Практическое занятие
Построение логической модели АСОИУ.

Часть 1. Диаграмма автоматов.

Цель занятия: получить навыки построения диаграмм автоматов UML.

Теоретические сведения к практическому занятию

Диаграмма автоматов обычно используется для описания поведения объекта в зависимости от состояния. Объект по-разному реагирует на одно и то же событие в зависимости от того, в каком состоянии он находится. Диаграммы конечного автомата обычно применяются к объектам, но могут применяться к любому элементу, который имеет поведение по отношению к другим объектам, таким как: субъекты, варианты использования, методы, подсистемы, системы и т. д.

Диаграмма автоматов - это графической изображение состояний, которые бывают простыми и составными, и переходов, соединяющих состояния.

Рамбо определяет состояние: «Состояние - это абстракция значений атрибутов и связей объекта. Наборы значений сгруппированы вместе в состояние в соответствии со свойствами, которые влияют на общее поведение объекта».

Каждая диаграмма автоматов обычно начинается с темного круга, который указывает начальное состояние, и заканчивается кружком с рамкой, который обозначает конечное состояние.

Состояния представлены прямоугольниками со скругленными углами, на которых указано название состояния. Переходы отмечены стрелками, которые переходят из одного состояния в другое, показывая, как состояния меняются.

Рис. 1. Графические примитивы диаграммы автоматов UML

Основная нотация диаграммы автоматов представлена на рисунке 2.

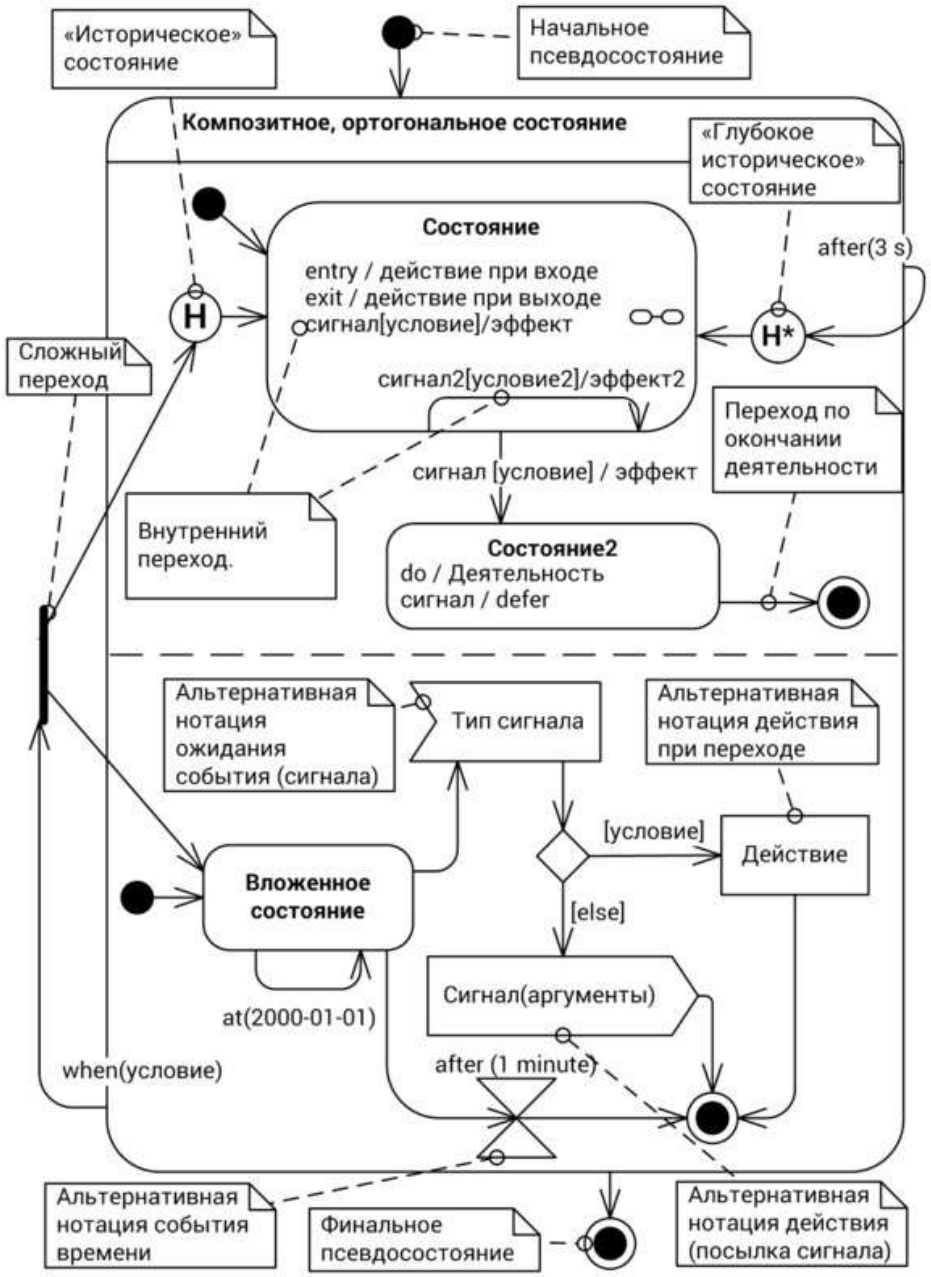

Рис.2. Основная нотация диаграммы состояний

Примеры

На рис. 7.3 показана контекстная диаграмма, моделирующая реакцию системы на выбор пользователем пунктов меню.

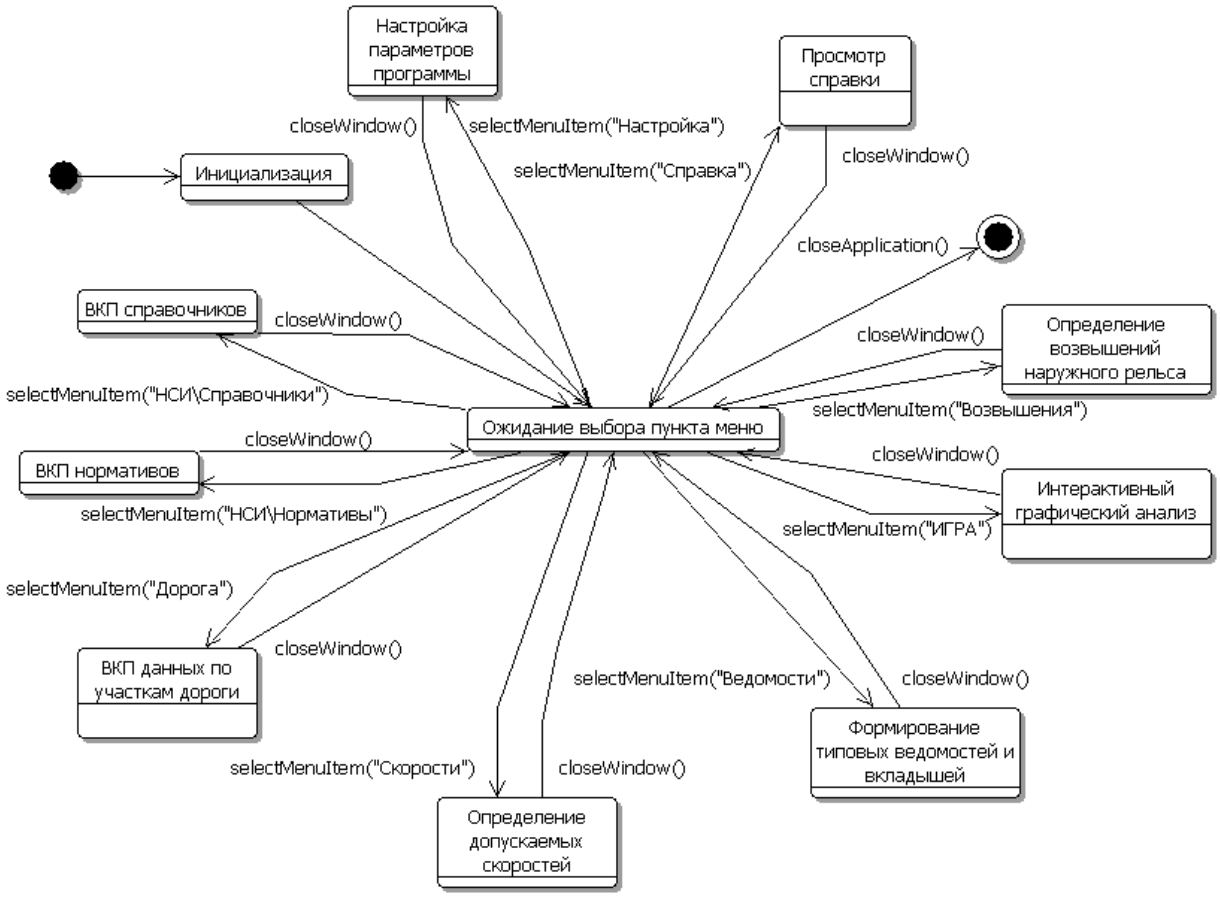

Рис. 3. Контекстная диаграмма автоматов системы определения допускаемых скоростей

На диаграмме приняты следующие условные обозначения:

- ВКП – ввод, корректировка или просмотр данных;
- selectMenuItem() – выбор пункта меню;
- closeWindow() – закрытие окна;
- closeApplication() – закрытие программы.

При запуске программы вначале осуществляется ее инициализация (чтение текущих настроек и отображение основного окна программы). Далее программа автоматически (по нетриггерному переходу) оказывается в состоянии ожидания выбора пункта меню. После выбора пункта меню осуществляется отображение соответствующего диалогового окна на экране, выполнение необходимых действий (как правило, в интерактивном режиме) и закрытие окна с возвратом в исходное состояние.

На следующем рисунке показана диаграмма декомпозиции для составного состояния «ВКП нормативов».

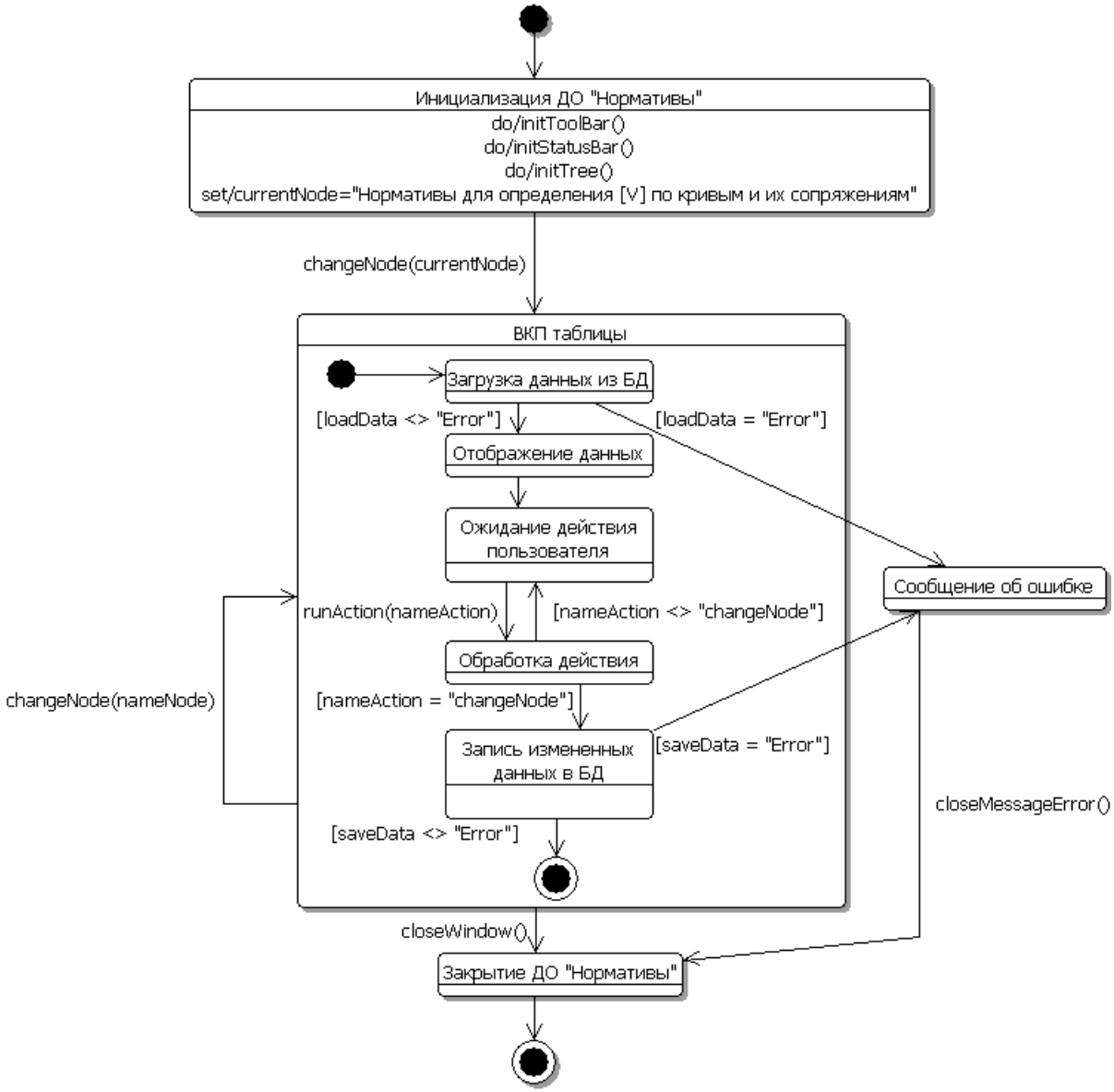

Рис. 4. Диаграмма декомпозиции для состояния «ВКП нормативов»

При выборе пункта меню «Нормативы» на экране отображается соответствующее диалоговое окно. В момент инициализации создаются панель инструментов initToolBar(), строка сообщений initStatusBar() и дерево с наименованиями таблиц в левой части диалогового окна initTree(). Исходным выбранным узлом дерева устанавливаются «Нормативы для определения [V] по кривым и их сопряжениям». При выборе узла дерева changeNode(), связанного с таблицей, система выполняет стандартные операции, начиная от загрузки данных, непосредственной обработки действий пользователя runAction() с таблицей или ее полями (редактирование значений, вставка новой записи, удаление записи, печать таблицы и т. д.) и заканчивая сохранением в базе всех измененных данных. Если в момент загрузки или записи данных возникает ошибка, то на экране отображается соответствующее сообщение. Закрытие окна «Нормативы» предусмотрено после закрытия сообщения об ошибке closeMessageError() или в результате стандартного действия пользователя closeWindow().

Пример состояния "Идентификация пользователя":

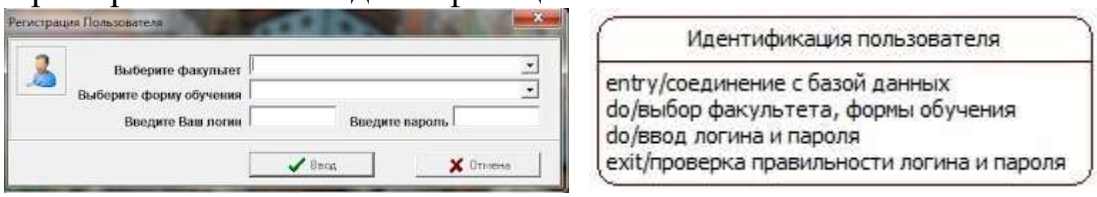

Рис.5. Фрагмент диаграммы автоматов для состояния "Идентификация пользователя"

Карманный калькулятор и конечный автомат с переходами Очистить и выключить:

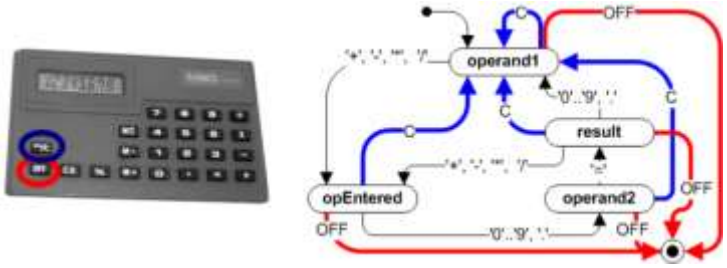

Рис.6. Фрагмент диаграммы автоматов карманного калькулятора

Карманный калькулятор и конечный автомат UML с вложением состояния:

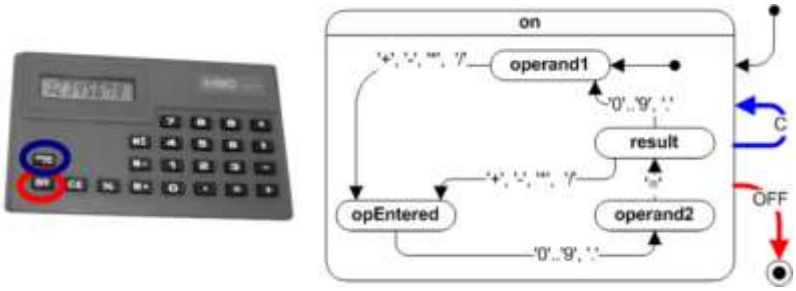

Рис.7. Фрагмент диаграммы автоматов карманного калькулятора с вложением состояния

Диаграмма автоматов для режима "Справочники" ИС "Учёт успеваемости студентов":

Рис.8. Диаграмма автоматов для режима "Справочники" ИС "Учёт успеваемости студентов"

Диаграмма автоматов для класса *ВедомостьМатериалов*

Выбранный класс может находиться в следующих состояниях:

- При создании документа класс переходит в состояние *Инициализация*, в котором выполняются расчет данных и заполнение этими данными полей документа;
- После завершения инициализации документ переходит в состояние *Действует*, в котором возможно ознакомиться с информацией в документе, отредактировать документ, распечатать или сохранить на ПК. Выход из этого состояния возможен, когда документ больше не используется для выполнения работ или когда он подлежит удалению.
- Документ теряет состояние действующего, когда наступает реальная (могут переноситься) дата завершения выполнения работ на объекте. Если документ может использоваться планово-экономическим отделом для анализа ТЭП организации или на основе этого документа будут разрабатываться новые проекты, то документ переходит в состояние *Архив*, в котором можно просматривать и извлекать информацию, однако для изменения он недоступен. При наступлении состояния документ находится в нем больше всего времени и выходит из этого состояния, когда необходимость в нем полностью исчезает и его удаляют.
- Если документ не удовлетворяет своему целевому назначению, потерял свою практическую пользу или содержит ошибочные данные, которые невозможно исправить инструментами программы (ошибка на уровне БД или других документов), то он переходит в состояние *Удален*. При выходе из этого состояния происходит удаление документа и всех содержащихся в нем данных.

Диаграмма автоматов класса *ВедомостьМатериалов* представлена на рис.9.

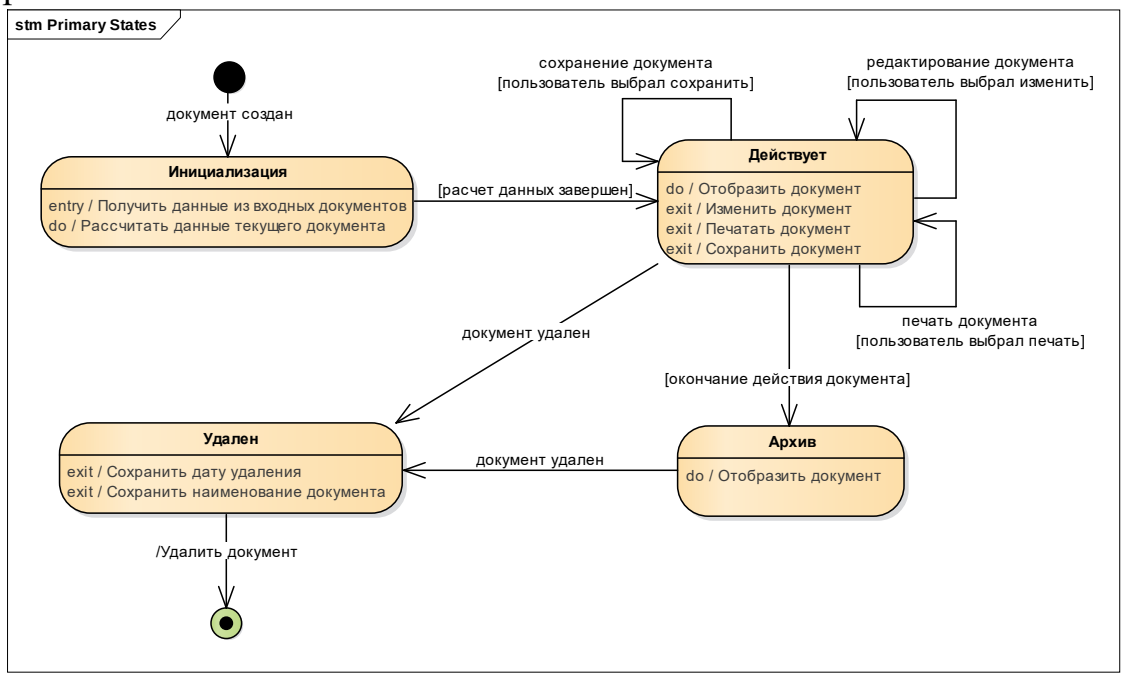

Рис.9. Диаграмма состояний для класса *ВедомостьМатериалов*

Первым состоянием на диаграмме является начальное состояние. При выполнение события «документ создан» класс переходит в состояние *Инициализация*. При входе в это состояние выполняется входное действие «Получить данные из входных документов». Основное действие, которое будет выполняться в течение всего времени, пока класс будет находиться в этом состоянии, это «Рассчитать данные текущего документа». Переход из этого состояния в состояние *Действует* произойдет при выполнении сторожевого условия «расчет данных завершен».

В следующем состоянии *Действует* имеется несколько переходов в себя с событиями:

- «редактирование документа» со сторожевым условием «пользователь выбрал изменить»,
- «печать документа» со сторожевым условием «пользователь выбрал печать»
- «сохранение документа» со сторожевым условием «пользователь выбрал сохранить»

При выходе из состояния и выполнения соответствующего событию сторожевого условия, будет выполняться выходное действие «Изменить документ», «Печатать документ» или «Сохранить документ». Действие, которое выполняется все время, пока класс находится в этом состоянии – «Отобразить документ». Выход из этого состояния состоится в 2-х случаях – когда выполнится сторожевое условие «окончание действия документа» (при этом класс перейдет в состояние *Архив*), или когда наступит событие «документ удален» (при этом класс перейдет в состояние *Удален*).

- В состоянии *Архив* присутствует только внутреннее действие – «Отобразить документ». В данное состояние класс переходит при выполнении сторожевого условия «окончание действия документа», когда СМР на объекте полностью выполнены. Выход из данного состояния и переход в состояние *Удален* осуществляется при наступлении события «документ удален».
- В состояние Удален класс переходит из состояний *Действует* или *Архив* при наступлении события «документ удален». Тут выполняется выходное действие «Сохранить дату удаления» и действие «Сохранить наименование документа». Т.к. если удаление – результат преступления, то в этом случае можно отследить момент правонарушения. При переходе из этого состояния в конечное выполняется действие «Удалить документ».

Задачи для самостоятельной работы

7.1. Светофор *TrafficLights* после создания переходит в состояние выключен *Offline*. При включении *On*, светофор переходит во вложенное состояние «зеленый» *Green* состояния включен *Online*. По истечении 50 секунд, светофор переходит в состояние «желтый» *Yellow* в *Online*. Затем, через 3 секунды – в состояние «красный» *Red*. По истечении 50 секунд

светофор возвращается в состояние «зеленый».

а. Добавьте возможность выключить *Off* включенный светофор.

б. Доработайте модель, укажите, что интервалы *t* между переключениями сигналов светофора настраиваются вызовом операций *setGreen, setRed* и *setYellow* в выключенном состоянии.

в. Измените порядок включения светофора, используя сторожевые условия, укажите по аналогии с предыдущим пунктом, что начальный сигнал светофора при включении *initialGreen*, *initialYellow* или *initialRed* настраивается *setInitial* в выключенном состоянии.

7.2. После создания статья *Paper* является черновиком *Draft*. После отправки *sent* статья рассматривается программным комитетом конференции *OnReview*, при этом выполняется рецензирование статьи *reviewPaper*. При согласии комитета *approved*, статья принимается на конференцию *Accepted*. Если комитет не принял статью *declined*, статья становится черновиком *Draft*.

а. Уточните изменения состояний статьи при рецензировании. После получения статья проверяется *Checking*. Если обнаружены недостатки, статья направляется на доработку *correct* в состояние *OnCorrection*. Если недостатков нет, рецензирование статьи завершается. При повторной отправке статьи *sent* она проверяется повторно *Checked*.

б. Укажите, что если статья не была отправлена с исправлениями недостатков в течение 10 дней, рецензирование прекращается и статья становится черновиком *Draft*.

7.3. Простейшие цифровые часы состоят из дисплея и двух кнопок A и B. Часы могут работать в двух режимах: отображения и настройки. В режиме отображения часы показывают часы и минуты, между которыми мигает символ двоеточия.

Режим настройки состоит из двух подрежимов: настройка часов и настройка минут. Кнопка A позволяет выбрать режим. Каждый раз при ее нажатии происходит переход к очередному режиму в последовательности: отображение, установка часов, установка минут, отображение и т. д. Кнопка B позволяет увеличивать значение часов или минут на единицу при каждом нажатии в одном из режимов установки. Чтобы кнопка смогла породить новое событие, ее необходимо отпустить. Нарисуйте диаграмму состояний часов.

7.4. Независимая система управления всего устройства определяет, когда двигатель должен быть включен, и непрерывно подает сигнал ВКЛ на управляющий вход двигателя.

Когда на вход подается сигнал ВКЛ, система управления двигателя должна запустить двигатель и поддерживать его работу. Двигатель запускается подачей напряжения на пусковую и рабочую обмотки. Датчик, называемый стартовым реле, определяет момент запуска двигателя, после чего отключает пусковую обмотку. Напряжение остается только на рабочей обмотке. Когда сигнал ВКЛ пропадает, обе обмотки отключаются.

Электродвигатели, применяемые в бытовой технике, могут перегреваться из-за чрезмерной нагрузки или невозможности запуска. Для защиты от перегрева в систему управления двигателем часто добавляется датчик

превышения температуры. Если двигатель нагревается слишком сильно, система управления снимает напряжение с обеих обмоток и игнорирует сигнал ВКЛ до тех пор, пока двигатель не остынет и не будет нажата клавиша сброса.

Добавьте на диаграмму следующие элементы. Деятельность: подать напряжение на пусковую обмотку, подать напряжение на рабочую обмотку. События: двигатель перегрелся, подан сигнал ВКЛ, снят сигнал ВКЛ, двигатель работает, сброс. Условие: двигатель не перегрет.

7.5. При выполнении проекта *Run* руководитель проекта (РП) осуществляет отслеживание текущего статуса *Monitor*. Если выявляется проблема *problem*, то РП устраняет ее *ManageProblem*, и возвращается *succeed* к отслеживанию. Если РП не удается решить проблему *fail*, то происходит завершение проекта *Shutdown* с выполнением остановки *stop*, по завершении которой поведение руководителя проекта завершается.

а. Добавьте этап работы РП по подготовке *Initiate* проекта. На этом этапе при необходимости *estim* производится оценка *preparePlan* для проектов с бюджетом *budget* более 100K. Для остальных проектов готовится *sketch* набросок. По утверждении *approve* проекта РП переходит к управлению им *Run*.

б. Покажите на диаграмме, что при управлении проектом РП взаимодействует с другими *Communicate*. Если поступает запрос на собрание m*eetingRequest*, то РП его назначает *schedule*. По завершении *done* собрания *RunMeeting*, РП возвращается к взаимодействию.

в. Добавьте возможность РП приостановить *suspend* работы в любой момент. Если работы приостановлены более года, то РП завершает проект *Shutdown*. При возобновлении *resume*, выполнение проекта продолжается с того момента, когда он был приостановлен.

7.6. Моделируется мобильное приложение – электронный словарь, имеющее несколько окон. Схему перехода между окнами будем моделировать с помощью диаграммы схем состояний.

а. Добавьте на диаграмму состояния *Список слов*, *Перевод*, *Словоформы слова*. Переход между *Список слов* и *Перевод* по событию *Выбрано слово*, обратный переход по событию *Назад*. Переход между *Перевод* и *Словоформы слова* по событию *Словоформы*, обратный переход по событию *Назад*.

б. В некоторых карточках перевода есть ссылки на другие карточки. При переходе по ссылке мы попадаем в то же состояние *Перевод*, но для другого слова. Добавьте переход в себя из состояния *Перевод*, активируемый событием *Переход по ссылке*.

в. Возврат после перехода по ссылке по событию *Назад* должен возвращать пользователя в предыдущую карточку. Для этого добавим к переходу по ссылке действие *Добавить в стек*. Добавим переход в себя из состояния *Перевод* по событию *Назад* со сторожевым условием *[стек не пуст]* и действием *Убрать из стека*. В переход из состояния *Перевод* в состояние *Список слов* по событию назад нужно добавить сторожевое условие *[else]*.

г. Перед запуском приложения нужно загрузить словари. На время загрузки словарей пользователю будет показан экран приветствия. Добавим

ортогональное состояние *Инициализация*. В нем выделим два региона. В первом регионе добавим начальное и конечное псевдосостояния. Между ними добавим состояние *Загрузка* словарей с выполняемой при нахождении в состоянии деятельностью *Загрузить словари*. Ожидается переход в конечное состояние по завершении деятельности. Во втором регионе мы добавим переход из начального состояния в конечное по событию времени *after (3 s)*. Это необходимо, чтобы при быстрой загрузке пользователь успел прочитать содержимое экрана приветствия. Из ортогонального состояния *Инициализация* переход по завершению в *Список слов*.

д. Уточним поведение в окне списка слов. *Список слов* становится составным состоянием. Дальнейшие состояния являются вложенными в *Список слов*. Из начального псевдосостояния происходит переход в состояние *История запросов*. По событию *Ввод* из *Истории запросов* происходит переход в псевдосостояние выбора. Далее при условии *поле ввода пустое* происходит переход (возврат) в состояние *История запросов*. Иначе происходит переход во вложенное составное состояние *Поиск слова*. По событию *Ввод* происходит переход из *Поиск слова* в описанное выше псевдосостояние выбора. В составном состоянии *Поиск слова* непосредственно процедура поиска откладывается с помощью события времени *after (1 s)*. После этого события происходит переход из начального псевдосостояния в состояние *Поиск в словаре*. По завершению происходит переход из *Поиск в словаре* в *Отображение списка*.

Часть 2. Диаграмма деятельности.

Цель занятия: получить навыки построения диаграмм деятельности.

Теоретические сведения к практическому занятию

Диаграмма деятельности — технология, позволяющая описывать логику процедур, бизнес-процессы и потоки работ

Основным отличием диаграмм деятельности от блок-схем является активная поддержка параллельных процессов, что объясняет применение диаграммы деятельности для моделирования потоков работ.

Некоторые из наиболее распространенных компонентов диаграммы активности включают:

Действие: этап действия, на котором пользователи или программное обеспечение выполняют заданную задачу. Действия обозначаются прямоугольниками с закругленными краями.

Узел принятия решения: условная ветвь в потоке, представленная ромбом. Он включает один вход и два или более выходов.

Потоки управления: другое название соединителей, которые показывают поток между шагами на схеме.

Начальный узел: символизирует начало занятия. Начальный узел представлен черным кружком.

Конечный узел: представляет последний этап действия. Конечный узел представлен обведенным черным кружком.

Для моделирования бизнес-процесса можно использовать дорожки –

диаграмма с разделением деятельности.

Основная нотация диаграммы представлена на рисунке 10

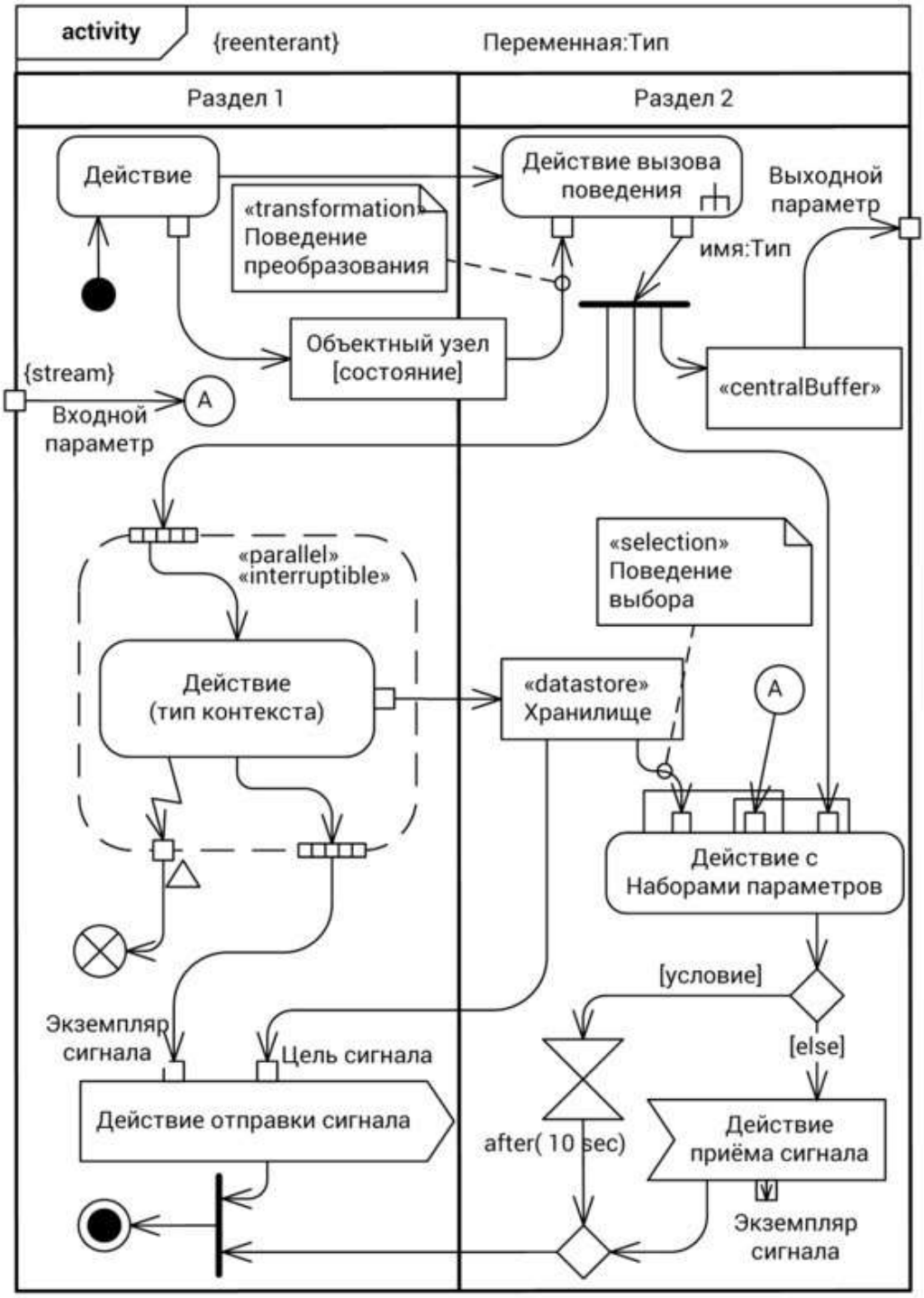

Рис. 10. Основная нотация диаграммы деятельности

Примеры

На рис. 11 показана диаграмма деятельности для подпотока «Составить документ».

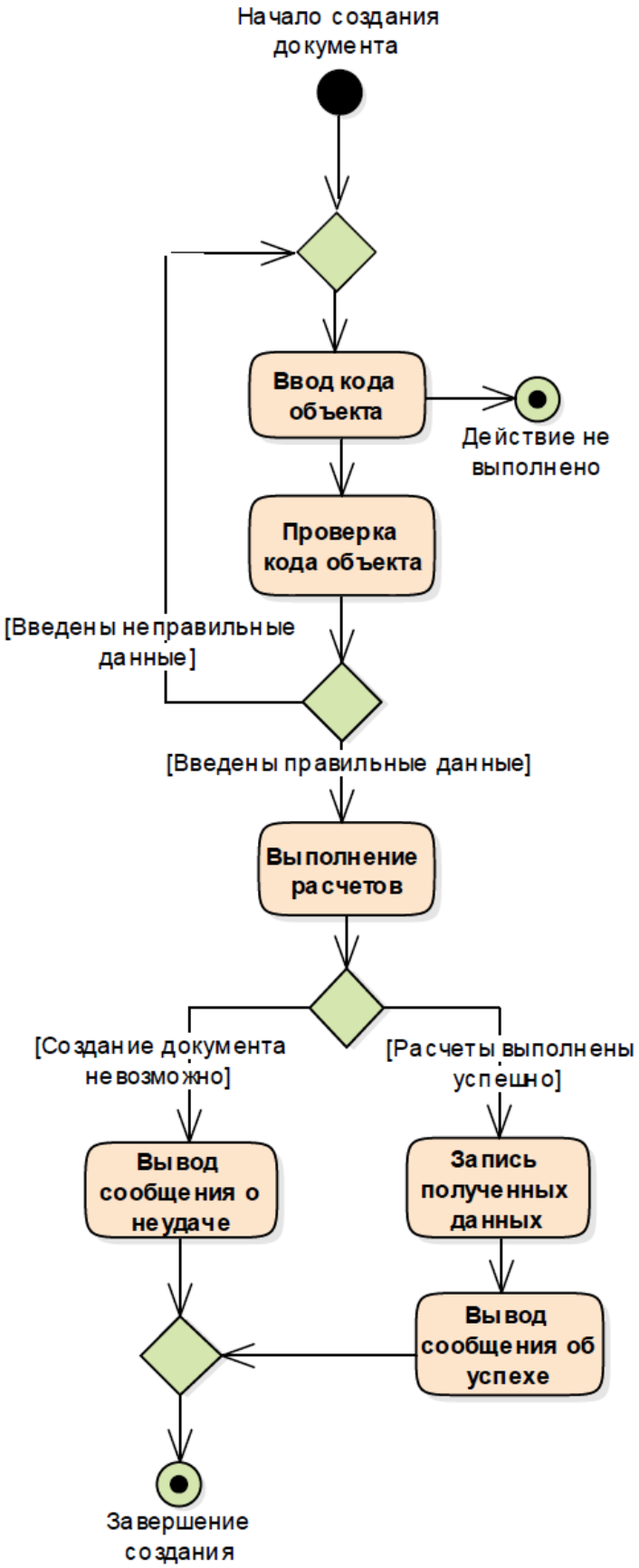

Рис. 11. Диаграмма деятельности для действия «Составить документ»

На рис. 12 показана диаграмма деятельности для подпотока входа в систему:

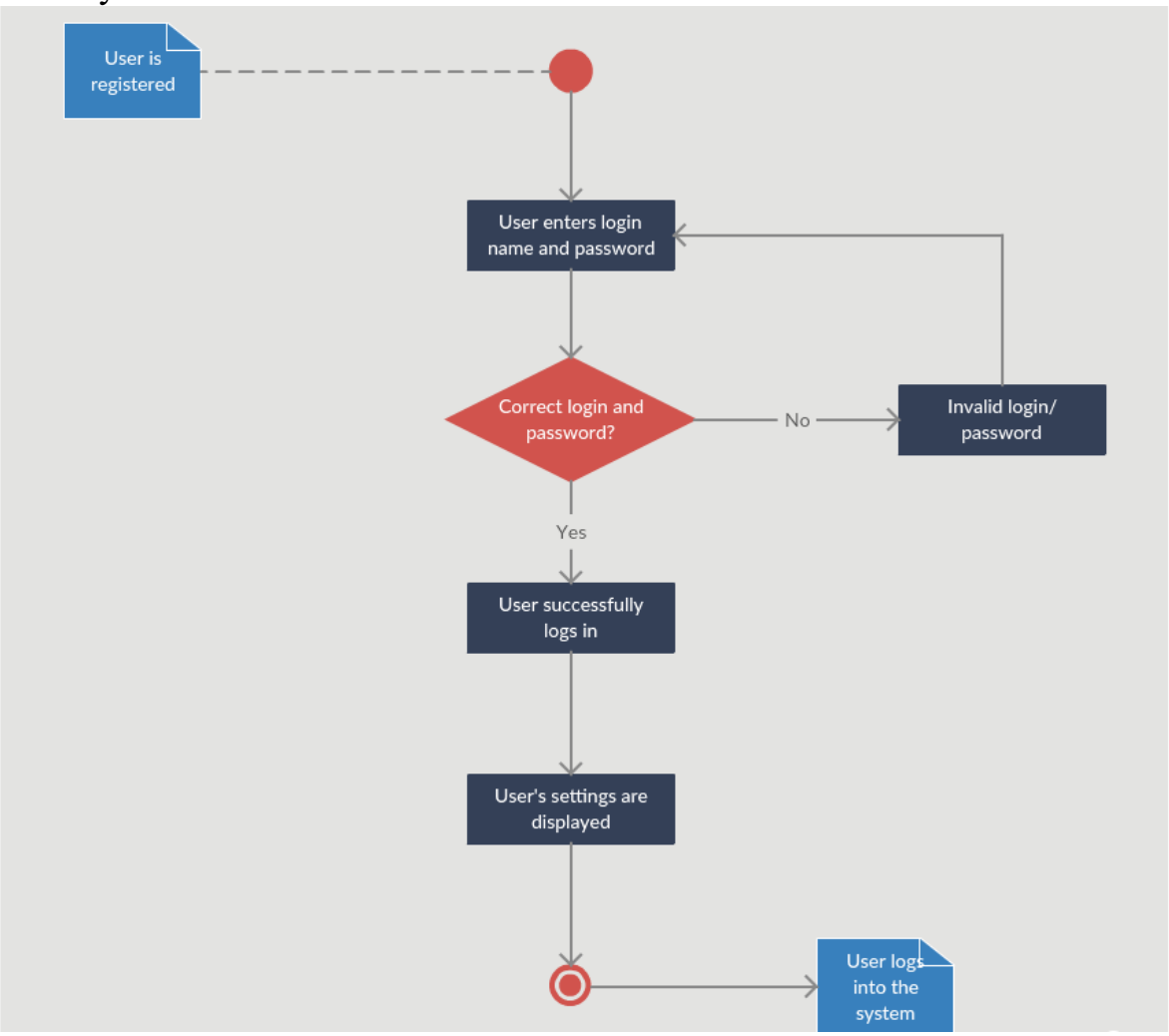

Рис. 12. Диаграмма деятельности для действия Вход в систему

На рис. 13 показана диаграмма деятельности для процесса «Учет расходных материалов»:

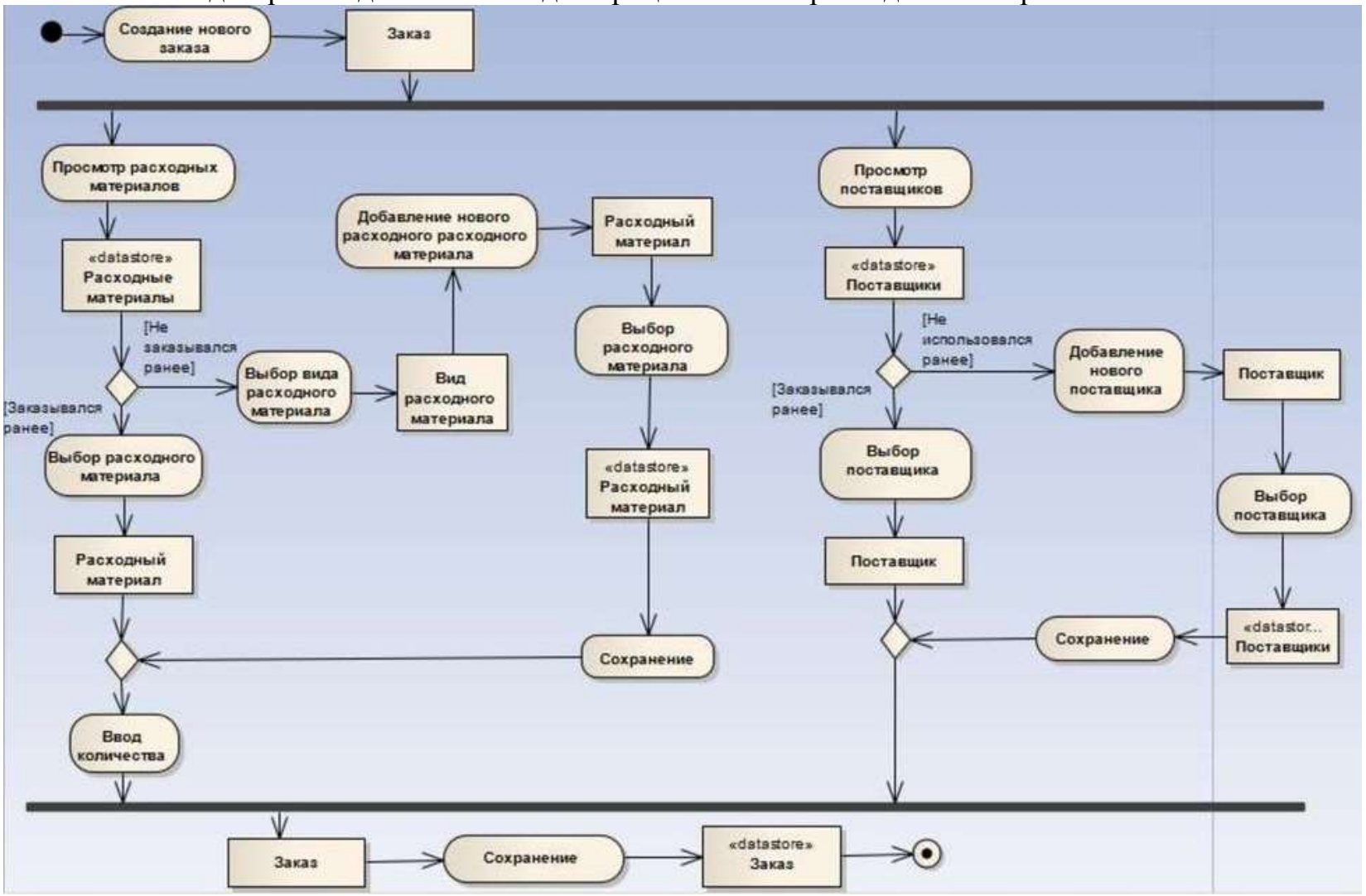

Рис. 13. Диаграмма деятельности для процесса «Учет расходных материалов»

На рис. 14 показана диаграмма деятельности для системы управление школой»:

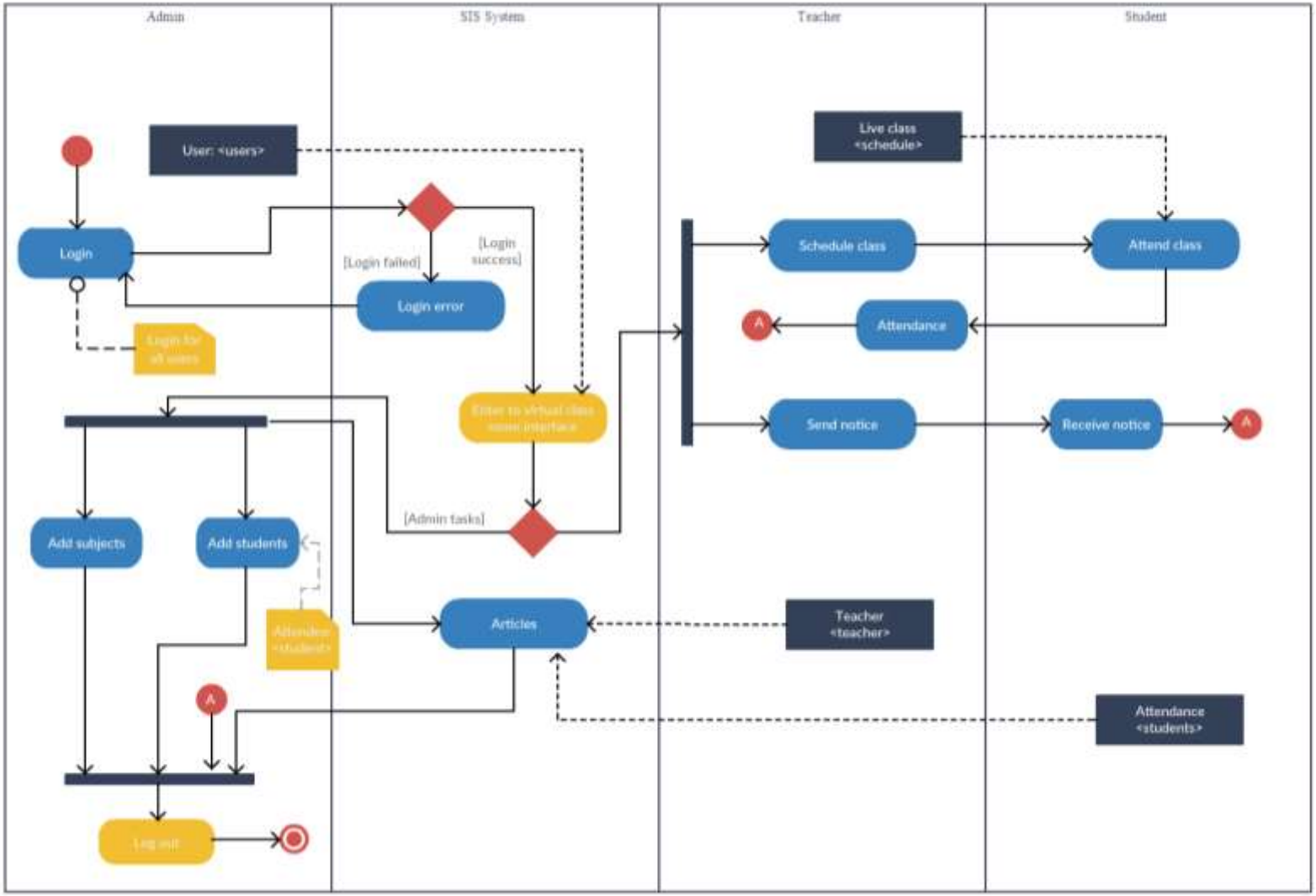

Рис. 14. Диаграмма деятельности для процесса управления школой

На рис. 15 показана диаграмма деятельности для системы управления документами

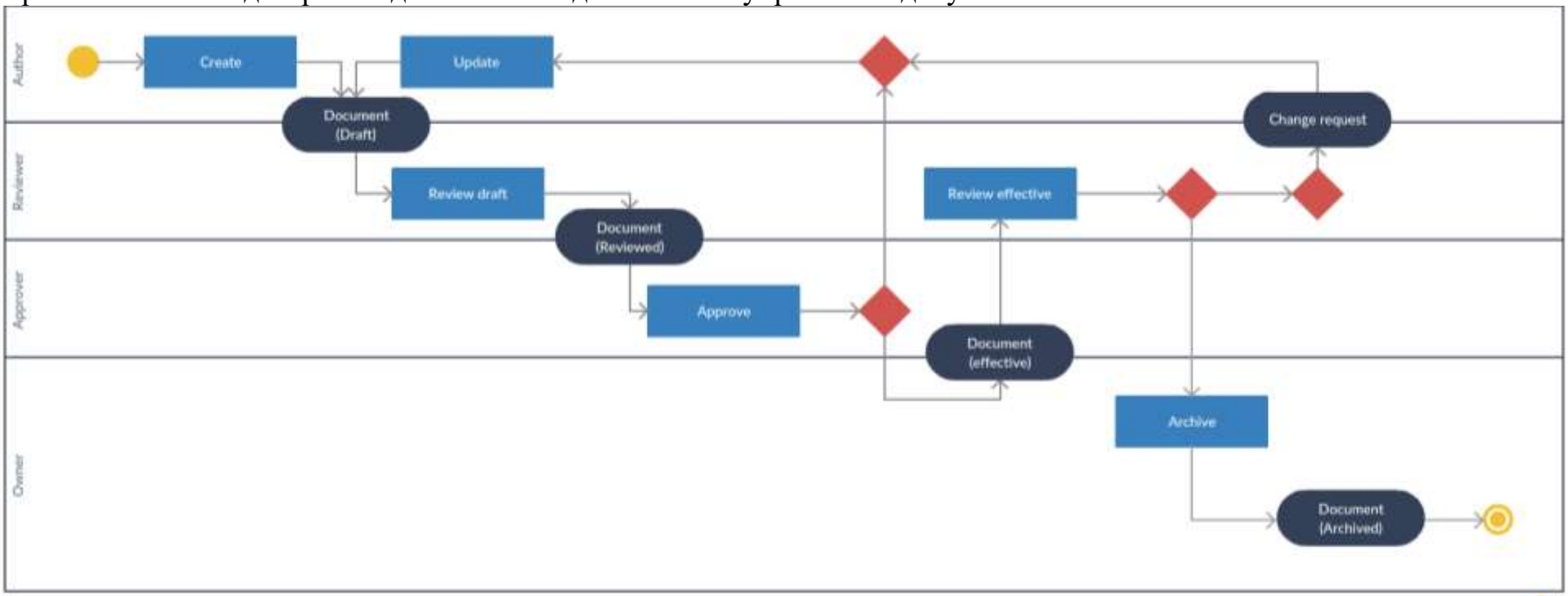

Рис. 15. Диаграмма деятельности для процесса управления документами

Задачи для самостоятельной работы

8.1. Первым действием деятельности является добавление элемента в заказ *AddItem*. Если заказ сформирован (условие *ready*), то он направляется получателю действием *ShipOrder* и деятельность завершается. В противном случае добавляется еще один элемент заказа.

а. Используя управляющие узлы деятельности, добавьте действие отправки счета *SendInvoice* так, чтобы оно исполнялось после того, как заказ сформирован, независимо от отправки заказа. При этом деятельность не завершается без отправки счета.

б. Предоставьте возможность после внесения элемента, добавить еще элемент в заказ при условии *continue* или завершить выполнение деятельности при условии *cancel*.

82. Деятельность *Обслуживание резервуаров* по работе гидропонной теплицы начинается с действия садовода (*Gardener*) *Проверить уровни резервуаров.* Если все уровни в пределах нормы, то деятельность завершается. Иначе *Gardene* откладывает все операции *Отложить операции*. Если все операции отложены, *Gardene* устанавливает уровни резервуаров действием *Установить уровни резервуаров* и процесс переходит к *WaterTank* (Поток воды) действием *Отключить обогрев* и *NutrientTank* (Резервуар удобрений) действием *Наполнить*. Если температура становиться ниже безопасной, то *WaterTank* (Поток воды) начинает наполнять резервуар действием *Наполнить*. Если бак наполнен, *WaterTank* (Поток воды) включает обогрев действием *Включить обогрев*.

Садовод возобновляет операции *Возобновить операции* после того как включен обогрев и резервуар удобрений наполнен. Далее деятельность завершается.

8.3. Деятельность *ImplementSolution* по реализации автоматизированной системы (АС) начинается с общения *Communication* аналитика *Analyst* с заказчиком. Созданное описание проекта *SoW* используется аналитиком для моделирования *Modeling*. Разработанная в результате модель *Model* сохраняется в репозитории проекта и передается в разработку *Development* и тестирование *Testing*, выполняемые проектной командой *DevTeam*. Результат разработки *Product* передается на тестирование, после этого производится его внедрение *Deployment* интегратором *Implementer*. Отчет о внедрении *Report* является результатом деятельности и деятельность завершается.

а. Покажите, если тестирование обнаружило дефекты в продукте, деятельность возвращается к разработке.

б. В условии задачи не сказано, что описание проекта используется для подготовки к внедрению интегратором. Добавьте данное действие и укажите, что внедрение выполняется только после завершения подготовки.

8.4. Рассмотрим процесс управления продуктовой линейкой *RunSPL*. Сначала заинтересованные стороны *stakeholders* ожидают поступления изменений *WaitChange*. При поступлении изменения *change*, оно передается

для оценки *ReviewApprove* комитету *ReviewCommittee*. Результатом оценки является спецификация *spec*, которая передается далее рабочей группе линейки *SPLTeam* для планирования *Plan*. В результате планирования получается проект *project*, который также является результатом процесса.

а. Сохраняйте все спецификации для последующего анализа.

б. Покажите, что если в результате оценки изменение было отклонено *rejected*, то выполнение процесса управления возвращается к ожиданию изменений. Если же изменение было принято *approved*, то процесс переходит к планированию спецификации и ожиданию новых изменений.

8.5. Подготовьте диаграмму деятельности, описывающую процесс загрузки выполненного задания на сайт.