Содержание

Введение

1. Общая часть

1.1 Характеристика структурного подразделения "Шахматный клуб"

1.2 Обзор шахматных систем-прототипов

1.3 Анализ достоинств и недостатков

1.4 Техническое задание на создание информационной системы "Шахматный клуб"

2. Специальная часть

2.1 Выбор основных методологий разработки программного обеспечения

2.1.1 Каскадная методология

2.1.2 Экстремальная методология

2.2 Выбор инструментальных средств

2.3 Содержательная постановка задачи создания СШПО

2.4 Разработка алгоритма решения задачи

2.5 Описание разработанного программного комплекса

2.5.1 Транслятор шахматных партий

2.5.2 Регистратор шахматных партий

2.6 Тестовые испытания и анализ результатов

3. Технико-экономическое обоснование проекта

3.1 Целесообразность и область применения разработки

3.2 Расчет затрат на разработку специализированного шахматного программного обеспечения

3.3 Расчет эксплуатационных затрат

3.4 Оценка экономической эффективности проекта

4. Безопасность и экологичность проекта

4.1 Безопасность труда

4.1.1 Анализ условий труда на рабочем месте инженера структурного подразделения "Шахматный клуб" ГОУ ВПО "СибГИУ"

4.1.2 Меры по безопасности труда

4.2 Чрезвычайные ситуации

4.2.1 Электробезопасность

4.2.2 Пожарная безопасность

4.2.3 Организационно-штатная структура по ГО и ЧС

4.2.4 Чрезвычайные ситуации, которые могут возникнуть на территории ГОУ ВПО "СибГИУ"

4.2.5 Способы оповещения при ЧС

4.2.6 Действия по видам сообщения при передаче сигнала "Внимание всем!"

4.2.7 Организация защиты сотрудников и студентов ГОУ ВПО "СибГИУ"

при возникновении ЧС мирного и военного времени

4.3 Экологическая безопасность

Заключение

Список использованных источников

Приложение А. Календарный план работ по созданию системы

Приложение Б. Комплектность документации на систему

Приложение В. Листинг программы-транслятора шахматных партий

Приложение Г. Листинг программы-регистратора шахматных партий

Приложение Д. Снимки экрана

Приложение Е. Протокол DGT

Приложение Ж. Общая структура транслятор шахматных партий

Приложение З. Общая структура регистратора шахматных партий

Приложение И. Анализ результатов тестовых испытаний

Приложение К. Листинг модуля вещания шахматных партий

Введение

В настоящее время во всем мире резко возросло число спортивных шахматных соревнований высокого и среднего уровня. Это способствует резкому росту общественного интереса к данному виду спорта и накладывает требование широкого освещения подобных событий в средствах массовой информации и сети Интернет. Для этих целей в настоящее время активно используются информационные системы, оснащенные электронными шахматными досками и оборудованием для отображения информации. Однако все подобные системы имеют ряд существенных недостатков. Все они вытекают в основном из отсутствия рынка информационных шахматных систем. Во-первых, большую сложность представляет закупка оборудования для информационных шахматных систем, так как его поставкой в мире занимается только 3 организации, и велик перечень требований, которые необходимо указать при заказе. Во-вторых, отсутствует свободно распространяемое программное обеспечения для подобных систем.

В данном дипломном проекте приводится пример создания информационной шахматной системы и делается попытка создания для нее программного обеспечения.

Целью дипломного проекта в первую очередь является обеспечение всемирной трансляции спортивных шахматных соревнований с применением разработанного в ходе проекта законченного программного продукта. В дальнейшем планируется коммерческое использование данной разработки.

1. Общая часть

1. Характеристика СП «ШК»

Структурное подразделение «Шахматный клуб» входит в состав государственного образовательного учреждения высшего профессионального образования «Сибирский государственный индустриальный университет» (ГОУ ВПО «СибГИУ»). Помещение шахматного клуба является полигоном кафедры «Физического воспитания, здоровья и спорта» (ФВЗиС) для проведения занятий по дисциплине «Шахматы» для студентов, по состоянию здоровья не пригодных к занятиям физической культурой в основной учебной группе. В настоящее время в помещении структурного подразделения «Шахматный клуб» (СП «ШК») занимается около 300 студентов и сотрудников университета. Изменение количества студентов, занимающихся в помещении СП «ШК» по дисциплине «Шахматы» за последние 5 лет отражает рисунок 1.



Рисунок 1 – Изменение числа студентов, обучающихся по дисциплине «Шахматы»

Таким образом, число студентов, занимающихся по дисциплине «Шахматы» за последние 5 лет стабильно не претерпевает резких изменений.

Помимо этого, помещение СП «ШК» является базой для проведения соревнований по шахматам различного уровня: от внутригородских до международных.

Учебный процесс по дисциплине «Шахматы» осуществляется в соответствии с учебным планом занятий кафедры ФВЗиС. Правила шахматной игры и правила проведения шахматных соревнований регулируются кодексом международной шахматной федерации ФИДЕ.

Процесс проведения соревнований по шахматам в помещении СП «ШК» осуществляется в соответствии с календарным планом проведения соревнований на текущий год, а также в сотрудничестве с городским муниципальным учреждением дополнительного образования детей «Специализированной детско-юношеской спортивной школой Олимпийского резерва по шахматам имени Б. А. Кустова» (МУДОД «СДЮСШОР по шахматам»).

Корпус СП «ШК» включает в себя следующие помещения: основной турнирный зал, судейская, раздевалка, преподавательская, малый турнирный зал, кабинет директора, подсобное помещение, туалетные комнаты. СП «ШК» оснащено следующим инвентарём: шахматные столы – 53 штуки; комплекты шахмат – 40 штук; шахматные часы – 40 штук. Обслуживающий персонал шахматного клуба: директор, ведущий инженер, две технички. Занятия в помещении СП «ШК» проводят 3 преподавателя кафедры физической культуры, здоровья и спорта. Во время проведения спортивных соревнований в случае нехватки инвентаря и оборудования все необходимое предоставляется в безвозмездное пользование на период соревнований городским МУДОД «СДЮСШОР по шахматам». Нехватка персонала на период спортивных состязаний компенсируется путем командировки работников городского МУДОД «СДЮСШОР по шахматам» на обслуживание и судейство соревновательного процесса, а также путем привлечения студентов-разрядников на волонтерских началах. Организационная структура СП «ШК», включающая работников смежных подразделений и учреждений, представлена на рисунке 2.



Рисунок 2 – Организационная структура СП «ШК»

За последнее время резко возросло число соревнований, проводимых в помещении СП «ШК». В сотрудничестве с городским МУДОД «СДЮСШОР по шахматам» в помещении СП «ШК» только за 2007 год были проведены следующие городские соревнования по шахматам:

* первая лига г.Новокузнецка по шахматам среди мужчин;
* первая лига г.Новокузнецка по шахматам среди женщин;
* лично-командный студенческий чемпионат ГОУ ВПО «СибГИУ» по шахматам среди факультетов;
* лично-командный преподавательский чемпионат ГОУ ВПО «СибГИУ» по шахматам среди факультетов;
* чемпионат г. Новокузнецка по шахматам среди ветеранов;
* традиционная летняя городская Спартакиада;
* турнир по быстрым шахматам среди студентов;
* первенство города по блицу среди студентов;

внутригородские шахматные соревнования, приуроченные к различным праздникам:

* традиционный новогодний блицтурнир;
* блиц турнир ко Дню Победы среди студентов;
* блиц турнир ко Дню Победы среди ветеранов;
* рождественский шахматный турнир;
* турнир, посвященный 8 марта;
* турнир, посвященный 23 февраля;

соревнования от областного до международного уровня:

* чемпионат СФО по шахматам среди мальчиков до 8 лет;
* чемпионат СФО по шахматам среди девочек до 8 лет;
* чемпионат СФО по шахматам среди мальчиков до 10 лет;
* чемпионат СФО среди мальчиков до 10 лет по быстрым шахматам;
* чемпионат СФО по шахматам среди девочек до 10 лет;
* чемпионат СФО среди девочек до 10 лет по быстрым шахматам;
* чемпионат СФО по шахматам среди мальчиков до 12 лет;
* чемпионат СФО среди мальчиков до 12 лет по быстрым шахматам;
* чемпионат СФО по шахматам среди девочек до 12 лет;
* чемпионат СФО среди девочек до 12 лет по быстрым шахматам;
* чемпионат СФО по шахматам среди мужчин;
* чемпионат СФО по шахматам среди женщин;
* областная спартакиада ВУЗов Кузбасса (студенты);
* областная спартакиада ВУЗов Кузбасса (преподаватели);
* областной командный турнир ветеранов в честь Дня Победы;
* традиционный международный турнир памяти Толстогузова, мужчины;
* традиционный международный турнир памяти Толстогузова, женщины;
* традиционный международный турнир памяти Толстогузова, преподаватели ВУЗов;
* первенство области по шахматам среди мальчиков до 8 лет;
* первенство области по шахматам среди девочек до 8 лет;
* первенство области по шахматам среди мальчиков до 10 лет;
* первенство области среди мальчиков до 10 лет по быстрым шахматам;
* первенство области по шахматам среди девочек до 10 лет;
* первенство области среди девочек до 10 лет по быстрым шахматам;
* первенство области по шахматам среди мальчиков до 12 лет;
* первенство области среди мальчиков до 12 лет по быстрым шахматам;
* первенство области по шахматам среди девочек до 12 лет;
* первенство области среди девочек до 12 лет по быстрым шахматам;

За последнее время работниками СП «ШК» в сотрудничестве с городским МУДОД «СДЮСШОР по шахматам» были проведены следующие крупнейшие международные шахматные форумы:

* лично-командный чемпионат России по шахматам среди студентов ВУЗов;
* лично-командный чемпионат Мира по шахматам среди студентов ВУЗов (во время написания дипломной работы данный шахматный форум находился на стадии подготовки).

На чемпионате России по шахматам среди студентов ВУЗов команде ГОУ ВПО «СибГИУ» (в составе которой был и автор данной дипломной работы; тренер команды – мастер международного класса по шахматам, преподаватель кафедры физического воспитания, здоровья и спорта Зайцев В. В.) удалось стать бронзовыми призерами. А, немного забегая вперед, на лично-командном чемпионате Мира по шахматам среди студентов ВУЗов сборная студенческая команда России, в составе которой выступали от ГОУ ВПО «СибГИУ» два представителя, выступила очень успешно, с крупным отрывом обогнав ближайших конкурентов, сборную команду Китая, и победив на этом грандиозном спортивном состязании.

В ближайшее время в г. Новокузнецке планируется провести еще ряд крупнейших шахматных состязаний, вот только несколько из них:

* чемпионат СФО по шахматам среди юниоров по классическим и быстрым шахматам (все возрастные категории – до 8, до 10, до 12, до 14, до 16, до 18 лет; и среди юношей и среди девушек. Таким образом – 22 турнира, включая турниры по быстрым шахматам. Причем одновременно будет проводиться сначала 12 турниров по классическим шахматам, а затем 10 турниров по быстрым шахматам);
* высшая лига чемпионата России среди мужчин и женщин (один из сильнейших шахматных форумов России и мира, раньше традиционно проводящийся в г.Сочи).

Помимо всего вышесказанного, в ходе работы структурного подразделения «Шахматный клуб» в сотрудничестве с городским МУДОД «СДЮСШОР по шахматам» сложилась хорошая традиция – отправлять в качестве поощрения лучших студентов спортсменов на международные соревнования за границу. Для реализации данной идеи привлекаются спонсорские средства, средства ГОУ ВПО «СибГИУ», средства городского МУДОД «СДЮСШОР по шахматам». География шахматных турниров, в которых участвовали спортсмены ГОУ ВПО «СибГИУ» действительно впечатляет:

* Малайзия. 2003 год. Лично-командный чемпионат Мира по шахматам среди студентов ВУЗов;
* Чехия. 2005 год. Европейский международный турнир;
* Хорватия 2006 год. Европейский международный турнир;
* Албания 2007 год. Европейский международный турнир;
* Греция 2008 год. Европейский международный турнир (планируется).

Кроме этого, команда ГОУ ВПО «СибГИУ» в полном составе регулярно выезжает на крупные спортивные состязания на территории России. Вот только некоторые традиционно посещаемые командой ГОУ ВПО «СибГИУ» по шахматам спортивные форумы:

* областная спартакиада ВУЗов Кузбасса (проводится попеременно в г.Новокузнецке и в г.Кемерово);
* чемпионат России по шахматам среди студентов ВУЗов (проводится каждый раз в разных городах. Несколько последних турниров проводилось в: г.Новокузнецке, г.Нижнем Тагиле, г.Уфа);
* традиционный международный шахматный фестиваль «Маэстро», г. Берцк;
* чемпионат СФО по шахматам среди мужчин и женщин (проводится каждый раз в разных городах. Несколько последних турниров проводилось в: г. Новокузнецке, г. Ангарске, г. Улан-Удэ, г.Томске).

В СП «ШК» занятия по дисциплине «Шахматы» ведутся весьма квалифицированными преподавателями. Двое преподавателей имеют высокие международные звания. Это мастер международного класса по шахматам Сорокина Т. Н. и мастер международного класса по шахматам Зайцев В. В. Преподаватели СП «ШК» активно участвуют в спортивной жизни города, области, страны и мира, показывая наглядный пример студентам ВУЗа. География соревнований, в которых участвовали и участвуют преподаватели ГОУ ВПО «СибГИУ» достаточно широка и насчитывает более десятка стран.

На основании анализа итогов работы СП «ШК» ГОУ ВПО «СибГИУ» был сделан вывод о необходимости создания специализированной информационной шахматной системы, обеспечивающей нормальный процесс проведения спортивных соревнований, освещения их для широкого круга наблюдателей, подготовки студентов-спортсменов к высшим спортивным достижениям.

1. Обзор шахматных систем прототипов

В настоящее время рынок шахматных информационных систем практически отсутствует. Поэтому отсутствует и массовое серийное производство подобных систем. Электронное шахматное оборудование во всем мире можно приобрести только у 3 организаций:

* Фирма «Шахком», генеральный директор Борис Ешан, Санкт-Петербург.
* Информационный шахматный центр российской шахматной федерации, Москва.
* Международная шахматная федерация ФИДЕ.

Электронное шахматное оборудование в свободной продаже практически отсутствует. Его необходимо заказывать индивидуально, указывая необходимую комплектацию и свои пожелания. Сроки поставки оборудования могут варьироваться от нескольких дней до нескольких месяцев, в зависимости от дальности поставки. Бесплатное сервисное обслуживания для этого оборудования не предусмотрено. Все эти обстоятельства сильно осложняют создание подобных систем в географически удаленных от центральной части России регионах.

До настоящего времени подавляющее большинство создаваемых информационных шахматных систем представляло собой миниатюрный вариант, состоящий из небольшого числа электронных шахматных досок (ЭШД), от 2 до 10 штук, соединенных или несоединенных между собой и подключаемых к компьютеру через стандартный сom-порт; одного или двух проекторов, с помощью которых собственно отображались шахматные партии. Если это крупнейшие международные шахматные форумы, обслуживаемые международной шахматной федерацией ФИДЕ или информационным шахматным центром России, то подобные соревнования освещаются и транслируются в сети Интернет на таких шахматных сайтах, как:

* www.chesspro.ru;
* www.fide.com;
* www.russiachess.org;
* www.chesscenter.com;
* www.64.ru;
* www.russiachess.org.

Однако не в интересах специалистов международной шахматной федерации ФИДЕ и информационного шахматного центра России предоставлять в пользование или продавать специализированное программное обеспечение (СПО) для трансляции соревнований в сети Интернет, так как подобное СПО является объектом их интеллектуальной собственности и используется ими для обслуживания турниров в случае приглашения сотрудников этой организации.

В качестве систем прототипов информационной системы «Шахматный клуб» (ИС «ШК») приведем две системы.

1. Программно-технический комплекс, обеспечивающий трансляцию шахматных соревнований для присутствующей аудитории, Нижний Тагил, 2005 год, чемпионат России по шахматам среди студентов

Данный программно-технический комплекс (ПТК) представляет собой набор из следующих технических и программных средств:

* + 4 ЭШД DGT;
  + 4 комплекта фигур для ЭШД;
  + 4 единицы электронных шахматных часов;
* персональный компьютер (ПК);
* 1 проектор;
* соединительные провода и переходники;
* программный пакет ChessAssistant9.1;
* драйвер для электронной шахматной доски dgtnix.

Электронные шахматные доски и электронные шахматные часы при помощи поставляемых в комплекте с ними переходников с com интерфейса на стандартный сетевой кабель, подключаются к магистральному проводу, который при помощи также поставляемого в комплекте переходника подключается к компьютеру через стандартный com-порт. В качестве магистрального провода используется кабель RJ45. Поступающая с электронных шахматных досок информация пакетами поступает на компьютер в программный продукт ChessAssistant9.1, причем процесс опроса стандартного com-порта инициируется драйвером dgtnix для электронных шахматных досок, а передача данных осуществляется с использованием протокола DGT. Поступающая информация сохраняется в отдельной базе данных программного продукта ChessAssistant9.1. Она включает в себя такие данные о шахматной партии, как:

* фамилия, имя, отчество игроков;
* результат партии;
* классификация дебюта шахматной партии;
* дата шахматной партии;
* город, страна где игралась шахматная партия;
* номер тура;
* название, ранг, статус соревнований;
* комментарии;
* международные звания и рейтинги игроков;
* непосредственно сам текст шахматной партии и т.д.

Далее информация, поступившая в базу данных программного продукта ChessAssistant9.1, выводится на проектор и отображается в зрительном зале для присутствующей аудитории.

1. Программно-технический комплекс, обеспечивающий трансляцию шахматных соревнований для присутствующей аудитории и в сети Интернет, Москва, 2007 год, международный шахматный турнир «Аэрофлот-опен 2007»

Данный ПТК создавался и обслуживался специалистами информационного шахматного центра российской шахматной федерации. Поэтому полная информация о данном ПТК отсутствует. В качестве программного обеспечения для данной системы использовалось СПО, являющееся объектом интеллектуальной собственности.

Данный ПТК представляет собой набор из следующих технических и программных средств:

* + 16 ЭШД DGT;
  + 16 комплектов фигур для ЭШД;
  + 16 единиц электронных шахматных часов;
* 2 ноутбука;
* 2 проектора;
* соединительные провода и переходники;
* СПО, обеспечивающее трансляцию соревнований для присутствующей аудитории и в сети Интернет;
* драйвер для электронной шахматной доски dgtnix.

Данный ПТК отличается от предыдущего бόльшим количеством комплектов ЭШД и единиц оборудования, а также возможностью трансляции спортивных соревнований в сети Интернет.

В представленной системе используется 2 магистральных провода, к каждому из которых подключается по 8 ЭШД. Магистральные провода, в качестве которых также используются кабели RJ45, подключаются к 2 ноутбукам, имеющим выход в сеть Интернет. Трансляция в сети интернет осуществляется с использованием СПО, являющегося объектом интеллектуальной собственности. Весь процесс соревнований также транслируется для присутствующей аудитории с использованием 2 проекторов. Схема подключения ЭШД подобна уже рассмотренной схеме подключения, технология передачи данных аналогична.

В общем виде работа рассмотренного ПТК представлена на рисунке 4.



Магистральный провод 1



ЭШД 1

ЭШД 2

ЭШД 8

…

Магистральный провод 2



ЭШД 1

ЭШД 2

…



ЭШД 8



Ноутбук 1

с СПО

Ноутбук 2

с СПО



Рисунок 4 – Иллюстрация работы ПТК, обеспечивающего трансляцию соревнований для присутствующей аудитории и в сети интернет

Стенд 2

Стенд 1

* 1. Анализ достоинств и недостатков

Рассмотренные шахматные системы прототипы обладают рядом достоинств и недостатков. Некоторые из них свойственны только этому классу систем, а некоторые характерны для любых подобных систем.

Достоинствами подобных систем являются:

* шахматные ИС способствуют повышению интереса общественности к данному интеллектуальному виду спорта, привлечению молодежи к спортивной жизни страны и мира;
* шахматные ИС способствуют привлечению спонсорских средств, что делает подобные системы экономически привлекательными;
* ИС подобного класса способствуют привлечению большого числа людей для участия в крупных спортивных состязаниях, что является большим плюсом для экономики города и региона, в котором проводятся подобные форумы;
* подобные ИС способствуют повышению общей культуры проведения спортивных состязаний, эргономики и судейской деятельности;
* информационные шахматные системы (ИШС) способны организовывать моментальный перенос играющихся партий в электронную форму и осуществлять транслирование спортивных состязаний в режиме он-лайн (on-line) по всему миру через сеть Интернет;
* ИШС являются незаменимым помощником в случае возникновения спорных ситуаций и конфликтов, так как позиции играющихся партий сохраняются даже в случае полного нарушения расстановки фигур на шахматной доске;
* ИШС позволяют преодолеть ограниченность вмещаемости помещения, в котором проводятся спортивные состязания;
* данные ИС позволяют издавать подробные отчеты об итогах спортивных состязаний практически на следующий день после окончания турнира;
* подобные системы делают процесс, происходящий на шахматной доске, предельно понятным даже для людей, не знакомых хорошо с принципами игры; это достигается подключением анализируемых игровых и комментирующих программных модулей, что предусмотрено возможностями подобных систем.

К недостаткам рассмотренных шахматных систем прототипов, а также большинства подобных систем, относятся:

* сложность создания подобных систем, обусловленная следующими факторами:
* отсутствие оборудования для подобных систем в свободной продаже;
* большой перечень требований, которые необходимо указать при заказе необходимого оборудования;
* сложности по доставке оборудования в регионы, удаленные от центральной части страны;
* отсутствие бесплатного сервисного обслуживания;
* сложность замены бракованного оборудования вследствие большой удаленности фирмы производителя и фирмы поставщика;
* очень ограниченное число эксплуатируемых ЭШД, что делает использование подобных систем узконаправленным на трансляцию только нескольких первых партий соревнования; подобное обстоятельство делает невозможным проведение полномасштабных транслируемых соревнований и затрудняет создание всевозможных печатных отчетов об итогах турниров;
* так как рынок подобных систем является монополизированным, то фирмы монополисты сами устанавливают стоимость лицензии на использование оборудования;
* отсутствует в свободной продаже СПО для подобных систем, поэтому приходится надеяться на работоспособность СПО, поставляемого в комплекте с ЭШД.

1.4 Техническое задание на создание ИС «ШК»

1.4.1 Общие сведения

а) Наименование системы – информационная система «Шахматный клуб» ГОУ ВПО «Сибирский государственный индустриальный университет».

б) Разработчик:

Студент

Группа АИС-03, ГОУ ВПО «СибГИУ»

Ширяев А.С.

Адрес: 654041, г. Новокузнецк, ул. Циолковского 32-22,

Телефон: (8-3843) 71-27-63

E-mail: jamert3@yandex.ru

Заказчики:

ГОУ ВПО «Сибирский государственный индустриальный университет»

МУДОД «СДЮСШОР по шахматам им. Б. А. Кустова»

Адреса заказчиков:

г. Новокузнецк, ул. Кирова 42,

г. Новокузнецк, ул. Орджоникидзе 23.

Телефоны заказчиков: (8-3843) 46-33-35, (8-3843) 45-36-98.

в) Система создается в связи с необходимостью оснащения СП ГОУ ВПО «СибГИУ» оборудованием для проведения учебного процесса и процесса проведения соревнований.

г) Плановые сроки начала работ по созданию системы: 01.08.07.

Плановые сроки окончания работ по созданию системы: 13.03.08.

д) Результаты работ по созданию системы представлены в приложении 1.

1.4.2 Назначение и цели создания системы

1.4.2.1 Назначение системы

ИС «ШК» предназначена для выполнения функций, обеспечивающих более эффективное проведение учебного процесса, а также соревнований, в помещении СП «ШК» ГОУ ВПО «СибГИУ», а именно:

* + - трансляция шахматных партий в режиме реального времени для присутствующей аудитории и в сети Интернет;
    - снижение трудозатрат на освещение турниров в средствах массовой информации;
    - предоставление исчерпывающей информации по каждому участнику соревнований по первому запросу;
    - организация моментального переноса записи шахматных партий в электронную форму;
    - отображение шахматных партий, турнирного положения, различного рода статистической информации на специализированных экранах и электронных стендах для широкой аудитории;
    - хранение данных о каждом участнике соревнований, турнирной и прочей информации;
    - оперативная обработка всей поступающей информации и выдача всех необходимых для соревнований данных;
    - организация выдачи в оперативном режиме информации по спорным вопросам, возникающим в ходе игрового процесса;
    - организация компьютеризированного процесса обучения игре в шахматы.

1.4.2.2 Цели создания системы

Цель складывается из следующих составляющих:

* + - информатизация процесса шахматной игры;
    - автоматизация:
    - процесса сборки,
    - отображения,
    - хранения,
    - обработки специализированной шахматной информации;
    - облегчение работы судейской коллегии в процессе соревнований;
    - автоматизация процесса обучения.

1.4.3 Характеристика объекта информатизации

Объектом информатизации является процесс шахматной игры. Правила шахматной игры и правила проведения шахматных соревнований регулируются кодексом международной шахматной федерации ФИДЕ.

Объектом информатизации также является учебный процесс на кафедре ФКЗиС ГОУ ВПО «СибГИУ» по дисциплине «Шахматы». Данный учебный процесс осуществляется в соответствии с учебным планом занятий.

Процесс проведения соревнований по шахматам осуществляется в соответствии с календарным планом проведения соревнований на текущий год.

Учебный процесс по дисциплине «Шахматы» проводится в помещении СП «ШК» ГОУ ВПО «СибГИУ». Помещение СП «ШК» ГОУ ВПО «СибГИУ» включает в себя следующие помещения: турнирный зал, судейский кабинет, раздевалка, помещение для инвентаря, комната анализа, кабинет директора. Шахматный клуб ГОУ ВПО «СибГИУ» оснащен следующим инвентарём: шахматные столы – 53 штук; комплекты шахмат – 40 штук; шахматные часы – 40 штук.

1.4.4 Требования к системе

1.4.4.1 Требования к системе в целом

а) Требования к структуре и функционированию системы

ИС «ШК» представлена пятью подсистемами:

* + - интегрирующая подсистема;
    - подсистема сбора информации;
    - подсистема обработки и хранения информации;
    - подсистема отображения информации;
    - обучающая подсистема.

Интегрирующая подсистема в общем виде представляет собой сетевую структуру, охватывающую все остальные подсистемы. Поэтому она является наиболее важной из всех остальных подсистем. Конфигурация оборудования подсистемы следующая:

* + - 1 сервер для хранения информации и обработки запросов;
    - соединительные провода;
    - 1 коммутатор D-Link DGS-1008D;
    - 6 рабочих компьютеров для работы с информацией, поступающей на сервер в оперативном режиме,
    - периферийные устройства:
    - многофункциональное устройство,
    - микрофон,
    - веб-камера и др.

Подсистема сбора информации в общем виде представляет собой все устройства, на которые поступает первичная и вторичная информация. Эта подсистема включает в себя следующие части:

* + - ЭШД в количестве 30 штук,
    - интернет и локальнаю сеть ГОУ ВПО «СибГИУ»,
    - фотоаппаратура и прочее оборудование.

ЭШД представляют собой комплекты шахматных фигур с досками и шахматными часами, соединённые при помощи соединительных проводов и переходников с компьютером. Передача данных, поступающих с шахматных досок, осуществляется при помощи протокола DGT. Процесс, происходящий во время игры за шахматной доской, полностью переносится в электронную форму при помощи СПО. Количество подключаемых ЭШД определяется потребностями соревновательного процесса, но общее их число 30 штук. Интернет и локальная сеть ГОУ ВПО «СибГИУ» – необходимые источники информации. Отсюда будет поступать информации самого различного рода: отзывы, комментарии, предложения, любительское фото и видео.

Подсистема обработки и хранения информации включает в себя:

* обработку и хранение шахматной информации;
* обработку и хранение прочей информации.

Подсистема обработки и хранения шахматной информации представляет собой набор СПО для проведения турниров, хранения и классифицирования шахматной информации, СПО для обслуживания электронных досок. В качестве СПО для проведения турниров используется набор из следующих программных продуктов:

* судейская программа SwissMaster 5.5 в количестве 2 лицензионных версий;
* многофункциональный пакет ChessAssistant9.1 в количестве 6 лицензионных версий
* многофункциональный пакет ChessBase9 в количестве 3 лицензионных версий.

Для обслуживания ЭШД используется программное обеспечение, поставляемое с ними в комплекте.

Подсистема обработки и хранения прочей информации представляет собой пакет стандартных программ для работы с графикой.

Подсистема отображения информации включает в себя:

* трансляцию соревнований в сети Интернет;
* трансляцию соревнований для присутствующей аудитории.

Так как используется СПО для ЭШД, поставляемое с ними в комплекте, то не возникает необходимости в дополнительном СПО для трансляции соревнований в сети Интернет по той причине, что означенное СПО уже содержит все для этого необходимое.

В качестве технических средств, обеспечивающих трансляцию соревнований для присутствующей аудитории, используется существующее оборудование, а именно – плазменные мониторы, установленные по университету, электронные стенды и проекторы: 1 электронный стенд и проектор в турнирном зале и 1 электронный стенд и проектор на территории ГОУ ВПО «СибГИУ».

Подсистема отображения информации, помимо выполнения своей главной функции – трансляции соревнований для присутствующей аудитории и в сети Интернет – может быть использована в рекламных целях.

Обучающая подсистема представляет собой набор программных средств для обучения шахматной игре, тренировки, игры в шахматы через сеть Интернет и по локальной сети ГОУ ВПО «СибГИУ». В качестве программного обеспечения используется:

* программный пакет ChessAssistant9.1 (имеющееся в наличии);
* программный пакет ChessBase9 (имеющееся в наличии);
* обучающая программа Studies2.0 (6 лицензионных версий);
* обучающая программа «Шахматные комбинации» (6 лицензионных версий);
* обучающая программа «Шахматная Стратегия» (6 лицензионных версий);
* обучающая программа «Шахматная школа» (6 лицензионных версий);
* программа для игры в шахматы по сети Интернет «ChessPlanet».

Более наглядно общая структура ИС «ШК» представлена в графической части дипломного проекта, на листе 3.

Структура ИС «Шахматный клуб» может быть дополнена новыми подсистемами, модулями и оборудованием, если на то возникнет необходимость. Численность единиц техники и программного обеспечения может меняться в зависимости от объемов финансирования.

б) Требования к численности и квалификации персонала системы и режиму его работы

К эксплуатации допускается персонал, изучивший инструкцию по эксплуатации.

Квалификация персонала должна обеспечивать эффективное функционирование системы во всех заданных режимах. Численность персонала должна быть достаточной для обеспечения выполнения всех функций системы.

Персонал должен быть подготовлен к выполнению своих обязанностей в соответствии с инструкциями организационного обеспечения.

Каждое лицо, входящее в состав персонала, должно уметь применять соответствующие информационные модели и работать с используемыми им техническими средствами и документацией, определяющей порядок его деятельности.

в) Показатели назначения

Время реализации системы на инициативы пользователя по запросам на выдачу информации, ввод информации, обновление не должно превышать 3 секунд.

Время запаздывания системы при переводе шахматной партии в электронную форму и отображении ее для широкого круга наблюдателей в процессе соревнований не должно превышать 0,5 секунд, если заранее не предусмотрена регламентированная задержка.

Структура комплекса технических средств, функциональные возможности и программное обеспечение должны быть реализованы с учетом возможности дальнейшего развития.

г) Требования к надежности

Технические средства системы должны работать в условиях реального времени в процессе проведения соревнований в течении всего рабочего дня с периодическим техническим обслуживанием.

Технические средства системы должны работать в оперативном режиме в ходе учебного процесса, что подразумевает обращение к средствам системы по мере возникновения необходимости.

Среднее время восстановления функций системы в процессе проведения соревнований не должно превышать 10 минут.

Среднее время восстановления функций системы в ходе учебного процесса не должно превышать 1 суток.

Время наработки на отказ технических и программных средств не должно превышать 5 лет.

Требования по надежности функционирования должны уточняться на последующих этапах проектирования.

д) Требования безопасности

Требования по безопасности при монтаже, наладке и эксплуатации оборудования должны соответствовать следующим документам:

* + - ГОСТ 12.1.004-85 «ССБТ. Пожарная безопасность. Общие правила».
    - СанПин 2.2.2.542-96 «Гигиенические требования к видеодисплейным терминалам вычислительных машин».
    - Нормы освещенности должны быть обеспечены в соответствии с СНиП 11-4-79.

е) Требования к эргономике и технической эстетике

Размещение технических средств, а также формы представления оперативной информации должны соответствовать ГОСТу 22.269-76.

Общие эргономические требования к микроклимату рабочих помещений персонала должны соответствовать ГОСТу 12.1.005-76.

Оборудование системы должно быть скомпоновано из серийно выпускаемых технических средств.

Способ и форма представления информации оперативному персоналу должна соответствовать требованиям эргономики ГОСТ 22.269-76.

ж) Требования по эксплуатации, техническому обслуживанию, ремонту и хранению

При разработке ИС «ШК» необходимо предусмотреть проведение технического обслуживания используемого оборудования на месте его эксплуатации. Техническое обслуживание проводится с целью предупреждения отказов в работе системы.

Периодичность проведения технического обслуживания зависит от вида и назначения устройств и устанавливается согласно техническому описанию и инструкции по эксплуатации для каждого элемента. Для быстрой замены вышедших из строя устройств и блоков необходимо предусмотреть запасные блоки, резервные каналы данных.

з) Требования по сохранности информации при авариях

Для обеспечения сохранности информации при резких изменениях и исчезновениях напряжения в питающей сети, отказах сервера, попаданиях в систему вируса, отказах компьютеров и других технических средств необходимо предусмотреть:

* + - автоматическое архивирование;
    - использование средств защиты информации;
    - анализ отказов и восстановление данных.

и) Требования к защите от влияния внешних воздействий

Защита технических средств ИС «ШК» от воздействия внешних электрических и магнитных полей, а также помех по цепям питания должна быть достаточной для надежного функционирования системы. В ИС «ШК» должна быть предусмотрена антивирусная защита и защита от несанкционированного доступа, достаточная для надежного, бесперебойного функционирования системы.

к) Требования к патентной чистоте

Разрабатываемая система не предназначена для экспорта, поэтому проверка используемых технических решений на патентную чистоту не требуется.

1.4.4.2 Требования к функциям системы

ИС «ШК» в соответствии с ГОСТ 24.104-85 должна выполнять:

* + - сбор, обработку и анализ информации (сигналов, сообщений, документов и т.п.) о состоянии объекта управления;
    - выработку управляющих воздействий (программ, планов и т.п.);
    - передачу управляющих воздействий на исполнение и их контроль;
    - реализацию и контроль выполнения управляющих воздействий;
    - обмен информацией (документами, сообщениями и т.п.) с взаимосвязанными автоматизированными системами.

Состав автоматизированных функций и функций по представлению информации ИС «ШК» должен обеспечивать возможность управления соответствующим объектом и представлением информации.

Состав автоматизированных функций, функций по представлению информации ИС «ШК» и степень информативности отображаемых данных должны быть технико-экономически и социально обоснованы с учетом необходимости освобождения персонала от выполнения повторяющихся действий и создания условий для использования его творческих способностей в процессе работы.

1.4.4.3 Требования к видам обеспечения

а) Требования к информационному обеспечению

Информационное обеспечение ИС «ШК» должно быть достаточным для выполнения всех автоматизированных и информативных функций данной системы.

Информационное обеспечение ИС «ШК» должно быть совместимо с информационным обеспечением систем, взаимодействующих с ней, по содержанию, системе кодирования, методам адресования, форматам данных и форме представления информации, получаемой и выдаваемой ИС «ШК».

б) Требования к программному обеспечению

Программное обеспечение ИС «ШК» должно быть достаточным для выполнения всех функций этой системы, реализуемых с применением средств вычислительной техники, а также иметь средства организации всех требуемых процессов обработки и представления данных, позволяющие своевременно выполнять все автоматизированные и информативные функции во всех регламентированных режимах функционирования ИС «ШК».

в) Требования к техническому обеспечению

Комплекс технических средств ИС «ШК» должен быть достаточным для выполнения всех автоматизированных функций и функций по представлению информации.

В комплексе технических средств ИС «ШК» должны использоваться технические средства серийного производства. При необходимости допускается применение технических средств единичного производства.

Технические средства ИС «ШК» должны быть размещены с соблюдением требований, содержащихся в технической, в том числе эксплуатационной, документации на них, и так, чтобы было удобно использовать их при функционировании ИС «ШК» и выполнять техническое обслуживание.

Технические средства ИС «ШК», используемые при ее взаимодействии с другими системами, должны быть совместимы по интерфейсам с соответствующими техническими средствами этих систем и используемых систем связи.

В ИС «ШК» должны быть использованы технические средства со сроком службы не менее пяти лет.

В ИС «ШК» должны быть использованы средства вычислительной техники, удовлетворяющие общим техническим требованиям по ГОСТу 22552-84.

г) Требования к лингвистическому обеспечению

Лингвистическое обеспечение системы должно быть достаточным для общения различных категорий пользователей в удобной для них форме со средствами ИС «ШК» и для осуществления процедур преобразования и машинного представления обрабатываемой информации.

В лингвистическом обеспечении ИС должны быть:

* + - предусмотрены языковые средства для описания любой используемой информации;
    - унифицированы используемые языковые средства;
    - стандартизированы описания однотипных элементов информации и записи синтаксических конструкций;
    - обеспечены удобство, однозначность и устойчивость общения пользователей со средствами автоматизации и представления информации;
    - предусмотрены средства исправления ошибок, возникающих при общении пользователей с техническими средствами ИС.

Лингвистическое обеспечение ИС «ШК» должно быть отражено в документации организационного обеспечения системы в виде правил общения пользователей с техническими средствами ИС «ШК» во всех режимах функционирования системы.

д) Требования к организационному обеспечению

Организационное обеспечение ИС «ШК» должно быть достаточным для эффективного выполнения персоналом возложенных на него обязанностей при осуществлении автоматизированных и связанных с ними неавтоматизированных функций системы, а также функций по представлению информации.

Организационная структура ИС «ШК» должна позволять выполнять все функции с учетом их распределения по уровням управления.

Инструкции организационного обеспечения ИС «ШК» должны определять действия, необходимые для выполнения каждой автоматизированной функции или функции по представлению информации, во всех режимах функционирования ИС «ШК», с учетом заданных требований по безошибочности и быстродействию реализации персоналом своих функциональных обязанностей, а также содержать конкретные указания о действиях в случае возникновения аварийных ситуаций или нарушения нормальных условий функционирования ИС «ШК».

1.4.5 Состав и содержание работ по созданию системы

Состав и содержание работ по созданию системы представлены в приложении 1.

1.4.6 Порядок контроля и приемки системы

1.4.6.1 Предварительные испытания системы

Предварительные испытания системы проводят для определения ее работоспособности и решения вопроса о возможности приемки ИС «ШК» в опытную эксплуатацию.

Предварительные испытания системы организует заказчик и проводит разработчик и заказчик совместно. Содержание программы испытаний должно соответствовать ГОСТу 24.208-80.

В «Протоколе испытаний», составленном по результатам предварительных испытаний системы, приводят заключение о возможности приемки системы в опытную эксплуатацию, а также перечень необходимых доработок.

1.4.6.2 Опытная эксплуатация

Результаты приемки ИС «ШК» в опытную эксплуатацию оформляют «Актом приемки в опытную эксплуатацию», составленным на основании «Протокола испытаний» комиссией, проводившей предварительные испытания системы.

Продолжительность опытной эксплуатации системы определяют по срокам, необходимым для проверки правильности функционирования системы при выполнении каждой автоматизированной функции, функции по представлению информации и готовности персонала к участию в выполнении всех автоматизированных функций и функций по представлению информации ИС «ШК».

Во время опытной эксплуатации системы ведется рабочий журнал, в который заносятся сведения: о продолжительности функционирования системы, о результатах наблюдения за правильностью функционирования системы, об отказах, сбоях, аварийных ситуациях, проводимых корректировках технической документации.

По результатам опытной эксплуатации системы составляют акт о завершении работ по проверке системы в режиме опытной эксплуатации.

1.4.6.3 Приемочные испытания системы

Приемочные испытания системы проводят для определения ее соответствия техническому заданию, требованиям стандарта и определения возможности ввода ИС «ШК» в действие.

Приемочной комиссии заказчик и разработчик предъявляют следующую документацию:

* + - техническое задание на систему;
    - проект программы приемочных испытаний;
    - протокол предварительных испытаний;
    - акт приемки системы в опытную эксплуатацию;
    - рабочие журналы опытной эксплуатации системы;
    - акт о завершении работ по проверке системы в режиме опытной эксплуатации;
    - техническая документация на систему.

По результатам приемочных испытаний приемочная комиссия составляет протокол испытаний и акт о вводе системы в действие.

1.4.7 Требования к составу и содержанию работ по подготовке объекта информатизации к вводу системы в действие

Одновременно с разработкой системы должно быть обеспечено проведение следующих мероприятий:

* + - определить конкретных лиц, ответственных от СП «ШК» ГОУ ВПО «СибГИУ» за подготовку системы к эксплуатации;
    - обозначить места установки технических средств и обеспечить их сохранность;
    - организовать подготовку персонала для работы с системой.

1.4.8 Требования к документированию

Эксплуатационная документация на разрабатываемую систему должна быть достаточной для ввода комплекса в действие эффективной работы при его эксплуатации.

Документация должна содержать сведения, необходимые для быстрого и качественного освоения и правильной эксплуатации средств автоматизации и представления информации системы, содержать указания по действиям персонала в аварийных ситуациях или при нарушениях нормальных условий функционирования комплекса, не содержать сведений, допускающих неоднозначное толкование.

Комплект документации по системе приведен в приложении 2.

1.4.9 Заключение

В ходе разработки ИС «ШК» было установлено, что в комплекте с ЭШД было поставлено неработоспособное программное обеспечение. Но так как необходимо было в срочном порядке закончить работы по созданию системы, а на выяснение причин и дополнительную поставку ушло бы значительное время вследствие того, что оборудование поставлялось из Нидерландов, было принято решение в срочном порядке пригласить специалистов из информационного шахматного центра российской шахматной федерации, владеющих собственным программным обеспечением для ЭШД, на время проведения соревнований, что повлекло за собой большие расходы.

После разработки ИС «ШК», ее приемки и проведения опытной эксплуатации руководством ГОУ ВПО «СибГИУ» в тесном сотрудничестве с администрацией г. Новокузнецка и администрацией Кемеровской области было принято решение перенести ИС «ШК» в помещение культурного центра «Западносибирского металлургического комбината» («ЗСМК») на время проведения Всемирной шахматной универсиады, которая проходила в г. Новокузнецке с 3 марта по 11 марта 2008 года. Перенесение ИС «ШК» в помещение культурного центра «ЗСМК» не составило больших трудностей, так как все блоки системы легко отсоединяемы и транспортируемы.

Вследствие того обстоятельства, что работоспособное программное обеспечения для ЭШД так и не было поставлено, а специалисты из информационного шахматного центра отказались от коммерческой продажи СПО, являющегося их интеллектуальной собственностью, было принято решение о самостоятельной разработке подобного СПО. Данное направление является приоритетным для написания дипломной работы студентом Ширяевым А.С.

2. Специальная часть

2.1 Выбор основных методологий разработки программного обеспечения

Проект – это уникальный процесс, в ходе выполнения которого получают уникальный продукт. Таким образом, для разработки продукта в проекте, скорее всего, должен применяться уникальный процесс. Вместо создания каждого проекта «с нуля», руководитель проекта может воспользоваться обобщенной, проверенной на практике методикой, адаптировав ее для конкретного проекта. Как правило, всегда есть возможность выбора среди нескольких «начальных» жизненных циклов.

Выбор и адаптация жизненного цикла разработки проекта оказывает влияние на методики разработки продукта, навыки менеджмента проектов и навыки менеджмента персонала. Что касается методов разработки продукта, руководитель проекта должен, прежде всего, иметь представление о стандартах процесса, уметь оценить их применимость по отношению к данному проекту, оценить альтернативные процессы и при необходимости адаптировать процесс жизненного цикла к текущим потребностям. На выбор методов и инструментальных средств также может оказывать влияние выбор жизненного цикла.

Модель жизненного цикла разработки программного обеспечения (ПО) является единственным видом процесса, в котором представлен порядок его осуществления. Модель жизненного цикла разработки ПО (Software Life Cycle Model, SLCM) схематически объясняет, каким образом будут выполняться действия по разработке программного продукта, посредством описания «последовательности» этих действий. Такая последовательность может быть или не быть линейной, поскольку фазы могут следовать друг за другом, повторяться или происходить одновременно. На рисунке 5 представлена простая обобщенная схема процесса.

Жизненный цикл

Процесс

Фаза

План

Спецификация

Разработка

Эксплуатация

Действие

Разработка

проекта

Код

Тестирование модуля

Тестирование системы

Планы

тестирования,

результаты

тестирования

Планы

тестирования,

результаты

тестирования

Показатель LOC

Рисунок 5 – Обобщенная схема процесса

Модель SLCM – это схема (или основа), используемая разработчиком ПО для определения повторяющегося процесса при создании программного продукта. Она определяет точные инструкции, которые разработчик может использовать для создания только высококачественных программных систем. Понятие жизненного цикла ПО относится ко всем программным проектам, причем независимо от их размеров.

Жизненный цикл – это своего рода «карта-путеводитель» для всех участников проекта, которая помогает им понять, не выходят ли они за определенные для них границы. Для управления программным проектом возникает необходимость в некотором роде карты для планирования действий и хронологий их выполнения.

В стандарт были включены описания причин, объясняющих необходимость выполнения стандартизированного процесса. Этот стандарт помогает достичь следующих целей.

* Улучшение и обеспечение качества:
  + с помощью стандартизированной процедуры можно наилучшим образом гарантировать завершенность результатов, которые необходимо предоставить;
  + определение промежуточных результатов обеспечивает возможность ускорить выполнение оценочных процедур;
  + контекст однородных продуктов облегчает их восприятие, а также работу с процедурами оценки.
* Возможность проверки затрат на выполнение полного жизненного цикла:
  + упрощает процесс создания стандартов разработки для определенного проекта и его оценка;
  + стандартизированные процедуры повышают степень «прозрачности» операций по определению затрат и позволяют более эффективно распознавать возможные риски, связанные с затратами;
  + одинаковые стандарты уменьшают риск возникновения разногласий между клиентом и разработчиком, а также между главным разработчиком и субподрядчиком;
  + в случае применения стандартизированной процедуры становятся «прозрачными» универсальные подходы к методам решения, а следовательно, их можно использовать повторно;
  + нежелательный ход процесса разработки, возможно, выявить на ранней стадии;
  + уменьшаются затраты на подготовку персонала.
* Улучшается обмен информацией между различными сторонами, участвующими в процессе разработки; происходит снижение зависимости клиента от подрядчика:
  + использование определенных терминов уменьшает разногласия, возникающие между всеми задействованными в проекте сторонами;
  + пользователь, покупатель и разработчик получают поддержку при формулировании своих требований, а также при описании своих ролей или полученных результатов;
  + промежуточные и окончательные результаты стандартизируются таким образом, что другие задействованные в проекте стороны или персонал других компаний могут в случае необходимости подключиться к процессу разработки, не прилагая при этом больших дополнительных усилий.

«Каркасом» процесса разработки ПО служит модель зрелости функциональных возможностей (Capability Maturity Model, CMM). Она основана на практических действиях, отображает лучшие результаты и определяет потребности индивидов, работающих над усовершенствованием процесса разработки ПО и выполняющих оценочный анализ этого процесса. Модель СММ представляет собой схему, по которой этапы разработки соответствуют пяти уровням развития функциональных возможностей, на основе которых осуществляется непрерывное усовершенствование процесса разработки.

Исходный. Процесс разработки ПО можно охарактеризовать как специальный, подобранный для определенного случая процесс, а иногда и как хаотический. Определить можно лишь небольшое количество процессов, и успех зависит от приложенных усилий и предпринимаемых решительных действий.

Повторяющийся. Основные процессы управления проектом создаются для того, чтобы отслеживать затраты, график работы и функциональные возможности. Здесь соблюдается необходимый порядок выполнения процесса, предназначенный для повторения достижений, полученных ранее при выполнении подобных проектов.

Определенный. Во всех проектах используется испытанная, адаптированная версия стандартного процесса разработки ПО данной организации.

Управляемый. Собираются детальные показатели процесса разработки ПО и качественные характеристики продукта. Управление процессом разработки программных продуктов осуществляется на количественном уровне.

Уровень оптимизации. Непрерывное усовершенствование процесса разработки достигается с помощью количественной обратной связи, достигаемой при осуществлении самого процесса, а также на базе новаторских идей и технологий.

Определение процесса включает в себя разработку и сопровождение стандартного процесса разработки определенной организации, а также относящиеся к нему ценные свойства процесса, такие как описательные характеристики жизненных циклов разработки ПО, руководящие принципы адаптации процесса и его критерии.

Цель определения организационной структуры процесса заключается в разработке и сопровождении стандартного процесса разработки ПО для данной организации.

Действия, формулирующие процесс построения организационной структуры, включают документирование и сопровождение описательных характеристик жизненных циклов разработки ПО, которые одобрены для использования в проектах. Руководящие принципы и критерии адаптации описывают выбор и адаптацию жизненного цикла разработки ПО и характеристик данного проекта.

Наиболее известными и широко используемыми жизненными циклами разработки ПО можно назвать следующие: каскадная модель и экстремальная модель. Есть и ряд других методологий, но в ходе реализации данного дипломного проекта они не исследовались. Рассмотрим указанные модели подробнее.

2.1.1 Каскадная методология

Классическая каскадная модель, несмотря на полученную в последнее время негативную оценку, исправно служила специалистам по программному инжинирингу многие годы. Понимание ее сильных сторон и недостатков улучшает оценочный анализ других, зачастую более эффективных моделей жизненного цикла, основанных на данной модели.

В первые годы практики программирования сначала записывался программный код, а затем происходила его отладка. Общепринятым считалось правило начинать работу не с разработки плана, а с общего ознакомления с продуктом. Без лишних формальностей можно было спроектировать, закодировать, отладить и протестировать ПО еще до того, как оно будет готово к выпуску. Это напоминало процесс, изображенный на рисунке 6.

Выходы

Входы

Кодирование и тестирование

Делать, пока не будет сделано

###### Рисунок 6 – Модель процесса "делать, пока, не будет сделано”

В структуре такого процесса есть несколько "неправильностей" (или недостатков). Во-первых, поскольку изначально не существовало официального проекта или анализа, невозможно было узнать о моменте завершения процесса. Также отсутствовал способ определения соответствия требованиям относительно достижения качества.

В 1970 году каскадная модель была впервые определена как альтернативный вариант метода разработки ПО по принципу кодирование-устранение ошибок, который был широко распространен в то время. Это была первая модель, которая формализовала структуру этапов разработки ПО, придавая особое значение исходным требованиям и проектированию, а также созданию документации на ранних этапах процесса разработки.

Начальный этап выполнения каскадной модели показан в левой верхней части рисунка 7. Продолжение процесса выполнения реализуется с помощью упорядоченной последовательности шагов. В модели предусмотрено, что каждая последующая фаза начинается лишь тогда, когда полностью завершено выполнение предыдущей фазы. Каждая фаза имеет определенные критерии входа и выхода: входные и выходные данные.

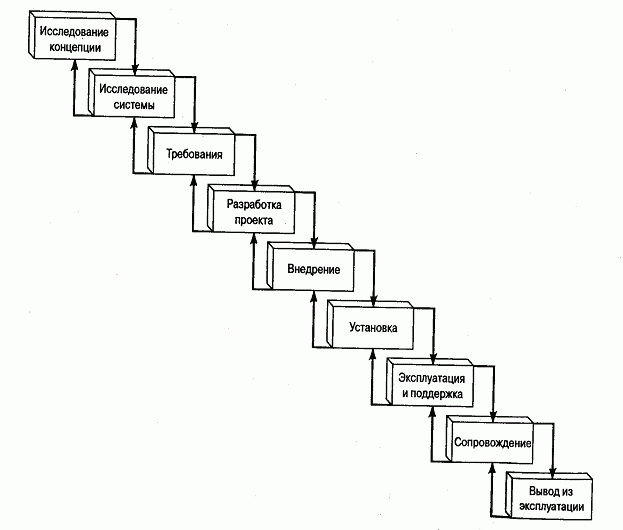


Рисунок 7 – Классическая каскадная модель с обратной связью

В результате выполнения генерируются внутренние или внешние данные проекта, включай документацию и ПО. Документы по анализу требований впоследствии передаются системным специалистам, которые в свою очередь передают их разработчикам программных систем более высокого уровня. Программисты передают детальные технические характеристики программистам, которые уже представляют готовый код тестерам.

Переход от одной фазы к другой осуществляется посредством формального обзора. Таким образом, клиент получает общее представление о процессе разработки, кроме того происходит проверка качества программного продукта. Как правило, прохождение стадии обзора указывает на договоренность между командой разработчиков и клиентом о том, что текущая фаза завершена и можно перейти к выполнению следующей фазы. Окончание фазы удобно принимать за стадию в процессе выполнения проекта.

В результате завершения определенных фаз формируется базовая линия, которая в данной точке "замораживает" продукты разработки. Если возникает потребность в их изменении, тогда для внесения изменений используется формальный процесс изменений.

В критических точках каскадной модели формируются базовые линии, последняя из которых является базовой линией продукта. После формирования заключительной базовой линии производится обзор приемки.

Попытки оптимизации каскадной модели привели к возникновению других циклов разработки ПО. Прототипирование программ позволяет обеспечить полное понимание требований, в то время как инкрементные и спиральные модели позволяют повторно возвращаться к фазам, соотнесенным с классической каскадной моделью, прежде чем полученный продукт будет признан окончательным.

Отличительным свойством каскадной модели можно назвать то, что она представляет собой формальный метод, разновидность разработки "сверху вниз", она состоит из независимых фаз, выполняемых последовательно, и подвержена частому обзору.

### а) Краткое описание фаз каскадной модели

Приведенная ниже характеристика представляет собой краткое описание каждой фазы каскадной модели (включая фазы интеграции):

* исследование концепции — происходит исследование требований на системном уровне с целью определения возможности реализации концепции;
* процесс системного распределения — может быть пропущен для систем по разработке исключительно ПО. Для систем, в которых необходима разработка как аппаратного, так и программного обеспечения, требуемые функции применяются к ПО и оборудованию в соответствии с общей архитектурой системы;
* процесс определения требований — определяются программные требования для информационной предметной области системы, предназначение, линии поведения, производительность и интерфейсы. (В случае необходимости в процесс также включено функциональное распределение системных требований к аппаратному и программному обеспечению.);
* процесс разработки проекта— разрабатывается и формулируется логически последовательная техническая характеристика программной системы, включая структуры данных, архитектуру ПО, интерфейсные представления и процессуальную (алгоритмическую) детализацию;
* процесс реализации — в результате его выполнения эскизное описание ПО превращается в полноценный программный продукт. При этом создается исходный код, база данных и документация, которые лежат в основе физического преобразования проекта. Если программный продукт представляет собой приобретенный пакет прикладных программ, основными действиями по его реализации будут являться установка и тестирование пакета программ. Если программный продукт разрабатывается на заказ, основными действиями являются программирование и код-тестирование;
* процесс установки — включает установку ПО, его проверку и официальную приемку заказчиком для операционной среды;
* процесс эксплуатации и поддержки - подразумевает запуск пользователем системы и текущее обеспечение, включая предоставление технической помощи, обсуждение возникших вопросов с пользователем, регистрацию запросов пользователя на модернизацию и внесение изменений, а также корректирование или устранение ошибок;
* процесс сопровождения— связан с разрешением программных ошибок, неисправностей, сбоев, модернизацией и внесением изменений, генерируемых процессом поддержки. Состоит из итераций разработки и предполагает обратную связь по предоставлению информации об аномалиях;
* процесс вывода из эксплуатации — вывод существующей системы из ее активного использования либо путем прекращения ее работы, либо благодаря ее замене новой системой или модернизированной версией существующей системы;
* интегральные задачи — включают начало работы над проектом, мониторинг проекта и его управление, управление качеством, верификацию и аттестацию, менеджмент конфигурации, разработку документации и профессиональную подготовку на протяжении всего жизненного цикла.

### б) Преимущества каскадной модели

Нетрудно заметить, что каскадная модель имеет множество преимуществ, если ее использовать в проекте, для которого она достаточно приемлема. Ниже перечислены эти преимущества:

* модель хорошо известна потребителям, не имеющим отношения к разработке и эксплуатации программ, и конечным пользователям (она часто используется другими организациями для отслеживания проектов, не связанных с разработкой ПО);
* она упорядоченее справляется со сложностями и хорошо срабатывает для тех проектов, которые достаточно понятны, но все же трудно разрешимы;
* она весьма доступна для понимания, так как преследуется простая цель — выполнить необходимые действия;
* она проста и удобна в применении, так как процесс разработки выполняется поэтапно;
* ее структурой может руководствоваться даже слабо подготовленный в техническом плане или неопытный персонал;
* она отличается стабильностью требований;
* она представляет собой шаблон, в который можно поместить методы для выполнения анализа, проектирования, кодирования, тестирования и обеспечения;
* она хорошо срабатывает тогда, когда требования к качеству доминируют над требованиями к затратам и графику выполнения проекта;
* она способствует осуществлению строгого контроля менеджмента проекта;
* при правильном использовании модели дефекты можно обнаружить на более ранних этапах, когда их устранение еще не требует относительно больших затрат;
* она облегчает работу руководителю проекта по составлению плана и Комплектации команды разработчиков;
* она позволяет участникам проекта, завершившим действия на выполняемой ими фазе, принять участие в реализации других проектов;
* она определяет процедуры по контролю за качеством. Каждые полученные данные подвергаются обзору. Такая процедура используется командой разработчиков для определения качества системы;
* стадии модели довольно хорошо определены и понятны;
* ход выполнения проекта легко проследить с помощью использования временной шкалы (или диаграммы Ганта), поскольку момент завершения каждой фазы используется в качестве стадии.

### в) Недостатки каскадной модели

Но при использовании каскадной модели для проекта, который трудно назвать подходящим для нее, проявляются следующими недостатки:

* в основе модели лежит последовательная линейная структура, в результате чего каждая попытка вернуться на одну или две фазы назад, чтобы исправить какую-либо проблему или недостаток, приведет к значительному увеличению затрат и сбою в графике;
* она не может предотвратить возникновение итераций между фазами, которые так часто встречаются при разработке ПО, поскольку сама модель создается согласно стандартному циклу аппаратного инжиниринга;
* она не отображает основное свойство разработки ПО, направленное на разрешение задач. Отдельные фазы строго связаны с определенными действиями, что отличается от реальной работы персонала или коллективов;
* она может создать ошибочное впечатление о работе над проектом. Выражение типа "35 процентов выполнено" — не несет никакого смысла и не является показателем для руководительа проекта;
* интеграция всех полученных результатов происходит внезапно в завершающей стадии работы модели. В результате такого единичного прохода через весь процесс, связанные с интегрированием проблемы, как правило, дают о себе знать слишком поздно. Следовательно, проявятся не обнаруженные ранее ошибки или конструктивные недостатки, повысить степень риска при небольшом задаче времени на восстановление продукта;
* у клиента едва ли есть возможность ознакомиться с системой заранее, это происходит лишь в самом конце жизненного цикла. Клиент не имеет возможности воспользоваться доступными промежуточными результатами, и отзывы пользователей нельзя передать обратно разработчикам. Поскольку готовый продукт не доступен вплоть до окончания процесса, пользователь принимает участие в процессе разработки только в самом начале — при сборе требований, и в конце — во время приемочных испытаний;
* пользователи не могут убедиться в качестве разработанного продукта до окончания всего процесса разработки. Они не имеют возможности оценить качество, если нельзя увидеть готовый продукт разработки;
* у пользователя нет возможности постепенно привыкнуть к системе. Процесс обучения происходит в конце жизненного цикла, когда ПО уже запущено в эксплуатацию;
* проект можно выполнить, применив упорядоченную каскадную модель, и привести его в соответствие с письменными требованиями, что, однако, не гарантирует его запуска в эксплуатацию;
* каждая фаза является предпосылкой для выполнения последующих действий, что превращает такой метод в рискованный выбор для систем, не имеющих аналогов, так как он не поддается гибкому моделированию;
* для каждой фазы создаются результативные данные, которые по его завершению считаются замороженными. Это означает, что они не должны изменяться на следующих этапах жизненного цикла продукта. Если элемент результативных данных какого-либо этапа изменяется (что встречается весьма часто), на проект окажет негативное влияние изменение графика, поскольку ни модель, ни план не были рассчитаны на внесение и разрешение изменения на более поздних этапах жизненного цикла;
* все требования должны быть известны в начале жизненного цикла, но клиенты редко могут сформулировать все четко заданные требования на этот момент разработки. Модель не рассчитана на динамические изменения в требованиях на протяжении всего жизненного цикла, так как получаемые данные "замораживаются". Использование модели может повлечь за собой значительные затраты, если требования в недостаточной мере известны или подвержены динамическим изменениям во время протекания жизненного цикла;
* возникает необходимость в жестком управлении и контроле, поскольку в модели не предусмотрена возможность модификации требований;
* модель основана на документации, а значит, количество документов может быть избыточным;
* весь программный продукт разрабатывается за один раз. Нет возможности разбить систему на части. В результате взятых разработчиками обязательств разработать целую систему за один раз могут возникнуть проблемы с финансированием проекта. Происходит распределение больших денежных средств, а сама модель едва ли позволяет повторно распределить средства, не разрушив при этом проект в процессе его выполнения;
* отсутствует возможность учесть переделку и итерации за рамками проекта.

### г) Область применения каскадной модели

Из-за недостатков каскадной модели ее применение необходимо ограничить ситуациями, в которых требования и их реализация максимально четко определены и понятны.

Каскадная модель хорошо функционирует при ее применении в циклах разработки программного продукта, в которых используется неизменяемое определение продукта и вполне понятные технические методики.

Если компания имеет опыт построения определенного рода системы — автоматизированного бухгалтерского учета, начисления зарплаты, ревизии, компиляции, производства, — тогда в проекте, ориентированном на построение еще одного продукта такого же типа, возможно, даже основанного на существующих разработках, можно эффективно использовать каскадную модель. Другим примером надлежащего применения модели может служить создание и выпуск новой версии уже существующего продукта, если вносимые изменения вполне определены и управляемы. Перенос уже существующего продукта на новую платформу часто приводят в качестве идеального примера использования каскадной модели в проекте.

При всей справедливости критики этой модели все же следует признать, что модифицированная версия каскадной модели является в значительной степени менее жесткой, чем ее первоначальная форма. Здесь включаются итерации между фазами, параллельные фазы и менеджмент изменений. Обратные стрелки предполагают возможность существования итераций между действиями в рамках фаз. Чтобы отобразить согласованность между этапами, их объединяют прямоугольниками или под прямоугольниками перечисляют выполняемые на данных этапах действия, чтобы продемонстрировать согласованность между ними. Несмотря на то, что модифицированная каскадная модель является значительно более гибкой, чем классическая модель, она все же не является наилучшим выбором для выполнения проектов по ускоренной разработке.

Каскадные модели на протяжении всего времени их существования используются при выполнении больших проектов, в которых задействовано несколько больших команд разработчиков.

2.1.2 Экстремальная методология

Отцом-идеологом экстремального программирования (XP) считают Кента Бека (Kent Beck). XP является достаточно молодой методологией, оценки которой весьма противоречивы — от восторженных до резко негативных. Основными принципами являются:

* Простота решений (simplicity).
* Интенсивная разработка малыми группами (не больше 10 человек), активное общение в группе и между группами (communication).
* Обратная связь с клиентом (feedback), который фактически вовлечен в процесс разработки.
* Достаточная степень смелости (courage) и желание идти на риск.

Первый фактор ускорения разработки — итеративность: разработка ведется короткими итерациями при наличии активной взаимосвязи с заказчиком. XP — это итеративный процесс разработки, который сам по себе не является революционным. Итерации как таковые предлагается делать короткими, рекомендуемая длительность — 2-3 недели и не более 1 месяца. За одну итерацию группа программистов обязана реализовать несколько свойств системы, каждое из которых описывается в пользовательской истории (user story). Пользовательские истории в данном случае являются начальной информацией, на основании которой создается модуль. Пользовательские истории отличаются от прецедентов (use case): пользовательская история коротка — 1-2 абзаца, тогда как прецеденты обычно пишут достаточно подробными, с основным и альтернативными потоками — таким образом, получается примерно страница плюс схема (наиболее распространенная формализация в настоящее время предложена в UML); истории пользователей пишутся самими пользователями (которые в XP являются частью команды) в отличие от прецедентов, которые обычно пишет системный аналитик. Отсутствие формализации описания входных данных проекта в XP стремятся компенсировать посредством активного включения в процесс разработки заказчика как полноправного члена команды и за счет наличия постоянного контакта с заказчиком (активное общение и непрерывная поддержка обратной связи). В данном случае extreme — это степень привлечения заказчика к программистской кухне, что обусловлено стремлением сжать сроки разработки за счет коммуникации и обратной связи.

Второй фактор ускорения разработки продукта — наличие малых групп и парное программирование (когда два программиста вместе создают код на одном общем рабочем месте). Все это нацелено на достижение высокого уровня общения в группе, а также на как можно более раннее обнаружение проблем (как ошибок, так и срыва сроков). Парное программирование преследует цель стабилизации проекта, так как при данной методологии высок риск потери кода по причине ухода программиста, не выдержавшего интенсивного графика работы. В этом случае второй программист из пары играет роль «наследника» кода (что в классических методиках реализуется в технической документации). Немаловажно и то, как именно распределены группы в рабочем пространстве — в XP используется открытое рабочее пространство, которое предполагает быстрый и свободный доступ всех ко всем; как правило, рабочее пространство строится на основе круга.

Третий фактор ускорения разработки процесса — принятие первого наипростейшего рабочего решения. В данном случае экстремальность метода связана с высокой степенью риска решения, обусловленного поверхностностью анализа и жестким временным графиком. Реализуется минимальный набор главных функций системы на первой и каждой последующей итерации; функциональность расширяется на каждой итерации.

### а) Практики экстремальной методологии

Обычно XP характеризуют набором из 12 действий (практик), которые необходимо выполнять для достижения хорошего результата. Практики XP не определяют сам процесс XP, но XP определяет эти практики — то есть выполнение практик не гарантирует результата. Ни одна из практик не является принципиально новой, но в XP они собраны вместе.

* Планирование процесса (planning game). Вся команда собирается вместе, принимается коллективное решение о том, какие свойства системы будут реализованы в ближайшей итерации. Набор свойств определяется пользовательскими историями. XP-трудоемкость каждого свойства определяется самими программистами.
* Тесное взаимодействие с заказчиком (feed-back, on-site customer). Заказчик должен быть членом XP-команды (on-site customer). Он пишет пользовательские истории, выбирает истории, которые будут реализованы в конкретной итерации, и отвечает на вопросы, касающиеся бизнеса. Заказчик должен быть экспертом в автоматизируемой предметной области. Необходимо постоянное наличие обратной связи с заказчиком (feed-back).
* Метафора системы (system metaphor). Хорошая метафора системы означает простоту именования классов и переменных. В реальной жизни поиск метафоры — крайне сложное занятие; найти хорошую метафору непросто. В любом случае команда должна иметь единые правила именования.
* Простая архитектура (simple design). Любое свойство системы должно быть реализовано как можно проще. Программисты в XP-команде работают под девизом: «Ничего лишнего!». Принимается первое наипростейшее работающее решение, реализуется необходимый уровень функциональности на данный момент. Тем самым экономится время программиста.
* Стандарты кодирования (coding conventions). Стандарты кодирования нужны для обеспечения других практик: коллективного владения кодом, парного программирования и рефакторинга. Без единого стандарта выполнять эти практики как минимум сложнее, а в реальности вообще невозможно: группа будет работать в режиме постоянной нехватки времени. Детальные стандарты не требуются, необходимо стандартизировать только важные вещи. Определение наиболее важных объектов стандартизации в XP субъективно.
* Рефакторинг (refactoring). Рефакторинг — это оптимизация существующего кода в сторону упрощения, что предусматривает постоянную работу по упрощению кода. Сохраняя код прозрачным и определяя его элементы всего один раз, программисты сокращают число ошибок, которые впоследствии придется устранять. При реализации каждого нового свойства системы программист должен подумать над тем, можно ли упростить существующий код и как это поможет реализовать новое свойство. Кроме того, нельзя совмещать рефакторинг с дизайном: если создается новый код, рефакторинг надо отложить.
* Парное программирование (pair programming) — одна из самых известных XP-практик. Все программисты должны работать в парах: один пишет код, другой смотрит. Таким образом, необходимо размещать группу программистов в одном месте, что легче всего сделать на территории заказчика (все необходимые члены команды географически находятся в одном месте); XP наиболее успешно работает в нераспределенных коллективах программистов и пользователей.
* 40-часовая рабочая неделя. Программист не должен работать более 8 часов в день. Необходимость сверхурочной работы (overtime) — это четкий индикатор проблемы на данном конкретном направлении разработки; к тому же заказчик не платит за сверхурочную работу в XP. Поиск причин сверхурочной работы и их скорейшее устранение — одно из основных правил.
* Коллективное владение кодом (collective code ownership). Каждый программист в коллективе XP должен иметь доступ к коду любой части системы и вносить изменения в любой код. Обязательное правило: если программист внес изменения и система после этого работает некорректно, то именно этот программист должен исправить ошибки. В противном случае работа системы уподобится тотальному хаосу.
* Частая смена версий (small releases). Минимальная итерация — один день, максимальная — месяц; чем чаще осуществляются релизы, тем больше недостатков системы будет выявлено. Первые релизы помогают выявить недостатки на самых ранних стадиях, далее функциональность системы расширяется (на основании тех же пользовательских историй). Поскольку пользователь включается в процесс разработки начиная с первого релиза, то он оценивает систему и выдает пользовательскую историю плюс feedback. На основании этого определяется следующая итерация: каким будет новый релиз. В XP все направлено на обеспечение непрерывной обратной связи с пользователями.
* Непрерывная интеграция (continuous integration). Интеграция новых частей системы должна происходить как можно чаще, как минимум раз в несколько часов. Основное правило интеграции следующее: интеграцию можно производить, если все тесты проходят успешно. Если тесты не проходят, то программист должен либо внести исправления и тогда интегрировать составные части системы, либо вообще не интегрировать их. Правило это жесткое и однозначное — если в созданной части системы имеется хотя бы одна ошибка, то интеграцию производить нельзя. Частая интеграция позволяет быстрее получить готовую систему, вместо того чтобы тратить на сборку неделю.
* Тестирование (testing). В отличие от большинства остальных методологий тестирование в XP — одно из важнейших составляющих. Экстремальный подход заключается в том, что тесты пишутся до написания кода. Каждый модуль обязан иметь unit test — тест данного модуля; таким образом, в XP осуществляется regression testing (возвратное тестирование, «неухудшение качества» при добавлении функциональности). Большинство ошибок исправляются на стадии кодирования. Тесты пишут сами программисты; любой программист имеет право написать тест для любого модуля. Еще один важный принцип: тест определяет код, а не наоборот (такой подход носит название test-driven development), то есть кусок кода кладется в хранилище тогда и только тогда, когда все тесты прошли успешно, в противном случае данное изменение кода отвергается.

Итак, XP крайне пренебрежительно относится ко всем артефактам процесса разработки, кроме исходного кода. Процесс XP является в высшей степени неформальным, но требует высокого уровня самодисциплины. Если это правило не выполняется, то XP мгновенно превращается в хаотичный и неконтролируемый процесс. XP не требует от программистов написания множества отчетов и построения массы моделей. В XP каждый программист считается квалифицированным работником, который профессионально и с большой ответственностью относится к своим обязанностям. Если в команде этого нет, то внедрять XP абсолютно бессмысленно — лучше для начала заняться перестройкой команды. Риск разработки снижается только в команде, которой XP подходит идеально, во всех остальных случаях XP — это процесс разработки с наиболее высокой степенью риска, поскольку другие методы снижения коммерческих рисков, кроме банального человеческого фактора, в XP просто отсутствуют.

б) Существующие риски применения методологии

Следует выделить риски XP, способные завалить проект, если не учитывать и не предотвращать их.

* Этап планирования (planning game). Программисты реализуют только те функции, которые необходимы для возможностей, выбранных на данной итерации заказчиком. В результате такого решения за кадром остается развитие системы, вследствие чего при разработке возникает необходимость строить «заглушки» и переписывать код.
* Постоянное участие заказчика (on-site customer). Представитель заказчика в период работы над системой находится в команде разработчиков, причем требования к квалификации этого человека или команды весьма высоки. Если заказчик не согласился предоставить персонал уровня экспертов, то проект попадает в группу наиболее высокого риска.
* Метафора (metaphor). Общий вид системы определяется при помощи метафоры или набора метафор, над которыми совместно работают заказчик и программисты. Если не ведется журнал данного процесса и структура наименований не стандартизована, то такой процесс может оказаться бесконечно итерационным.
* Простая архитектура (simple design). В каждый момент времени разрабатываемая система выполняет все тесты и поддерживает все взаимосвязи, определяемые программистом, не имеет дубликатов кода и содержит минимально возможное количество классов и методов. Это правило кратко можно выразить так: «Каждую мысль формулируй один и только один раз». Данный принцип вступает в противоречие с быстротой написания кода. Без наличия высокой самодисциплины и жестких стандартов кода система немедленно попадает в группу риска.
* Частая смена версий (small releases). Систему запускают в эксплуатацию уже через несколько месяцев после начала реализации, не дожидаясь окончательного разрешения всех поставленных проблем. Периодичность выпуска новых версий может варьироваться от ежедневной до ежемесячной. Протестировать за такой срок более-менее сложный компонент невозможно; заказчик фактически выступает в роли бета-тестера. Системы, к которым предъявляется требование непрерывной надежной работы (так называемое требование 24Ѕ7), входят в группу риска.
* Переработка системы (refactoring). Архитектура системы постоянно эволюционирует. Текущий проект трансформируется, при этом гарантируется правильное выполнение всех тестов. Экстремальное программирование исходит из того, что переделать часть системы всегда можно, причем без особых затрат. Однако практика довольно часто свидетельствует об обратном.
* Непрерывная интеграция (continuous integration). Новый код интегрируется в существующую систему не позднее чем через несколько часов. После этого система вновь собирается в единое целое и прогоняются все тесты. Если хотя бы один из них не выполняется корректно, внесенные изменения отменяются. В данном случае не всегда понятно, кто именно будет исправлять ошибки, причем не только локальные, но и наведенные неправильным кодом. Проведение комплексных тестов на данном этапе не предполагается; кроме того, изменения сохраняются даже в том случае, когда ошибка обнаружена.
* Программирование в паре (pair programming). Весь код проекта пишут группы по два человека, использующих одно рабочее место. Человеческий фактор в данном случае играет определяющую роль: пара или работает или нет, третьего не дано.
* 40-часовая неделя (40-hour weeks). Объем сверхурочных работ не может превышать по длительности одну рабочую неделю. Даже отдельные случаи сверхурочных работ, повторяющиеся слишком часто, служат признаком серьезных проблем, которые требуют безотлагательного решения. Как показывает практика применения экстремального программирования (несмотря на целый ряд положительных примеров, приводимых сторонниками данного метода), сверхурочная работа при таком подходе — это правило, а не исключение, и борьба с проблемами в данном случае — явление постоянное. Усиливается она в период замены текущей сырой версии продукта очередной — менее сырой. Если заказчик не получает постоянных доказательств улучшения системы, значит, у вас возникли серьезные проблемы.
* Коллективное владение (collective ownership). Каждый программист имеет возможность при необходимости в любое время усовершенствовать любую часть кода в системе. Без стандарта контроля исходного кода процесс разработки приобретает абсолютно неконтролируемый характер.
* Открытое рабочее пространство (open workspace). Команда разработчиков располагается в большом помещении, окруженном комнатами меньшей площади. В центре рабочего пространства устанавливаются компьютеры, на которых работают пары программистов (причем в соответствии с вышеизложенными принципами, все это должно располагаться на территории заказчика, поскольку он весьма активно привлекается к процессу разработки). При наличии территориально распределенной группы разработчиков и заказчиков проект требует стандартизации протокола взаимодействия (быстро, надежно, безотказно) или попадает в группу риска.
* Тесты (tests). Программисты постоянно пишут тесты для модулей (unit tests). Собранные вместе, эти тесты должны работать корректно. Для этапов итерации заказчики пишут функциональные тесты (functional tests), от которых также требуется правильная работа. Однако на практике это не всегда достижимо. Чтобы принять верное решение, необходимо понять, во что обойдется сдача системы с заранее известным дефектом, и сравнить это с ценой задержки на его устранение. Тесты, написанные самими программистами (особенно в условиях сверхурочных работ), не являются полнофункциональными и уж тем более не учитывают особенностей многопользовательской работы. На более продвинутые тесты у разработчиков обычно не хватает времени. Решается данная проблема путем привлечения на определенный срок контакторов, что связано с большой ролью человеческого фактора: поскольку техническая документация изначально отсутствует, то информация передается посредством общения программистов. Хотя, конечно, можно построить систему разработки таким образом, что от начала до конца всем будут заниматься одни и те же люди. К сказанному необходимо добавить, что тестирование системы вовсе не исчерпывается тестами компонентов (units); не менее важны тесты взаимодействия между ними, это же относится и к тестам надежности работы. И тем не менее метод экстремального программирования не предусматривает создания тестов данного класса. Это объясняется тем, что сами подобные тесты могут представлять достаточно сложный код (особенно это касается тестов — имитаторов реальной работы системы). В данной технологии также никак не учитывается еще один важный класс тестов — тесты поведения системы при росте объемов обрабатываемой информации. При высокой частоте изменения версий выполнить такой тест технологически невозможно, поскольку его проведение требует стабильного и неизменного кода проекта, например в течение недели. В таком случае придется или приостанавливать разработку компонентов, или создавать на время проведения теста параллельную версию проекта, которая будет сохраняться неизменной, тогда как другая при этом будет изменяться. Затем нужно будет выполнить процесс слияния кода. Но в этом случае тест придется создавать заново, так как методы экстремального программирования просто не предусматривают разработку средств, позволяющих прогнозировать поведение системы при тех или иных изменениях. Решать данные проблемы в XP предлагается посредством все того же человеческого фактора и самодисциплины.
* Не более чем правила (just rules). Члены коллектива, работающего по технологии экстремального программирования, обязуются выполнять изложенные правила. Однако это не более чем правила, и команда может в любой момент изменить их, если ее члены достигнут принципиального соглашения по поводу внесенных изменений. Данный принцип серьезно зависит от человеческого фактора; нарушение дисциплины разработки влечет за собой срывы сроков и в результате ведет к краху проекта.

В итоге мы получаем метод, потенциально обладающий высокой адаптацией к серьезно и часто изменяющимся требованиям к проекту, но в то же время не свободный от ряда принципиальных недостатков и в очень высокой степени зависимый от человеческого фактора.

Таким образом, результат применения метода экстремального программирования может получиться либо экстремально хорошим, либо экстремально плохим.

2.2 Выбор инструментальных средств

В качестве единой среды разработки, как для транслятора партий так и для регистратора была выбрана среда NetBeans IDE. NetBeans IDE — свободная интегрированная среда разработки приложений (IDE) на языке программирования Java, Ruby, C++ и ряде других. Среда разработки NetBeans по умолчанию поддерживает разработку для платформ J2SE и J2EE. Для разработки программ в среде NetBeans и для успешной инсталляции и работы самой среды NetBeans должен быть предварительно установлен Sun JDK или J2EE SDK подходящей версии. Для поддержки разработки в среде NetBeans для мобильных платформ (J2ME) необходимо установить отдельно распространяемый (и также бесплатный) NetBeans Mobility Pack (доступен только для Linux и Windows). Проект NetBeans IDE поддерживается и спонсируется фирмой Sun Microsystems, однако разработка NetBeans ведется независимо сообществом разработчиков-энтузиастов (NetBeans Community) и компанией NetBeans Org. По качеству и возможностям последние версии NetBeans IDE не уступают лучшим коммерческим (платным) интегрированным средам разработки для языка Java, таким, как IntelliJ IDEA, поддерживая рефакторинг, профилирование, выделение синтаксических конструкций цветом, автодополнение набираемых конструкций на лету, множество предопределённых шаблонов кода и др. В версии NetBeans IDE 6.0 декларируется поддержка UML, SOA, языка программирования Ruby (включая поддержку Ruby on Rails), а также средства для создания приложений на J2ME для мобильных телефонов (Linux, Windows). NetBeans IDE поддерживает плагины, позволяя разработчикам расширять возможности среды. На идеях, технологиях и в значительной части на исходном коде NetBeans IDE базируются предлагаемые фирмой Sun коммерческие интегрированные среды разработки для Java — Sun Java Studio Creator, Sun Java Studio Enterprise и Sun Studio (для ведения разработки на C, C++ или Фортран). Сравнительно недавно Sun стала предлагать эти среды разработки бесплатно для зарегистрировавшихся в Sun Developer Network (SDN) разработчиков, сама же регистрация на сайте бесплатна и не требует никаких предварительных условий, кроме согласия с лицензией CDDL. NetBeans IDE доступна в виде готовых дистрибутивов (прекомпилированных бинарников) для платформ Microsoft Windows, GNU/Linux, FreeBSD, Mac OS X и Solaris (как для SPARC, так и для x86 — Intel и AMD). Для всех остальных платформ доступна возможность собрать NetBeans самостоятельно из исходных текстов.

Корпорация Sun Microsystems добавила поддержку Ruby к своей интегрированной среде разработки NetBeans и расширила платформу JRuby. Первая версия NetBeans Ruby Pack содержит подключаемый модуль для свободно распространяемой среды разработки NetBeans, поддерживающей Ruby и JRuby. Последний представляет собой Java-реализацию Ruby, которая работает с виртуальной машиной Java. Платформа NetBeans в первую очередь ориентирована на Java, но может быть расширена и до Ruby. Как правило, разработчики, которые пишут программы на этом языке, не используют интегрированные среды разработки. Однако, как подчеркнул Тор Норби, старший инженер Sun, «предложенное корпорацией решение представляет собой значительно более производительную среду, чем все, что существовало для Ruby ранее».

2.3 Содержательная постановка задачи создания СШПО

Предметная область

Специализированное шахматное программное обеспечение (СШПО) предназначено для переноса играющихся шахматных партий в электронную форму, трансляции игр для присутствующей на соревнованиях аудитории и в сети Интернет. Пользователями СШПО будет являться персонал структурного подразделения «Шахматный клуб» ГОУ ВПО «СибГИУ» и персонал МУДОД «СДЮСШОР по шахматам».

Дано:

Действующая ИС «Шахматный клуб».

Прототипы информационной системы.

Множество моделей жизненного цикла (МЖЦ) разработки программного обеспечения.

Инструментальные средства разработки ПО.

Общие требования к СШПО:

* + работоспособность во всех современных операционных системах (ОС),
  + непрерывность работы в ходе всего соревновательного процесса,
  + возможность получения корректной и оперативной информации о соревнованиях за пределами турнирного зала и из любой точки земного шара по сети Интернет,
  + накопление и сохранение информации по соревновательному процессу,
  + экономия времени и средств на перенос партий в электронную форму,
  + обеспечение прохождения соревновательного процесса в рамках действующих игровых правил,
  + учет возможности исправления неправильного течения игрового процесса.

Ограничения:

При выборе МЖЦ ограничиться каскадной и экстремальной МЖЦ разработки программного обеспечения.

1. При выборе инструментальных средств разработки СШПО ограничиться следующими программными продуктами: NetBeans IDE 6.0, MySQLAdministrator.

2. В ходе функционирования СШПО во время соревновательного процесса должны выполняться следующие условия:

* + СШПО должно функционировать без простоев в течении всего игрового дня.
  + Отображаемый ход должен полностью соответствовать ходу, сделанному на электронной шахматной доске.
  + Временная задержка трансляции, если она заранее не предусмотрена, не должна превышать 0,5 секунды.
  + Временная задержка переноса игрового процесса в электронную форму не должна превышать 0,5 секунды.

Требуется

Реализовать СШПО с учетом всех требований и ограничений

2.4 Разработка алгоритма решения задачи

Регистратор шахматных партий (РШП) реализуется на языке Java (j2se). РШП реализует протокол обмена данных DGT шахматных электронных досок, который в свою очередь базируется на прокотоле обмена через последовательный порт RS-232. В качестве компонента для работы с последовательным портом в Java была выбрана библиотека rxtx версии 1.72. Протокол DGT приведен в приложении 6 в виде заголовочного C файла (header). Задача РШП осуществлять трансляцию партий, при этом изменения позиции партий сохраняются в базу данных, откуда эти данные получает Транслятор шахматных партий (ТШП). Формат записи, в котором записываются шахматные ходы в базу данных, следующий:

[фигура{K(король),Q(ферзь),N(конь),B(слон),R(ладья),’ ’(пешка)}][вертикаль исходного поля][горизонталь исходного поля]-[вертикаль поля назначения][горизонталь поля назначения].

Например:

Kg8-g7

Ng1-f3

e2-e4

Данные, получаемые РШП от электронной доски, интерпретируются согласно описанию в протоколе DGT. Например, дамп доски получается в виде 64 ASCII символов (информативная часть сообщения) – ‘rnbqkbnrpppppppp PPPPPPPPRNBQKBNR’ преобразуется в вид:



Рисунок 8 – Результат преобразования информативной части сообщения от ЭШД

ТШП реализован на технологии Ruby on Rails. Rails — это полноценный, многоуровневый фреймворк для построения веб-приложений, использующих базы данных, который основан на архитектуре Модель-Представление-Контроллер (Model-View-Controller, MVC). Динамичный AJAX-интерфейс, обработка запросов и выдача данных в контроллерах, предметная область, отраженная в базе данных, — для всего этого Rails предоставляет однородную среду разработки на Ruby. Все, что необходимо для начала — база данных и веб-сервер. Rails отлично работает со многими веб-серверами и СУБД. В качестве веб-сервера можно использовать Apache или lighttpd как с FastCGI, так и с SCGI. В качестве СУБД можно использовать MySQL, PostgreSQL, SQLite, Oracle, SQL Server, DB2 или Firebird. Использовать Rails можно на практически любой операционной системе.

Задача ТШП создавать трансляции и вещать шахматные партии. В ТШП предусмотрена система авторизации, что позволяет гибко настраивать права пользователей зарегистрированных в системе, по умолчанию существуют три профиля пользователей: Администратор (права на все), Руководитель (ему принадлежат права на создание/редактирование online трансляций, турниров, комментирование партий и т.д.) и Гость (только просмотр партий).

Модуль вещания партий реализован при помощи скриптов JavaScript, при этом обновление позиции запрашивается с сервера через AJAX запросы, без обновления всей страницы.

Листинг модуля вещания партий представлен в приложении 10.

2.5 Описание разработанного программного комплекса

2.5.1 Транслятор шахматных партий

В общем виде транслятор шахматных партий (ТШП) представляет собой следующие структуры:

* модели данных (models);
* представления (views);
* контроллеры (controllers);
* помощники (helpers).

Модели данных содержат объектные представления, задачи в виде классов бизнес-логики. Здесь описываются классы, к которым будут отнесены реальные данные. Бизнес-логика управляется одноименным контроллером, например, класс Cities (города) управляется одноименным контроллером cities\_controller.rb. Модель может иметь одно или несколько представлений, которые отвечают за то, в каком виде будут отображаться данные. Помимо контроллеров всех классов существует главный контроллер main\_controller.rb (в качестве главного может быть назначен любой контроллер). Он выполняет все функции по обслуживанию шахматного интернет-портала:

* отображение главной веб-страницы;
* возврат к предыдущей веб-странице;
* показ партий в режиме реального времени (online) и архива турнирных партий (offline);
* авторизация пользователей;
* вход в личный кабинет пользователя;
* напоминание при потере пароля или логина;
* показ трансляции;
* выгрузка шахматных партий в формате pgn (portable game notation);
* выход пользователя;
* интерфейс регистрации и т.д.

Снимки экрана (Screenshots) главной страницы rDGT-сервера, страницы авторизации пользователя, страницы просмотра текущих online трансляций и страницы просмотра шахматных партий представлены в приложении 5.

Скрипт трансляции шахматной партии реализован на языке Javascript. Обновление позиции осуществляется через асинхронные javascript-запросы к rDGT-серверу при использовании технологии Ajax без обновления всей страницы.

Клиентское приложение, содержит все необходимые функции, обеспечивающие корректную трансляцию шахматных партий. Оно создает отображение шахматной доски, фигур, времени часов, отображает динамику изменения позиции в соответствии с поступающими от rDGT-сервера данными, позволяет пользователю просматривать текущие партии, осуществлять навигацию по последовательности ходов в любом порядке.

Веб-интерфейс реализован на платформе Ruby on Rails, в соответствии с идеологией изложенной в разделе 2.4 . Портал можно разделить на несколько узловых разделов:

* Раздел трансляций партий (online режим).
* Раздел просмотра партий, сохраненных на сервере (offline режим).
* Раздел редактирования данных портала, подразумевает авторизацию пользователя, которому доступны:
* добавление, удаление, изменение турниров;
* добавление, удаление, изменение игроков;
* добавление, удаление, изменение трансляций;
* добавление, удаление, изменение данных различных справочников (регламент проведения турнира, часовые регламенты, страны, города).

Общая структура транслятора шахматных партий, все атрибуты и методы его структурных элементов представлены в приложении 7

2.5.2 Регистратор шахматных партий

Алгоритм работы:

Пользователь в веб-интерфейсе формирует трансляцию партий, при этом он указывает следующие параметры:

* Название последовательного порта (serial port), к которому подключены доски, с которых будет происходить трансляция шахматных партий («COM1», «COM2» и т.п. для операционной системы Windows и «/dev/ttyS0», «/dev/ttyS1» для операционной системы Linux).
* Время начала трансляции.
* Для каждой доски указывается игрок, играющий белым цветом, и игрок, играющий черным цветом.
* Ассоциирует данную трансляцию с заранее заданным турниром.

Веб-сервер после составления заявки на трансляцию партий, запускает приложение jgdtnix, параметризуя его введенными параметрами. Jgdtnix до назначенного времени переходит в «спящий» режим, а за несколько минут до начала трансляции jgdtnix начинает опрашивать порт на наличие доступных досок и регистрирует их. Во время трансляции jgdtnix с определенной периодичностью опрашивает порт, получает обновления позиции и данные с шахматных часов, и заносит ходы в базу данных.

Общая структура регистратора шахматных партий, все атрибуты и методы его структурных элементов представлены в приложении 8.

2.6 Тестовые испытания и анализ результатов

Тестовые испытания проводились в помещении СП «ШК» в сроки с 5 июня по 11 июня 2008 года на Чемпионате области по шахматам среди юношей до 12 лет. Количество используемых в испытаниях ЭШД и электронных шахматных часов – 5 штук,. Трансляция соревновательного процесса осуществлялась в течение всего указанного срока с 11:00 до 15:00, что соответствует длительности игрового дня. Схема подключения ЭШД и электронных шахматных часов аналогична рисунку 3 пункта «Обзор шахматных систем-прототипов» с той лишь разницей, что трансляция шахматных партий осуществлялась не только для присутствующей аудитории, но и в сети Интернет.

Анализ результатов тестовых испытаний представлен в приложении 9.

В качестве сравнительного показателя используется общее число шахматных ходов, сделанных на всех досках в течение всего соревновательного процесса. Так как число игровых дней – 7, число учитываемых досок – 5, а среднее число ходов в шахматной партии 30, то общее число шахматных ходов равно равно 1050.

http://www.compress.ru/Archive/CP%5C2003%5C10%5C65/http:/www.cpress.ru/

# Технико-экономическое обоснование проекта

## Целесообразность и область применения разработки

В данном технико-экономическом обосновании рассматривается специализированное шахматное программное обеспечение, разработанное для автоматизации деятельности информационной системы структурного подразделения «Шахматный клуб» в рамках внедрения комплексной информационной системы структурного подразделения «Шахматный клуб» ГОУ ВПО «Сибирский государственный индустриальный университет».

### Эффект от внедрения информационных систем

Информатизация ради информатизации не имеет смысла. Информационная система – это экономический товар, и как любой товар она имеет цену. Одним из первых критериев при внедрении такой системы является сопоставление выгод, приносимых ею, с затратами на ее внедрение, сопровождение и поддержку, то есть цель должна оправдывать средства.

Выгоду от внедрения информационной системы можно разделить на две составляющие - организационную и экономическую. Первая связана с общими изменениями учебного и соревновательного процессов, внедрением более прогрессивных методов проведения спортивных состязаний, повышением общей культуры управления, снижением бумажного документооборота, использованием новых, более оптимальных схем построения учебного и соревновательного процессов. Это, так сказать, качественные улучшения, и они являются скорее следствием, чем целью информатизации. Улучшения с ними связанные, представляют набор управленческих решений, позволяющий успешнее достигать целей, поставленных руководством.

Помимо повышения общей культуры управления, информатизация помогает учреждению в построении взаимоотношений с потенциальными партнерами и инвесторами. Не секрет, что спонсоры заинтересованы в экономически привлекательных информационных системах, которые могут быть использованы не только по непосредственному назначению, но и в рекламных целях.

Под экономическими выгодами от внедрения подразумевается получение реальной экономической отдачи от использования всего пакета приложений или отдельных функциональных блоков. Согласно статистике внедрение информационных систем и систем информационного сопровождения может обеспечить следующую отдачу:

* устранение ручной подготовки и сопровождения документов;
* уменьшение затрат на административно-управленческий аппарат;
* более точный учет затрат;
* более полное информационное обеспечение конечных пользователей и потребителей;
* уменьшение сроков информирования конечных пользователей и потребителей.

Даже самая лучшая в мире компьютерная система не в состоянии выполнить роль «волшебной палочки», как по мановению снимающей все накопившиеся проблемы. Внедрение информационной системы – это совместный труд, как поставщика, так и заказчика. Такая система призвана в первую очередь поддерживать регулярный менеджмент. Если же в учреждении такового не существует или нет хотя бы попыток его наладить, информационная система изначально обречена быть «инородным телом». Поэтому реальную отдачу от информационной системы можно получить только при правильной организации проекта внедрения и при желании первых лиц организации создать прибыльную, эффективно работающую структуру информационного сопровождения.

### Информатизация деятельности СП «ШК»

В настоящее время на рынке программных продуктов существуют различные прикладные информационные решения для информационного обеспечения деятельности структурного подразделения «Шахматный клуб». Однако у всех у них один общий и весьма существенный недостаток – отсутствие поддержки трансляций соревновательного процесса в сети Интернет и локальных сетях. Программное обеспечение подобного рода зачастую является объектом интеллектуальной собственности и распространяется по решению владельцев. Программное обеспечение, поставляемое в комплекте с электронными шахматными досками, не отвечает необходимым требованиям и во многих случаях является нерабочим.

Из этой ситуации есть 3 выхода: приглашать специалистов из международных шахматных организаций на обслуживание процесса соревнований, что влечет за собой весьма значительные расходы; пытаться найти качественное программное обеспечение на рынке, что весьма проблематично, так как владельцы интеллектуальной собственности не хотят с ним расставаться, а предпочитают пользоваться им самостоятельно, зарабатывая деньги на обслуживании соревнований; разрабатывать собственное программное обеспечение.

Первые два пути были уже опробованы. На обслуживание Всемирной студенческой универсиады, проводившейся с 3 по 11 марта 2008 года были приглашены специалисты из информационного центра российской шахматной федерации. В результате эти услуги обошлись организаторам примерно в 100000 рублей, что весьма накладно. После проведения этого крупного спортивного форума организаторами была предпринята попытка наладить коммерческие связи с обслуживающими турнир специалистами с целью покупки программного обеспечения для личного использования, однако это предложение было проигнорировано. В связи с этими обстоятельствами автором дипломного проекта по согласованию с руководством структурного подразделения «Шахматный клуб» ГОУ ВПО «СибГИУ» и руководством МУДОД «СДЮСШОР по шахматам имени Б. А. Кустова» было принято решение о самостоятельной разработке специализированного шахматного программного обеспечения с целью личного и коммерческого использования.

* 1. Расчет затрат на разработку специализированного шахматного программного обеспечения

Просчитаем экономическую эффективность разработки специализированного программного обеспечения в денежном эквиваленте.

Затраты на разработку специализированного программного обеспечения складываются из заработной платы разработчикам, расходов на приобретение, монтаж и эксплуатацию оборудования, материалов, расходов на приобретение лицензионных программных продуктов, необходимых для разработки.

Разработчиками специализированного программного обеспечения являются: руководитель проекта – администратор МУДОД «СДЮСШОР по шахматам им. Б.А.Кустова» и инженер структурного подразделения «Шахматный клуб».

В таблице 1 представлены основные этапы проведения работы.

Таблица 1 - Оценка загрузки исполнителей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  этапа | Наименование работ | Длитель-ность, дни | Исполнитель | Коэффициент загрузки, проценты |
| 1 | Постановка задачи | 3 | руководитель | 50% |
| инженер | 50% |
| 2 | Сбор исходных материалов | 3 | руководитель | 75% |
| инженер | 25% |
| 3 | Определение требований к программе | 3 | руководитель | 50% |
| инженер | 50% |
| 4 | Согласование и утверждение ТЗ | 2 | руководитель | 50% |
| инженер | 50% |
| 5 | Изучение литературы по теме задания | 7 | инженер | 100% |
| 6 | Уточнение структуры и формы представления данных.  Разработка структуры программы | 7 | инженер | 100% |
| 7 | Разработка плана мероприятий по внедрению программы,  пояснительной записки. | 3 | инженер | 100% |
| 8 | Программирование | 20 | инженер | 100% |
| 9 | Тестирование программы | 7 | инженер | 100% |
| 10 | Написание программной документации | 7 | инженер | 100% |
| 11 | Подготовка и передача программы и программной документации для сопровождения и утверждения. | 7 | инженер | 25% |
| руководитель | 75% |
|  | Итого | 69 |  |  |

* + 1. Затраты на оплату труда разработчиков программы

Размер месячного оклада для инженера структурного подразделения «Шахматный клуб» составляет 3000 рублей.

Размер месячного оклада для руководителя составляет 4000 рублей.

Полагая, что в месяце 21 рабочий день, рассчитаем стоимость одного рабочего дня. Расчет оклада разработчиков программы за один рабочий день приведен в таблице 2.

Расчет основной заработной платы приведен в таблице 3.

Таблица 2 - Расчет оклада сотрудников за один рабочий день

|  |  |  |
| --- | --- | --- |
| Должность | Оклад в месяц, руб. | Стоимость одного рабочего дня, руб. |
| Руководитель | 4 000,00 | 190,48 |
| Программист | 3 000,00 | 142,86 |

Таблица 3 - Расчет основной заработной платы сотрудников за период разработки программы

|  |  |  |  |
| --- | --- | --- | --- |
| Исполнитель | Трудоемкость, дней | Стоимость одного рабочего дня, руб. | Сумма, руб. |
| Руководитель | 11,5 | 190,48 | 2190,52 |
| Программист | 57,5 | 142,86 | 8214,45 |
| Итого | 69 | 333,34 | 10404,97 |

* + 1. Затраты на отчисление единого социального налога

Отчисления по единому социальному налогу (ЕСН), который составляет 26 % от заработной платы, равны:

ЗЕСН=10404,97\*0.26=2705,29 руб.

* + 1. Затраты на приобретение оборудования для разработки специализированного программного обеспечения

Затраты на приобретение оборудования определяются по формуле:

(1)



где – стоимость оборудования;



– затраты на транспортировку (8% от );



– затраты на монтаж (10% от ).



Перечень приобретенного оборудования и затраты на него приведены в таблице 4.

Таблица 4 - Перечень приобретенного оборудования и затраты на него

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование оборудования | Количество | , руб. | , руб. | , руб. | , руб. |
| Принтер (HP LaserJet 1010) | 1 | 5200 | 416 | 520 | 6136 |
| Манипулятор Logitech | 2 | 1200 | 96 | 120 | 1416 |
| Итого: |  | 6 200 | 496 | 620 | 7552 |

* + 1. Затраты на приобретение материалов

На эту статью относится стоимость материалов, покупных изделий, полуфабрикатов и других материальных ценностей, расходуемых непосредственно в процессе выполнения проекта. Цена материальных ресурсов определяется по соответствующим ценникам. В стоимость материальных затрат включаются транспортные расходы (5% от прейскурантной цены). Расчет статьи «Материалы» приводится в таблице 5.

Таблица 5 - Материалы и покупные изделия и услуги

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование | Единица измерения | Количество | Цена за единицу, руб | Стоимость, руб. |
| Бумага для принтера | пачка | 1 | 120 | 120 |
| Чернила | мл | 100 | 0,8 | 80 |
| Лазерные диски | шт | 5 | 15 | 75 |
| Оплата за интернет | час | 15 | 30 | 450 |
| Итого: | | | | 725 |

* + 1. Затраты на приобретение лицензионных программных продуктов

Для разработки специализированного шахматного программного обеспечения требуются следующие лицензионные программные продукты:

* среда разработки NetBeans IDE 6.0;
* программный пакет для создания и разработки баз данных MySQLAdministrator;
* программный пакет Ruby-186-26;
* драйвер опроса электронных шахматных досок dgtnix.

Все перечисленные программные продукты являются абсолютно бесплатными. Первый, второй и третий из них распространяются по лицензии с открытым исходным кодом LGPL, последний из них также распространяется бесплатно, но по лицензии с открытым исходным кодом GPL.

* + 1. Затраты на потребляемую электроэнергию

К этой статье относится стоимость потребляемой электроэнергии компьютером за время разработки программы.

Стоимость электроэнергии, потребляемой при разработке специализированного программного обеспечения, определяется по формуле (2):

(2)



где – суммарная мощность ЭВМ, кВт;



– время работы компьютера, час;



– стоимость 1 кВт\*ч электроэнергии, руб.



Поскольку стандартный рабочий день равен 8 часам, то

,



где – время эксплуатации компьютера при создании специализированного программного обеспечения, дней.



Следовательно, стоимость электроэнергии за период работы компьютера во время создания специализированного шахматного программного обеспечения будет вычисляться по формуле (5):

, (3)



Согласно техническому паспорту ЭВМ Рэвм=0,25 кВт, а стоимость 1 кВт/ч электроэнергии по городу =0,96 руб.



Тогда расчетное значение затрат на электроэнергию:



* + 1. Накладные расходы

Накладные расходы включаются в сумму затрат на разработку специализированного программного обеспечения и принимаются 10% от затрат на заработную плату и ЕСН.



* + 1. Смета затрат на разработку системы

Смета затрат на разработку системы приведена в таблице 6.

Таблица 6 - Смета затрат на разработку системы

|  |  |
| --- | --- |
| Наименование затрат | Сумма, руб. |
| Затраты на заработную плату разработчикам | 10404,97 |
| Затраты на ЕСН | 2705,29 |
| Затраты на приобретения оборудования для разработки системы | 7552 |
| Затраты на приобретение материалов | 725 |
| Затраты на приобретение лицензионных программных продуктов | – |
| Затраты на электроэнергию | 178,56 |
| Накладные расходы | 1311,03 |
| Итого: | 22876,85 |

* 1. Расчет эксплуатационных затрат

Эксплуатационные затраты определяются по формуле:

(4)



где – затраты на заработную плату;



– затраты на амортизационные отчисления;



– затраты на электроэнергию;



– затраты на материалы;



– затраты на текущий ремонт оборудования и устройств системы;



– накладные расходы.



* + 1. Затраты на заработную плату

За работоспособностью специализированного шахматный программного обеспечения будет следить инженер структурного подразделения «Шахматный клуб», доля заработной платы которого составляет 500 рублей в месяц.

Отчисления ЕСН (26%) составят: 500\*0,26 = 130 рублей/месяц

* + 1. Затраты на амортизационные отчисления

Амортизационные отчисления рассчитываются по формуле:

(5)



где - стоимость оборудования, руб.;



- норма амортизации.



Расчет затрат на амортизацию приведен в таблице 7.

Таблица 7 - Затраты на амортизационные отчисления

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование | Кол. | Общая стоимость, руб. | Норма амортизации, %. | Сумма, руб. |
| Принтер (НР LaserJet 1010) | 1 | 6 136 | 12,5 | 767 |
| Манипулятор Logitech | 2 | 1416 | 12,5 | 177 |
| Итого: | | | | 944 |

* + 1. Затраты на электроэнергию

Затраты на электроэнергию рассчитываются по формуле:

(6)



где – мощность i-го оборудования;



n – количество токоприемников;

– время работы i-того оборудования, час.



(7)



где – количество рабочих дней в году;



– норматив среднесуточной загрузки, ч.;



– коэффициент использования i-того оборудования;



– стоимость 1 кВт/ч электроэнергии.



Расчет затраты на электроэнергию приведены в таблице 8.

Таблица 8 - Расчет затрат на электроэнергию

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Наименование устройства | Кол. | Р, Вт | Дг, дней | ч. | Т, ч. | К | Сэл (кВт/ч), руб. | Зэл, руб. |
| ПК (Celeron 1.7 GHz) | 1 | 250 | 366 | 8 | 2920 | 0,5 | 0,96 | 350,4 |
| Монитор (Samsung 17") | 1 | 80 | 366 | 8 | 2920 | 0,5 | 0,96 | 112,13 |
| Принтер (НР LaserJet 1010) | 1 | 200 | 366 | 6 | 2190 | 0,1 | 0,96 | 42,05 |
| Итого: | | | | | | | | 504,58 |

* + 1. Затраты на материалы

Перечень расходных материалов и затраты на них приведен в таблице 9.

Таблица 9 - Перечень расходных материалов и затраты на них

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование материала | Количество | Стоимость за 1 ед., руб. | Доставка (8%),  руб. | Сумма, руб. |
| Бумага офисная (формат А4) | 4 | 120 | 9,6 | 518,4 |
| Тонер для принтера | 3 | 180 | 14,4 | 583,2 |
| Итого: | | | | 1 101,6 |

* + 1. Затраты на текущий ремонт оборудования и устройств системы

Затраты на текущий ремонт оборудования составляют 5% в год от его стоимости (таблица 10).

Таблица 10 - Расчет затрат на текущий ремонт оборудования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование оборудования | Кол. | Общая стоимость, руб. | Норматив затрат на ремонт, % | Сумма, руб. |
| Принтер (НР LaserJet 1010) | 1 | 5 200 | 0,05 | 260 |
| Манипулятор Logitech | 2 | 1 200 | 0,05 | 60 |
| Итого: | | | | 320 |

* + 1. Смета эксплуатационных затрат

Смета эксплуатационных затрат приведена в таблице 11.

Таблица 11 - Смета эксплуатационных затрат

|  |  |
| --- | --- |
| Наименование затрат | Сумма, руб. |
| Затраты на заработную плату | 6000 |
| Затраты на ЕСН | 1560 |
| Затраты на амортизационные отчисления | 944 |
| Затраты на электроэнергию | 504,58 |
| Затраты на материалы | 1 101,6 |
| Затраты на текущий ремонт оборудования и устройств системы | 320 |
| Итого: | 10430,18 |

* 1. Оценка экономической эффективности проекта

Для выяснения экономической выгоды от разработки специализированного шахматного программного обеспечения необходимо проанализировать и оценить его преимущества.

Данная разработка предназначена для информационного обеспечения учебного процесса и спортивных состязаний, проводимых в структурном подразделении «Шахматный клуб» ГОУ ВПО «СибГИУ», а также спортивных состязаний по шахматам любого уровня, проводимых с использованием электронных шахматных досок DGT. Она обеспечит эффективное информационное сопровождение проводимых в помещении структурного подразделения «Шахматный клуб» спортивных состязаний, освещение соревновательного процесса в сети Интернет и локальной сети ГОУ ВПО «СибГИУ», позволит полностью отказаться от периодических услуг наемных рабочих. Кроме того, данная разработка будет приносить прибыль при коммерческом использовании.

Годовая экономия от внедрения системы определяется по формуле:

(8)



где Э1, Э2, Э3 – экономия в год по факторам;

Сэкс – эксплуатационные затраты в год.

3.4.1 Исключение расходов на услуги наборщиков

Каждый год в помещении структурного подразделения «Шахматный клуб» проводится в среднем 10 крупных спортивных состязаний, на период проведения которых нанимается 2 наборщика, в обязанности которых входит перевод записи шахматных партий из бумажной формы в электронную. Величина оплаты услуг одного наборщика за один турнир составляет 2000р. Таким образом, общие расходы на услуги наборщиков составляют 40000р. в год.

Данное специализированное программное обеспечение позволяет полностью автоматизировать перевод записи шахматных партий в электронную форму.

Э1=40 000 руб.

3.4.2 Исключение затрат на освещение соревнований в сети Интернет

Все значительные соревнования, проводящиеся в помещении структурного подразделения «Шахматный клуб» согласно регламенту должны быть освещены в сети Интернет. Освещение соревновательного процесса в сети Интернет входит в обязанности главного арбитра соревнований. Главный арбитр командируется городским МУДОД «СДЮСШОР по шахматам им. Б.А. Кустова» в шахматный клуб ГОУ ВПО «СибГИУ» на проведение турнира. Расходы по оплате работы главного судьи по проведению спортивных состязаний берет на себя городское МУДОД «СДЮСШОР по шахматам им. Б. А. Кустова». В тех случаях, когда необходимо освещение спортивного форума в сети Интернет, данная услуга оплачивается отдельно шахматным клубом ГОУ ВПО «СибГИУ». Из 10 крупных турниров, проводящихся за год в помещении структурного подразделения «Шахматный клуб» минимум 5 освещаются в сети Интернет.

Расходы на освещение соревнований в сети Интернет представлены в таблице 12.

Таблица 12. Расходы на освещение соревнований в сети Интернет

|  |  |  |
| --- | --- | --- |
| Наименование затрат | Затраты за  1 турнир (руб) | Годовой объем затрат (руб) |
| Выкладка всей информации о соревнованиях в сети интернет | 2 000 | 10 000 |
| Дополнительные затраты за пользование Интернетом (руб/час) | 300 | 1 500 |
| Итого | | 11 500 |

Разработанное специализированное шахматное программное обеспечение уже включает в себя web-сайт, на котором в режиме реального времени ведется трансляция соревнований, проводящихся в помещении структурного подразделения «Шахматный клуб». На данном сайте отображается вся информация о соревнованиях, сохраняются все текущие и предыдущие партии. Таким образом, отпадает необходимость оплаты дополнительных услуг главного арбитра соревнований по выкладке информации в сети Интернет, а также отпадает необходимость в дополнительных затратах за пользование Интернетом, так как трансляция ведется в режиме реального времени и отсутствует необходимость в дополнительных временных затратах.

Э2=11 500 руб.

3.4.3 Прибыль от коммерческого использования разработки

В настоящее время в Сибирском федеральном округе проводится ежегодно около сотни крупных шахматам форумов. Каждый такой турнир нуждается в информационном сопровождении и услугах наборщиков шахматных партий. В среднем на информационное сопровождение и набор партий одного крупного шахматного турнира тратится от 10000 до 100000 рублей (в зависимости от масштабов соревнований), включая услуги наборщиков, освещение соревнований в сети Интернет и средствах массовой информации. Использование разработанного специализированного программного обеспечения в сочетании с электронными шахматными досками DGT позволит полностью или частично отказаться от услуг наборщиков, автоматизировать процесс трансляции соревнований в сети Интернет и локальных сетях, упростить процесс освещения спортивных состязаний в средствах массовой информации. Таким образом, в сфере проведения спортивных шахматных состязаний данная разработка будет востребованной.

Полная оценка размеров прибыли от коммерческого использования данной разработки весьма проблематична, так как должна включать в себя детальные маркетинговые исследования, учет затрат на оформление авторских прав, рекламу, сервисное обслуживание, хостинг и т.д. Данную оценку еще только предстоит произвести, что является одним из приоритетных направлений дальнейшей деятельности. Но уже сейчас ясно, что данная разработка будет приносить прибыль, так как ею заинтересовались специалисты из информационного центра российской шахматной федерации.

Факторы годовой экономии при внедрении системы (без учета прибыли от коммерческого использования разработки) приведены в таблице 13.

Таблица 13 - Факторы годовой экономии при внедрении системы

|  |  |  |
| --- | --- | --- |
| Эi | Фактор экономии | Сумма, руб. |
| Э1 | Исключение расходов на услуги наборщиков | 40 000 |
| Э2 | Исключение затрат на освещение соревнований в сети Интернет | 11 500 |
| Итого: | | 51 500 |

Таким образом, годовая экономия может составить:

Э=51 500 – 10 430,18 = 41 069,82 руб.

Экономическая эффективность единовременных затрат на создание системы определяется показателями Тр и Ер, где Тр – расчетный срок окупаемости единовременных затрат на создание системы в годах, Ер – расчетный коэффициент эффективности единовременных затрат на создание системы:



Сопоставляя расчетный коэффициент и срок окупаемости системы с нормативными, находим, что и . Следовательно, данная разработка экономически эффективна и целесообразна.



3.5 Технико-экономические показатели проекта

Технико-экономические показатели проекта представлены в таблице 14.

Таблица 14. Технико-экономические показатели проекта

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование показателя | Ед. изм. | Значение показателя |
| 1 | Проектные затраты | Руб. | 22 876,85 |
| 1.1 | Затраты на оборудование | Руб. | 7552 |
| 1.2 | Затраты на приобретение материалов | Руб. | 725 |
| 1.3 | Затраты на заработную плату разработчикам | Руб. | 10 404,97 |
| 1.4 | Затраты на отчисления ЕСН | Руб. | 2 705,29 |
| 1.5 | Затраты на электроэнергию | Руб. | 132,48 |
| 1.6 | Накладные расходы | Руб. | 1 311,03 |
| 2 | Эксплуатационные затраты | Руб. | 10 430,18 |
| 2.1 | Затраты на амортизационные отчисления | Руб. | 944 |
| 2.2 | Затраты на текущий ремонт оборудования и устройств системы | Руб. | 320 |
| 2.3 | Затраты на ЕСН | Руб. | 1 560 |
| 2.4 | Затраты на заработную плату | Руб. | 6 000 |
| 2.5 | Затраты на электроэнергию | Руб. | 504,58 |
| 2.6 | Затраты на материалы | Руб. | 1 101,6 |
| 3 | Годовая экономия | Руб. | 51 500 |
| 3.1 | Исключение расходов на услуги наборщиков | Руб. | 40 000 |
| 3.2 | Исключение затрат на освещение соревнований в сети Интернет | Руб. | 11 500 |
| 4 | Экономия при внедрении разработанного проекта | Руб. | 41 069,82 |
| 5 | Срок окупаемости проекта | Мес. | 0,56 |

# 4. Безопасность и экологичность проекта

## 4.1 Безопасность труда

### 

### 4.1.1 Анализ условий труда на рабочем месте инженера структурного подразделения «Шахматный клуб» ГОУ ВПО «СибГИУ»

Объектом анализа являются условия труда на рабочем месте инженера структурного подразделения «Шахматный клуб» ГОУ ВПО «СибГИУ», расположенном по адресу ул. Кирова, 42.

Анализ условий труда выполнен для рабочего кабинета инженера структурного подразделения «Шахматный клуб» ГОУ ВПО «СибГИУ», которое располагается в пристройке к главному корпусу СибГИУ на третьем этаже над помещением столовой. Параметры рабочего кабинета инженера: площадь – 12 м2, периметр – 14 м. В рабочем кабинете инженера располагается 3 компьютера. Помещение оборудовано системой отопления и приточно-вытяжной вентиляцией, соответствующей [1].

Для внутренней отделки интерьера рабочего кабинета инженера с ВДТ и ПЭВМ использованы материалы с коэффициентом отражения для потолка - 0,7; для стен - 0,5; для пола - 0,3, соответствующие [2].

Поверхность пола в рабочем кабинете инженера ровная, без выбоин, нескользкая, удобная для очистки и влажной уборки, обладает антистатическими свойствами.

Пыль, оседающая на экране, снижает его освещенность, ухудшает видимость изображения и способствует накоплению статического электричества.

В помещении с ПЭВМ ежедневно проводиться влажная уборка, что соответствует [2].

При работе с компьютером пользователь подвергается воздействию ряда опасных и вредных производственных факторов, которые по природе действия подразделяются на следующие группы [3]:

* поражение электрическим током;
* механические повреждения;
* электромагнитное излучение;
* инфракрасное излучение;
* опасность пожара;
* повышенный уровень шума и вибрации.

Нормативные документы [1] - [7] предусматривают создание на рабочем месте условий труда, при которых влияние опасных и вредных факторов на работающих либо допустимо, либо находится в допустимых пределах.

Проведем оценку условий труда на рабочем месте (таблица 15).

Таблица 15 - Оценка условий труда на рабочем месте

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Фактор | Оценка условий труда | | | | | |
| Фактич. условия | Норматив. значение | | | Ссылка на норматив | Величина соотв./несоотв. |
| Микроклимат | | | | | | |
| Температура,0С  холодный период года  теплый период года | 20-22  22-25 | 20-24  21-28 | | | [4] | соответствует |
| Относительная  влажность воздуха,% | 50-60 | 60 | | | соответствует |
| Скорость движения воздуха, м/с | 0,1 | 0,1-0,2 | | | соответствует |
| Освещение | | | | | | |
| Естественное, % | 1,6 | > 1,5 | | | [5] | соответствует |
| Искусственное, лк | 300 | 300-500 | | | соответствует |
| Шумовое воздействие | | | | | | |
| Уровень шума, дБ | 60 | 61 | | | [6] | соответствует |
| Электромагнитное | | | | | | |
| Напряженность электрического поля |  | |  | [7] | |  |
| 5 Гц-2 кГц | 24 В/м | | 25 В/м | соответствует |
| 2кГц-400 кГц | 2.3 В/м | | 2.5 В/м | соответствует |
| Плотность магнитного потока |  | |  |  |
| 5 Гц-2 кГц | 238 нТл | | 250 нТл | соответствует |
| 2кГц-400 кГц | 22 нТл | | 25 нТл | соответствует |

Работа инженера структурного подразделения «Шахматный клуб» ГОУ ВПО «СибГИУ» относится к категории 1б, т.е. это работа, производимая сидя, стоя или связана с ходьбой и сопровождается некоторым физическим напряжением, при которых расход энергии составляет от 140 до 232 Дж/с [2].

Анализ этой таблицы позволяет охарактеризовать условия работы как оптимальные. В целях оптимизации условий труда проектом разработаны мероприятия по безопасности труда.

### 4.1.2 Мероприятия по безопасности труда

Разработанные проектом мероприятия соответствуют требованиям, предъявляемым при работе за компьютером [2].

Таблица 16 – Факторы воздействия на рабочем месте и используемые средства защиты

|  |  |
| --- | --- |
| Вредные факторы | Используемые средства защиты |
| Электромагнитное излучение | Соблюдение расстояния от глаз оператора до монитора не менее 70 см |
| Статическое электричество | Заземление компьютеров и электрооборудования, влажная уборка |
| Перенапряженность зрительных анализаторов | Правильная ориентация компьютера, чтобы свет падал слева.  Наличие жалюзи. |
| Монотония | При 8-ми часовом рабочем дне работа за компьютером не более 6 часов.  Скорость набора не более 120 знаков в минуту. |

4.1.2.1 Организация рабочих мест

Высота рабочего стола составляет 720 мм. Размеры столешницы составляют 1400 х 800 мм, что соответствует оптимальному размеру. Под столешницей рабочего стола есть свободное пространство для ног с размерами по высоте не менее 600 мм, по ширине не менее 500 мм, по глубине - 650 мм, что соответствует [2].

Конструкция одноместного стола для работы с ПЭВМ и ВДТ должна предусматривает две раздельные поверхности: одна - горизонтальная для размещения ПЭВМ или ВДТ и вторая - для клавиатуры. Отсутствуют ящики.

Корпус ВДТ и ПЭВМ, клавиатура и другие блоки и устройства ПЭВМ имеют матовую поверхность одного цвета, не имеет блестящих деталей, способных создавать блики. В конструкции ВДТ предусмотрены кнопки регулировки яркости и контраста. Они обеспечивают возможность регулировки этих параметров от минимальных до максимальных значений.

Размер экрана должен быть не менее 17 дюймов по диагонали, при этом расстояние от глаз до экрана должно быть в пределах 40 – 80 см.

Клавиатура исполнена в виде отдельного устройства с возможностью свободного перемещения. Конструкция клавиатуры предусматривает опорное приспособление, позволяющее изменять угол наклона поверхности клавиатуры в пределах от 5 до 15 градусов. Высоту среднего ряда клавиш не более 30 мм.

При защите от электромагнитных и электростатических полей допускается применение приэкранных фильтров, специальных экранов и других средств индивидуальной защиты, прошедших испытания в аккредитованных лабораториях, и имеющих соответствующий гигиенический сертификат.

Все описанные параметры и характеристики соответствуют [2].

4.1.2.2 Освещенность рабочего места

В рабочем кабинете инженера структурного подразделения «Шахматный клуб» ГОУ ВПО «СибГИУ» предусмотрено естественное и искусственное освещение. Естественное освещение в соответствии с требованиями [5] осуществляется через светопроемы, ориентированные на север и северо-восток. Освещенность на поверхности стола и клавиатуре должна быть не менее 300 люкс, а вертикальная освещенность экрана - всего 100-250 люкс.

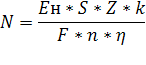
В качестве источников света при искусственном освещении применяются люминесцентные лампы типа ЛБ – 40 в количестве 3 штук. Светильники располагаются локализовано над рабочим столом ближе к его переднему краю, обращенному к оператору.

4.1.2.3 Расчет освещенности рабочего кабинета инженера структурного подразделения «Шахматный клуб» ГОУ ВПО «СибГИУ».

Высота – 4 м, длина – 4 м, ширина – 3 м

Потолок и стены рабочего кабинета инженера структурного подразделения «Шахматный клуб» ГОУ ВПО «СибГИУ» побелены. Используются лампы ЛБ-40.

Рассчитаем количество светильников, необходимое в рабочем кабинете инженера структурного подразделения «Шахматный клуб» ГОУ ВПО «СибГИУ», по методу коэффициента использования светового потока [5].



Ен = 300 лк

S = 4\* 3 = 12



Z = 1.1

k = 1.3

F = 3000 лм

n = 2

I = (4\*3)/(2.78\*(4+3))= 0,62

h = 4 – 0.5 – 0.72 = 2.78 м

hc = 0.5 м (высота светильника)

hр = 0.72 м (высота рабочего стола)

η = 0,29

(шт)



Для освещения рабочего кабинета инженера структурного подразделения «Шахматный клуб» ГОУ ВПО «СибГИУ» размерами 4м х 3м х 4 м потребуется 3 светильника с люминесцентными лампами типа ЛБ – 40. В рабочем кабинете инженера установлено 3 светильника, что соответствует [5].

## 

## 4.2 Чрезвычайные ситуации

### 4.2.1 Электробезопасность

При пользовании средствами вычислительной техники и периферийным оборудованием каждый пользователь должен внимательно и осторожно обращаться с электропроводкой, приборами и аппаратами и всегда помнить, что пренебрежение правилами безопасности угрожает и здоровью, и жизни человека.

Во избежание поражения электрическим током необходимо твердо знать и выполнять следующие правила безопасного пользования электроэнергией.

Необходимо постоянно следить на своем рабочем месте за исправным состоянием электропроводки, выключателей, штепсельных розеток, при помощи которых оборудование включается в сеть, и заземления. При обнаружении неисправности немедленно обесточить электрооборудование, оповестить администрацию. Продолжение работы возможно только после устранения неисправности.

Во избежание повреждения изоляции проводов и возникновения коротких замыканий не разрешается:

* + вешать что-либо на провода;
  + закрашивать и белить шнуры и провода;
  + закладывать провода и шнуры за газовые и водопроводные трубы, за батареи отопительной системы;
  + выдергивать штепсельную вилку из розетки за шнур, усилие должно быть приложено к корпусу вилки.

Для исключения поражения электрическим током запрещается:

* + часто включать и выключать компьютер без необходимости;
  + прикасаться к экрану и к тыльной стороне блоков компьютера;
  + работать на средствах вычислительной техники и периферийном оборудовании мокрыми руками;
  + работать на средствах вычислительной техники и периферийном оборудовании, имеющих нарушения целостности корпуса, нарушения изоляции проводов, неисправную индикацию включения питания, с признаками электрического напряжения на корпусе;
  + класть на средства вычислительной техники и периферийное оборудование посторонние предметы.

Запрещается очищать от пыли и загрязнения электрооборудование, находящееся под напряжением.

Запрещается проверять работоспособность электрооборудования в неприспособленных для эксплуатации помещениях с токопроводящими, сырыми полами, не позволяющими заземлить доступные металлические части.

Ремонт электроаппаратуры производится только специалистами-техниками с соблюдением необходимых технических требований.

Недопустимо под напряжением проводить ремонт средств вычислительной техники и периферийного оборудования.

Во избежание поражения электрическим током, при пользовании электроприборами нельзя касаться одновременно каких-либо трубопроводов, батарей отопления, металлических конструкций , соединенных с землей.

При пользовании электроэнергией в сырых помещениях соблюдать особую осторожность.

При обнаружении оборвавшегося провода необходимо немедленно сообщить об этом администрации, принять меры по исключению контакта с ним людей. Прикосновение к проводу опасно для жизни.

Спасение пострадавшего при поражении электрическим током главным образом зависит от быстроты освобождения его от действия током.

Во всех случаях поражения человека электрическим током немедленно вызывают врача. До прибытия врача нужно, не теряя времени, приступить к оказанию первой помощи пострадавшему.

Необходимо немедленно начать производить искусственное дыхание, наиболее эффективным из которых является метод “рот в рот”, а также наружный массаж сердца.

Искусственное дыхание пораженному электрическим током производится вплоть до прибытия врача.

### 4.2.2 Пожарная безопасность

В соответствии с [8] кабинеты с ВДТ по пожарной опасности относятся к категории “В”, т.к. в помещении находятся электронная аппаратура, офисная техника и бумага.

Причины, которые могут повести за собой возгорание различных предметов, заключены в следующем:

* перегрузки электрических сетей;
* неисправность электропроводки, электрооборудования и неумелое их использование;
* самовозгораемость некоторых веществ;
* несоблюдение правил техники безопасности;
* курение и применение открытого огня в неположенном месте;
* короткое замыкание в электрооборудовании;
* диверсионно-террористические акции.

Порядок и последовательность действий:

* сообщение о пожаре;
* эвакуация студентов из горящих помещений;
* сверка списочного состава с фактическим наличием эвакуированных;
* тушение возникшего пожара.

Необходимо знать, что:

* при обнаружении пожара, загорании или задымлении немедленно звонить по телефону 01;
* при возникновении пожара эвакуация производится через запасные выходы корпуса, столовой и здания главного корпуса «СибГИУ»;
* при возникновении пожара в части здания, имеющей выход на улицу, эвакуацию производить через запасные выходы столовой, главного корпуса и поточных аудиторий;
* все эвакуированные пересчитываются и сверяются по списку;
* тушение загорания производится немедленно с момента его обнаружения.

Исполнители:

* первый заметивший дым и дежурный по корпусу;
* дежурный помещений и преподаватели, проводившие занятия;
* преподаватели и старосты групп;
* члены ДПД.

В СибГИУ имеется и успешно функционирует собственная противопожарная система, которая состоит из противопожарной службы, средств пожаротушения и организационно-технических мероприятий.

Противопожарная служба состоит из штатных работников отдела обеспечения порядка, которые своими силами способны тушить пожар любой степени сложности. Численность работников отдела обеспечения порядка 30 человек. Главное в их работе – предотвратить пожар. Если возгорание произошло, то основная задача – остановить его, не дать перерасти в пожар. Основной причиной возгорания в здании университета является незатушенная сигарета. Повреждение электропроводки также может привести к возгоранию, а затем и к пожару.

К средствам пожаротушения относятся средства коллективной и индивидуальной защиты.

Средства коллективной защиты:

* огнетушители порошковые ОП-3;
* гидранты;
* пожарные рукава.

Средства индивидуальной защиты:

* противогазы ГП5;
* респираторы.

Все виды средств защиты в достаточном количестве имеются на постах службы обеспечения порядка.

В главном корпусе СибГИУ располагается пожарный пост №1, где находятся первичные средства пожаротушения (на начальной стадии развития пожара): огнетушители, песок, войлок, кошма, асбестовое полотно, ведра, лопаты и др. Там же находится и форма.

Источники водоснабжения:

* пожарные гидранты;
* пожарные краны – 58 шт.

Пожарные краны размещены в шкафчиках, на дверцы которых нанесен буквенный индекс “ПК” и порядковый номер крана. Каждый пожарный кран оборудован рукавом длиной 20 м и пожарным стволом со спрыском. Пожарные рукава бывают всасывающие и напорные. При помощи всасывающих рукавов забирают воду из водоисточников пожарным насосом. Напорные рукава применяют для подачи воды к месту пожара от водоисточника или водопроводного крана.

Пожарная сигнализация в главном корпусе находится только в компьютерных классах, в виде:

* дымовых датчиков;
* термических датчиков.

Основным мероприятием по защите людей при пожаре является полная экстренная эвакуация. Планы эвакуации висят на всех этажах во всех корпусах университета.

### 4.2.3 Организационно-штатная структура по ГО и ЧС.

В руководящий состав входят начальник ГОЧС (Кулагин Николай Михайлович), начальник штаба ГОЧС (Герасимова Анна Алексеевна), председатель эвакуационной комиссии (Галевский Геннадий Владиславович) и начальник СЭП (Рябцев Вадим Олегович).

Силы ГОЧС составляют спасательная команда (105 человек), группы оповещения и связи (30 человек), команда охраны общественного порядка (45 человек), санитарные дружины (72 человека) и звенья пожаротушения (20 человек).

### 4.2.4 Чрезвычайные ситуации, которые могут возникнуть на территории ГОУ ВПО «СибГИУ».

Чрезвычайные ситуации, которые могут возникнуть на территории ГОУ ВПО «СибГИУ», связанные с опасностью для здоровья, гибелью и увечьем студентов и персонала:

* пожаро-взрывоопасная ситуация;
* утечка хлора в плавательном бассейне спорткомплекса университета;
* производственные аварии на Левобережном водозаборе, на его технологических линиях и емкостях с хлором, при неблагоприятных метеоусловиях и направлении ветра от места аварии в сторону университета;
* аварии на железной дороге, пролегающей в непосредственной близости к ГОУ ВПО «СибГИУ» с разливом АХОВ или взрывом особо опасного груза;
* неблагополучная обстановка в городе, в центральном районе, связанная с возникновением очагов особо опасных инфекционных заболеваний;
* ураганный ветер, землетрясения, наводнения и другие стихийные бедствия.

### 4.2.5 Способы оповещения при ЧС

Звучание сирен, прерывистые гудки предприятий и транспортных средств означает подачу предупредительного сигнала “Внимание всем!”.

Услышав его, следует немедленно включить громкоговоритель, радио и телеприемник (в любое время суток), прослушайте экстренное сообщение управления по делам ГО и ЧС.

После сигнала “Внимание всем!” может последовать и другая информация, например, об угрозе возникновения ЧС, о надвигающейся опасности радиоактивного или химического заражения, возникновения “Воздушной опасности”. В этих случаях будет передано краткое сообщение о порядке действий и правилах поведения при данной ЧС.

В рабочее время можно получить информацию о передаче сигнала “Внимание всем!” от руководителей университета (ГО и ЧС). Действовать необходимо согласно их указаниям.

Следует соблюдать спокойствие и порядок. Необходимо быть внимательным к сообщениям работников гражданской обороны.

### 4.2.6 Действия по видам сообщения при передачи сигнала «Внимание всем!»

При аварии с выбросом АХОВ и неблагоприятных метеоусловиях необходимо:

* провести герметизацию помещений;
* отключить вентиляционные устройства, электронагревательные приборы, перекрыть воду, газ;
* подготовить и проверить средства индивидуальной защиты органов дыхания и кожи, при их отсутствии изготовить простейшие средства индивидуальной защиты;
* взять с собой документы, деньги, продукты;
* покинуть свое местонахождение и выйти по кратчайшему пути в направлении, указанном ГОЧС.

Все должны выйти в безопасное место в кротчайшее время до подхода зараженного хлором воздушного облака.

Если при аварии на любом из химически опасных объектов района или на железной дороге и направлении ветра в сторону объекта или жилых домов получена информация о грозящей опасности попадания в зону заражения АХОВ, то необходимо немедленно покинуть помещения университета, выйти из очага заражения и двигаться в направлении перпендикулярном движению ветра.

Время выхода из очага заражения весьма ограничено и не должно превышать 15-20 минут с момента аварии, расстояние выхода около 200 – 300 метров. Выход осуществлять, дыша через смоченную в воде матерчатую повязку.

При возникновении “Воздушной опасности” (при военном положении) необходимо:

* отключить свет, газ, воду;
* взять средства индивидуальной защиты, документы, запас питания и воды;
* помочь больному выйти на улицу;
* как можно быстрее дойти до защитного сооружения или укрыться в складках местности.

При отбое действия по сигналам ГО по указанию ГОЧС или представителя охраны общественного порядка выйти из укрытий и возвратиться к месту работы. Снимать средства индивидуальной защиты – по особому распоряжению. В дальнейшем действовать по указанию управления ГОЧС или старшего на рабочем месте.

При других ЧС действовать применительно к случившейся ЧС.

### 4.2.7 Организация защиты сотрудников и студентов СибГИУ при возникновении ЧС мирного и военного времени

Инженерная защита: студенты и персонал университета укрываются в защитных сооружениях – подвальных помещениях корпусов, спорткомплексе, столовой.

Обеспечение средствами индивидуальной защиты:

* в экстренных случаях (при авариях с АХОВ) изготавливаются простейшие средства защиты органов дыхания (матерчатые повязки) из подручного материала, могут быть использованы промышленные респираторы или ватно-марлевые повязки, имеющиеся на рабочем месте;
* при поступлении распоряжения управления ГО и ЧС на получение средств индивидуальной защиты (СИЗ) из запасов объекта (или централизованного) в помещениях главного корпуса, блока поточных аудиторий, горно-технологического корпуса разворачиваются “Пункты выдачи СИЗ”.

Выдача СИЗ осуществляется поочередно для всех структурных подразделений по графику, составленному штабом ГОЧС университета.

Организация эвакуации университета:

* в мирное время (экстренная эвакуация):
  + при авариях на предприятиях города (района) или ж/д с выбросом АХОВ – в места, указанные при получении сигнала “Внимание всем!” в направлении, перпендикулярном движению зараженного облака (с учетом особенностей АХОВ);
  + при повышении радиационного фона свыше 60 мкр/час – по распоряжению управления ГОЧС района или руководителя объектов в места, определенные администрацией области;
* в военное время (плановая эвакуация):
  + с получением распоряжения управления ГОЧС эвакуацию осуществляет эвакуационная комиссия университета. Эвакуация работников и студентов СибГИУ и членов их семей осуществляется по железной дороге в населенный пункт г. Гурьевск Кемеровской области в том числе: железнодорожным транспортом, через СЭП №20, разворачиваемый в помещении блока поточных аудиторий;
  + в загородных зонах размещение происходит в помещениях 3-х автохозяйств и жилых домах (подселением) согласно плану управления ГОЧС.

## 4.3 Экологичность проекта

С точки зрения энергоснабжения мониторы, используемые в рабочем процессе, имеют встроенную систему экономии электроэнергии – PowerSaver. Эта система сохраняет электроэнергию посредством переключения монитора в режим с низким потреблением энергии, когда он не используется в течение определенного периода времени.

Возможные режимы работы – рабочий, Stand-by, Suspend, выключен. Данная система успешно работает с VESA DPMS – совместимой видеокартой, установленной в компьютере. Более детальная информация сведена в таблице 3.

Таблица 17 – Характеристики режимов сохранения электроэнергии [10].

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Состояние | Рабочее | Режим сохранения электроэнергии EPA/NUTEK | | |
| Режим Stand-by | Режим Suspend | Режим Power-off |
| Горизонтальная синхр. | Активный | Неактивный | Активный | Неактивный |
| Вертикальная синхр. | Активный | Активный | Неактивный | Неактивный |
| Видео | Активный | Темный экран | Темный экран | Темный экран |
| Потребляемая мощность | 73 Вт (макс)  55 Вт (номин) | 50 Вт (номин) | Менее 15 Вт | Менее 5 Вт |

Монитор автоматически возвращается к режиму нормального функционирования при восстановлении горизонтальной и вертикальной развертки.

Заключение

В результате выполнения дипломного проекта главная задача, поставленная в разделе «Постановка задачи» – реализовать специализированное шахматное программное обеспечение, отвечающее всем критериям и ограничениям – была выполнена полностью.

Для решения поставленной задачи были использованы методологии разработки программного обеспечения, детально рассмотренные в дипломном проекте, а также стандартные средства программных продуктов, представленных в настоящей работе. В связи с невозможностью использования прототипов специализированного шахматного программного обеспечения в ходе разработки были применены приемы экстремальной методологии разработки ПО.

По данным анализа результатов тестовых испытаний, представленных в приложении 8, было выявлено полное соответствие реализованного специализированного шахматного программного обеспечения всем критериям и ограничениям.

Список использованных источников

1. Санитарные нормы и правила: СНиП 41.01-03. Отопление, вентиляция и кондиционирование. - Введен 01.01 2004. Технорматив: информационная система. – 58 с.
2. Санитарные правила и нормы: СанПиН 2.2.2/2.4.1340-03. Гигиенические требования к персональным электронно-вычислительным машинам и организация работы. – Введен 30.06.2003. Технорматив: информационная система – 26 с.
3. Система стандартов безопасности труда: ГОСТ 12.0.003-74. Опасные и вредные факторы. – Введен 18.11.1974. М.: - Госстандарт СССР, 1974г., - 55с.
4. Система стандартов безопасности труда: ГОСТ 12.005-88. Общие санитарно-гигиенические требования к воздуху рабочей зоны. – Введен 29.09.1988. М.: - Государственный комитет СССР по стандартам, 1988г. - 28с
5. Санитарные правила и нормы: СанПиН 2.2.1/2.2.2.1278-03. Гигиенические требования к естественному, искусственному и совмещенному освещению жилых и общественных зданий. – Введен 15.06.2003.Технорматив: информационная система. – 27 с.
6. Санитарные нормы: СН 2.2.4/2.1.8.562-96. Шум на рабочих местах, в помещениях жилых, общественных зданий и на территории жилой застройки. – Введен 31.10.1996. М.: - Госкомсанэпиднадзора России , 1996г. - 34с.
7. Санитарные правила и нормы: СанПиН 2.2.4.1191-03 Электромагнитные поля в производственных условиях - Введен 01.05.2003.СПб.: Минздрав РФ, 2003г. - 17 с.
8. Нормы пожарной безопасности: НПБ 105-2003. Определение категорий помещений, зданий и наружных установок по взрывопожарной и пожарной опасности. - Введен 18.06.2003. М.: МЧС РФ, 2003г. - 26с.
9. Санитарные нормы и правила: СНиП 21.01-97. Пожарная безопасность зданий и сооружений. - Введен 01.01.1998. Технорматив: информационная система. – 23 с.
10. Система стандартов безопасности труда: ГОСТ Р50949-96. Средства отображения информации индивидуального пользования. Методы измерений и оценки эргономических параметров и параметров безопасности. – Введен 01.07.1997. М: Госстандарт России, 1996 – 27 с.
11. Фултон Х. Программирование на языке Ruby. Идеалогия языка, теория и практика приминения. / Х.Фултон – М.: ДМК-Пресс, 2007. – 688 с.
12. Томас Д. Гибкая разработка веб-приложений в среде Rails. / Томас Д., Хэнссон Д.Х. – СПб.:Питер, 2008. – 716 с.
13. Монахов В.В. Язык программирования Java и среда NetBeans. / Монахов В.В. – СПб.: БХВ-Петербург, 2008. – 640 с.
14. Шильдт Х. Java. Методики программирования Шилдта. / Шильдт Х. – М.: Вильямс, 2008. – 512 с.
15. Гудман Д. JavaScript. Библия пользователя. Издание 5-е. / Гудман Д., Моррисон М. – М.: Вильямс, 2006. – 1184 с.
16. Флэнаган Д. JavaScript. Подробное руководтство. Издание 4-е. / Флэнаган Д. – М.: Символ-плюс, 2004. – 960 с.
17. Муссиано Ч. HTML и XHTML. Подробное руководство. Издание 6-е. / Муссиано Ч., Кеннеди Б. – М.: Символ-плюс, 2008. – 752 с.
18. Лазаро И.К. Полный справочник по HTML, CSS и JavaScript. Справочник профессионала. / Лазаро И.К. – М.: Третий Рим, 2007. – 1168 с.
19. Бобко Т.В. Методические указания к выполнению экономической части дипломных проектов. / Бобко Т.В. – Новокузнецк: СибГИУ, 2007. – 47 с.
20. Бек К. Экстремальное программирование. / Бек К. – СПб.:Питер, 2002. -224с.
21. Избачков Ю.С. Информационные системы: Учебник для вузов. 2-е издание. / Избачков Ю.С., Петров В.Н. – СПб.: Питер, 2005. – 656 с.

Приложение А

Календарный план работ по созданию системы

|  |  |  |  |
| --- | --- | --- | --- |
| № | Наименование этапа | Сроки исполнения | Исполнитель |
| 1 | Разработка ТЗ на ИС «Шахматный клуб» по пунктам 1,2 | 01.08.2007 –  01.09.2007 | Ширяев А.С. |
| 2 | Разработка ТЗ на ИС «Шахматный клуб» по пунктам 3-5 | 01.09.2007 –  01.11.2007 | Ширяев А.С. |
| 3 | Разработка ТЗ на ИС «Шахматный клуб» по пунктам 6-9 | 01.11.2007 –  01.12.2007 | Ширяев А.С. |
| 4 | Закупка оборудования, создание системы, разработка необходимой документации на систему | 01.12.2007 –  01.02.2008 | Персонал шахматного клуба ГОУ ВПО «СибГИУ», персонал МУДОД «СДЮСШОР по шахматам им. Б. А. Кустова» |
| 5 | Предварительные испытания системы | 02.02.2008 –  03.02.2008 | Персонал шахматного клуба ГОУ ВПО «СибГИУ», персонал МУДОД «СДЮСШОР по шахматам им. Б. А. Кустова»,  Ширяев А.С. |
| 6 | Опытная эксплуатация | 03.02.2008 –  03.03.2008 | Персонал шахматного клуба ГОУ ВПО «СибГИУ», персонал МУДОД «СДЮСШОР по шахматам им. Б. А. Кустова» |

Приложение Б

Комплектность документации на систему

|  |
| --- |
| Наименование документации |
| Документация по общесистемным решениям:  Схема функциональной структуры  Описание автоматизируемых функций и функций по представлению информации  Общее описание системы |
| Документация по организационному обеспечению:  Описание организационной структуры  Организационно-техническая инструкция |
| Документация по техническому обеспечению:  Описание комплекса технических средств  Структурная схема комплекса технических средств  Документация на технические средства |
| Документация по информационному обеспечению:  Перечень входных сигналов и данных  Перечень выходных сигналов и данных  Описание информационного обеспечения системы  Описание организации информационной базы |
| Документация по программному обеспечению:  Описание программного обеспечения  Документация на программное обеспечение |

Приложение В

Листинг программы-транслятора шахматных партий

Контроллеры (controller)

main\_controller.rb

---

# Основной контроллер портала

class MainController < ApplicationController

# Редирект на главную страницу

def index

render :action => 'logo'

end

# Главная страница

def logo

end

# Старая страница

def main

render :action => 'logo'

end

# Показ партий в онлайн режиме (трансляции)

def online

@online\_games = CurrentGame.find :all

end

# Показ партий в оффлайн режиме (уже закончившиеся)

def offline

end

# Напоминание при потере пароля или логина

def remind

end

# Вход в личный кабинет пользователя

def enter

unless session[:user\_id].nil?

redirect\_to "/users/room/#{session[:user\_id]}"

end

end

# Показ трансляции

def game

unless params[:id].nil?

@game = Game.find params[:id]

else

render :action => 'logo'

end

end

# Возвращает текст партии в условленном формате

def game\_moves

@game = Game.find(params[:id])

render\_text @game.moves

end

# Авторизация

def login

session[:user\_id] = nil

user = User.authenticate(params[:login], params[:password])

if user.nil?

redirect\_to :action => 'enter'

else

session[:user\_id] = user.id

redirect\_to "/users/room/#{user.id}"

end

end

# Выход пользователя

def unlogin

session[:user\_id] = nil

redirect\_to :action => 'logo'

end

# Переход к регистрации

def register

redirect\_to '/users/new'

end

end

cities\_controller.rb

---

class CitiesController < ApplicationController

layout 'main'

scaffold :cities

end

clock\_reglaments\_controller.rb

---

class ClockReglamentsController < ApplicationController

layout 'main'

scaffold :clock\_reglament

end

countries\_controller.rb

---

class CountriesController < ApplicationController

layout 'main'

scaffold :countries

end

game\_infos\_controller.rb

---

class GameInfosController < ApplicationController

layout 'main'

scaffold :game\_info

end

games\_controller.rb

---

class GamesController < ApplicationController

layout 'main'

scaffold :game

end

moves\_controller.rb

---

class MovesController < ApplicationController

layout 'main'

scaffold :move

end

players\_controller.rb

---

class PlayersController < ApplicationController

layout 'main'

scaffold :player

end

tournament\_views\_controller.rb

---

class TournamentViewsController < ApplicationController

layout 'main'

scaffold :tournament\_view

end

tournaments\_controller.rb

---

class TournamentsController < ApplicationController

layout 'main'

scaffold :tournament

def index

render :action => 'list'

end

end

users\_controller.rb

---

class UsersController < ApplicationController

layout 'main'

scaffold :users

def index

render :action => 'room'

end

def room

end

end

Помощники (helper)

main\_helper.rb

---

module MainHelper

def print\_current\_tournament

html = ""

Tournament.find\_all\_by\_status('open').each do |tournament|

city = tournament.city\_id.nil? ? nil : City.find(tournament.city\_id)

html << "<tr>"

html << "<td></td>"

html << "<td>" + tournament.name + "</td>"

html << "<td>" + (city.nil? ? "" : city.name) + "</td>"

html << "</tr>"

end

html

end

def render\_game\_info

"Тестовая партия из тестового турнира"

end

def render\_moves

moves = "<div id=\"moves\"></div>"

moves = "<table width='100%' align='center'>"

moves += "<tr><th width='2%'></th><th width='45%'></th><th ></th><th width='45%'></th><th width='2%'></th></tr>"

moves += "<tr><td>1.</td><td>b</td><td>c</td><td>d</td><td>e</td></tr>"

moves += "<tr><td>2.</td><td>b</td><td>c</td><td>d</td><td>e</td></tr>"

moves += "<tr><td>3.</td><td>b</td><td>c</td><td>d</td><td>e</td></tr>"

moves += "<tr><td>4.</td><td>b</td><td>c</td><td>d</td><td>e</td></tr>"

moves += "<tr><td>5.</td><td>b</td><td>c</td><td>d</td><td>e</td></tr>"

moves += "<tr><td>6.</td><td>b</td><td>c</td><td>d</td><td>e</td></tr>"

moves += "<tr><td>7.</td><td>b</td><td>c</td><td>d</td><td>e</td></tr>"

moves += "<tr><td>8.</td><td>b</td><td>c</td><td>d</td><td>e</td></tr>"

moves += "<tr><td>9.</td><td>b</td><td>c</td><td>d</td><td>e</td></tr>"

moves += "<tr><td>10.</td><td>b</td><td>c</td><td>d</td><td>e</td></tr>"

moves += "<tr><td>11.</td><td>b</td><td>c</td><td>d</td><td>e</td></tr>"

moves += "<tr><td>12.</td><td>b</td><td>c</td><td>d</td><td>e</td></tr>"

moves += "<tr><td>13.</td><td>b</td><td>c</td><td>d</td><td>e</td></tr>"

moves += "<tr><td>14.</td><td>b</td><td>c</td><td>d</td><td>e</td></tr>"

moves += "<tr><td>15.</td><td>b</td><td>c</td><td>d</td><td>e</td></tr>"

moves += "<tr><td>16.</td><td>b</td><td>c</td><td>d</td><td>e</td></tr>"

moves += "<tr><td>17.</td><td>b</td><td>c</td><td>d</td><td>e</td></tr>"

moves += "<tr><td>18.</td><td>b</td><td>c</td><td>d</td><td>e</td></tr>"

moves += "<tr><td>19.</td><td>b</td><td>c</td><td>d</td><td>e</td></tr>"

moves += "<tr><td>20.</td><td>b</td><td>c</td><td>d</td><td>e</td></tr>"

moves += "<tr><td colspan='5' align='center'>"

moves += image\_tag('go\_begin.jpg', :id => 'go\_begin', :onclick => 'goBegin();'), :onmouseover => 'new Effect.Scale("go\_begin", 110)'), :onmouseout => 'new Effect.Scale("go\_begin", 90)') + '&nbsp;'

moves += image\_tag('go\_prev.jpg', :id => 'go\_prev', :onclick => 'goPrev();') :onmouseover => 'new Effect.Scale("go\_prev", 110)', :onmouseout => 'new Effect.Scale("go\_prev", 90)') + '&nbsp;'

moves += image\_tag('go\_refresh.jpg', :id => 'go\_refresh', :onclick => 'goRefresh();') :onmouseover => 'new Effect.Scale("go\_refresh", 110)', :onmouseout => 'new Effect.Scale("go\_refresh", 90)') + '&nbsp;'

moves += image\_tag('go\_next.jpg', :id => 'go\_next', :onclick => 'goNext();'):onmouseover => 'new Effect.Scale("go\_next", 110)', :onmouseout => 'new Effect.Scale("go\_next", 90)') + '&nbsp;'

moves += image\_tag('go\_end.jpg', :id => 'go\_end', :onclick => 'goEnd();')#:onmouseover => 'new Effect.Scale("go\_end", 110)', :onmouseout => 'new Effect.Scale("go\_end", 90)') + '&nbsp;'

moves += "</td></tr>"

moves += "</table>";

moves

end

def render\_board

return "<table class=\"board\" id=\"board\">

<tr>

<td class=\"board\_label\_digit\">8</td>

<td class=\"board\_white\_field\" id=\"fieldA8\"></td>

<td class=\"board\_black\_field\" id=\"fieldB8\"></td>

<td class=\"board\_white\_field\" id=\"fieldC8\"></td>

<td class=\"board\_black\_field\" id=\"fieldD8\"></td>

<td class=\"board\_white\_field\" id=\"fieldE8\"></td>

<td class=\"board\_black\_field\" id=\"fieldF8\"></td>

<td class=\"board\_white\_field\" id=\"fieldG8\"></td>

<td class=\"board\_black\_field\" id=\"fieldH8\"></td>

</tr>

<tr>

<td class=\"board\_label\_digit\">7</td>

<td class=\"board\_black\_field\" id=\"fieldA7\"></td>

<td class=\"board\_white\_field\" id=\"fieldB7\"></td>

<td class=\"board\_black\_field\" id=\"fieldC7\"></td>

<td class=\"board\_white\_field\" id=\"fieldD7\"></td>

<td class=\"board\_black\_field\" id=\"fieldE7\"></td>

<td class=\"board\_white\_field\" id=\"fieldF7\"></td>

<td class=\"board\_black\_field\" id=\"fieldG7\"></td>

<td class=\"board\_white\_field\" id=\"fieldH7\"></td>

</tr>

<tr>

<td class=\"board\_label\_digit\">6</td>

<td class=\"board\_white\_field\" id=\"fieldA6\"></td>

<td class=\"board\_black\_field\" id=\"fieldB6\"></td>

<td class=\"board\_white\_field\" id=\"fieldC6\"></td>

<td class=\"board\_black\_field\" id=\"fieldD6\"></td>

<td class=\"board\_white\_field\" id=\"fieldE6\"></td>

<td class=\"board\_black\_field\" id=\"fieldF6\"></td>

<td class=\"board\_white\_field\" id=\"fieldG6\"></td>

<td class=\"board\_black\_field\" id=\"fieldH6\"></td>

</tr>

<tr>

<td class=\"board\_label\_digit\">5</td>

<td nowrap class=\"board\_black\_field\" id=\"fieldA5\"></td>

<td nowrap class=\"board\_white\_field\" id=\"fieldB5\"></td>

<td nowrap class=\"board\_black\_field\" id=\"fieldC5\"></td>

<td nowrap class=\"board\_white\_field\" id=\"fieldD5\"></td>

<td nowrap class=\"board\_black\_field\" id=\"fieldE5\"></td>

<td nowrap class=\"board\_white\_field\" id=\"fieldF5\"></td>

<td nowrap class=\"board\_black\_field\" id=\"fieldG5\"></td>

<td nowrap class=\"board\_white\_field\" id=\"fieldH5\"></td>

</tr>

<tr>

<td class=\"board\_label\_digit\">4</td>

<td class=\"board\_white\_field\" id=\"fieldA4\"></td>

<td class=\"board\_black\_field\" id=\"fieldB4\"></td>

<td class=\"board\_white\_field\" id=\"fieldC4\"></td>

<td class=\"board\_black\_field\" id=\"fieldD4\"></td>

<td class=\"board\_white\_field\" id=\"fieldE4\"></td>

<td class=\"board\_black\_field\" id=\"fieldF4\"></td>

<td class=\"board\_white\_field\" id=\"fieldG4\"></td>

<td class=\"board\_black\_field\" id=\"fieldH4\"></td>

</tr>

<tr>

<td class=\"board\_label\_digit\">3</td>

<td class=\"board\_black\_field\" id=\"fieldA3\"></td>

<td class=\"board\_white\_field\" id=\"fieldB3\"></td>

<td class=\"board\_black\_field\" id=\"fieldC3\"></td>

<td class=\"board\_white\_field\" id=\"fieldD3\"></td>

<td class=\"board\_black\_field\" id=\"fieldE3\"></td>

<td class=\"board\_white\_field\" id=\"fieldF3\"></td>

<td class=\"board\_black\_field\" id=\"fieldG3\"></td>

<td class=\"board\_white\_field\" id=\"fieldH3\"></td>

</tr>

<tr>

<td class=\"board\_label\_digit\">2</td>

<td class=\"board\_white\_field\" id=\"fieldA2\"></td>

<td class=\"board\_black\_field\" id=\"fieldB2\"></td>

<td class=\"board\_white\_field\" id=\"fieldC2\"></td>

<td class=\"board\_black\_field\" id=\"fieldD2\"></td>

<td class=\"board\_white\_field\" id=\"fieldE2\"></td>

<td class=\"board\_black\_field\" id=\"fieldF2\"></td>

<td class=\"board\_white\_field\" id=\"fieldG2\"></td>

<td class=\"board\_black\_field\" id=\"fieldH2\"></td>

</tr>

<tr>

<td class=\"board\_label\_digit\">1</td>

<td class=\"board\_black\_field\" id=\"fieldA1\"></td>

<td class=\"board\_white\_field\" id=\"fieldB1\"></td>

<td class=\"board\_black\_field\" id=\"fieldC1\"></td>

<td class=\"board\_white\_field\" id=\"fieldD1\"></td>

<td class=\"board\_black\_field\" id=\"fieldE1\"></td>

<td class=\"board\_white\_field\" id=\"fieldF1\"></td>

<td class=\"board\_black\_field\" id=\"fieldG1\"></td>

<td class=\"board\_white\_field\" id=\"fieldH1\"></td>

</tr>

<tr>

<td class=\"board\_lebel\_letter\">&nbsp;</td>

<td class=\"board\_lebel\_letter\">A</td>

<td class=\"board\_lebel\_letter\">B</td>

<td class=\"board\_lebel\_letter\">C</td>

<td class=\"board\_lebel\_letter\">D</td>

<td class=\"board\_lebel\_letter\">E</td>

<td class=\"board\_lebel\_letter\">F</td>

<td class=\"board\_lebel\_letter\">G</td>

<td class=\"board\_lebel\_letter\">H</td>

</tr>

</table>"

end

def render\_result

"1/2 - 1/2"

end

def render\_black\_time

"-:--:--"

end

def render\_white\_time

"-:--:--"

end

end

Представление (view)

layout/main.rhtml

---

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>

<meta http-equiv="content-type" content="text/html;charset=UTF-8" />

<title>Стартовая страница: <%= controller.action\_name %></title>

<%= stylesheet\_link\_tag 'scaffold' %>

<%= stylesheet\_link\_tag 'main' %>

<%= stylesheet\_link\_tag 'game' %>

<%= javascript\_include\_tag :defaults %>

<%= javascript\_include\_tag 'game' %>

</head>

<body style='background-color: #000000'>

<table bgcolor="#f4f4f4" border="0" width="80%" align="center" cellpadding="0" cellspacing="0" >

<tr>

<td width='3%'></td>

<td>

<h1><%= link\_to(image\_tag('dgt.gif', :size => '42x25', :border => 0, :alt => 'http://dgtprojects.com'), 'http://dgtprojects.com') %>&nbsp;&nbsp;&nbsp;

Сервер трансляции шахматных партий <%= link\_to 'rDGT', 'http://rdgt.org' %>&nbsp;&nbsp;&nbsp;

<%= link\_to(image\_tag('home.gif', :size => '12x12', :border => 0), '/main/index') %>

<%= link\_to(image\_tag('email.gif', :size => '12x12', :border => 0), '') %>

<%= link\_to(image\_tag('search.gif', :size => '12x12', :border => 0), '')%>

</h1><hr />

</td>

<td width='3%'></td>

</tr>

<tr>

<td></td>

<td>

<p style="color: green"><%= flash[:notice] %></p>

<%= yield %>

</td>

<td></td>

</tr>

<tr>

<td></td>

<td align="right"><br /><hr />

<%= mail\_to 'zarstudio@gmail.com', '[ Разработчики ]' %><br />&nbsp;

</td>

<td></td>

</tr>

</table>

</body>

</html>

main/enter.rhtml

---

<% form\_tag '/main/login' do %>

<table align='center' width='100%'>

<tr>

<td width='41%' align="right"><strong>Имя:&nbsp;<strong></td>

<td align="left"><%= text\_field\_tag :login, params[:login] %></td>

</tr>

<tr>

<td align="right"><strong>Пароль:&nbsp;</strong></td>

<td align="left"><%= password\_field\_tag :password, params[:password] %></td>

</tr>

<tr>

<td align="center" colspan="2"><%= submit\_tag 'Войти' %></td>

</tr>

<tr>

<td align="center" colspan="2"><%= link\_to '[ Регистрация ]', { :action => 'register' }%></td>

</tr>

<tr>

<td align="center" colspan="2"><%= link\_to '[ Забыли логин или пароль? ]', { :action => 'remind' }%></td>

</tr>

</table>

<% end %>

<br />

main/game.rhtml

---

<%= render\_game\_info %>

<br />

<br />

<table width='100%' cellspacing="0" cellpadding="0" border="0" id='table\_board'>

<tr align="center">

<td width='1%'></td>

<td width='50%' align="left"><%= image\_tag 'seek\_black.png', :size => '16x16' %></td>

<td></td>

<td width='45%'><%= link\_to @game.black\_player.to\_s, { :controller => 'players', :action => 'show', :id => @game.black\_player.id } %></td>

<td width='1%'></td>

</tr>

<tr align="center">

<td></td>

<td></td>

<td></td>

<td><%= render\_black\_time %></td>

<td></td>

</tr>

<tr>

<td></td>

<td><%= render\_board %></td>

<td></td>

<td style="background-color: #dadada"><%= render\_moves %></td>

<td></td>

</tr>

<tr align="center">

<td></td>

<td></td>

<td></td>

<td><%= render\_white\_time%></td>

<td></td>

</tr>

<tr align="center">

<td></td>

<td align="left"><%= image\_tag 'seek\_white.png', :size => '16x16' %></td>

<td></td>

<td><%= link\_to @game.white\_player.to\_s, { :controller => 'players', :action => 'show', :id => @game.white\_player.id} %></td>

<td></td>

</tr>

<tr align="center">

<td></td>

<td></td>

<td></td>

<td><%= render\_result %></td>

<td></td>

</tr>

<tr>

<td colspan="5"></td>

</tr>

</table>

<%= link\_to\_remote 'Обновить', :update => 'moves', :url => { :action => 'game\_moves', :id => @game.id } %>

<div id='debug'></div>

main/logo.rhtml

---

<table align='center'><tr><td align='center'>

<%= image\_tag 'logo.jpg', :alt => 'Шахматный портал rDGT', :width => '400px', :height => '280px' %>

<br />

<br /><%= link\_to '[ Трансляции ]', { :action => 'online' }%>

<br /><%= link\_to '[ Турниры ]' , { :action => 'offline' }%>

<br /><%= link\_to '[ Кабинет ]', { :action => 'enter' } %>

</td></tr></table>

main/online.rhtml

---

<!-- <%= link\_to '[ Просмотр ]', { :action => 'game' }%>-->

<table width='80%' align='center'>

<tr>

<th></th>

<th width='10%'>Начало</th>

<th width='25%'>Турнир</th>

<th width='30%'>Белые</th>

<th width='30%'>Черные</th>

</tr>

<% @online\_games.each do |game| %>

<tr align='center'>

<td class='td\_info'><%= link\_to(image\_tag('eye.gif', :size => '16x16', :onclick => '', :alt => 'Просмотр трансляции'), { :action => 'game', :id => game.id }) %></td>

<td class='td\_info'><%= game.begin.to\_s %></td>

<td class='td\_info'><%= link\_to game.tournament.to\_s, { :controller => 'tournaments', :action => 'show', :id => game.tournament.id } %></td>

<td class='td\_info'><%= link\_to game.white.to\_s, { :controller => 'players', :action => 'show', :id => game.white.id } %></td>

<td class='td\_info'><%= link\_to game.black.to\_s, { :controller => 'players', :action => 'show', :id => game.black.id } %></td>

</tr>

<% end %>

</table>

Модель (model)

---

class Cities < ActiveRecord::Base

end

class City < ActiveRecord::Base

end

class ClockReglament < ActiveRecord::Base

end

class Country < ActiveRecord::Base

end

class CurrentGame < ActiveRecord::Base

def init

@game = Game.find(self.game\_id) if @game.nil?

end

def tournament

init

@game.tournament

end

def white

init

@game.white\_player

end

def black

init

@game.black\_player

end

end

class Game < ActiveRecord::Base

def moves

result = ""

@moves = Move.find\_all\_by\_game\_id self.id

@moves.sort { |movea, moveb| movea.number <=> moveb.number }.each do |move|

result += move.to\_s + "|"

end

result

end

def tournament

info = GameInfo.find self.game\_info\_id unless self.game\_info\_id.nil?

t = Tournament.find info.tournament\_id

end

def white\_player

player = Player.new

player.id= 1

player.first\_name = "Иван"

player.second\_name = "Рыков"

player

end

def black\_player

player = Player.new

player.id= 1

player.first\_name = "Сергей"

player.second\_name = "Бедарев"

player

end

end

class GameInfo < ActiveRecord::Base

end

class LastMove < ActiveRecord::Base

end

class Move < ActiveRecord::Base

def to\_s

((self.white\_move.nil? or self.white\_move == "") ? "\*\*\*\*\*\*" : self.white\_move) +

((self.black\_move.nil? or self.black\_move == "") ? "\*\*\*\*\*\*" : self.black\_move) + ":" +

(self.white\_clock.nil? ? "" : self.white\_clock) + "-" + (self.black\_clock.nil? ? "" : self.black\_clock)

end

end

class Player < ActiveRecord::Base

def full\_name

return self.second\_name + " " + self.first\_name

end

def to\_s

full\_name

end

end

class Tournament < ActiveRecord::Base

def to\_s

self.name

end

end

class TournamentView < ActiveRecord::Base

end

require 'digest/md5'

class User < ActiveRecord::Base

validates\_presence\_of :login

validates\_uniqueness\_of :login

def self.authenticate(login, password)

user = User.find\_by\_name(login)

if user

expected\_password = encrypted\_password(password)

if user.password != expected\_password

user = nil

end

end

user

end

def self.encrypted\_password(password)

Digest::MD5.hexdigest("sdjkvkjeho2ijij2o3d2kn3dl2kn3dn23dkn2ld3n" + password)

end

end

Миграция (db migration)

---

ActiveRecord::Schema.define(:version => 12) do

create\_table "cities", :force => true do |t|

t.column "name", :string, :limit => 30

t.column "country\_id", :integer

t.column "description", :text

t.column "image", :binary

end

create\_table "clock\_reglaments", :force => true do |t|

t.column "title", :string, :limit => 30

t.column "total\_time", :integer

t.column "add", :boolean, :limit => nil, :default => false

t.column "add\_time", :integer, :default => 0

t.column "desciprion", :text

end

create\_table "countries", :force => true do |t|

t.column "name", :string, :limit => 30

t.column "code", :string, :limit => 30

t.column "image", :binary

t.column "description", :text

end

create\_table "current\_games", :force => true do |t|

t.column "game\_id", :integer

t.column "begin", :datetime

t.column "end", :datetime

t.column "description", :text

end

create\_table "game\_infos", :force => true do |t|

t.column "game\_id", :integer

t.column "white\_id", :integer

t.column "black\_id", :integer

t.column "tournament\_id", :integer

t.column "result", :integer

t.column "clock\_reglament\_id", :integer

t.column "round", :integer

t.column "debut\_code", :string, :limit => 30

t.column "description", :text

t.column "status", :string, :limit => 30

end

create\_table "games", :force => true do |t|

t.column "game\_info\_id", :integer

t.column "begin", :datetime

t.column "end", :datetime

t.column "desciption", :text

end

create\_table "last\_moves", :force => true do |t|

t.column "move\_id", :integer

t.column "move\_time", :datetime

end

create\_table "moves", :force => true do |t|

t.column "number", :integer

t.column "game\_id", :integer

t.column "white\_move", :string, :limit => 6

t.column "black\_move", :string, :limit => 6

t.column "white\_clock", :float

t.column "black\_clock", :float

t.column "white\_comment", :text

t.column "black\_comment", :text

end

create\_table "players", :force => true do |t|

t.column "first\_name", :string, :limit => 30

t.column "second\_name", :string, :limit => 30

t.column "country\_id", :integer

t.column "burn\_date", :datetime

t.column "fide\_rating", :integer

t.column "photo", :binary

t.column "description", :text

end

create\_table "tournament\_views", :force => true do |t|

t.column "name", :string, :limit => 30

t.column "system", :string, :limit => 30

t.column "is\_match", :boolean, :limit => nil, :default => false

t.column "is\_command", :boolean, :limit => nil, :default => false

t.column "total\_games", :integer, :default => 0

t.column "total\_rounds", :integer, :default => 0

t.column "description", :text

end

create\_table "tournaments", :force => true do |t|

t.column "name", :string, :limit => 30

t.column "category", :integer

t.column "city\_id", :integer

t.column "begin\_date", :datetime

t.column "end\_date", :datetime

t.column "tournament\_view\_id", :integer

t.column "status", :string, :limit => 30

t.column "description", :text

end

create\_table "users", :force => true do |t|

t.column "login", :string, :limit => 30

t.column "password", :string, :limit => 50

t.column "name", :string, :limit => 30

t.column "is\_admin", :boolean, :limit => nil, :default => false

t.column "email", :string, :limit => 30

t.column "webpage", :string, :limit => 30

t.column "desciption", :text

end

end

Javascript (javascript) – осуществляют демонстрацию шахматной партии, при помощи технологии AJAX обращаясь за обновлениями позиции на rDGT сервер.

game.js

---

var images = 'http://localhost:3000/images/classic/';

var moves\_url = '/main/game\_moves/1';

// Фигура

function figure(id, color, type, current\_field, last\_field, alive, board) {

this.id = id;

this.color = color;

this.type = type;

this.current\_field = current\_field;

this.last\_field = last\_field;

this.alive = alive;

this.set\_field = set\_field;

this.do\_move = do\_move;

this.reload = reload;

this.img = document.createElement('img');

this.img.id = this.id;

this.board = board;

this.reload();

//trace('create figure [' + this.id + ']');

}

// Поле

function field(id, color, vertical, horizontal, figure, board) {

this.id = id;

this.color = color;

this.vertical = vertical;

this.horizontal = horizontal;

this.figure = figure;

this.board = board;

this.repaint = repaint;

//trace('create field [' + this.id + ']');

}

// Ход

function move(number, color, from\_field, to\_field, figure, alive\_figure, is\_short\_castling, is\_long\_castling, prev\_move, next\_move, board, time) {

this.number = number;

this.color = color;

this.from\_field = from\_field;

this.to\_field = to\_field;

this.figure = figure;

this.alive\_figure = alive\_figure;

this.forward = forward;

this.backward = backward;

this.is\_short\_castling = is\_short\_castling;

this.is\_long\_castling = is\_long\_castling;

this.prev\_move = prev\_move;

this.next\_move = next\_move;

this.board = board;

this.white\_time = white\_time;

this.black\_time = black\_time;

}

// Коллекция ходов

function move\_collection(board) {

this.board = board;

this.get\_move = function(color, number) {

return color == 'white' ? this.moves[(number \* 2) - 1] : this.moves[number \* 2];

}

this.exists\_move = function(color, number) {

return get\_move(color, number) != null;

}

this.get\_current\_move = function() {

return this.current\_move;

}

this.add\_move = function(move) {

trace('add move: ' + move.number + " " + moveBy.color);

if(current\_move == null) {

this.first\_move = move;

this.current\_move = move;

} else {

move.prev\_move = this.current\_move;

this.current\_move.next\_move = move;

this.current\_move = move;

}

this.moves.push(move);

}

this.first\_move = null;

this.current\_move = null;

this.moves = new Array();

}

// Коллекция фигур

function figure\_collection(board) {

this.board = board;

this.wpA = new figure('wpA', 'white', 'pawn', null, null, true, board);

this.wpB = new figure('wpB', 'white', 'pawn', null, null, true, board);

this.wpC = new figure('wpC', 'white', 'pawn', null, null, true, board);

this.wpD = new figure('wpD', 'white', 'pawn', null, null, true, board);

this.wpE = new figure('wpE', 'white', 'pawn', null, null, true, board);

this.wpF = new figure('wpF', 'white', 'pawn', null, null, true, board);

this.wpG = new figure('wpG', 'white', 'pawn', null, null, true, board);

this.wpH = new figure('wpH', 'white', 'pawn', null, null, true, board);

this.wrA = new figure('wrA', 'white', 'rook', null, null, true, board);

this.wrH = new figure('wrH', 'white', 'rook', null, null, true, board);

this.whB = new figure('whB', 'white', 'horse', null, null, true, board);

this.whG = new figure('whG', 'white', 'horse', null, null, true, board);

this.wbC = new figure('wbC', 'white', 'bishop', null, null, true, board);

this.wbF = new figure('wbF', 'white', 'bishop', null, null, true, board);

this.wq = new figure('wq', 'white', 'queen', null, null, true, board);

this.wk = new figure('wk', 'white', 'king', null, null, true, board);

this.bpA = new figure('bpA', 'black', 'pawn', null, null, true, board);

this.bpB = new figure('bpB', 'black', 'pawn', null, null, true, board);

this.bpC = new figure('bpC', 'black', 'pawn', null, null, true, board);

this.bpD = new figure('bpD', 'black', 'pawn', null, null, true, board);

this.bpE = new figure('bpE', 'black', 'pawn', null, null, true, board);

this.bpF = new figure('bpF', 'black', 'pawn', null, null, true, board);

this.bpG = new figure('bpG', 'black', 'pawn', null, null, true, board);

this.bpH = new figure('bpH', 'black', 'pawn', null, null, true, board);

this.brA = new figure('brA', 'black', 'rook', null, null, true, board);

this.brH = new figure('brH', 'black', 'rook', null, null, true, board);

this.bhB = new figure('bhB', 'black', 'horse', null, null, true, board);

this.bhG = new figure('bhG', 'black', 'horse', null, null, true, board);

this.bbC = new figure('bbC', 'black', 'bishop', null, null, true, board);

this.bbF = new figure('bbF', 'black', 'bishop', null, null, true, board);

this.bq = new figure('bq', 'black', 'queen', null, null, true, board);

this.bk = new figure('bk', 'black', 'king', null, null, true, board);

}

// Коллекция полей шахматной доски

function field\_collection(board) {

this.get\_field = get\_field;

this.board = board;

this.toArray = function() {

return [this.A1, this.A2, this.A3, this.A4, this.A5, this.A6, this.A7, this.A8,

this.B1, this.B2, this.B3, this.B4, this.B5, this.B6, this.B7, this.B8,

this.C1, this.C2, this.C3, this.C4, this.C5, this.C6, this.C7, this.C8,

this.D1, this.D2, this.D3, this.D4, this.D5, this.D6, this.D7, this.D8,

this.E1, this.E2, this.E3, this.E4, this.E5, this.E6, this.E7, this.E8,

this.F1, this.F2, this.F3, this.F4, this.F5, this.F6, this.F7, this.F8,

this.G1, this.G2, this.G3, this.G4, this.G5, this.G6, this.G7, this.G8,

this.H1, this.H2, this.H3, this.H4, this.H5, this.H6, this.H7, this.H8];

}

this.A1 = new field('fieldA1', 'black', 'A', '1', null, board);

this.B1 = new field('fieldB1', 'white', 'B', '1', null, board);

this.C1 = new field('fieldC1', 'black', 'C', '1', null, board);

this.D1 = new field('fieldD1', 'white', 'D', '1', null, board);

this.E1 = new field('fieldE1', 'black', 'E', '1', null, board);

this.F1 = new field('fieldF1', 'white', 'F', '1', null, board);

this.G1 = new field('fieldG1', 'black', 'G', '1', null, board);

this.H1 = new field('fieldH1', 'white', 'H', '1', null, board);

this.A2 = new field('fieldA2', 'white', 'A', '2', null, board);

this.B2 = new field('fieldB2', 'black', 'B', '2', null, board);

this.C2 = new field('fieldC2', 'white', 'C', '2', null, board);

this.D2 = new field('fieldD2', 'black', 'D', '2', null, board);

this.E2 = new field('fieldE2', 'white', 'E', '2', null, board);

this.F2 = new field('fieldF2', 'black', 'F', '2', null, board);

this.G2 = new field('fieldG2', 'white', 'G', '2', null, board);

this.H2 = new field('fieldH2', 'black', 'H', '2', null, board);

this.A3 = new field('fieldA3', 'black', 'A', '3', null, board);

this.B3 = new field('fieldB3', 'white', 'B', '3', null, board);

this.C3 = new field('fieldC3', 'black', 'C', '3', null, board);

this.D3 = new field('fieldD3', 'white', 'D', '3', null, board);

this.E3 = new field('fieldE3', 'black', 'E', '3', null, board);

this.F3 = new field('fieldF3', 'white', 'F', '3', null, board);

this.G3 = new field('fieldG3', 'black', 'G', '3', null, board);

this.H3 = new field('fieldH3', 'white', 'H', '3', null, board);

this.A4 = new field('fieldA4', 'white', 'A', '4', null, board);

this.B4 = new field('fieldA4', 'black', 'B', '4', null, board);

this.C4 = new field('fieldC4', 'white', 'C', '4', null, board);

this.D4 = new field('fieldD4', 'black', 'D', '4', null, board);

this.E4 = new field('fieldE4', 'white', 'E', '4', null, board);

this.F4 = new field('fieldF4', 'black', 'F', '4', null, board);

this.G4 = new field('fieldG4', 'white', 'G', '4', null, board);

this.H4 = new field('fieldH4', 'black', 'H', '4', null, board);

this.A5 = new field('fieldA5', 'black', 'A', '5', null, board);

this.B5 = new field('fieldB5', 'white', 'B', '5', null, board);

this.C5 = new field('fieldC5', 'black', 'C', '5', null, board);

this.D5 = new field('fieldD5', 'white', 'D', '5', null, board);

this.E5 = new field('fieldE5', 'black', 'E', '5', null, board);

this.F5 = new field('fieldF5', 'white', 'F', '5', null, board);

this.G5 = new field('fieldG5', 'black', 'G', '5', null, board);

this.H5 = new field('fieldH5', 'white', 'H', '5', null, board);

this.A6 = new field('fieldA6', 'white', 'A', '6', null, board);

this.B6 = new field('fieldB6', 'black', 'B', '6', null, board);

this.C6 = new field('fieldC6', 'white', 'C', '6', null, board);

this.D6 = new field('fieldD6', 'black', 'D', '6', null, board);

this.E6 = new field('fieldE6', 'white', 'E', '6', null, board);

this.F6 = new field('fieldF6', 'black', 'F', '6', null, board);

this.G6 = new field('fieldG6', 'white', 'G', '6', null, board);

this.H6 = new field('fieldH6', 'black', 'H', '6', null, board);

this.A7 = new field('fieldA7', 'black', 'A', '7', null, board);

this.B7 = new field('fieldB7', 'white', 'B', '7', null, board);

this.C7 = new field('fieldC7', 'black', 'C', '7', null, board);

this.D7 = new field('fieldD7', 'white', 'D', '7', null, board);

this.E7 = new field('fieldE7', 'black', 'E', '7', null, board);

this.F7 = new field('fieldF7', 'white', 'F', '7', null, board);

this.G7 = new field('fieldG7', 'black', 'G', '7', null, board);

this.H7 = new field('fieldH7', 'white', 'H', '7', null, board);

this.A8 = new field('fieldA8', 'white', 'A', '8', null, board);

this.B8 = new field('fieldB8', 'black', 'B', '8', null, board);

this.C8 = new field('fieldC8', 'white', 'C', '8', null, board);

this.D8 = new field('fieldD8', 'black', 'D', '8', null, board);

this.E8 = new field('fieldE8', 'white', 'E', '8', null, board);

this.F8 = new field('fieldF8', 'black', 'F', '8', null, board);

this.G8 = new field('fieldG8', 'white', 'G', '8', null, board);

this.H8 = new field('fieldH8', 'black', 'H', '8', null, board);

}

function board() {

this.moves = new move\_collection(this);

this.figures = new figure\_collection(this);

this.fields = new field\_collection(this);

this.init = init;

this.last\_move = null;

// Обновляет позицию

this.refresh\_moves = function() {

// Ajax запрос к серверу

new Ajax.Request(moves\_url, {

method: 'get',

onSuccess: function(transport) {

all\_moves = transport.responseText;

trace('ajax: ' + all\_moves);

}

});

trace('call => parse\_moves(' + all\_moves + ')')

function existsWhite(move) {

return getWhiteMove(move) != "\*\*\*\*\*\*";

}

function existsBlack(move) {

return getBlackMove(move) != "\*\*\*\*\*\*";

}

function getWhiteMove(move) {

return move.substring(0,6);

}

function getWhiteTime(move) {

return move.substring(13, 19);

}

function getBlackMove(move) {

return move.substring(6,12);

}

function getBlackTime(move) {

return move.substring(20, 26);

}

function getFrom(color, move) {

}

function getTo(color, move) {

}

function isShortCastling(color, move) {

}

function isLongCastling(color, move) {

}

moves\_split = all\_moves.split('|');

moves\_strip = new Array();

for(i = 0; i < moves\_split.length; i++) {

if(moves\_split[i] != "") {

moves\_strip.push(moves\_split[i]);

trace('split: ' + moves\_split[i]);

}

}

var number = moves\_strip.length;

trace('number = ' + number);

var current\_move = this.moves.get\_current\_move();

trace('current\_move = ' + current\_move);

var current\_number = current\_move != null ? current\_move.number : 0;

trace('current\_number = ' + current\_number);

if(number == current\_number) {

if((current\_move.color == 'white') && existsBlack(moves\_strip[number - 1])) {

var move\_annotation = getBlackMove(moves\_strip[number - 1]);

var move\_time = getBlackTime(moves\_strip[number - 1]);

var fieldFrom = getFrom('black', move\_annotation);

var fieldTo = getTo('black', move\_annotation);

var move = new move(number, 'black', fieldFrom, fieldTo, fieldFrom.figure, fieldTo.figure, isShortCastling('black', move\_annotation), isLongCastling('black', move\_annotation), current\_move, null, this, move\_time);

this.moves.add\_move(move);

}

}

if(number > current\_number){

for(iMove = current\_move; iMove < number; iMove++) {

var move\_annotation = getWhiteMove(moves\_strip[number - 1]);

if(existsWhite(move\_annotation)) {

var move\_time = getWhiteTime(moves\_strip[number - 1]);

var fieldFrom = getFrom('white', move\_annotation);

var fieldTo = getTo('white', move\_annotation);

var move = new move(iMove + 1, 'white', fieldFrom, fieldTo, fieldFrom.figure, fieldTo.figure, isShortCastling('white', move\_annotation), isLongCastling('white', move\_annotation), current\_move, null, this, move\_time);

this.moves.add\_move(move);

}

move\_annotation = getBlackMove(moves\_strip[number - 1]);

if(existsBlack(move\_annotation)) {

var move\_time = getBlackTime(moves\_strip[number - 1]);

var fieldFrom = getFrom('black', move\_annotation);

var fieldTo = getTo('black', move\_annotation);

var move = new move(iMove + 1, 'black', fieldFrom, fieldTo, fieldFrom.figure, fieldTo.figure, isShortCastling('black', move\_annotation), isLongCastling('black', move\_annotation), current\_move, null, this, move\_time);

this.moves.add\_move(move);

}

}

}

}

this.refresh\_figures = refresh\_figures;

this.go\_begin = go\_begin;

this.go\_end = go\_end;

this.go\_next = go\_next;

this.go\_preview = go\_preview;

this.go\_custom = go\_custom;

this.repaint = function() {

}

// Парсит данные пришедшие от сервера

this.parse\_moves = function() {

var moves = this.all\_moves;

trace('call => parse\_moves(' + moves + ')')

function existsWhite(move) {

return getWhiteMove(move) != "\*\*\*\*\*\*";

}

function existsBlack(move) {

return getBlackMove(move) != "\*\*\*\*\*\*";

}

function getWhiteMove(move) {

return move.substring(0,6);

}

function getWhiteTime(move) {

return move.substring(13, 19);

}

function getBlackMove(move) {

return move.substring(6,12);

}

function getBlackTime(move) {

return move.substring(20, 26);

}

function getFrom(color, move) {

}

function getTo(color, move) {

}

function isShortCastling(color, move) {

}

function isLongCastling(color, move) {

}

moves\_split = moves.split('|');

moves\_strip = new Array();

for(i = 0; i < moves\_split.length; i++) {

if(moves\_split[i] != null) {

moves\_strip.push(moves\_split[i]);

}

}

var number = moves\_strip.length;

var current\_move = this.moves.get\_current\_move;

var current\_number = current\_move != null ? current\_move.number : 0;

if(number == current\_number) {

if((current\_move.color == 'white') && existsBlack(moves\_strip[number - 1])) {

var move\_annotation = getBlackMove(moves\_strip[number - 1]);

var move\_time = getBlackTime(moves\_strip[number - 1]);

var fieldFrom = getFrom('black', move\_annotation);

var fieldTo = getTo('black', move\_annotation);

var move = new move(number, 'black', fieldFrom, fieldTo, fieldFrom.figure, fieldTo.figure, isShortCastling('black', move\_annotation), isLongCastling('black', move\_annotation), current\_move, null, this, move\_time);

this.moves.add\_move(move);

}

}

if(number > current\_number){

for(iMove = current\_move; iMove < number; iMove++) {

var move\_annotation = getWhiteMove(moves\_strip[number - 1]);

if(existsWhite(move\_annotation)) {

var move\_time = getWhiteTime(moves\_strip[number - 1]);

var fieldFrom = getFrom('white', move\_annotation);

var fieldTo = getTo('white', move\_annotation);

var move = new move(iMove + 1, 'white', fieldFrom, fieldTo, fieldFrom.figure, fieldTo.figure, isShortCastling('white', move\_annotation), isLongCastling('white', move\_annotation), current\_move, null, this, move\_time);

this.moves.add\_move(move);

}

move\_annotation = getBlackMove(moves\_strip[number - 1]);

if(existsBlack(move\_annotation)) {

var move\_time = getBlackTime(moves\_strip[number - 1]);

var fieldFrom = getFrom('black', move\_annotation);

var fieldTo = getTo('black', move\_annotation);

var move = new move(iMove + 1, 'black', fieldFrom, fieldTo, fieldFrom.figure, fieldTo.figure, isShortCastling('black', move\_annotation), isLongCastling('black', move\_annotation), current\_move, null, this, move\_time);

this.moves.add\_move(move);

}

}

}

}

// Настройки

this.all\_moves = "";

this.moves\_delay = 3;

this.moves\_url = '/main/game\_moves/1';

this.figures\_delay = 2;

}

// Инициализация доски

function init() {

trace('call => init()');

// Расставляем шахматы в начальную позицию

// белые пешки

this.figures.wpA.set\_field(this.fields.A2);

this.figures.wpB.set\_field(this.fields.B2);

this.figures.wpC.set\_field(this.fields.C2);

this.figures.wpD.set\_field(this.fields.D2);

this.figures.wpE.set\_field(this.fields.E2);

this.figures.wpF.set\_field(this.fields.F2);

this.figures.wpG.set\_field(this.fields.G2);

this.figures.wpH.set\_field(this.fields.H2);

// белые фигуры

this.figures.wrA.set\_field(this.fields.A1);

this.figures.wrH.set\_field(this.fields.H1);

this.figures.whB.set\_field(this.fields.B1);

this.figures.whG.set\_field(this.fields.G1);

this.figures.wbC.set\_field(this.fields.C1);

this.figures.wbF.set\_field(this.fields.F1);

this.figures.wq.set\_field(this.fields.D1);

this.figures.wk.set\_field(this.fields.E1);

// черные пешки

this.figures.bpA.set\_field(this.fields.A7);

this.figures.bpB.set\_field(this.fields.B7);

this.figures.bpC.set\_field(this.fields.C7);

this.figures.bpD.set\_field(this.fields.D7);

this.figures.bpE.set\_field(this.fields.E7);

this.figures.bpF.set\_field(this.fields.F7);

this.figures.bpG.set\_field(this.fields.G7);

this.figures.bpH.set\_field(this.fields.H7);

// черные фигуры

this.figures.brA.set\_field(this.fields.A8);

this.figures.brH.set\_field(this.fields.H8);

this.figures.bhB.set\_field(this.fields.B8);

this.figures.bhG.set\_field(this.fields.G8);

this.figures.bbC.set\_field(this.fields.C8);

this.figures.bbF.set\_field(this.fields.F8);

this.figures.bq.set\_field(this.fields.D8);

this.figures.bk.set\_field(this.fields.E8);

// Создаем Ajax опрашиватель сервера

new PeriodicalExecuter(this.refresh\_moves, this.moves\_delay);

new PeriodicalExecuter(this.refresh\_figures, this.figures\_delay);

}

// Перерисовка позиции (делает один ход)

function refresh\_figures() {

//trace('call => refresh\_figures');

if(this.exists\_move) {

this.current\_move = this.moves.get\_current\_move();

this.current\_move.forward();

}

}

// Перейти к началу

function go\_begin() {

}

// Перейти к концу

function go\_end() {

}

// Следующий ход

function go\_next() {

}

// Предыдущий ход

function go\_preview() {

}

// Перейти к произвольному ходу

function go\_custom(number, color) {

}

// сделать ход

function do\_move(from\_field, to\_field) {

}

// Возвращает поле

function get\_field(vertical, horizontal) {

}

// Установить фигуру на поле

function set\_field(field\_to\_set) {

this.last\_field = field\_to\_set;

field\_to\_set.figure = this;

field\_to\_set.repaint();

}

// Возвращает ход по номеру и цвету

function get\_move(number, color) {

}

// Возвращает текуший ход

function get\_current\_move() {

return this.current\_move;

}

// Возвращает true если есть ход который нужно совершить

function exists\_move() {

}

function repaint() {

$(this.id).innerHTML = '';

if(this.figure != null) {

$(this.id).appendChild(this.figure.img);

}

}

function reload() {

this.img.src = images + this.color[0] + this.type[0] + '.gif';

//trace('reload figure [' + this.id + '].src = \"' + this.img.src + '\"');

}

// Ход вперед !! предыдущую фигуру удалить

function forward(){

if(this.to\_field.figure != null) {

}

this.figure.do\_move(this.from\_field, this.to\_field);

}

// Ход назад

function backward() {

this.figure.do\_move(this.to\_field, this.from\_field);

this.alive\_figure.set\_field(this.to\_field);

}

function trace(message) {

$("debug").innerHTML += message + '<br />';

}

window.onload=function() {

// Запуск

var main\_board = new board();

main\_board.init();

// setBeginingPosition();

// refreshGame();

// new PeriodicalExecuter(refreshGame, periodicalDelay);

// new PeriodicalExecuter(refreshBoard, 1);

}

Приложение Г

Листинг программы-регистратора шахматных партий

rdgtChess.java

---

public class rdgtChess implements Runnable

static boolean debug = false;

String comport = "COM1";

public static void main(String[] args) {

rdgtChess t;

if(args.length > 0)

t = new rdgtChess(args[0]);

else

t = new rdgtChess();

}

public rdgtChess() {

(new Thread(this)).start();

}

public rdgtChess(String comport) {

this.comport = comport;

(new Thread(this)).start();

}

// Destructor

public void finalize()

{

rdgtMysql.disconnect();

}

// This is the main loop. It samples all boards and tries at regular intervals

// to find new boards and to mark non-answering boards as, well, non-answering.

Это головная часть. Она опрашивает все доски с регулярным интервалом времени,

чтобы найти новые доски и отметить неотвечающие доски как неотвечающие

public void run() {

rdgtController c = new rdgtController();

if (c.connect(comport) == false) return;

System.out.println(rdgtMysql.connect());

int i = 0;

while (true) {

if ((i==0) || (c.db.count\_alive()==0)) {

while (c.find\_all\_boards()==0) {} c.dump\_all\_active\_boards();

i = 15;

} else {

try {Thread.sleep(1000);} catch(Throwable t) {}}

c.poll\_all\_active\_boards();

i--;

}

}

}

rdgtChessboard.java

---

class rdgtChessboard {

Integer address;

rdgtHistory history = new rdgtHistory(this);

rdgtSnapshot snapshot = new rdgtSnapshot(this);

rdgtFenhax myFenhax = new rdgtFenhax();

boolean alive;

boolean dumped;

boolean updated;

rdgtDatabase db;

rdgtChessBoardForm board = null;

public rdgtChessboard(Integer \_address, rdgtDatabase \_db) {

address = \_address;

db = \_db;

alive = true;

dumped = false;

updated = false;

update\_html();

board = new rdgtChessBoardForm();

new Thread(new Runnable() {

public void run() {

board.setVisible(true);

}

}).start();

}

rdgtSnapshot get\_snapshot() {

return snapshot;

}

int get\_address() {

return address.intValue();

}

public void print\_position() {

board.printPosition(snapshot.debugprint());

board.printClock(snapshot.debugprint\_clock());

}

String print\_figure(int figure) {

if(figure == rdgtProtocol.EMPTY) {

return " ";

} else if (figure == rdgtProtocol.WPAWN) {

return "P";

} else if(figure == rdgtProtocol.WROOK) {

return "R";

} else if(figure == rdgtProtocol.WKNIGHT) {

return "N";

} else if(figure == rdgtProtocol.WBISHOP) {

return "B";

} else if(figure == rdgtProtocol.WQUEEN) {

return "Q";

} else if(figure == rdgtProtocol.WKING) {

return "K";

} else if (figure == rdgtProtocol.BPAWN) {

return "p";

} else if(figure == rdgtProtocol.BROOK) {

return "r";

} else if(figure == rdgtProtocol.BKNIGHT) {

return "n";

} else if(figure == rdgtProtocol.BBISHOP) {

return "b";

} else if(figure == rdgtProtocol.BQUEEN) {

return "q";

} else if(figure == rdgtProtocol.BKING) {

return "k";

} else {

return " ";

}

}

void set\_alive(boolean x) {

alive = x;

update\_html();

}

void set\_dumped(boolean x) {

dumped = x;

}

void set\_updated(boolean x) {

updated = x;

history.append(snapshot);

}

boolean get\_alive() {

return alive;

}

boolean get\_dumped() {

return dumped;

}

boolean get\_updated() {

return updated;

}

void set\_clockdata(boolean running, boolean batteryLow, boolean frontViewLeftSideHigh, boolean blacksTurn, boolean whitesTurn, int secW, int secB) {

snapshot.set\_clockdata(running, batteryLow, frontViewLeftSideHigh, blacksTurn, whitesTurn, secW, secB);

snapshot.debugprint\_clock();

}

void update\_html() {

rdgtMysql.update\_snapshot(snapshot);

String fen = rdgtMysql.snapshot2fen(snapshot);

myFenhax.add(fen);

rdgtMysql.update\_moves(rdgtFenhax.getMoves(true, false), rdgtFenhax.getMoves(true, true), address.intValue());

}

void set\_emptyboard() {

snapshot.set\_emptyboard();

history.append(snapshot);

}

void set\_boarddump(int[] all64, int startpos) {

snapshot.set\_boarddump(all64, startpos);

dumped = true;

history.append(snapshot);

}

void set\_fieldupdate(int piece, int pos) {

snapshot.set\_fieldupdate(piece, pos);

updated = true;

history.append(snapshot);

}

}

rdgtConfig.java

---

class rdgtConfig {

String getDbServer() { return server; }

String getDbUser() { return user; }

String getDbPass() { return pass; }

String getDbBase() { return dbase; }

boolean useDbServer() { return mysql; }

boolean usePgnWriter() { return pgn; }

String getPgnFileName() { return pgnfile; }

String server= "localhost";

String user= "root";

String pass= "zar1562";

String dbase= "rdgt\_development";

boolean mysql = true;

boolean pgn = false;

String pgnfile = "default.pgn";

}

rdgtController.java

---

class rdgtController extends rdgtProtocol {

rdgtDatabase db;

rdgtInterpreter ipr;

rdgtReceiver rec;

rdgtSerialport ser;

rdgtSender snd;

public rdgtController() {

db = new rdgtDatabase();

ipr = new rdgtInterpreter(db);

rec = new rdgtReceiver(ipr);

ser = new rdgtSerialport(rec);

snd = new rdgtSender(ser);

}

boolean connect(String comport) {

if (ser.open(comport)==false) {

System.out.println("Could not open "+comport);

return false;

}

System.out.println("COM port opened: "+comport);

return true;

}

int find\_all\_boards() {

System.out.println("Scanning for boards...");

db.unalive\_all();

int alive = 0;

while (true) {

try {Thread.sleep(300);} catch(Throwable t) {}

snd.send(DGT\_BUS\_PING, 0);

try {Thread.sleep(1100);} catch(Throwable t) {}

int alivenow = db.count\_alive();

if (alivenow == alive) break;

alive = alivenow;

for (Enumeration e = db.get\_boards() ; e.hasMoreElements() ;) {

rdgtChessboard b = (rdgtChessboard)(e.nextElement());

if (b.get\_alive()==true) {

snd.send(DGT\_BUS\_IGNORE\_NEXT\_BUS\_PING, b.get\_address());

}

}

}

System.out.println("Scanning completed, found "+Integer.toString(alive)+" boards.\n");

return alive;

}

void dump\_all\_active\_boards() {

for (Enumeration e = db.get\_boards() ; e.hasMoreElements() ;) {

rdgtChessboard b = (rdgtChessboard)(e.nextElement());

if (b.get\_alive()==true) {

b.set\_dumped(false);

for (int i=0; i<3; i++) { if (b.get\_dumped()==true) break;

snd.send(DGT\_BUS\_SEND\_BRD, b.get\_address());

try {Thread.sleep(200);} catch(Throwable t) {}

}

if (b.get\_dumped()==false) {

System.out.println("Board "+Integer.toString(b.get\_address())+" does not reply to dump\_board.");

b.set\_alive(false);

}

}

}

}

void poll\_all\_active\_boards() {

int TIMEOUT = 100;

for (Enumeration e = db.get\_boards() ; e.hasMoreElements() ;) {

rdgtChessboard b = (rdgtChessboard)(e.nextElement());

if (b.get\_alive()==true) {

snd.send(DGT\_BUS\_SEND\_CLK, b.get\_address());

try {Thread.sleep(TIMEOUT);} catch(Throwable t) {}

b.set\_updated(false);

snd.send(DGT\_BUS\_SEND\_CHANGES, b.get\_address());

try {Thread.sleep(TIMEOUT);} catch(Throwable t) {}

for (int i=0; i<2; i=i+1) {

if (b.get\_updated()==true) break;

snd.send(DGT\_BUS\_REPEAT\_CHANGES, b.get\_address());

try {Thread.sleep(TIMEOUT);} catch(Throwable t) {}}

if (b.get\_updated()==false) {

System.out.println("Board "+Integer.toString(b.get\_address())+" does not reply anymore.");

b.set\_alive(false);

} else {

b.print\_position();

}

}

}

}

}

rdgtDatabase.java

---

class rdgtDatabase {

Hashtable boards;

public rdgtDatabase() {

boards = new Hashtable();

}

rdgtChessboard get\_board(int address) {

Integer \_address = new Integer(address);

rdgtChessboard b = (rdgtChessboard)(boards.get(\_address));

if (b==null) {

System.out.println("New board found: ID="+\_address.toString());

b = new rdgtChessboard(\_address, this);

boards.put(\_address, b);

}

return b;

}

int get\_boardcount() {

return boards.size();

}

Enumeration get\_boards() {

return boards.elements();

}

void unalive\_all() {

for (Enumeration e = boards.elements() ; e.hasMoreElements() ;) {

((rdgtChessboard)(e.nextElement())).set\_alive(false);

}

}

int count\_alive() {

int x = 0;

for (Enumeration e = boards.elements() ; e.hasMoreElements() ;) {

if (((rdgtChessboard)(e.nextElement())).get\_alive()==true) x++;

}

return x;

}

}

rdgtFenboard.java

---

class rdgtFenboard {

int[] itsBoard;

String itsFen;

public rdgtFenboard(){

itsBoard = new int[64];

for(int i=0;i<64;i++)

itsBoard[i]='.';

itsFen="";

}

public rdgtFenboard(String fen){ setBoard(fen); }

public rdgtFenboard(int[] board) { setBoard(board); }

public boolean setBoard(String fen){

if(!validateFen(fen))

return false;

itsBoard=fen2board(fen);

itsFen=fen;

return true;

}

public int getPieceAt(int x, int y){

if(x<0 || x>7 || y<0 || y>7) return 0;

return itsBoard[y\*8+x];

}

public boolean setBoard(int[] board){

if(board.length != 64)

return false;

itsBoard=board;

return true;

}

public void print(){

printBoard(itsBoard);

}

public int[] getBoard() { return itsBoard; }

public rdgtFenboard mask(rdgtFenboard b){

rdgtFenboard output;

output=new rdgtFenboard(mask(getBoard(),b.getBoard()));

return output;

}

public int diffCount(rdgtFenboard b){

rdgtFenboard m = mask(b);

int[] table=m.getBoard();

int output=0;

for(int i=0;i<64;i++){

if(table[i]!='.')

output++;

}

return output;

}

int[] fen2board(String aFen){

int[] board = new int[64];

int square=0;

int chr=0;

while(square<64){

char c = aFen.charAt(chr);

if(c=='r'||c=='R'||c=='n'||c=='N'||c=='b'||c=='B'||c=='q'||c=='Q'||c=='k'||c=='K'||c=='p'||c=='P'){

board[square]=c;

square++;

}

if(c>='1' && c<='8'){

for(int i=0;i<c-'1'+1;i++){

board[square]='.';

square++;

}

}

chr++;

}

return board;

}

String[] outputBoard(int[] board){

String s="";

String[] out = new String[8];

for(int chr=0;chr<64;chr++){

if(board[chr]<0)

s += "-" + (char)(Math.abs(board[chr]));

else if(board[chr]==0)

s += ", ";

else

s += (char)board[chr] + " ";

if((chr+1)%8==0){

out[chr/8]=s;

s="";

}

}

return out;

}

void printBoard(int[] board){

String[] out = outputBoard(board);

for(int i=0;i<8;i++){

System.out.println(out[i]);

}

System.out.println();

}

int[] mask(int[] b1, int[] b2){

int[] output=new int[64];

for(int i=0;i<64;i++){

if(b2[i]!=b1[i]){

if(b2[i]=='.')

output[i]=-b1[i];

else

output[i]=b2[i];

}

else

output[i] = '.';//b2[i]-b1[i];

}

return output;

}

boolean validateFen(String aFen){

/\* Only check board part of FEN \*/

if(aFen.indexOf(' ')>0){

aFen = aFen.substring(0,aFen.indexOf(' '));

}

int s=0;

for(int c=0;c<aFen.length();c++){

if(aFen.charAt(c) == '/') s++;

}

if(s!=7) return false;

int x=0;

for(int c=0;c<aFen.length();c++){

if(aFen.charAt(c) == '/'){

if(x!=8) return false;

else x=0;

}

else

{

if(aFen.charAt(c) >= '1' && aFen.charAt(c)<= '8')

x+= aFen.charAt(c)-'1' ;

x++;

}

}

if(x!=8) return false;

return true;

}

}

rdgtFendiff.java

---

class rdgtFendiff {

char piece;

boolean added;

int square;

rdgtFenboard before;

rdgtFenboard after;

public rdgtFenboard getBefore() { return before; }

public rdgtFenboard getAfter() { return after; }

public char getPiece() { return piece; }

public boolean isAdded() { return added; }

public int getSquare() { return square; }

public boolean isWhite(){

if(piece>'A' && piece < 'Z')

return true;

else

return false;

}

public boolean equals(rdgtFendiff other){

if(getPiece() != other.getPiece()) return false;

if(getSquare() != other.getSquare()) return false;

if(isAdded() != other.isAdded()) return false;

return true;

}

public rdgtFendiff() {

secured=false;

}

public rdgtFendiff(char piece, boolean added, int square){

set(piece,added,square);

}

public rdgtFendiff(int piece, int square){

if(piece<0){

this.piece = (char)(-piece);

added=false;

}

else {

this.piece=(char)piece;

added=true;

}

this.square=square;

}

public rdgtFendiff(int piece,int square, rdgtFenboard before, rdgtFenboard after){

this.before=before;

this.after=after;

if(piece<0){

this.piece = (char)(-piece);

added=false;

}

else {

this.piece=(char)piece;

added=true;

}

this.square=square;

}

public void set(char piece, boolean added, int square){

this.piece=piece;

this.added=added;

this.square=square;

}

public String toString() {

String output = ""+piece;

if(added) output += "@";

else output += "&";

output += (char)((square)%8+'a') + "" + (8-square/8);

return output;

}

}

rdgtFenhax.java

---

public class rdgtFenhax {

static boolean debug=false;

static boolean rotated=false;

public rdgtFenhax(){

boards=new ArrayList();

moves=new movelist();

diffTable=new ArrayList();

result="?-?";

plyCount=0;

}

static String getMoves(boolean shortNotation, boolean nice) {

calcMoves();

return moves.getAll(shortNotation,nice);

}

static String getMoves(){

return getMoves(false,false);

}

static int getPlyCount(){

calcMoves();

return moves.size();

}

static void clear(){

boards.clear();

moves.clear();

diffTable.clear();

result="?-?";

}

static String getResult() { return result; }

static boolean add(String aFen){

rdgtFenboard f = new rdgtFenboard();

if(!f.setBoard(aFen))

return false;

boards.add(f);

if(aFen.indexOf("rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR")>=0){

clear();

boards.add(f);

}

if((aFen.indexOf("rnbq1bnr/pppppppp") >=0 && aFen.indexOf("PPPPPPPP/RNBQ1BNR") >= 0)||

(aFen.indexOf("rnbqkbnr/pppppppp") >=0 && aFen.indexOf("PPPPPPPP/RNBQ1BNR") >= 0)||

(aFen.indexOf("rnbq1bnr/pppppppp") >=0 && aFen.indexOf("PPPPPPPP/RNBQKBNR") >= 0)){

int[] b=f.getBoard();

boolean black=false;

boolean white=false;

for(int i=16;i<=47;i++){

if(b[i] == 'k') black=true;

if(b[i] == 'K') white=true;

}

if(black && white) result="1/2-1/2";

else if(black) result="0-1";

else if(white) result="1-0";

}

if(boards.size()>=2){

rdgtFenboard last = (rdgtFenboard)boards.get(boards.size()-1);

rdgtFenboard nextlast = (rdgtFenboard)boards.get(boards.size()-2);

if(last.diffCount(nextlast) < 5){

rdgtFenboard m=nextlast.mask(last);

int[] mask = m.getBoard();

for(int i=0;i<64;i++){

if(mask[i] != '.' && mask[i]<0){

diffTable.add(new rdgtFendiff(mask[i],i,nextlast,last));

}

}

for(int i=0;i<64;i++){

if(mask[i] != '.' && mask[i]>0){

diffTable.add(new rdgtFendiff(mask[i],i,nextlast,last));

}

}

}

else {

clear();

boards.add(f);

}

}

return true;

}

static void calcMoves(){

moves.clear();

if(debug){

for(int i=0;i<diffTable.size();i++){

System.out.print(i + ":" + ((rdgtFendiff)diffTable.get(i)).toString() + " ");

}

System.out.print("\n");

}

int offset=0;

int length=diffTable.size();

while(offset<length){

int start=offset;

int end=-1;

boolean turn = isWhite( ((rdgtFendiff)diffTable.get(offset)).getPiece() );

int i=start;

while(end==-1){

rdgtFendiff test = (rdgtFendiff)diffTable.get(i);

if(turn != isWhite(test.getPiece()) && test.isAdded()){

end=i-1;

}

else{

i++;

if(i>=length)

end=length-1;

}

if(turn==isWhite(test.getPiece()) && test.isAdded()){

offset=i;

}

}

if(offset==start) offset++;

if(debug) System.out.print(start + "-"+end+" ");

movelist chunkMoves = new movelist(true,true);

for(int s=start;s<end;s++){

rdgtFendiff a = (rdgtFendiff)diffTable.get(s);

for(int e=s+1;e<=end;e++){

rdgtFendiff b = (rdgtFendiff)diffTable.get(e);

if(a.getPiece() == b.getPiece() && a.getSquare() != b.getSquare()){

if(!a.isAdded() && b.isAdded()){

String special="";

if(b.getPiece()=='P' && b.getSquare()/8==0){

for(int j=e+1;j<=end;j++){

rdgtFendiff prom = (rdgtFendiff)diffTable.get(j);

if(prom.isAdded() && prom.isWhite()){

if(a.getSquare()%8 != b.getSquare()%8)

special+=pos2xy(a.getSquare()).substring(0,1)+"x";

special+=pos2xy(b.getSquare())+"="+prom.getPiece();

b=prom;

}

}

}

if(b.getPiece()=='p' && b.getSquare()/8==7){

for(int j=e+1;j<=end;j++){

rdgtFendiff prom = (rdgtFendiff)diffTable.get(j);

if(prom.isAdded() && prom.isWhite()){

if(a.getSquare()%8 != b.getSquare()%8)

special+=pos2xy(a.getSquare()).substring(0,1)+"x";

special+=pos2xy(b.getSquare())+"="+prom.getPiece();

b=prom;

}

}

}

chunkMoves.add(new move(a,b,special));

}

}

}

}

if(chunkMoves.size()==0){

rdgtFendiff a = (rdgtFendiff)diffTable.get(start);

for(int s=start;s<=end;s++){

rdgtFendiff b = (rdgtFendiff)diffTable.get(s);

if(b.isAdded()){

if(a.getPiece()=='P' && a.getSquare()/8==1 && !a.isAdded()){

String special="";

if(a.getSquare()%8 != b.getSquare()%8)

special+=pos2xy(a.getSquare()).substring(0,1)+"x";

special+=pos2xy(b.getSquare())+"="+b.getPiece();

chunkMoves.add(new move(a,b,special));

}

else if(a.getPiece()=='p' && a.getSquare()/8==6 && !a.isAdded()){

String special="";

if(a.getSquare()%8 != b.getSquare()%8)

special+=pos2xy(a.getSquare()).substring(0,1)+"x";

special+=pos2xy(b.getSquare())+"="+b.getPiece();

chunkMoves.add(new move(a,b,special));

}

else

chunkMoves.add(new move(a,b,""+a.getPiece()+"@"+pos2xy(a.getSquare())));

}

}

}

for(int j=0;j<chunkMoves.size();j++){

if(debug) System.out.print( (chunkMoves.get(j)).toString() + " ");

moves.add(chunkMoves.get(j));

}

if(debug) System.out.print("\n");

}

}

static String pos2xy(int pos){

String s = "" + (char)((pos)%8+'a') + "" + (8-pos/8);

return s;

}

static boolean isWhite(char piece){

if(piece>'A' && piece < 'Z')

return true;

else

return false;

}

static movelist moves;

static ArrayList boards;

static ArrayList diffTable;

static String result;

static int plyCount;

}

class move {

rdgtFendiff from;

rdgtFendiff to;

boolean take;

String special;

public move(rdgtFendiff from, rdgtFendiff to){

this.from=from;

this.to=to;

take=false;

special="";

}

public move(rdgtFendiff from, rdgtFendiff to, String special){

this.from=from;

this.to=to;

take=false;

this.special=special;

}

public rdgtFendiff getFrom(){ return from; }

public rdgtFendiff getTo(){ return to; }

public char getPiece(){ return from.getPiece(); }

public int getFromSquare() { return from.getSquare(); }

public int getToSquare() { return to.getSquare(); }

public String getFromCor() { return pos2xy(getFromSquare()); }

public String getToCor() { return pos2xy(getToSquare()); }

public boolean isWhite() { return from.isWhite(); }

public String longNotation() { return moveString(false); }

public String shortNotation() { return moveString(true); }

public void setTake(boolean take){ this.take=take; }

public boolean getTake() { return take; }

String moveString(boolean shortNotation){

if(!special.equals("")) {

if(isCheck(isWhite()))

return special + "+";

else

return special;

}

boolean ep=false;

char takeBit='-';

if(take) takeBit='x';

String s="";

if(getPiece()!='p' && getPiece()!='P')

s+=new String(""+getPiece()).toUpperCase();

if(!shortNotation)

s+=getFromCor()+takeBit;

else {

if(getPiece()=='R' || getPiece()=='r')

if(findRookLength(getToSquare()%8,getToSquare()/8,getPiece()))

s+=getFromCor()+takeBit;

if(getPiece()=='N' || getPiece()=='n')

if(findNightLength(getToSquare()%8,getToSquare()/8,getPiece()))

s+=getFromCor()+takeBit;

if(getPiece()=='B' || getPiece()=='b')

if(findBishopLength(getToSquare()%8,getToSquare()/8,getPiece()))

s+=getFromCor()+takeBit;

if(getPiece()=='Q' || getPiece()=='q')

if(findQueenLength(getToSquare()%8,getToSquare()/8,getPiece()))

s+=getFromCor()+takeBit;

if(getPiece()=='P' || getPiece()=='p')

if( !getFromCor().substring(0,1).equals( getToCor().substring(0,1) ) ) {

s+=getFromCor().substring(0,1)+"x";

if(!take) ep=true;

}

if(take && s.length()>=1){

if(s.charAt(s.length()-1) != 'x')

s+="x";

}

}

s+=getToCor();

if(ep)

s+=" ep.";

/\*\* Is it a check? \*\*/

if(isCheck(isWhite())) s+='+';

return s;

}

boolean isCheck(boolean white){

int k=-1;

int K=-1;

for(int i=0;i<64;i++){

if(to.getAfter().getPieceAt(i%8,i/8) == 'k') k=i;

if(to.getAfter().getPieceAt(i%8,i/8) == 'K') K=i;

}

boolean check=false;

if(white){

if(findQueenLength(k%8,k/8,'Q')) check=true;

if(findRookLength(k%8,k/8,'R')) check=true;

if(findBishopLength(k%8,k/8,'B')) check=true;

if(findNightLength(k%8,k/8,'N')) check=true;

if(findPawnLength(k%8,k/8,'P')) check=true;

}

else{

if(findQueenLength(K%8,K/8,'q')) check=true;

if(findRookLength(K%8,K/8,'r')) check=true;

if(findBishopLength(K%8,K/8,'b')) check=true;

if(findNightLength(K%8,K/8,'n')) check=true;

if(findPawnLength(K%8,K/8,'p')) check=true;

}

return check;

}

/\*\* Is there a night on a night-distance from square? \*\*/

boolean findNightLength(int x, int y,char n){

rdgtFenboard b = to.getAfter();

if(b.getPieceAt(x-1,y-2) == n) return true;

if(b.getPieceAt(x+1,y-2) == n) return true;

if(b.getPieceAt(x-2,y-1) == n) return true;

if(b.getPieceAt(x+2,y-1) == n) return true;

if(b.getPieceAt(x-2,y+1) == n) return true;

if(b.getPieceAt(x+2,y+1) == n) return true;

if(b.getPieceAt(x-1,y+2) == n) return true;

if(b.getPieceAt(x+1,y+2) == n) return true;

return false;

}

boolean findRookLength(int x, int y, char r){

rdgtFenboard b = to.getAfter();

int xx=0;

int yy=0;

do{

xx++;

if(b.getPieceAt(x+xx,y+yy) == r) return true;

}while(b.getPieceAt(x+xx,y+yy) == '.');

xx=0;

do{

xx--;

if(b.getPieceAt(x+xx,y+yy) == r) return true;

}while(b.getPieceAt(x+xx,y+yy) == '.');

xx=0;

do{

yy--;

if(b.getPieceAt(x+xx,y+yy) == r) return true;

}while(b.getPieceAt(x+xx,y+yy) == '.');

yy=0;

do{

yy++;

if(b.getPieceAt(x+xx,y+yy) == r) return true;

}while(b.getPieceAt(x+xx,y+yy) == '.');

return false;

}

boolean findBishopLength(int x, int y, char r){

rdgtFenboard b = to.getAfter();

int xx=0;

int yy=0;

do{

xx++;

yy++;

if(b.getPieceAt(x+xx,y+yy) == r) return true;

}while(b.getPieceAt(x+xx,y+yy) == '.');

xx=0;

yy=0;

do{

xx--;

yy--;

if(b.getPieceAt(x+xx,y+yy) == r) return true;

}while(b.getPieceAt(x+xx,y+yy) == '.');

xx=0;

yy=0;

do{

yy--;

xx++;

if(b.getPieceAt(x+xx,y+yy) == r) return true;

}while(b.getPieceAt(x+xx,y+yy) == '.');

yy=0;

xx=0;

do{

yy++;

xx--;

if(b.getPieceAt(x+xx,y+yy) == r) return true;

}while(b.getPieceAt(x+xx,y+yy) == '.');

return false;

}

boolean findQueenLength(int x, int y, char q){

if(findRookLength(x,y,q)) return true;

if(findBishopLength(x,y,q)) return true;

return false;

}

boolean findPawnLength(int x, int y, char p){

rdgtFenboard b = to.getAfter();

if(p=='P'){

if(b.getPieceAt(x-1,y-1)==p) return true;

if(b.getPieceAt(x+1,y-1)==p) return true;

}

else {

if(b.getPieceAt(x-1,y-1)==p) return true;

if(b.getPieceAt(x+1,y-1)==p) return true;

}

return false;

}

public String toString(){

return ""+from.toString() + "-" + to.toString();

}

public boolean equals(move other){

if(!from.equals( other.getFrom() )) return false;

if(!to.equals( other.getTo() )) return false;

return true;

}

String pos2xy(int pos){

String s = "" + (char)((pos)%8+'a') + "" + (8-pos/8);

return s;

}

}

class movelist {

ArrayList moves;

boolean unique;

boolean noSmooth;

public void clear(){moves.clear(); }

public void remove(int n) { moves.remove(n); }

public movelist(){ moves=new ArrayList(); unique=false; }

public movelist(boolean unique, boolean noSmooth) {

moves=new ArrayList();

this.unique=unique;

this.noSmooth=noSmooth;

}

public move get(int n){ return (move)moves.get(n); }

public int size() { return moves.size(); }

public String toString() { return getAll(false,false); }

public void print(boolean shortNotation) {

System.out.println("Moves");

System.out.print(getAll(shortNotation,true));

}

void fixTakes(){

for(int i=1;i<moves.size();i++){

int toSquare = ((move)moves.get(i)).getToSquare();

char p = (char) ((move)moves.get(i-1)).getTo().getAfter().getBoard()[toSquare];

if(p!='.'){

move m=(move)moves.get(i);

m.setTake(true);

moves.set(i,m);

}

}

}

public String getAll(boolean shortNotation,boolean nice){

fixTakes();

String output="";

boolean lastColor=false;

int n=1;

for(int i=0;i<moves.size();i++){

move m=(move)moves.get(i);

if(m.isWhite() && lastColor==false){

if(nice)

output += n+".\t";

else

output += n+". ";

n++;

}

if(n==1 && !m.isWhite()) {

if(nice)

output += "1.\t...\t\t";

else

output += "1. ... ";

}

lastColor=m.isWhite();

int length=output.length();

if(shortNotation)

output += m.shortNotation()+" ";

else

output += m.longNotation()+" ";

if(nice){

if(!m.isWhite())

output += "\n";

else{

if(output.length()-length < 8) output += "\t";

if(output.length()-length < 12) output += "\t";

}

}

}

return output;

}

public void add(move aMove){

if(!unique){

moves.add(aMove);

return;

}

for(int i=0;i<moves.size();i++){

if( ((move)moves.get(i)).equals(aMove))

return;

}

if(moves.size()>=1){

move m=(move)moves.get(moves.size()-1);

// O-O

if(m.moveString(false).equals("Ke1-g1") && aMove.moveString(false).equals("Rh1-f1")){

moves.remove(moves.size()-1);

moves.add(new move(aMove.getFrom(),aMove.getTo(),"O-O"));

return;

}

// O-O-O

if(m.moveString(false).equals("Ke1-c1") && aMove.moveString(false).equals("Ra1-d1")){

moves.remove(moves.size()-1);

moves.add(new move(aMove.getFrom(),aMove.getTo(),"O-O-O"));

return;

}

// O-O

if(m.moveString(false).equals("Ke8-g8") && aMove.moveString(false).equals("Rh8-f8")){

moves.remove(moves.size()-1);

moves.add(new move(aMove.getFrom(),aMove.getTo(),"O-O"));

return;

}

// O-O-O

if(m.moveString(false).equals("Ke8-c8") && aMove.moveString(false).equals("Ra8-d8")){

moves.remove(moves.size()-1);

moves.add(new move(aMove.getFrom(),aMove.getTo(),"O-O-O"));

return;

}

}

moves.add(aMove);

}

public void removeSmooth(){

for(int a=0;a<moves.size();a++){

move aMove=(move)moves.get(a);

for(int i=0;i<moves.size();i++){

move test = (move)moves.get(i);

if( test.getPiece() == aMove.getPiece()){

if( aMove.getFrom().getSquare() == test.getTo().getSquare() ||

aMove.getTo().getSquare() == test.getFrom().getSquare() ){

if(a==i){

moves.remove(i);

i=moves.size();

a=0;

}

else if(i>a){

moves.remove(i);

moves.remove(a);

i=moves.size();

a=0;

}

else {

moves.remove(a);

moves.remove(i);

i=moves.size();

a=0;

}

}

}

}

}

}

}

rdgtHistory.java

---

class rdgtHistory {

Vector history = new Vector();

rdgtSnapshot lastadded = null;

rdgtSnapshot last = null;

rdgtChessboard board;

public rdgtHistory(rdgtChessboard \_board) {

board = \_board;

}

void append(rdgtSnapshot x) {

if(history.size()==0){

history.addElement(x.copy());

board.update\_html();

}

else if(x.sameas((rdgtSnapshot)history.lastElement())==false ){

history.addElement(x.copy());

board.update\_html();

}

}

void debugprint() {

System.out.println("======= History for board: "+board.address.toString());

for (Enumeration e = history.elements(); e.hasMoreElements() ;) {

((rdgtSnapshot)(e.nextElement())).debugprint();

}

}

}

rdgtHtml.java

---

class rdgtHtml extends rdgtProtocol {

static String extension = ".html";

static String indexfilename = "board\_index";

static String historyprefix = "boardhistory\_";

static String snapshotprefix = "boardsnapshot\_";

static void print\_index(rdgtDatabase db) {

StringBuffer s = new StringBuffer(2000);

s.append("<html>\n<head>\n <meta http-equiv=refresh content=3>\n</head>\n");

s.append("<body><h1>Chessboards</h1>\n<table border=1 cellspacing=0 cellpadding=10>\n");

s.append("<tr><th>Board id<th>Status<th>Time of last move<th>Snapshot<th>History\n");

for (Enumeration e = db.get\_boards() ; e.hasMoreElements() ;) {

rdgtChessboard b = (rdgtChessboard)(e.nextElement());

s.append("<tr>\n<td valign=top>");

s.append(Integer.toString(b.get\_address()));

s.append("\n<td>");

if (b.get\_alive()==true) s.append("Online"); else s.append("Offline");

s.append("<td>");

s.append(b.get\_snapshot().get\_time().toString());

s.append("\n<td><a href=\"");

s.append(snapshotprefix);

s.append(Integer.toString(b.get\_address()));

s.append(extension);

s.append("\">Link</a><td><a href=\"");

s.append(historyprefix);

s.append(Integer.toString(b.get\_address()));

s.append(extension);

s.append("\">Link</a>\n");

}

s.append("</table>\n</body>\n</html>\n");

if (writefile(indexfilename+extension, s.toString()) == false) {

System.err.println("Error: Failed to write indexfile.");

return;

}

}

static void print\_history(rdgtHistory h) {

StringBuffer s = new StringBuffer(2000);

s.append("<html>\n<head></head>\n");

s.append("<body><h1>History for Chessboard: ");

s.append(Integer.toString(h.board.get\_address()));

s.append("</h1>\n");

int i = 0;

for (Enumeration e = h.history.elements() ; e.hasMoreElements() ;) {

rdgtSnapshot ss = (rdgtSnapshot)(e.nextElement());

i=i+1;

s.append("<p><hr size=1><p><h3>Move: ");

s.append(Integer.toString(i));

s.append(" &nbsp; &nbsp;");

s.append(ss.get\_time().toString());

s.append(boardview(ss));

s.append("\n");

}

s.append("</body>\n</html>\n");

if (writefile(historyprefix + Integer.toString(h.board.get\_address()) + extension, s.toString()) == false) {

System.err.println("Error: Failed to write historyfile.");

return;

}

}

static String boardview(rdgtSnapshot b) {

StringBuffer s = new StringBuffer(1000);

s.append("<table border=0 cellpadding=0 cellspacing=0>\n");

boolean filled = true;

for (int i=0; i<8; i=i+1) {

s.append("<tr>");

for (int j=0; j<8; j=j+1) {

s.append("<td><img src=\"images/");

int p = b.pieces[(i\*8)+j];

if (p==EMPTY ) s.append("empty");

else if (p==WPAWN ) s.append("w\_pawn");

else if (p==WROOK ) s.append("w\_rook");

else if (p==WKNIGHT) s.append("w\_knight");

else if (p==WBISHOP) s.append("w\_bishop");

else if (p==WKING ) s.append("w\_king");

else if (p==WQUEEN ) s.append("w\_queen");

else if (p==BPAWN ) s.append("b\_pawn");

else if (p==BROOK ) s.append("b\_rook");

else if (p==BKNIGHT) s.append("b\_knight");

else if (p==BBISHOP) s.append("b\_bishop");

else if (p==BKING ) s.append("b\_king");

else if (p==BQUEEN ) s.append("b\_queen");

if (filled) s.append("1"); else s.append("2");

filled = !filled;

s.append(".gif\">");

}

filled = !filled;

s.append("</tr>\n");

}

s.append("\n</table>");

return s.toString();

}

static void print\_snapshot(rdgtSnapshot ss) {

StringBuffer s = new StringBuffer(2000);

s.append("<html>\n<head>\n <meta http-equiv=refresh content=4>\n</head>\n");

s.append("<body><h1>Snapshot of Chessboard: ");

s.append(Integer.toString(ss.board.get\_address()));

s.append("</h1>\n<h3>");

s.append(ss.get\_time().toString());

s.append("</h3>\n");

s.append(boardview(ss));

s.append("\n</body>\n</html>\n");

if (writefile(snapshotprefix + Integer.toString(ss.board.get\_address()) + extension, s.toString()) == false) {

System.err.println("Error: Failed to write snapshotfile.");

return;

}

}

static boolean writefile(String filename, String content) {

FileWriter f = null;

for (int i=0; i<10; i=i+1) {

try {

f = new FileWriter(filename);

break;

}

catch (Throwable t) {

System.err.println("Error opening "+filename+" for writing. Retrying...");

}

}

if (f==null) {

System.out.println("Failed to open file "+filename);

return false;

}

try {

f.write(content);

f.close();

}

catch (Exception e) {

System.out.println("Error writing file "+filename+". ("+e.toString()+")");

return false;

}

return true;

}

}

rdgtInterpreter.java

---

class rdgtInterpreter extends rdgtProtocol {

rdgtDatabase db;

public rdgtInterpreter(rdgtDatabase \_db) {

db = \_db;

}

public boolean interpret(int[] data) {

int len = ((int) data[1] \* 0x80) + (int) data[2];

int address = ((int) data[3] \* 0x80) + (int) data[4];

String tmp = "Len:" + Integer.toString(len) + " Address:" + Integer.toString(address);

if (data[0] == DGT\_BUS\_PING) {

System.out.println("Received message: PING " + tmp);

} else if (data[0] == DGT\_BUS\_UPDATE) {

System.out.println("Received message: UPDATE " + tmp);

} else if (data[0] == DGT\_BUS\_BWTIME) {

System.out.println("Received message: BWTIME " + tmp);

} else {

System.out.println("Received message: (some other message) " + tmp);

}

if (check\_checksum(data) == false) {

System.out.println("Message failed checksum test");

return false;

}

if (data[0] == DGT\_BUS\_PING) {

if (data.length != 6) { // address+data

System.out.println("Illegal: Got a BUS\_PING with wrong size");

return false;

}

db.get\_board(address).set\_alive(true);

return true;

} else if (data[0] == DGT\_BUS\_BRD\_DUMP) {

if (data.length != (6 + 64)) {

System.out.println("Illegal: Got a BUS\_BRD\_DUMP with wrong size");

return false;

}

rdgtChessboard c = db.get\_board(address);

c.set\_boarddump(data, 5);

return true;

} else if (data[0] == DGT\_BUS\_UPDATE) {

byte pr[] = new byte[data.length - 5];

for (int i = 5; i < data.length; i++) {

pr[i - 5] = (byte) data[i];

}

System.out.println(new String(pr));

rdgtChessboard c = db.get\_board(address);

for (int i = 5; i < data.length - 1; i++) {

if (data[i] == EE\_POWERUP) {

}

else if (data[i] == EE\_EOF) {

}

else if (data[i] == EE\_FOURROWS) {

}

else if (data[i] == EE\_EMPTYBOARD) {

c.set\_emptyboard();

} else if (data[i] == EE\_DOWNLOADED) {

}

else if (data[i] == EE\_BEGINPOS) {

}

else if (data[i] == EE\_BEGINPOS\_ROT) {

}

else if (data[i] == EE\_START\_TAG) {

}

else if (data[i] == EE\_WATCHDOG\_ACTION) {

}

else if (data[i] == EE\_NOP) {

}

else if (data[i] >= 0x40 && data[i] <= 0x5F) { // 2-byte piece update

c.set\_fieldupdate(data[i] & 0x0F, data[i + 1]);

i++;

} else if ((data[i] >= 0x60 && data[i] <= 0x69) || (data[i] >= 0x70 && data[i] <= 0x79)) {

i += 2;

} else {

System.out.println("Unknown data in DGT\_BUS\_UPDATE command.");

return false;

}

}

c.set\_updated(true);

return true;

}

else if (data[0] == DGT\_BUS\_BWTIME) {

if (data.length != 13) {

System.out.println("Illegal: Got a DGT\_BUS\_BWTIME with wrong size (!=10)");

return false;

}

if ((data[11] & 0x20) != 0) {

System.out.println("(No clock connected)");

return true;

}

boolean running = (data[11] & 0x01) != 0;

boolean batteryLow = (data[11] & 0x04) != 0;

boolean frontViewLeftSideHigh = (data[11] & 0x02) != 0;

boolean blackPlayersTurn = (data[11] & 0x08) != 0;

boolean whitePlayersTurn = (data[11] & 0x10) != 0;

int hoursW = data[5] & 0x0F;

int minutesW = (data[6] & 0x0F) + ((data[6] >> 4) \* 10);

int secondsW = (data[7] & 0x0F) + ((data[7] >> 4) \* 10);

int hoursB = data[8] & 0x0F;

int minutesB = (data[9] & 0x0F) + ((data[9] >> 4) \* 10);

int secondsB = (data[10] & 0x0F) + ((data[10] >> 4) \* 10);

secondsW = (((hoursW \* 60) + minutesW) \* 60) + secondsW;

secondsB = (((hoursB \* 60) + minutesB) \* 60) + secondsB;

db.get\_board(address).set\_clockdata(running, batteryLow, frontViewLeftSideHigh,

blackPlayersTurn, whitePlayersTurn, secondsW, secondsB);

return true;

} else if (data[0] == DGT\_BUS\_FROM\_START) {

} else if (data[0] == DGT\_BUS\_START\_GAME\_WRITTEN) {

} else if (data[0] == DGT\_BUS\_VERSION) {

}

System.err.println("Unknown message (checksum was ok).");

return true;

}

boolean check\_checksum(int[] data) {

int sum = 0;

for (int i = 0; i < (data.length - 1); i = i + 1) {

sum = (sum + data[i]) & 0x7F;

}

if (sum == data[data.length - 1]) {

return true;

}

return false;

}

}

rdgtMysql.java

---

class rdgtMysql extends rdgtProtocol {

static String server= "localhost";

static String user= "root";

static String pass= "zar1562";

static String dbase= "rdgt\_development";

static String update\_table= "games";

static Connection conn = null;

static Statement stmt = null;

static boolean rotated=false;

static String connect()

{

try

{

Class.forName("com.mysql.jdbc.Driver").newInstance();

conn = DriverManager.getConnection("jdbc:mysql://"+server+"/"+dbase+"?user="+user+"&password="+pass);

return "";

}

catch (SQLException e)

{

return e.getSQLState();

}

catch (Exception e)

{

return "0";

}

}

static boolean disconnect()

{

if (conn != null) {

try

{

conn.close();

return true;

}

catch (SQLException SQLE)

{

return false;

}

}

else

return false;

}

static String execute(String expr)

{

try

{

stmt = null;

stmt = conn.createStatement();

stmt.executeUpdate(expr);

return "";

}

catch (SQLException e)

{

return e.getSQLState();

}

catch (Exception e)

{

return "0";

}

finally

{

if (stmt != null) {

try

{

stmt.close();

}

catch (SQLException SQLE)

{

;

}

}

}

}

static String snapshot2fen (rdgtSnapshot b)

{

StringBuffer s = new StringBuffer(100);

int counter = 0;

for (int i=0; i<8; i++)

{

for (int j=0; j<8; j++)

{

if ((counter != 0) && (b.pieces[(i\*8)+j] != EMPTY))

{

s.append(counter);

counter = 0;

}

switch(b.pieces[(i\*8)+j])

{

case WPAWN:s.append("P"); break;

case WROOK: s.append("R"); break;

case WKNIGHT: s.append("N"); break;

case WBISHOP: s.append("B"); break;

case WKING: s.append("K"); break;

case WQUEEN: s.append("Q"); break;

case BPAWN: s.append("p"); break;

case BROOK: s.append("r"); break;

case BKNIGHT: s.append("n"); break;

case BBISHOP: s.append("b"); break;

case BKING: s.append("k"); break;

case BQUEEN: s.append("q"); break;

default: counter++; break;

}

}

if (counter != 0)

s.append(counter);

s.append("/");

counter = 0;

}

String outfen= s.toString().substring(0,s.toString().length()-1);

switch(b.board.address.intValue()){

case 3737:

case 3811:

case 2896:

case 3740:

case 3588:

case 2897:

default:

rotated=false; break;

}

if(rotated){

String newFen="";

for(int i=outfen.length()-1;i>=0;i--){

newFen += outfen.charAt(i);

}

outfen=newFen;

}

return outfen;

}

static void update\_snapshot(rdgtSnapshot b)

{

if (conn != null)

{

String fen = snapshot2fen(b);

String turn;

if(rotated){

if(b.isWhiteTurn()) turn="Black";

else turn="White";

}

else{

if(b.isWhiteTurn()) turn="White";

else turn="Black";

}

String sql\_command = "UPDATE "+update\_table+" SET fen = '"+fen+"',wtime='"+b.getTimeWhite()+"',btime='"+b.getTimeBlack()+"',turn='"+turn+"' WHERE dgt\_active like '%Fen%' AND dgt\_board = "+b.board.address;

execute(sql\_command);

System.out.println(b.board.address+" "+fen+" "+b.getTimeWhite()+" "+b.getTimeBlack()+" "+turn.substring(0,1));

}

}

static void update\_moves(String moves, String movestable, int board\_id)

{

if (conn != null)

{

int t=0;

if(moves.length()>20) t=moves.length()-20;

System.out.println(board\_id + " Moves: " + moves.substring(t));

String sql\_command = "UPDATE games SET moves = '"+moves+"' WHERE dgt\_active like '%Moves%' AND dgt\_board = "+board\_id;

execute(sql\_command);

}

}

}

rdgtProtocol.java

---

class rdgtProtocol {

static int DGT\_TO\_BUSMODE = 0x4a;

static int DGT\_BUS\_SEND\_CLK = 0x01 | 0x80;

static int DGT\_BUS\_SEND\_BRD = 0x02 | 0x80;

static int DGT\_BUS\_SEND\_CHANGES = 0x03 | 0x80;

static int DGT\_BUS\_REPEAT\_CHANGES = 0x04 | 0x80;

static int DGT\_BUS\_SET\_START\_GAME = 0x05 | 0x80;

static int DGT\_BUS\_SEND\_FROM\_START = 0x06 | 0x80;

static int DGT\_BUS\_PING = 0x07 | 0x80;

static int DGT\_BUS\_END\_BUSMODE = 0x08 | 0x80;

static int DGT\_BUS\_RESET = 0x09 | 0x80;

static int DGT\_BUS\_IGNORE\_NEXT\_BUS\_PING = 0x0a | 0x80;

static int DGT\_BUS\_SEND\_VERSION = 0x0b | 0x80;

static int DGT\_BUS\_BRD\_DUMP = 0x03 | 0x80;

static int DGT\_BUS\_BWTIME = 0x04 | 0x80;

static int DGT\_BUS\_UPDATE = 0x05 | 0x80;

static int DGT\_BUS\_FROM\_START = 0x06 | 0x80;

static int DGT\_BUS\_START\_GAME\_WRITTEN = 0x08 | 0x80;

static int DGT\_BUS\_VERSION = 0x09 | 0x80;

static boolean is\_busmessage(int m) {

return (m== DGT\_BUS\_BRD\_DUMP || m == DGT\_BUS\_BWTIME ||

m == DGT\_BUS\_UPDATE || m == DGT\_BUS\_FROM\_START || m == DGT\_BUS\_PING ||

m == DGT\_BUS\_START\_GAME\_WRITTEN || m == DGT\_BUS\_VERSION);

}

static final int EMPTY = 0x00;

static final int WPAWN = 0x01;

static final int WROOK = 0x02;

static final int WKNIGHT = 0x03;

static final int WBISHOP = 0x04;

static final int WKING = 0x05;

static final int WQUEEN = 0x06;

static final int BPAWN = 0x07;

static final int BROOK = 0x08;

static final int BKNIGHT = 0x09;

static final int BBISHOP = 0x0a;

static final int BKING = 0x0b;

static final int BQUEEN = 0x0c;

boolean is\_piece(int p) {

return (p==EMPTY || p==WPAWN || p==WROOK ||

p==WKNIGHT || p==WBISHOP || p==WKING ||

p==WQUEEN || p==BPAWN || p==BROOK ||

p==BKNIGHT || p==BBISHOP || p==BKING ||

p==BQUEEN);

}

static int EE\_POWERUP = 0x6a;

static int EE\_EOF = 0x6b;

static int EE\_FOURROWS = 0x6c;

static int EE\_EMPTYBOARD = 0x6d;

static int EE\_DOWNLOADED = 0x6e;

static int EE\_BEGINPOS = 0x6f;

static int EE\_BEGINPOS\_ROT = 0x7a;

static int EE\_START\_TAG = 0x7b;

static int EE\_WATCHDOG\_ACTION = 0x7c;

static int EE\_NOP = 0x7f;

}

rdgtReceiver.java

---

class rdgtReceiver extends rdgtProtocol {

int messagetype;

int messagelen\_msb;

int messagelen\_lsb;

int[] data;

int useddata;

int state;

rdgtInterpreter interpreter;

public rdgtReceiver(rdgtInterpreter \_interpreter) {

state = 0;

interpreter = \_interpreter;

}

void receive(int[] d) {

char c[] = {'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};

StringBuffer x = new StringBuffer("Received: ");

for (int i=0; i<d.length; i=i+1) {

x.append(c[d[i]>>4]).append(c[d[i] & 0x0F]).append(" ");

}

if (rdgtChess.debug) System.out.println(x.toString());

\_receive (d, 0);

}

void \_receive(int[] d, int start) {

if (start == d.length) return;

if (state == 0) { // Find a known messageid.

while (start<d.length) {

if (is\_busmessage(d[start])) {

messagetype = d[start];

state = 1;

\_receive(d,start+1);

return;

}

System.err.println("Did not understand 1 byte of incoming data.");

start = start + 1;

}

return;

}

if (state == 1) {

if ((d[start] & 0x80) > 0) {

System.err.println("Did not understand 2 bytes of incoming data.");

state = 0;

\_receive(d, start);

return;

}

messagelen\_msb = d[start];

state = 2;

start = start + 1;

}

if (start == d.length) return;

if (state == 2) {

if ((d[start] & 0x80) > 0) {System.err.println("Did not understand 3 bytes of incoming data.");

state = 0;

\_receive(d, start);

return;

}

messagelen\_lsb = d[start];

state = 3;

start = start + 1;

int newlen = ((int)messagelen\_msb \* 0x80) + (int)messagelen\_lsb;

if (newlen <5) {

System.out.println("Too small message length: " + Integer.toString(newlen));

state = 0;

\_receive(d, start);

return;

}

data = new int[newlen];

data[0] = messagetype;

data[1] = messagelen\_msb;

data[2] = messagelen\_lsb;

useddata = 3;

}

if (start == d.length) return;

if (state == 3) {

while (start<d.length && useddata<data.length) {

data[useddata]=d[start];

start = start + 1;

useddata = useddata + 1;

}

if (useddata == data.length) {

interpreter.interpret(data);

start = start + 1;

data = null;

state = 0;}

}

\_receive(d, start);

}

}

rdgtSender.java

---

class rdgtSender extends rdgtProtocol {

rdgtSerialport com;

public rdgtSender(rdgtSerialport \_com) {

com = \_com;

}

void send(int message, int address) {

String tmp = "Address:"+Integer.toString(address);

System.out.println();

if (message==DGT\_TO\_BUSMODE) System.out.println("Sending message: SWITCH TO BUSMODE "+tmp);

else if (message==DGT\_BUS\_SEND\_BRD) System.out.println("Sending message: SEND BOARD "+tmp);

else if (message==DGT\_BUS\_SEND\_CHANGES) System.out.println("Sending message: SEND CHANGES "+tmp);

else if (message==DGT\_BUS\_REPEAT\_CHANGES) System.out.println("Sending message: REPEAT CHANGES "+tmp);

else if (message==DGT\_BUS\_PING) System.out.println("Sending message: PING "+tmp);

else if (message==DGT\_BUS\_IGNORE\_NEXT\_BUS\_PING) System.out.println("Sending message: IGNORE NEXT PING "+tmp);

else System.out.println("Sending message: (some other message)");

byte x[] = new byte[4];

x[0] = (byte)message;

x[1] = (byte)((address>>7) & 0x7F);

x[2] = (byte)(address & 0x7F);

x[3] = (byte)(((int)x[0]+(int)x[1]+(int)x[2]) & 0x7F);

com.write(x);

}

}

rdgtSerialport.java

---

class rdgtSerialport implements SerialPortEventListener {

static CommPortIdentifier portid;

InputStream inStream;

OutputStream outStream;

SerialPort serialPort;

rdgtReceiver receiver;

public rdgtSerialport(rdgtReceiver r) {

receiver = r;

}

public boolean open(String portname) {

CommPortIdentifier id = null;

Enumeration portList;

portList = CommPortIdentifier.getPortIdentifiers();

System.out.println(portList.toString());

while (portList.hasMoreElements()) {

System.out.println("2");

id = (CommPortIdentifier) portList.nextElement();

if (id.getPortType() == CommPortIdentifier.PORT\_SERIAL) {

System.out.println("Ports: "+id.getName());

if (id.getName().equals(portname)) break; // found

}

id = null;

}

if (id == null) return false; // not found

try {

serialPort = (SerialPort) id.open("ttchess", 2000);

inStream = serialPort.getInputStream();

outStream = serialPort.getOutputStream();

serialPort.addEventListener(this);

serialPort.notifyOnDataAvailable(true);

serialPort.setSerialPortParams(9600, SerialPort.DATABITS\_8,

SerialPort.STOPBITS\_1,SerialPort.PARITY\_NONE);

serialPort.setDTR(false);

serialPort.setRTS(false);

} catch (Exception e) { return false; }

return true;

}

public boolean write(byte[] data) {

char c[] = {'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};

StringBuffer x = new StringBuffer(" Sending: ");

for (int i=0; i<data.length; i=i+1) {

int d = data[i];

if (d<0) d=256+d;

x.append(c[d>>4]).append(c[d & 0x0F]).append(" ");

}

if (rdgtChess.debug) System.out.println(x.toString());

try {

outStream.write(data);

} catch (IOException e) { return false; }

return true;

}

public void serialEvent(SerialPortEvent event) {

switch(event.getEventType()) {

case SerialPortEvent.OUTPUT\_BUFFER\_EMPTY:

break;

case SerialPortEvent.DATA\_AVAILABLE:

byte[] readBuffer = new byte[200];

try {

while (inStream.available() > 0) {

int numBytes = inStream.read(readBuffer);

if (numBytes>0) {

int[] buf = new int[numBytes];

for (int i=0; i<numBytes; i=i+1) {

int tmp = readBuffer[i];

if (tmp<0) tmp=256+tmp;

buf[i]=tmp;

}

receiver.receive(buf);

}

}

} catch (IOException e) {}

break;

}

}

}

rdgtSnapshot.java

---

class rdgtSnapshot extends rdgtProtocol {

int[] pieces = new int[64];

Date time;

rdgtChessboard board;

boolean clkRunning, clkBatteryLow, clkFrontViewLeftSideHigh, clkBlackTurn, clkWhiteTurn;

int clkSecWhite, clkSecBlack;

public rdgtSnapshot(rdgtChessboard \_board) {

time = new Date();

board = \_board;

set\_emptyboard();

}

public rdgtSnapshot(rdgtSnapshot x) {

for (int i=0; i<64; i=i+1) pieces[i] = x.pieces[i];

time = x.time;

board = x.board;

clkRunning = x.clkRunning;

clkBatteryLow = x.clkBatteryLow;

clkFrontViewLeftSideHigh = x.clkFrontViewLeftSideHigh;

clkBlackTurn = x.clkBlackTurn;

clkWhiteTurn = x.clkWhiteTurn;

clkSecWhite = x.clkSecWhite;

clkSecBlack = x.clkSecBlack;

}

rdgtSnapshot copy() {

return new rdgtSnapshot(this);

}

void set\_clockdata(boolean running, boolean batteryLow, boolean frontViewLeftSideHigh,

boolean blackTurn, boolean whiteTurn, int secW, int secB) {

clkRunning = running;

clkBatteryLow = batteryLow;

clkFrontViewLeftSideHigh = frontViewLeftSideHigh;

clkBlackTurn = blackTurn;

clkWhiteTurn = whiteTurn;

clkSecWhite = secW;

clkSecBlack = secB;

}

public boolean sameas(rdgtSnapshot x) {

for (int i=0; i<64; i=i+1) if (pieces[i] != x.pieces[i]) return false;

return true;

}

Date get\_time() { return time; }

void set\_emptyboard() {

time = new Date();

for (int i=0; i<64; i=i+1) pieces[i]=EMPTY;

}

void set\_boarddump(int[] all64, int startpos) {

time = new Date();

for (int i=0; i<64; i=i+1) {

int p = all64[i+startpos];

if (is\_piece(p)==false) {

System.out.println("Confused: Boarddump contained an unknown piece");

pieces[i]=EMPTY;

} else {

pieces[i]=p;

}

}

}

void set\_fieldupdate(int piece, int pos) {

time = new Date();

if (pos <0 || pos > 63) {

System.out.println("Confused: Fieldupdate for pos outside 0..63");

return;

}

if (is\_piece(piece)==false) {

System.out.println("Confused: Fieldupdate with an unknown piece");

return;

}

pieces[pos]=piece;

}

String seconds2hms(int s) {

return Integer.toString(s/36000)+Integer.toString((s%36000)/3600)+":"+

Integer.toString((s%3600)/600)+Integer.toString((s%360)/60)+":"+

Integer.toString((s%60)/10)+Integer.toString(s%10);

}

String debugprint\_clock() {

StringBuffer s = new StringBuffer(50);

s.append("Clock: White "); s.append(seconds2hms(clkSecWhite));

s.append(" Black "); s.append(seconds2hms(clkSecBlack));

System.out.println(s);

return s.toString();

}

String getTimeWhite(){

return seconds2hms(clkSecWhite);

}

String getTimeBlack(){

return seconds2hms(clkSecBlack);

}

boolean isWhiteTurn(){

return clkFrontViewLeftSideHigh;

}

String debugprint() {

debugprint\_clock();

StringBuffer s = new StringBuffer(300);

for (int i=0; i<8; i=i+1) {

for (int j=0; j<8; j=j+1) {

int p = pieces[(i\*8)+j];

if (p==EMPTY ) s.append(".");

if (p==WPAWN ) s.append("P");

if (p==WROOK ) s.append("R");

if (p==WKNIGHT) s.append("N");

if (p==WBISHOP) s.append("B");

if (p==WKING ) s.append("K");

if (p==WQUEEN ) s.append("Q");

if (p==BPAWN ) s.append("p");

if (p==BROOK ) s.append("r");

if (p==BKNIGHT) s.append("n");

if (p==BBISHOP) s.append("b");

if (p==BKING ) s.append("k");

if (p==BQUEEN ) s.append("q");

}

s.append("\n");

}

System.out.println(s);

return s.toString();

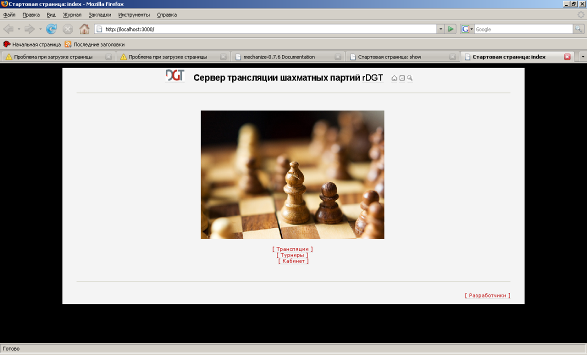
}

}

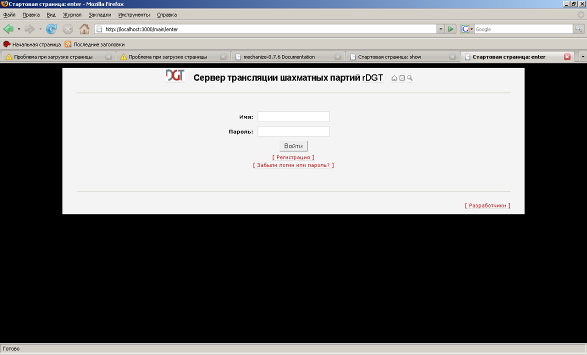
Приложение Д

Снимки экрана

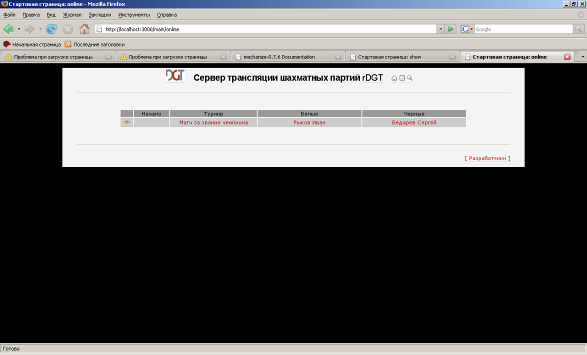
Д.1 Снимок экрана главной страницы сервера трансляции шахматных партий rDGT



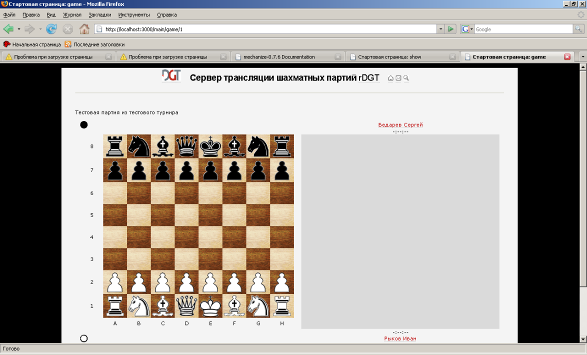
Д.2 Снимок экрана страницы авторизации пользователя



Д.3 Снимок экрана страницы просмотра текущих online трансляций



Д.4 Снимок экрана страницы просмотра шахматной партии



Приложение Е

Протокол DGT

#ifndef dgtbrd13

#define dgtbrd13

/\*

Protocol description for DGT chess board.

Copyright 1998 DGT Projects B.V

Version: 1.03 Single computer and bus support in one .h file

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

This protocol is protected under trade mark registration and copyrights.

It may not be used commercially without written permission

of DGT Projects B.V. It is illegal to transfer any registered trade mark

identifications by means of this protocol between any chessboard or other

application and any computer.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Main functionality of the DGT Electronic Chess Board

----------------------------------------------------

The DGT board is basically a sensor which senses the presense of the special

chess set pieces on the squares of the board. The situation on the board is

measured and can be communicated with an average maximum time delay of

200 mS.

Besides this detection function, the board communicates with an optional

DGT TopMatch Chess Clock, to give the data of the clock available to the

general interface.

Finally the board is equipped with an internal storage of the measured

piece positions.

The board supports two methods of communication: for single-board situations

a protocol for communication between one board and one computer is available.

For situations with many boards a network communications protocol is

available, where many boards can be connected in a bus structure. A separate

communication protocol is available for this bus structure.

The communication protocol for single board connections is described

in the following paragraph "Single board communication protocol". This

paragraph describes much more than only the communication protocol. All

developers should read this paragraph, even if they would only use bus

communication.

The special bus communication protocol is derived from the single board

communication and functionality, where the main added feature is the

possibility to address a specific board on a shared communication bus.

The commands and data contens are described in the paragraph "Bus

Communication Protocol", Note however that the contens can not be understood

without reading the single board communication paragraph.

Paragraph: Single board communication protocol

----------------------------------------------

The main function of the board is to transfer piece position information.

For this, three modes are available:

1. IDLE mode. This cancelles any of the two UPDATE modes. No automatic

transfer of moves.

2. UPDATE\_BOARD mode. On the moment that the board detects a removal, change

or placing of a piece, it outputs a DGT\_SEND\_UPDATE message

3. UPDATE mode. As UPDATE\_BOARD mode, where additional the clock data are send

regularly (at least every second)

The board accepts command codes from the computer RS232. The commands are

1-byte codes, sometimes followed by data (see code definition)

The board can send data to the computer. Data always carries a message header.

The message header contains a message code and the total message size in bytes.

The start of the incoming message can be recognised by the MSB of the message

identifier set to 1 (see definition).

Board to computer communication interfaces:

RS232 for communication with computer, receiving commands, sending data

- 9600 Baud, 1 stopbit, 1 startbit, no parity

- No use of handshaking, neither software nor hardware

Connection between Digital Game Timer TopMatch Clock and the board:

Based on NEC SBI protocol. Adaption of the definition given in

the DGT TopMatch documentation.

Connector assignments for DGT Electronic Board: See User

and Programmers Manual

Related to the before mentioned modes, and to piece position information

transfer, the following commands to the board are available:

1. DGT\_SEND\_RESET

puts the DGT Board in IDLE mode

2. DGT\_SEND\_CLK

on which the DGT board responds with a DGT\_MSG\_BWTIME message containing clock

information

3. DGT\_SEND\_BRD

on which the DGT Board responds with a DGT\_MSG\_BOARD\_DUMP message containing

the actual piece exising of all fields

4. DGT\_SEND\_UPDATE puts the DGT Board in the UPDATE mode, FRITZ5 compatible

5. DGT\_SEND\_UPDATE\_BRD puts the DGT Board in the UPDATE\_BOARD mode

6. DGT\_SEND\_UPDATE\_NICE puts the board in UPDATE mode, however transferring only clocktimes when any time info changed.

The DGT\_SEND\_CLK command and the DGT\_SEND\_BOARD command do not affect the current board

mode: i.e. when in UPDATE mode, it continues sending DGT\_SEND\_UPDATE messages.

Board Identification:

Each DGT Electronic Board carries a unique serial number,

a EEPROM configuration version number and a embedded program version number.

These data are unalterable by the users.

Current identification is:

"DGT Projects - This DGT board is produced by DGT Projects.\n

DGT Projects is a registered trade mark.\n

220798 ISP/bus/8KP/8KE/P6/Fritz5 Vs 1.00. Serial nr. 00137 1.0"

The board can be loaded by the user with a non-volatile one-byte bus number,

for future use with multiple board configurations.

On-board EEPROM:

The board carries a 8 kB cyclic non-volatile memory, in which all position

changes and clock information is stored during all power-up time. This

file can be read and processed.

Start of Definitions:

---------------------\*/

/\* COMMAND CODES FROM PC TO BOARD: \*/

/\* resulting in returning message(s): \*/

#define DGT\_SEND\_CLK 0x41

/\* results in a DGT\_MSG\_BWTIME message \*/

#define DGT\_SEND\_BRD 0x42

/\* results in a DGT\_MSG\_BOARD\_DUMP message \*/

#define DGT\_SEND\_UPDATE 0x43

/\* results in DGT\_MSG\_FIELD\_UPDATE messages and DGT\_MSG\_BWTIME messages

as long as the board is in UPDATE mode \*/

#define DGT\_SEND\_UPDATE\_BRD 0x44

/\* results in DGT\_MSG\_FIELD\_UPDATE messages

as long as the board is in UPDATE\_BOARD mode \*/

#define DGT\_RETURN\_SERIALNR 0x45

/\* results in a DGT\_MSG\_SERIALNR message \*/

#define DGT\_RETURN\_BUSADRES 0x46

/\* results in a DGT\_MSG\_BUSADRES message \*/

#define DGT\_SEND\_TRADEMARK 0x47

/\* results in a DGT\_MSG\_TRADEMARK message \*/

#define DGT\_SEND\_VERSION 0x4d

/\* results in a DGT\_MSG\_VERSION message \*/

#define DGT\_SEND\_UPDATE\_NICE 0x4b

/\* results in DGT\_MSG\_FIELD\_UPDATE messages and DGT\_MSG\_BWTIME messages,

the latter only at time changes,

as long as the board is in UPDATE\_NICE mode\*/

#define DGT\_SEND\_EE\_MOVES 0x49

/\* results in a DGT\_MSG\_EE\_MOVES message \*/

/\* not resulting in returning messages: \*/

#define DGT\_SEND\_RESET 0x40

/\* puts the board in IDLE mode, cancelling any UPDATE mode \*/

/\* DESCRIPTION OF THE MESSAGES FROM BOARD TO PC

A message consists of three header bytes:

MESSAGE ID one byte, MSB (MESSAGE BIT) always 1

MSB of MESSAGE SIZE one byte, MSB always 0, carrying D13 to D7 of the total message length, including the 3 header byte

LSB of MESSAGE SIZE one byte, MSB always 0, carrying D6 to D0 of the

total message length, including the 3 header bytes

followed by the data:

0 to ((2 EXP 14) minus 3) data bytes, of which the MSB is always zero.

\*/

/\* DEFINITION OF THE BOARD-TO-PC MESSAGE ID CODES and message descriptions \*/

/\* the Message ID is the logical OR of MESSAGE\_BIT and ID code \*/

#define MESSAGE\_BIT 0x80

/\* ID codes: \*/

#define DGT\_NONE 0x00

#define DGT\_BOARD\_DUMP 0x06

#define DGT\_BWTIME 0x0d

#define DGT\_FIELD\_UPDATE 0x0e

#define DGT\_EE\_MOVES 0x0f

#define DGT\_BUSADRES 0x10

#define DGT\_SERIALNR 0x11

#define DGT\_TRADEMARK 0x12

#define DGT\_VERSION 0x13

/\* Macros for message length coding (to avoid MSB set to 1) \*/

#define BYTE char

#define LLL\_SEVEN(a) ((BYTE)(a&0x7f)) /\* 0000 0000 0111 1111 \*/

#define LLH\_SEVEN(a) ((BYTE)((a & 0x3F80)>>7)) /\* 0011 1111 1000 0000 \*/

/\* DGT\_MSG\_BOARD\_DUMP is the message that follows on a DGT\_SEND\_BOARD

command \*/

#define DGT\_MSG\_BOARD\_DUMP (MESSAGE\_BIT|DGT\_BOARD\_DUMP)

#define DGT\_SIZE\_BOARD\_DUMP 67

/\* message format:

byte 0: DGT\_MSG\_BOARD\_DUMP

byte 1: LLH\_SEVEN(DGT\_SIZE\_BOARD\_DUMP) (=0 fixed)

byte 2: LLL\_SEVEN(DGT\_SIZE\_BOARD\_DUMP) (=67 fixed)

byte 3-66: Pieces on position 0-63

Board fields are numbered from 0 to 63, row by row, in normal reading

sequence. When the connector is on the left hand, counting starts at

the top left square. The board itself does not rotate the numbering,

when black instead of white plays with the clock/connector on the left hand.

In non-rotated board use, the field numbering is as follows:

Field A8 is numbered 0

Field B8 is numbered 1

Field C8 is numbered 2

..

Field A7 is numbered 8

..

Field H1 is numbered 63

So the board always numbers the black edge field closest to the connector

as 57.

Piece codes for chess pieces: \*/

#define EMPTY 0x00

#define WPAWN 0x01

#define WROOK 0x02

#define WKNIGHT 0x03

#define WBISHOP 0x04

#define WKING 0x05

#define WQUEEN 0x06

#define BPAWN 0x07

#define BROOK 0x08

#define BKNIGHT 0x09

#define BBISHOP 0x0a

#define BKING 0x0b

#define BQUEEN 0x0c

#define PIECE1 0x0d /\* future use: pointing device in rest \*/

#define PIECE2 0x0e /\* future use: pointing device right button \*/

#define PIECE3 0x0f /\* future use: pointing device left button \*/

/\* message format DGT\_MSG\_BWTIME \*/

#define DGT\_MSG\_BWTIME (MESSAGE\_BIT|DGT\_BWTIME)

#define DGT\_SIZE\_BWTIME 10

/\*

byte 0: DGT\_MSG\_BWTIME

byte 1: LLH\_SEVEN(DGT\_SIZE\_BWTIME) (=0 fixed)

byte 2: LLL\_SEVEN(DGT\_SIZE\_BWTIME) (=10 fixed)

byte 3:

D4: 1 = Flag fallen for left player, and clock blocked to zero

0 = not the above situation

D5: 1 = Time per move indicator on for left player ( i.e. Bronstein, Fischer)

0 = Time per move indicator off for left player

D6: 1 = Left players flag fallen and indicated on display

0 = not the above situation

(D7 is MSB)

D0-D3: Hours (units, 0-9 Binary coded) white player (or player at the A side of the board)

byte 4: Minutes (0-59, BCD coded)

byte 5: Seconds (0-59, BCD coded)

byte 6-8: the same for the other player

byte 9: Clock status byte: 7 bits

D0 (LSB): 1 = Clock running

0 = Clock stopped by Start/Stop

D1: 1 = tumbler position high on (white) player (front view: \ , left side high)

0 = tumbler position high on the other player (front view: /, right side high)

D2: 1 = Battery low indication on display

0 = no battery low indication on display

D3: 1 = Black players turn

0 = not black players turn

D4: 1 = White players turn

0 = not white players turn

D5: 1 = No clock connected; reading invalid

0 = clock connected, reading valid

D6: not used (read as 0)

D7: Always 0

The function of the information bits are derived from the full information

as described in the programmers reference manual for the DGT TopMatch

\*/

/\* message format DGT\_MSG\_FIELD\_UPDATE: \*/

#define DGT\_MSG\_FIELD\_UPDATE (MESSAGE\_BIT|DGT\_FIELD\_UPDATE)

#define DGT\_SIZE\_FIELD\_UPDATE 5

/\*

byte 0: DGT\_MSG\_FIELD\_UPDATE

byte 1: LLH\_SEVEN(DGT\_SIZE\_FIELD\_UPDATE) (=0 fixed)

byte 2: LLL\_SEVEN(DGT\_SIZE\_FIELD\_UPDATE) (=5 fixed)

byte 3: field number (0-63) which changed the piece code

byte 4: piece code including EMPTY, where a non-empty field became empty

\*/

/\* message format: DGT\_MSG\_TRADEMARK which returns a trade mark message \*/

#define DGT\_MSG\_TRADEMARK (MESSAGE\_BIT|DGT\_TRADEMARK)

/\*

byte 0: DGT\_MSG\_TRADEMARK

byte 1: LLH\_SEVEN(DGT\_SIZE\_TRADEMARK)

byte 2: LLL\_SEVEN(DGT\_SIZE\_TRADEMARK)

byte 3-end: ASCII TRADEMARK MESSAGE, codes 0 to 0x3F

The value of DGT\_SIZE\_TRADEMARK is not known beforehand, and may be in the

range of 0 to 256

Current trade mark message: ...

\*/

/\* Message format DGT\_MSG\_BUSADRES return message with bus adres \*/

#define DGT\_MSG\_BUSADRES (MESSAGE\_BIT|DGT\_BUSADRES)

#define DGT\_SIZE\_BUSADRES 5

/\*

byte 0: DGT\_MSG\_BUSADRES

byte 1: LLH\_SEVEN(DGT\_SIZE\_BUSADRES)

byte 2: LLL\_SEVEN(DGT\_SIZE\_BUSADRES)

byte 3,4: Busadres in 2 bytes of 7 bits hexadecimal value

Byte 3: 0bbb bbbb with bus adres MSB 7 bits

byte 4: 0bbb bbbb with bus adres LSB 7 bits

The value of the 14-bit busadres is het hexadecimal representation

of the (decimal coded) serial number

i.e. When the serial number is "01025 1.0" the busadres will be

byte 3: 0000 1000 (0x08)

byte 4: 0000 0001 (0x01)

\*/

/\* Message format DGT\_MSG\_SERIALNR return message with bus adres \*/

#define DGT\_MSG\_SERIALNR (MESSAGE\_BIT|DGT\_SERIALNR)

#define DGT\_SIZE\_SERIALNR 12

/\* returns 5 ASCII decimal serial number + space + 3 byte version string: \*/

/\* byte 0-5 serial number string, sixth byte is LSByte \*/

/\* byte 6: space \*/

/\* byte 7-9: Internal storage version nr: format "1.0" \*/

/\* Message format DGT\_MSG\_EE\_MOVES, which is the contens of the storage array \*/

/\* Message format DGT\_MSG\_VERSION return message with bus adres \*/

#define DGT\_MSG\_VERSION (MESSAGE\_BIT|DGT\_VERSION)

#define DGT\_SIZE\_VERSION 5

/\*

byte 0: DGT\_MSG\_VERSION

byte 1: LLH\_SEVEN(DGT\_SIZE\_VERSION)

byte 2: LLL\_SEVEN(DGT\_SIZE\_VERSION)

byte 3,4: Version in 2 bytes of 7 bits hexadecimal value

Byte 3: 0bbb bbbb with main version number MSB 7 bits

byte 4: 0bbb bbbb with sub version number LSB 7 bits

The value of the version is coded in binary

i.e. When the number is "1.02" the busadres will be

byte 3: 0000 0001 (0x01)

byte 4: 0000 0010 (0x02)

\*/

#define DGT\_MSG\_EE\_MOVES (MESSAGE\_BIT|DGT\_EE\_MOVES)

/\* DGT\_SIZE\_EE\_MOVES is defined in dgt\_ee1.h: current (0x2000-0x100+3) \*/

/\*

message format:

byte 0: DGT\_MSG\_EE\_MOVES

byte 1: LLH\_SEVEN(DGT\_SIZE\_EE\_MOVES)

byte 2: LLL\_SEVEN(DGT\_SIZE\_EE\_MOVES)

byte 3-end: field change storage stream: See defines below for contens

The DGT\_MSG\_EE\_MOVES message contains the contens of the storage,

starting with the oldest data, until the last written changes, and will

always end with EE\_EOF

\*/

/\*

Description of the EEPROM data storage and dump format

------------------------------------------------------

General: The internal EEPROM storage can be seen as a cyclic buffer with length

0x1f00 bytes, with one pointer, pointing to the last written byte in the buffer.

Only at this pointer location, data can be written, incrementing the pointer.

The written data always overwrites the oldest data.

In this buffer, sequentially messages are written. The messages are of various

length, from 1 byte to 5 bytes, specific for every message.

Various events generate a message that is written in the storage, in the

sequence as the events occur. When the buffer is downloaded and read, the event

messages can be found, starting with the oldest event, and the latest event in

the end of the buffer, followed by EE\_EOF.

- At power-on, three tags EE\_NOP are written, followed by a one-byte

EE\_POWERUP message.

After this, an UPDATE\_BOARD message is written (in virtually random sequence)

for every piece that is found on the board, at power-on.

When the board is equipped with a watchdog timer, and the watchdog times out,

an EE\_WATCHDOG\_ACTION is written and after that, the above described power-up

procedure takes place.

- When at any time a normal starting position for chess is found, with the

player for white having the board connector on his left hand, an EE\_BEGINPOS tag

is written, and an EE\_BEGINPOS\_ROT tag is written when white has the

connector at his right hand (rotated)

- When 16 chess figures are found on the board, all in the A, B, G and H row,

which are not(!) in a normal chess starting position, the one-byte

EE\_FOURROWS message is written, to be tolerant on erroneous placement and i.e. to be able to play the "Random Chess" as proposed by Bobby

Fischer. The exact position of the pieces has to be analyzed on the context: or found in the previous piece move messages, or found in the

coming piece move messages.

When an empty board is detected, the one-byte EE\_EMPTYBOARD message is

written.

The above described detection of begin positions or empty-board has a certain

hysteresis: only after more than two pieces have been out of the begin

positions the search for begin positions is restarted, resulting in possibly

new tag writing. This to avoid flushing the buffer full with data, only because

of one bad positioned and flashing piece.

When the data of the internal storage are sent upon reception of the

DGT\_SEND\_EE\_MOVES command, the one-byte EE\_DOWNLOADED message is sent

On every detected change of piece positions this change is written to EEPROM

in a 2-byte message, which cover exactly the same data as is sent to the PC

in the UPDATE\_BOARD mode.

The formatting of the 2-byte piece update tag is:

First byte: 0t0r nnnn (n is piece code, see before)

(piece code EMPTY when a piece is removed)

(t is recognition bit, always 1)

(r is reserved)

so the first byte of this tag is always in the

range 0x40 to 0x5f

Second byte: 00ii iiii (i = 0 to 63 (0x3f), the field number as

defined before)

NB: when one piece only is changing, the new value is overwrites the

piece update field described above, instead of generating a new message

in the internal storage.

The same kind of optimization is included for begin-position tags:

a EE\_BEGINPOS or EE\_BEGINPOS\_ROT or EE\_FOURROWS is not written, when

between the previous written tags and the new occurence of the begin-

situation only 2 or 1 piece were out of the tagged beginsituation.

On the pressing of the clock, the time of the halted clock is written in

a time message. It might be that when the moves are done very fast, the

storage is skipped. Note: the two clock sides are identified only by

left and right side of the clock: When the board is swapped, the clock

data are not (!) swapped.

The clock data are written on tumbler position change, so at the beginning

of the game, the game starting times will be lost.

Format of a three-byte time message:

First byte: 0uuf hhhh (f=1 for time in left clock screen, seen from the front)

( hhhh: hours, valued 0 to 9)

(uu recognition bits, both 1, so byte 0 has the

( value range of 0x60 to 0x69, or 0x70 to 0x79)

Second byte: 0MMM mmmm (MMM: Tens of minutes (range 0 to 5),

(mmmm: minute units, range 0 to 9)

Third byte: 0SSS ssss (SSS: tens of seconds, range 0-5)

(ssss: seconds units, range 0-9)

On the recognition of the first byte of a message: The above definitions

imply that the first byte of a message is always ranged

from 40-5f for a field change message, 60-69 or 70-79 for a time message,

and 6a to 6f, or 7a to 7f for a 1-byte message.

(all values hexadecimal)

\*/

/\* Definition of the one-byte EEPROM message codes \*/

#define EE\_POWERUP 0x6a

#define EE\_EOF 0x6b

#define EE\_FOURROWS 0x6c

#define EE\_EMPTYBOARD 0x6d

#define EE\_DOWNLOADED 0x6e

#define EE\_BEGINPOS 0x6f

#define EE\_BEGINPOS\_ROT 0x7a

#define EE\_START\_TAG 0x7b

#define EE\_WATCHDOG\_ACTION 0x7c

#define EE\_NOP 0x7f

/\* 7d and 7e reserved for future use\*/

/\*

Notes on the communication dynamics:

The squares of the board are sampled one by one, where a full scan takes

about 200-250 ms. Due to this scan time, it can be, that a piece that is

moved from square e.g. A2 to A3 can be seen on both squares, in the same

scan. On a next scan of course, the old position is not seen anymore.

When in UPDATE mode, this means, that the information on changes on the

squares can come in opposite sequence: first the new occurence is reported,

then the clearing of the old position is sent.

When a piece B on C4 is taken by piece A on B3, it can be that the following

changes are reported:

A on C4

Empty on B3

(and Empty on C4 is never seen)

An other extreme situation can occur e.g. when a queen on C1 takes a pawn

on C4. The reported changes can be (in sequence):

Empty on C4 (the pawn is taken by one hand)

Queen on C2

Queen on C3

Empty on C1

Empty on C2

Queen on C4

Empty on C3

For writing software it is important to take these dynamics into account.

Some effort needs to be made to reconstruct the actual moves of the pieces.

See also the programmers and users manual

Paragraph: Bus communication protocol

-------------------------------------

Differencens between busmode and single board mode:

\* In bus mode, RS232 input and RS232 output are connected to all boards.

The RS232 output of the board is configured as a pull-up driver: with a

pull-down resistor on the RS232 pull-up line. Now all boards receive all

commands from the computer, and can all send data to the computer.

\* In single board mode the board has a small incoming commands

buffer (10 bytes).

\* The bus mode has only a one-command incoming commands buffer

\* When entered in single board mode, the board status always switches to IDLE:

changes are not send automatically

Bus mode is default power up mode of the board. The board recognises

bus commands from the start.

However, single board commands are recognised and answered.

The board switches to single board mode on the moment, a single board

command is recognised. Switching back to bus mode is invoked by the

extra command DGT\_TO\_BUSMODE or by sending a busmode command. (NB This

busmode command causing the swithing is not processed!)

For all detailed hardware descriptions: call.

\*/

/\* one added functon for swiching to busmode by command: \*/

#define DGT\_TO\_BUSMODE 0x4a

/\*

This is an addition on the other single-board commands. This command is

recognised in single-board mode. The RS232 output goes in

pull-up mode and bus commands are immediatly recognised hereafter.

Note that when the board is in single-board mode, and eventually a bus

mode command is found, this command is not processed, but the board

switches to bus mode. The next (bus) command is processed regularly.\*/

/\* Bus mode commands: \*/

#define DGT\_BUS\_SEND\_CLK (0x01 | MESSAGE\_BIT)

#define DGT\_BUS\_SEND\_BRD (0x02 | MESSAGE\_BIT)

#define DGT\_BUS\_SEND\_CHANGES (0x03 | MESSAGE\_BIT)

#define DGT\_BUS\_REPEAT\_CHANGES (0x04 | MESSAGE\_BIT)

#define DGT\_BUS\_SET\_START\_GAME (0x05 | MESSAGE\_BIT)

#define DGT\_BUS\_SEND\_FROM\_START (0x06 | MESSAGE\_BIT)

#define DGT\_BUS\_PING (0x07 | MESSAGE\_BIT)

#define DGT\_BUS\_END\_BUSMODE (0x08 | MESSAGE\_BIT)

#define DGT\_BUS\_RESET (0x09 | MESSAGE\_BIT)

#define DGT\_BUS\_IGNORE\_NEXT\_BUS\_PING (0x0a | MESSAGE\_BIT)

#define DGT\_BUS\_SEND\_VERSION (0x0b | MESSAGE\_BIT)

// extra return headers for bus mode:

#define DGT\_MSG\_BUS\_BRD\_DUMP (0x03 | MESSAGE\_BIT)

#define DGT\_MSG\_BUS\_BWTIME (0x04 | MESSAGE\_BIT)

#define DGT\_MSG\_BUS\_UPDATE (0x05 | MESSAGE\_BIT)

#define DGT\_MSG\_BUS\_FROM\_START (0x06 | MESSAGE\_BIT)

#define DGT\_MSG\_BUS\_PING (0x07 | MESSAGE\_BIT)

#define DGT\_MSG\_BUS\_START\_GAME\_WRITTEN (0x08 | MESSAGE\_BIT)

#define DGT\_MSG\_BUS\_VERSION (0x09 | MESSAGE\_BIT)

// extra defines for bus length info:

#define DGT\_SIZE\_BUS\_PING 6

#define DGT\_SIZE\_BUS\_START\_GAME\_WRITTEN 6

#define DGT\_SIZE\_BUS\_VERSION 8 // was 6 up to version 1.2

/\* Definition of different commands&data

All commands DGT\_BUS\_xx have the following format:

byte 1: command, i.e. DGT\_BUS\_SEND\_BDR (D7 always 1)

byte 2: MSB of addressed board (D7 always 0)

byte 3: LSB of addressed board (D7 always 0)

byte 4: checksum: this is the sum of all bytes from start of the message

upto the last byte before the checksum. (D7 always 0)

I.e. message code 0x81 0x10 0x06 will carry checksum byte 0x17

DGT\_BUS\_SEND\_CLK

asks for clock information of addressed board.

Will result in a DGT\_MSG\_BUS\_BWTIME message from the board.

DGT\_BUS\_SEND\_BRD

asks for a board dump of addressed board.

Will result in a DGT\_MSG\_BUS\_BRD\_DUMP message from the board.

DGT\_BUS\_SEND\_CHANGES

asks for all stored information changes from the moment of the last

DGT\_BUS\_SEND\_CHANGES. Will result in a DGT\_MSG\_BUS\_UPDATE message from

the board.

In case these data do not arrive properly, the data can be asked again

with the DGT\_BUS\_REPEAT\_CHANGES command.

DGT\_BUS\_REPEAT\_CHANGES

Causes the board to send last sent packet of changes again.

DGT\_BUS\_SET\_START\_GAME

sets an EE\_START\_TAG tag in the internal board changes buffer, for use in the

following command DGT\_BUS\_SEND\_FROM\_START. After this EE\_START\_TAG the

positions of the pieces are all logged in the file.

The command is answered with a DGT\_MSG\_BUS\_START\_GAME\_WRITTEN message,

about 70 msec. after receipt of DGT\_BUS\_SET\_START\_GAME

DGT\_BUS\_SEND\_FROM\_START

causes the board to send a DGT\_MSG\_BUS\_FROM\_START message, containing

all update information starting with EE\_START\_TAG until the last registered

changes (excluding the moves that are to be sent with the DGT\_BUS\_SEND\_CHANGES

command). Remember that after the EE\_START\_TAG all piece positions are written

in the eeprom file.

DGT\_BUS\_PING

causes the addressed board to send a DGT\_MSG\_BUS\_PING message.

NB: when the DGT\_BUS\_PING command is sent with board address 0 (zero )

all connected boards will reply with a DGT\_MSG\_BUS\_PING message, randomly

spread over a 1100 msec. interval. This to avoid collision. For reliable

identification of all connected boards, this process should be repeated

sometimes with checking of checksums!

DGT\_IGNORE\_NEXT\_BUS\_PING

is used in the process of detecting connected boards on a bus. After this

command (which itself sends a DGT\_MSG\_BUS\_PING as handshake) the first

following DGT\_BUS\_PING with address zero (!) is ignored. This command

can be used to suppress response of already detected boards, and decreases

the chance of bus collisions.

This command responds immediately with a DGT\_MSG\_BUS\_PING.

DGT\_BUS\_END\_BUSMODE

causes the board to quit busmode, and go into single-connection mode.

Be careful not to use this command in a multiple board configuration!

NOTE: Any single-board command arriving during bus mode will

switch a board to single-board mode and will be processed. When sent

with address 0 the command is processed on all connected boards

DGT\_BUS\_RESET

forces a power-up reset procedure. NB: this procedure takes some seconds.

When sent with address 0 the command is processed on all conected boards

DEFINITION OF THE MESSAGE DATA FORMATS FROM BOARD TO PC

-------------------------------------------------------

General: the message format is:

byte 1: message type byte (MSB = 1)

byte 2: message length MSB (from byte 1 to checksum) (D7=0)

byte 3: message length LSB (containing D0 to D6 of the length (D7 = 0)

byte 4: board address MSB (D7=0)

byte 5: board address LSB (D7=0)

< data bytes: 0 to theoretically 16K-6 bytes >

last byte: checksum. This is the sum of all byte values from byte 1 to

the byte before this checksum. D7 is forced to 0

DGT\_MSG\_BUS\_BRD\_DUMP

the data area contains the piece codes from field 1 to field 64, in format

identical as the single board command

DGT\_MSG\_BUS\_BWTIME

the data area contains the clock time information, in format identical as

the single board command

DGT\_MSG\_BUS\_UPDATE

the data area contains a variable amount of change information, formatted

as described in the DGT\_DUMP\_EEMOVES message

DGT\_MSG\_BUS\_FROM\_START

the data area contains a variable amount of change information, formatted

as described in the DGT\_DUMP\_EEMOVES message

DGT\_MSG\_BUS\_PING

the data area is empty: a message of 6 bytes is returned.

DGT\_MSG\_BUS\_START\_GAME\_WRITTEN

The same format as for DGT\_MSG\_BUS\_PING

DGT\_MSG\_BUS\_VERSION

The two data bytes contain binary version number: first byte main number

second byte sub number.

Tips for usage of the bus mode:

A. On connection of power and starting of the communication process:

- Check communication with addressed DGT\_BUS\_PING commands to all expected

boards.

LOOP:

- Send DGT\_BUS\_IGNORE\_NEXT\_PING to all found boards

- Check eventually extra boards or unknown board busnumbers by using

a DGT\_BUS\_PING with address zero.

- Register the found boards, and go to LOOP:, until no more boards are found

B. At the start of an event: when all boards have pieces in starting

position, send DGT\_BUS\_SET\_START\_GAME commands to all boards.

- Read full clock times from all boards.

C. During a game: send DGT\_BUS\_SEND\_CHANGES to all boards sequentially,

once every few seconds. The returned data should logically match

with the expected piece positions. When any mismatch occurs: ask full

position with DGT\_BUS\_SEND\_BRD.

D. When previous data are lost: send a DGT\_BUS\_SEND\_FROM\_START command

which returns the full registered changes from the starting position.

On every sent message: The responce time at the board is within milliseconds

normally, except when the board is storing a measured change in the

internal EEPROM: Then the responce time can be up to 20 milliseconds.

So allow a time-out on sent messages of say 80 mS.

Checksum errors: when a received checksum does not match, resend the command

except on the DGT\_BUS\_SEND\_CHANGES: then the command DGT\_BUS\_REPEAT\_CHANGES

should be used, to avoid discarding of the changes sent by the board.

On clock data:

- reading data out of the clock and make them available

for communication causes a delay of up to 1 second between clock display

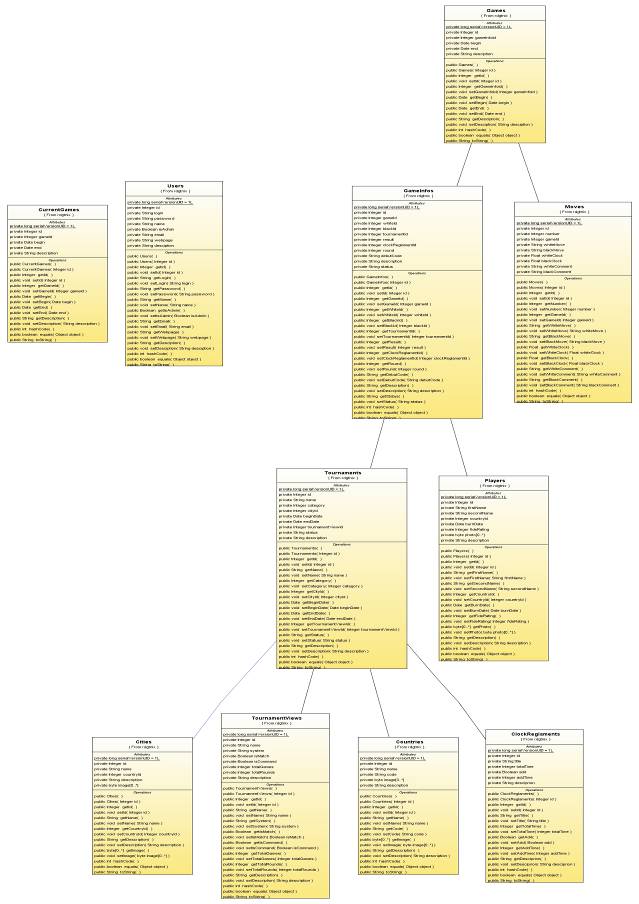
and received data.

\*/

#endif

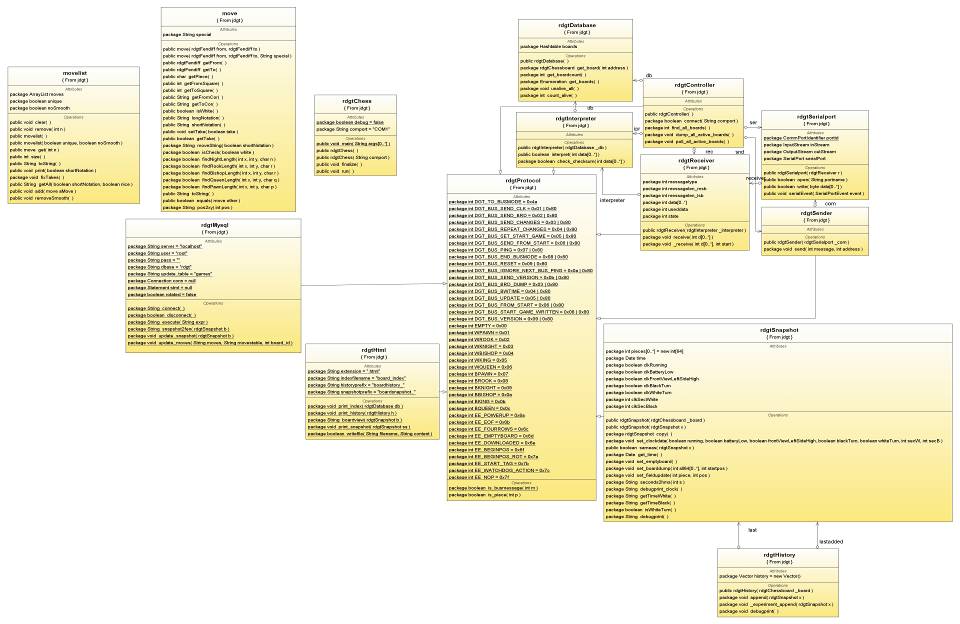
Приложение Ж

Общая структура транслятора шахматных партий



Приложение З

Общая структура регистратора шахматных партий



Приложение И

Анализ результатов тестовых испытаний

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № п/п | Показатель | Абсолютный  количественный показатель | Относительный количественный показатель в % | Степень соответствия условиям соревновательного процесса  (субъективная оценка главного арбитра соревнований) |
| 1 | Количество сбоев в работе СШПО в ходе всего сорев-новательного про-цесса | 0 | – | ПОЛНОСТЬЮ СООТВЕТСТВУЕТ |
| 2 | Количество вре-менных простоев в работе СШПО в ходе всего сорев-новательного про-цесса | 0 | – | ПОЛНОСТЬЮ СООТВЕТСТВУЕТ |
| 3 | Количество несоот-ветствий отобража-емого хода ходу, сделанному на ша-хматной доске | 5 | 0,48 | ПОЛНОСТЬЮ СООТВЕТСТВУЕТ |
| 4 | Количество вре-менных задержек в ходе отображения игрового процесса | 1 | 0,01 | ПОЛНОСТЬЮ СООТВЕТСТВУЕТ |
| 5 | Количество вре-менных задержек переноса шахмат-ных партий в элек-тронную форму | 1 | 0,01 | ПОЛНОСТЬЮ СООТВЕТСТВУЕТ |

Директор СП «ШК» Амзоров А.М.

Главный арбитр соревнований Федоров В.Н.

Приложение К

Листинг модуля вещания шахматных партий

// Фигура

var Figure = {

create: function(color, type, vertical) {

function figure() {}

var self = new figure();

self.id = color + type + vertical;

self.image = document.createElement('img');

self.image.src = get\_image\_path() + (color + type + '.gif').toLowerCase();

self.image.size = '8x8';

return self;

}

};

// Поле доски

var Field = {

create: function(vertical, horizontal) {

function field() {}

var self = new field();

// Идентификатор поля, соответствует ячейке в таблице, являющейся шахматной доской

self.id = "field" + vertical + horizontal;

self.horizontal = horizontal;

self.vertical = vertical;

self.cell = $(self.id);

//

function v(letter) {

letter = letter.toLowerCase();

if(letter == 'a') {

return 1;

}

if(letter == 'b') {

return 2;

}

if(letter == 'c') {

return 3;

}

if(letter == 'd') {

return 4;

}

if(letter == 'e') {

return 5;

}

if(letter == 'f') {

return 6;

}

if(letter == 'g') {

return 7;

}

if(letter == 'h') {

return 8;

}

return 0;

}

var vv = v(self.vertical);

var hh = parseInt(self.horizontal);

function isWhite(summa) {

var is\_white = (summa / 2) - (Math.floor(summa / 2)) == 0;

//log(summa + ' - ' + is\_white);

return !is\_white;

}

self.cell.className = isWhite(vv + hh) ? 'board\_white\_field' : 'board\_black\_field';

self.set\_figure = function(figure) {

if(figure == null) {

log('null in ' + self.id);

}

self.delete\_figure();

self.cell.appendChild(figure.image);

self.current\_figure = figure;

}

self.init\_figure = function(figure) {

self.set\_figure(figure);

self.first\_figure = figure;

self.last\_figure = figure;

}

self.delete\_figure = function() {

this.cell.innerHTML = "";

}

self.first\_figure = null;

self.current\_figure = null;

self.last\_figure = null;

return self;

}

};

// Ход шахматной партии

var Move = {

create: function() {

function move() {}

var self = new move();

self.color = '';

self.next = null;

self.prev = null;

self.number = 0;

self.field\_from = null;

self.field\_to = null;

self.figure = null;

self.figure\_dead = null;

self.is\_short\_castling = false;

self.is\_long\_castling = false;

self.forward = function() {

if(self.is\_short\_castling || self.is\_long\_castling) {

self.field\_from[0].delete\_figure();

self.field\_from[1].delete\_figure();

self.field\_to[0].set\_figure(self.figure[0]);

self.field\_to[1].set\_figure(self.figure[1]);

} else {

self.field\_from.delete\_figure();

self.field\_to.set\_figure(self.figure);

}

}

return self;

}

};

var Chess = {

version: "2.0",

refresh\_delay: 2, // интервал времени между обновлениями позиции в секундах

repaint\_delay: 0.5, // интервал времени между перерисовками позиции в секундах

moves: '/main/game\_moves/1',

createBoard: function() {

var board = Board.create();

board.init();

return board;

},

createField: function(vertical, horizontal) {

var self = Field.create(vertical, horizontal);

return self;

},

createFigure: function(color, type, vertical) {

var self = Figure.create(color, type, vertical);

return self;

},

createMove: function() {

var self = Move.create();

return self;

}

};

// Шахматная доска

var Board = {

create: function() {

function board() { }

var self = new board();

self.white\_number = 0;

self.black\_number = 0;

self.first\_move = null;

self.current\_move = null;

self.last\_move = null;

// Могут быть состояния

// listen - следить за ходом партий, отображаются актуальные ходы

// custom - наблюдатель выбрал произвольную позицию в партии и навигирует по партии свободно

self.status = 'listen';

self.fields = new Array(64);

self.figures = new Array(32);

self.all\_moves = '';

// Обновление позиции

self.refresh = function() {

function existsMove(color, annotation) {

var move = getMove(color, annotation);

var exists = move != "\*\*\*\*\*\*" && move != "";

return exists;

}

function getMove(color, annotation) {

var m = annotation.substring(color == 'w' ? 0 : 6, color == 'w' ? 6 : 12);

return m;

}

function isCastling(color, type, annotation) {

var move = getMove(color, annotation);

var castling = (type == 'short' ? move.substring(0, 3) == 'O-O' : move.substring(0, 5) == 'O-O-O');

return castling;

}

function getField(is\_short, is\_long, target, color, annotation) {

if(is\_short || is\_long) {

if(color == 'w') {

if(is\_short) {

return new Array(self.fields[target == 'from' ? 4 : 6], self.fields[target == 'from' ? 7 : 5]);

}

if(is\_long) {

return new Array(self.fields[target == 'from' ? 4 : 2], self.fields[target == 'from' ? 0 : 3]);

}

}

if(color == 'b') {

if(is\_short) {

return new Array(self.fields[target == 'from' ? 60 : 62], self.fields[target == 'from' ? 63 : 61]);

}

if(is\_long) {

return new Array(self.fields[target == 'from' ? 60 : 58], self.fields[target == 'from' ? 56 : 59]);

}

}

}

var field = annotation.substring(color == 'w' ? (target == 'from' ? 1 : 4) : (target == 'from' ? 7 : 10), color == 'w' ? (target == 'from' ? 3 : 6) : (target == 'from' ? 9 : 12));

var v = field.substring(0, 1).toLowerCase();

var h = field.substring(1, 2).toLowerCase();

//log('getField: v ' + v + ' h: ' + h);

var iField = (parseInt(h) - 1) \* 8;

//log('tmp ' + iField);

if(v == 'b') {

iField += 1;

}

if(v == 'c') {

iField += 2;

}

if(v == 'd') {

iField += 3;

}

if(v == 'e') {

iField += 4;

}

if(v == 'f') {

iField += 5;

}

if(v == 'g') {

iField += 6;

}

if(v == 'h') {

iField += 7;

}

//log('return ' + iField);

return self.fields[iField];

}

function getFigure(is\_short, is\_long, target, color, field) {

if(is\_short || is\_long) {

if(color == 'w') {

if(is\_short) {

return new Array(self.figures[0], self.figures[7]);

}

if(is\_long) {

return new Array(self.figures[0], self.figures[6]);

}

}

if(color == 'b') {

if(is\_short) {

return new Array(self.figures[16], self.figures[23]);

}

if(is\_long) {

return new Array(self.figures[16], self.figures[22]);

}

}

}

if(target == 'alive') {

return field.last\_figure;

}

if(target == 'dead') {

return field.last\_figure;

}

}

// Запрашиваем обновление позиции

new Ajax.Request(Chess.moves, {

method: 'get',

onSuccess: function(transport) {

self.all\_moves = transport.responseText;

}

});

// Анализируем ходы партии

var splits = self.all\_moves.split('|');

var moves = new Array();

for(i = 0; i < splits.length; i++) {

if(splits[i] != null) {

moves.push(splits[i

}

}

// Количество полученных ходов

var number = moves.length;

var current\_number = self.last\_move == null ? 1 : self.last\_move.number;

if(number == current\_number && self.last\_move != null && self.last\_move.color == 'w') {

var move\_annotation = moves[number - 1];

if(move\_annotation != '' && existsMove('b', moves[number - 1])) {

var move = Chess.createMove();

move.is\_short\_castling = isCastling('b', 'short', move\_annotation);

move.is\_long\_castling = isCastling('b', 'long', move\_annotation);

move.field\_from = getField(move.is\_short\_castling, move.is\_long\_castling, 'from' , 'b', move\_annotation);

move.field\_to = getField(move.is\_short\_castling, move.is\_long\_castling, 'to', 'b', move\_annotation);

move.figure = getFigure(move.is\_short\_castling, move.is\_long\_castling, 'alive', 'b', move.field\_from);

move.figure\_dead = getFigure(false, false, 'dead', 'b', move.field\_to);

move.number = self.last\_move != null ? ('b' == 'w' ? (self.last\_move.number + 1) : self.last\_move.number) : 1;

move.color = 'b';

move.field\_from.last\_figure = null;

move.field\_to.last\_figure = move.figure;

move.prev = self.last\_move;

if(self.last\_move != null) {

self.last\_move.next = move;

}

self.last\_move = move;

self.changes = true;

}

}

if(number > current\_number) {

var c = ['w', 'b'];

for(var iMove = current\_number; iMove <= number; iMove++) {

for(var ic = 0; ic < c.length; ic++) {

var move\_annotation = moves[iMove - 1];

if(existsMove(c[ic], move\_annotation)) {

var move = Chess.createMove();

if(self.first\_move == null) {

self.first\_move = move;

}

move.is\_short\_castling = isCastling(c[ic], 'short', move\_annotation);

move.is\_long\_castling = isCastling(c[ic], 'long', move\_annotation);

move.field\_from = getField(move.is\_short\_castling, move.is\_long\_castling, 'from' , c[ic], move\_annotation);

move.field\_to = getField(move.is\_short\_castling, move.is\_long\_castling, 'to', c[ic], move\_annotation);

move.figure = getFigure(move.is\_short\_castling, move.is\_long\_castling, 'alive', c[ic], move.field\_from);

move.figure\_dead = getFigure(move.is\_short\_castling, move.is\_long\_castling, 'dead', c[ic], move.field\_to);

move.number = self.last\_move != null ? (c[ic] == 'w' ? (self.last\_move.number + 1) : self.last\_move.number) : 1;

move.color = c[ic];

var ff = move.field\_from;

var ft = move.field\_to;

var fg = move.figure;

if(fg == null) {

continue;

}

move.field\_from.last\_figure = null;

move.field\_to.last\_figure = move.figure;

move.prev = self.last\_move;

if(self.last\_move != null) {

self.last\_move.next = move;

}

self.last\_move = move;

self.changes = true;

}

}

}

}

}

self.changes = false;

// Перерисовка позиции

self.repaint = function() {

if(self.current\_move == null) {

if(self.first\_move == null) {

return;

}

self.current\_move = self.first\_move;

}

if(self.current\_move.color == 'w' && self.white\_number == self.current\_move.number ||

self.current\_move.color == 'b' && self.black\_number == self.current\_move.number) {

if(self.current\_move.next != null) {

self.current\_move = self.current\_move.next;

}

return;

}

self.current\_move.forward();

if(self.current\_move.color == 'w') {

self.white\_number++;

}

if(self.current\_move.color == 'b') {

self.black\_number++;

}

if(self.current\_move.next != null) {

self.current\_move = self.current\_move.next;

}

}

// Инициализируем доску, создаем поля и фигуры

self.init = function() {

var v = new Array("A", "B", "C", "D", "E", "F", "G", "H");

var f = new Array("K", "Q", "B", "N", "R", "P");

var c = new Array("w", "b");

// Создаем представление доски - таблицу

var tableBoard = document.createElement('table');

tableBoard.border = 1;

tableBoard.align = "center";

tableBoard.width = "512px";

var tableBody = document.createElement('tbody');

tableBoard.appendChild(tableBody);

var tableDiv = $('board');

tableDiv.appendChild(tableBoard);

var rows = new Array(document.createElement('tr'), document.createElement('tr'), document.createElement('tr'), document.createElement('tr'),

document.createElement('tr'), document.createElement('tr'), document.createElement('tr'), document.createElement('tr'));

for(var iRow = 7; iRow >= 0; iRow--) {

tableBody.appendChild(rows[iRow]);

}

// создаем ячейки

for(var iVertical = 0; iVertical < 8; iVertical++) {

for(var iHorizontal = 8; iHorizontal > 0; iHorizontal--) {

var cell = document.createElement('td');

cell.width = '64px';//(100 / 8).toString() + "%";

cell.height = cell.width;

cell.align = 'center';

cell.id = "field" + v[iVertical] + iHorizontal.toString();

rows[iHorizontal - 1].appendChild(cell);

}

}

// поля

log('creating fields ...');

var iField = 0;

for(var iHorizontal = 1; iHorizontal < 9; iHorizontal++) {

for(var iVertical = 0; iVertical < 8; iVertical++) {

self.fields[iField] = Chess.createField(v[iVertical], iHorizontal.toString());

iField++;

}

}

log('creating figure ...');

// фигуры

self.figures[0] = Chess.createFigure(c[0], f[0], ''); // wk - 0

self.figures[1] = Chess.createFigure(c[0], f[1], ''); // wq - 1

self.figures[2] = Chess.createFigure(c[0], f[2], v[2]); // wbC - 2

self.figures[3] = Chess.createFigure(c[0], f[2], v[5]); // wbF - 3

self.figures[4] = Chess.createFigure(c[0], f[3], v[1]); // wnB - 4

self.figures[5] = Chess.createFigure(c[0], f[3], v[6]); // wnG - 5

self.figures[6] = Chess.createFigure(c[0], f[4], v[0]); // wrA - 6

self.figures[7] = Chess.createFigure(c[0], f[4], v[7]); // wrH - 7

self.figures[8] = Chess.createFigure(c[0], f[5], v[0]); // wpA - 8

self.figures[9] = Chess.createFigure(c[0], f[5], v[1]); // wpB - 9

self.figures[10] = Chess.createFigure(c[0], f[5], v[2]); // wpC - 10

self.figures[11] = Chess.createFigure(c[0], f[5], v[3]); // wpD - 11

self.figures[12] = Chess.createFigure(c[0], f[5], v[4]); // wpE - 12

self.figures[13] = Chess.createFigure(c[0], f[5], v[5]); // wpF - 13

self.figures[14] = Chess.createFigure(c[0], f[5], v[6]); // wpG - 14

self.figures[15] = Chess.createFigure(c[0], f[5], v[7]); // wpH - 15

self.figures[16] = Chess.createFigure(c[1], f[0], ''); // bk - 16

self.figures[17] = Chess.createFigure(c[1], f[1], ''); // bq - 17

self.figures[18] = Chess.createFigure(c[1], f[2], v[2]); // bbC - 18

self.figures[19] = Chess.createFigure(c[1], f[2], v[5]); // bbF - 19

self.figures[20] = Chess.createFigure(c[1], f[3], v[1]); // bnB - 20

self.figures[21] = Chess.createFigure(c[1], f[3], v[6]); // bnG - 21

self.figures[22] = Chess.createFigure(c[1], f[4], v[0]); // brA - 22

self.figures[23] = Chess.createFigure(c[1], f[4], v[7]); // brH - 23

self.figures[24] = Chess.createFigure(c[1], f[5], v[0]); // bpA - 24

self.figures[25] = Chess.createFigure(c[1], f[5], v[1]); // bpB - 25

self.figures[26] = Chess.createFigure(c[1], f[5], v[2]); // bpC - 26

self.figures[27] = Chess.createFigure(c[1], f[5], v[3]); // bpD - 27

self.figures[28] = Chess.createFigure(c[1], f[5], v[4]); // bpE - 28

self.figures[29] = Chess.createFigure(c[1], f[5], v[5]); // bpF - 29

self.figures[30] = Chess.createFigure(c[1], f[5], v[6]); // bpG - 30

self.figures[31] = Chess.createFigure(c[1], f[5], v[7]); // bpH - 31

// расставляем исходную позицию

self.set\_begin\_position();

}

self.set\_begin\_position = function() {

log('initialize begin position ...');

self.fields[56].init\_figure(self.figures[22]);

self.fields[57].init\_figure(self.figures[20]);

self.fields[58].init\_figure(self.figures[18]);

self.fields[59].init\_figure(self.figures[17]);

self.fields[60].init\_figure(self.figures[16]);

self.fields[61].init\_figure(self.figures[19]);

self.fields[62].init\_figure(self.figures[21]);

self.fields[63].init\_figure(self.figures[23]);

self.fields[48].init\_figure(self.figures[24]);

self.fields[49].init\_figure(self.figures[25]);

self.fields[50].init\_figure(self.figures[26]);

self.fields[51].init\_figure(self.figures[27]);

self.fields[52].init\_figure(self.figures[28]);

self.fields[53].init\_figure(self.figures[29]);

self.fields[54].init\_figure(self.figures[30]);

self.fields[55].init\_figure(self.figures[31]);

self.fields[8].init\_figure(self.figures[8]);

self.fields[9].init\_figure(self.figures[9]);

self.fields[10].init\_figure(self.figures[10]);

self.fields[11].init\_figure(self.figures[11]);

self.fields[12].init\_figure(self.figures[12]);

self.fields[13].init\_figure(self.figures[13]);

self.fields[14].init\_figure(self.figures[14]);

self.fields[15].init\_figure(self.figures[15]);

self.fields[0].init\_figure(self.figures[6]);

self.fields[1].init\_figure(self.figures[4]);

self.fields[2].init\_figure(self.figures[2]);

self.fields[3].init\_figure(self.figures[1]);

self.fields[4].init\_figure(self.figures[0]);

self.fields[5].init\_figure(self.figures[3]);

self.fields[6].init\_figure(self.figures[5]);

self.fields[7].init\_figure(self.figures[7]);

}

self.set\_end\_position = function() {

}

self.set\_position = function(number, color) {

}

self.next\_move = function() {

}

self.prev\_move = function() {

}

return self;

}

};

var board = Chess.createBoard();

// Периодическое обновление позиции

new PeriodicalExecuter(board.refresh, Chess.refresh\_delay);

// Периодическая перерисовка позиции

new PeriodicalExecuter(board.repaint, Chess.repaint\_delay)