O(N^3)$, где N – количество узлов графа. Расчет по алгоритму Флойда пришлось бы использовать слишком часто. Расчет пути для каждого перемещения агента с помощью алгоритмов Дейкстры [9] или A* [10] также представляется нецелесообразным, так как приводит к большому количеству повторных вычислений для однотипных маршрутов передвижения техники.

В ходе работы над проектом было принято решение использовать кэш рассчитанных с помощью алгоритма Дейкстры кратчайших путей для единиц оборудования каждого типа. После построения кратчайшего пути между некоторыми двумя вершинами результат расчета заносится в кэш в памяти модели, и повторное нахождение данного маршрута сводится к извлечению ранее рассчитанного пути по ключу. При изменении веса ребра некоторые элементы кэша могут потерять актуальность. В таких случаях необходима инвалидация – определение и исключение из кэша неактуальных элементов.

В таблице 3 сравнивается время имитации 3 месяцев работы рудника с 934 выработками, 24 подъемно-доставочными машинами, 22 самоходно-буровыми установками и 2 шахтными автосамосвалами с использованием стандартных средств расчета траектории движения агентов и кэширующего алгоритма с частичной инвалидацией кэша. В таблице видно, что даже при 112 событиях частичной инвалидации кэша доля путей, взятых из кэша, составляет 86%, что позволяет сократить время выполнения модели почти на 30%.

Таблица 3. Сравнение производительности модели с различными алгоритмами нахождения пути в транспортной сети рудников

Вариант	Доля путей, взятых из кэша, %	Длительность моделирования, сек	Количество событий частичной инвалидации кэша
Без кэширования	0%	203 ± 10	0
С кэшированием	86%	143 ± 6	112

Реализация алгоритма кэширования и эффективной инвалидации кэша кратчайших путей потребовала знаний в теории графов, а также владения программированием на высоком уровне, знаний особенностей классов стандартной библиотеки языка Java (в первую очередь, стандартных коллекций, таких как HashSet и TreeSet). Ускорение реализованного алгоритма потребовало владения профилировщиком Java-программ – инструментом, показывающим, какие части кода требуют наибольшего процессорного времени для выполнения.

Выводы

Модели для цифровых двойников целесообразно создавать в средах программирования общего назначения

Опыт показывает, что при разработке сложных интегрированных имитационных моделей нецелесообразно использовать инструменты имитационного моделирования. Эти инструменты нацелены на быстрое получение результата с помощью встроенных универсальных компонентов, функциональности которых часто недостаточно в сложных проектах.

На базе приведенных примеров можно сделать вывод, что разработку «цифровых двойников» целесообразно вести в средствах разработки общего назначения, которые позволяют разработчикам использовать весь диапазон технологий и средств базового языка программирования, в частности:

1. Создавать библиотеки логических компонентов для исключения повторных реализаций алгоритмов, используемых в имитационных моделях
2. Выполнять модульное и регрессионное тестирование компонентов библиотек и моделей

3. Эффективно организовывать совместную и параллельную разработку имитационных моделей
4. Быстро доставлять изменения и обновления конечным пользователям
5. Гибко расширять функциональность существующих компонентов
6. Профилировать и отлаживать программный код имитационных моделей.
7. Очевидно, что для разработки ИМ требуются специализированные средства, которых нет в составе инструментов общего назначения. К таким средствам можно отнести следующие:
8. Имитационный движок, дающий возможность планировать и выполнять команды в заданные моменты модельного времени
9. Библиотеки для геометрических вычислений и вычислений на графах
10. Средства для сбора статистики, агрегации и интерполяции данных – накопители, табличные и кусочно-линейные функции
11. Средства составления расписаний и планирования
12. Наличие предметно-ориентированных библиотек компонентов, часто применяемых в имитационных моделях.
13. Кроме того, для быстрого построения специфического пользовательского интерфейса при разработке практически любой ИМ требуются:
14. Средства быстрого создания таблиц, графиков, диаграмм, глубоко интегрированных с логикой имитационной модели
15. Средства быстрого создания 2D- и 3D-анимации.

Предлагаемый пример архитектуры средства разработки цифровых двойников на платформе Eclipse

На основании проектного опыта в компании Амальгама сформировалась платформа (рис. 5) для разработки сложных имитационных моделей, которая удовлетворяет большинству сформулированных в настоящей статье требований. В основе платформы – язык программирования Java и платформа Eclipse. Eclipse [11] – открытая платформа для разработки промышленных приложений, ориентированных на сложные предметные области. Все модули платформы разделены на 3 логических блока:

1) пользовательский интерфейс и визуализация;
2) логика имитационной модели;
3) управление данными имитационной модели.

Такое разделение позволяет отделять управление данными модели от логики имитации, а логику имитации – от внешнего представления. Этот подход обеспечивает высокую модульность программного кода «цифровых двойников» и возможность полностью независимой параллельной работы над различными блоками создаваемых решений.

Основной блок платформы – имитационное ядро (Simulation core), базовым элементом которого является имитационный движок (Engine), позволяющий планировать и исполнять события в модельном времени. Также имитационное ядро содержит библиотеки классов, позволяющих создавать логику прикладных имитационных моделей:

1. Utils Library – содержит классы для удобного форматирования строк, дат и чисел, вычислений с плавающим окном и манипуляций с кусочно-линейными функциями, а также функции для вычислений на графах, специально ориентированные на использование в имитационных моделях.
2. Graph Agent Library – позволяет моделировать взаимодействие агентов, передвигающихся по узлам и ребрам ориентированного графа.
3. Event Library – позволяет планировать сложные последовательности действий, выполняемых по расписанию.
4. Scheduling Library – предоставляет удобный интерфейс для планирования моделируемых процессов.
5. Discrete Rate Library – позволяет моделировать перемещение и смешивание непрерывных потоков веществ по трубопроводам, бункерам и конвейерам.

6. Trains Library – позволяет моделировать движение поездов, определяя правила выполнения железнодорожных операций любого уровня сложности. При этом в библиотеке уже реализованы такие типовые маневры как переезд с протяжкой, подъезд локомотива к составу, определение свободного пути до требуемой точки железнодорожной сети.
7. Crane Library – позволяет моделировать работу нескольких мостовых кранов в крановом пролете с учетом приоритетов крановых операций и сложных маршрутов перемещений грузов.

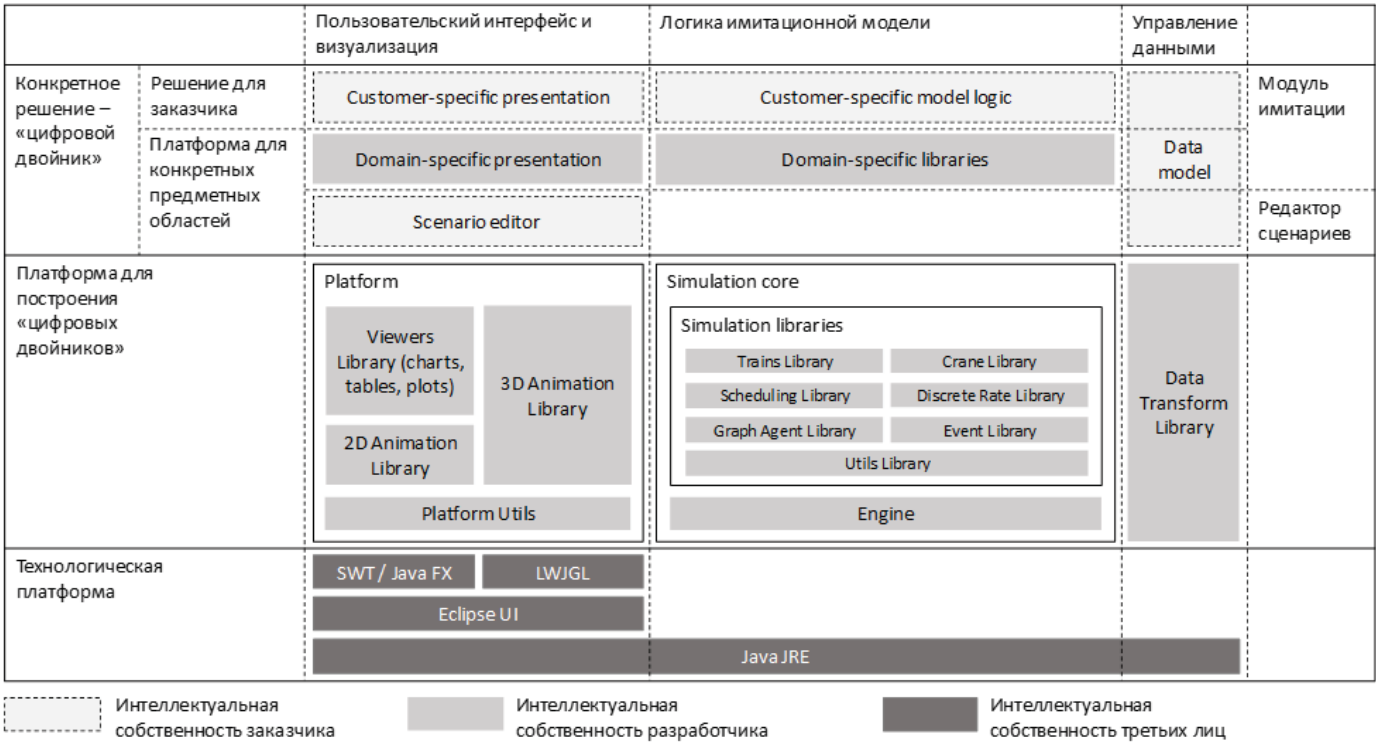


Рис. 5. Пример архитектуры платформы для разработки «цифровых двойников»

В отдельный блок – Platform – вынесены элементы, позволяющие создавать визуализации в форме таблиц, графиков, диаграмм, а также 2D- и 3D-анимации моделей. Для создания 3D-анимации используется библиотека LWJGL [12], которая обращается напрямую в графическую подсистему компьютера, снижая нагрузку на центральный процессор во время выполнения имитационных экспериментов.

Библиотека DataTransformLibrary позволяет загружать данные из текстовых файлов и файлов MS Excel, а также проверять их корректность и соответствие ограничениям модели данных. Для этой цели могут использоваться сторонние средства, такие как EMF [13] или Hibernate [14].

Требования к специалистам, создающим цифровые двойники

Опыт показывает, что для разработчиков «цифровых двойников» являются важными следующие навыки:

Способность понимать и декомпозировать различные предметные области, строить их логичное и непротиворечивое описание

Способность строить формальные постановки решаемых прикладных задач, применять существующие и проектировать собственные алгоритмы

Широкий кругозор в области математических методов, алгоритмов и структур данных

Владение современными инженерными приемами и средствами разработки программного обеспечения: контроль версий, автоматическая сборка, модульное и регрессионное тестирование

Уверенное владение всеми средствами языка программирования и платформы, лежащей в основе инструмента создания моделей.

Изучение конкретной предметной области человеком, обладающим перечисленными навыками, гораздо проще, чем приобретение этих навыков специалистами в конкретных предметных областях.

Литература

1. Sargent R.G. (1996) Verifying and Validating Simulation Models // Proceedings of the 1996 Winter Simulation Conference, pp.133–141.
2. Azad M. Madni, Carla C. Madni, Scott D. Lucero 2 (2019) Leveraging Digital Twin Technology in Model-Based Systems Engineering // Systems 2019, 7, 7; doi:10.3390/systems7010007
3. Enric Escorsa (2018) Digital Twins. A glimpse at the main patented developments // IALE Tecnologia August 2018
4. Glaessgen, Edward, Stargel (2012) The digital twin paradigm for future NASA and US Air Force vehicles // 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 20th AIAA/ASME/AHS Adaptive Structures Conference 14th AIAA. 2012
5. Söderberg, Rikard et al (2017). Toward a Digital Twin for real-time geometry assurance in individualized production // CIRP Annals 66.1 (2017): pp. 137-140
6. Saddik (2018). Digital Twins: The Convergence of Multimedia Technologies // IEEE MultiMedia. 25 (2): 87–92. doi:10.1109/MMUL.2018.023121167
A. A. Malykhanov, V. E. Chernenko (2016) Mining Simulation Tool to Support Planning and Operational Improvements // 2016 AnyLogic Conference, Nashville, TN, USA
7. Floyd, Robert W (1962) Algorithm 97: Shortest Path // Communications of the ACM. 5 (6): 345. doi:10.1145/367766.368168.
8. Dijkstra, E. W. (1959) A note on two problems in connexion with graphs // Numerische Mathematik. 1: 269–271. doi:10.1007/BF01386390.
9. Hart, P. E.; Nilsson, N. J.; Raphael, B. (1968) A Formal Basis for the Heuristic Determination of Minimum Cost Paths // IEEE Transactions on Systems Science and Cybernetics SSC4. 4 (2): 100–107. doi:10.1109/TSSC.1968.300136.
10. https://www.eclipse.org/eclipse/
11. https://www.lwjgl.org/
12. https://www.eclipse.org/modeling/emf/
13. https://hibernate.org/