ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК
КАФЕДРА ПРИКЛАДНОЙ ИНФОРМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

РАЗРАБОТКА НАБОРА ИНДИКАТОРОВ ПРИ РЕАЛИЗАЦИИ СТРАТЕГИИ ТОРГОВЛИ METATRADER

Выпускная квалификационная работа
обучающегося по направлению подготовки 09.03.03 Прикладная информатика
очной формы обучения, группы 07001404
Григоренко Владислава Евгеньевича

Научный руководитель
к.т.н. Путивцева Н.П.

БЕЛГОРОД 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Автоматизированные торговые системы набирают популярность среди большого количества людей торгующих на биржах со времени появления первых компьютеров. Наиболее популярным методом торговли на бирже через Интернет является так называемый Интернет-трейдинг. Популярность такого подхода к торговле растет с каждым годом, но даже на сегодняшний день до конца не изучены все стороны построения эффективной торговой системы.

С появлением Интернет-трейдинга не только увеличилась скорость осуществления торговых операций. Появились механические торговые системы (автоматические системы, торговые роботы) – программы, предназначенные для полной или частичной автоматизации деятельности трейдеров.

Результаты разработки данного программного обеспечения актуальны, так как из-за больших объемов валютных операция на бирже инвесторам некомфортно обрабатывать настолько большие массивы данных, из-за чего большинство из них не получают максимальную прибыль.

Актуальность данной системы также заключается в том, что невозможно быстро производить необходимые математические вычисления без использования специализированных программных средств, разрабатываемые индикаторы позволяют максимально своевременно выполнять все математические операции без вмешательства человека. Для получения максимальной прибыли в торговой стратегии необходимо использовать индикаторы, которые вычисляются на основе исторических данных торгового инструмента.

Существует ряд инструментов для осуществления торговли. Одним из наиболее популярных является программный продукт компании MetaQuotes, MetaTrader. Данный продукт является свободно распространяемым программным обеспечением. MetaTrader предоставляет возможность осуществлять торговлю различными финансовыми инструментами, а также

обладает широкими аналитическими возможностями. Так же имеет свой встроенный язык программирования, который позволяет разрабатывать скрипты, индикаторы и финансовые советники.

Индикатором называют математическое преобразование цены либо объема финансового инструмента для прогнозирования будущих цен. Индикаторы выполняют основные функции при осуществлении торговли финансовых советников. На основании значений индикаторов принимаются решения относительно купли или продажи финансового инструмента. Индикаторы в свою очередь делятся на два типа: осцилляторы и индикаторы тенденций. Их использование позволяет осуществить прогнозирование финансового актива за счет анализа изменений значений или пересечений между другими индикаторами.

Используемые в системах индикаторы часто ошибочно выдают рекомендации о том, стоит ли покупать или продавать финансовый инструмент. В связи с этим модификация и разработка новых индикаторов являются актуальной задачей.

Объектом исследования данной работы является финансовый рынок.

Предметом исследования является процесс торговли на финансовом рынке.

Целью данной работы является повышение финансовой прибыльности торговой стратегии.

Задачи, которые необходимы выполнить:

- произвести анализ предметной области и выявить недостатки в текущей торговой системе;
- осуществить разработку алгоритмов работы индикаторов;
- осуществить разработку набора индикаторов;
- выполнить модификацию торговой стратегии;
- произвести тестирование индикаторов и торговой стратегии с использованием индикаторов;

- проанализировать экономическую эффективность усовершенствованной торговой стратегии.

Пояснительная записка к выпускной квалификационной работе структурно состоит из: введения, трех разделов, заключения, списка библиографических источников, приложения.

Во введении рассмотрена актуальность работы, выбран предмет исследования и объект исследования, поставлена цель, а также определены задачи и описана структура выпускной квалификационной работы.

В первом разделе рассмотрена технико-экономическая характеристика предметной области, построение модели «как есть», анализ существующих торговых терминалов и выполнена постановка задачи.

Во втором разделе осуществляется обоснование проектных решений, проектирование информационной модели «как должно быть», проектирование технических индикаторов и алгоритма работы торгового советника.

В третьем разделе приведено описание программной реализации с описанием используемых средств, этапы разработки и оценка экономической эффективности усовершенствованной торговой стратегии.

В заключении приведен список выполненных задач при достижении цели, а также вывод по проделанной работе.

В приложениях приведен листинг разработанных индикаторов и листинг торговой стратегии.

Данная работа написана на 80 страницах, содержит 39 рисунков, 4 таблицы, 4 приложения.

1 Аналитическая часть

1.1 Технико-экономическая характеристика предметной области

Предметной областью данной работы является финансовый рынок и все основные понятия, связанные с этой отраслью. Перед описанием характеристики предметной области следует дать основные определения и провести краткий обзор плюсов и минусов использования торгового советника, тем самым изучив предметную область более досконально.

Финансовый рынок — структура, с помощью которой в условиях рыночной экономики создается возможность купли-продажи ценных бумаг, инвестиционных товаров, таких как драгоценные металлы.

Биржа – юридическое лицо, предоставляющая доступ к финансовому рынку, физическим, а также юридическим лицам [6].

Торговая стратегия (торговый советник, финансовый советник) – это совокупность инструментов анализа и правил, которых придерживается трейдер при работе на финансовом рынке [7].

Индикаторы - это математически просчитанное преобразование цен и объемов за определенный промежуток времени, на определенной валютной паре, вместе или раздельно, которое способно прогнозировать дальнейшие поведение рынка [6].

Особенностью электронной торговли является то, что сделки совершаются всегда в одном и том же месте, в строго отведенное время - во время проведения биржевой сессии (или биржевого сеанса) и по четко установленным, обязательным для всех участников правилам. Биржа создает четкую организационную структуру, четкий механизм заключения и исполнения сделок с биржевыми ценностями и высоконадежную систему контроля за ходом исполнения сделок.

Для международного характера товарных бирж необходим большой объем операций с соответствующим товаром, удовлетворяющий потребности всего мирового рынка, кроме того, предоставление свободного валютного, торгового и налогового режимов, что способствует участию в биржевой торговле иностранных участников. Региональные биржи ориентированы на биржевые операции с более узким кругом участников и обслуживают рынки нескольких стран. Национальные биржи действуют в пределах отдельно взятого государства. Они проводят операции, ориентированные на внутренний рынок, при этом часто имеют ограничения в торговом и налоговом режимах [19].

Финансовый советник отслеживает изменения котировок на финансовом рынке, основываясь на критериях, заложенных в нём. Он так же может выступать в роли системы с сигналами, а решение открывать сделку или нет принимает человек. Также советник может выполнять функции по осуществления контроля заявок и мониторинга, анализа торговли и предоставления графиков и отчётов. У торгового советника есть ряд положительных факторов:

- Отсутствие эмоций. Торговый советник работает строго по указанному алгоритму и не допускает отклонения от него, тем самым не боится понести убытки и дождаться системного выхода из позиции. Торговый советник позволит избавиться от несистемных сделок.

- Объективность. При создании финансового советника правила являются абсолютными, так как компьютер не имеет возможности анализировать ситуацию и совершает только те действия, которые точно указаны в алгоритме.

- «Железная» дисциплина. Дисциплина сохраняется даже при волатильной ситуации на финансовом рынке. Инвесторы периодически допускают ошибки из-за страха понести убытки или из-за жадности слишком долго не выходят из позиции для получения более высокой доходности.

Торговый советник исключает данные негативные факторы, поскольку установлены четкие правила.

- Последовательность. Большой проблемой при торговле является выбор между планированием торговли и торговлей по плану. Даже когда трейдер знает, что его система в долгосрочной перспективе приносит прибыль, краткосрочная череда убыточных сделок может заставить его отойти от плана, убивая, таким образом, все ожидания от стратегии. Торговые стратегии обеспечивают четкую последовательность действий и избавляют от возможных ошибок ввода данных при открытии позиции.

- Скорость. Так как компьютер моментально осуществляют вычисления и реагирует на изменения рыночных условий, торговая система без промедления осуществит открытие позиции на одну или даже несколько торговых инструментов. Если осуществлять отклонение на несколько секунд от торгового плана результаты могут сильно измениться.

- Диверсификация. Торговые стратегии позволяют торговать сразу на нескольких счетах или инструментах единовременно. Это позволяет распределить риски. То, что кажется невозможным для человека, торговый советник это выполнит за секунды. Система позволяет единовременно отслеживать несколько рынков и контролировать много позиций.

Недостатки, которые может дать торговый советник. Торговый советник имеет много плюсов, но не исключает минусов. Ниже приведены минусы, которые необходимо знать при использовании торговых советников:

- Недостаток технологичности. Теоретически торговые советники работают просто: устанавливается программное обеспечение, устанавливаются правила и осуществляется торговля. Однако в действительности, это сложный процесс, который полагается на многоуровневые технологии. Зависимо от торговой платформы, позиция может сохраняться на компьютере, а не на сервере. Если связь с интернет утрачена, то позиция не будет отправлена на рынок, что за собой может понести большие убытки.

- Контроль. Торговые стратегии требуют постоянного контроля со стороны пользователя, так как могут произойти технические неполадки (поломка компьютера, проблемы с подключением, сбой электроэнергии, или другие сбои системы). Могут возникнуть внутренние проблемы с советником, непреднамеренная отправка позиции или двойных объемов. При условии контроля системы, такие сбои легко устраняются.

- Переоптимизация. Частой проблемой являются переоптимизация советника. Трейдеры, которые подбирают точные параметры торгового советника по историческим данным после осуществления тестирования. После чего на реальных данных можно понести колоссальные убытки.

Автоматизированные торговые системы или торговые советники с каждым годом собирают все большие объемы на финансовых рынках. Но, прежде чем учиться их создавать, нужно иметь достаточно основательные знания в биржевой торговле и понимание финансовых рынков.

1.2 Обоснование необходимости и цели использования вычислительной техники для решения задачи

На сегодняшний день более 80% всех операций на финансовом рынке выполняется на электронных площадках при использовании специализированных терминалов для осуществления торговли. Около 60% от этого значения приходится на торговых советников. Более половины от этого значения приходится на высокочастотных торговых советников.

Высокочастотный трейдинг - это трейдинг, осуществляемый компьютерами, которые проводят миллионы вычислительных действий в секунду и на основе этих операций самостоятельно выполняют все сделки купли-продажи ценных бумаг. Специалисты по аналитике утверждают, что данный вид трейдинга стабилизирует использование рынков и снижает затраты

на товарооборот. Но у этого способа есть множество минусов. Высокочастотный трейдинг ориентирован на кратковременные, даже секундные операции, приносящие прибыль столь незначительную и работающие на мелком товарообороте, что это не раз приводило к обвалу рынков.

Данный тип торговли создает множество проблем для инвесторов, а главной проблемой является скорость. Инвестор физически не успеет подать заявку на позицию у брокера при помощи звонка или сообщения. За счет чего инвесторы теряют прибыль, а прибыль является основной целью. Из чего можно сделать вывод – для осуществления прибыльной торговли необходимо использовать компьютер со специализированным программным обеспечением.

После улучшения условий для торговли необходимая скорость не достигается, так как расчеты индикаторов приходится осуществлять вручную. Чтобы решить данную проблему, необходимо использовать торговые терминалы, внутри которых находятся различные стандартные индикаторы (скользящее среднее, Bollinger Bands, SAR и т.д.), но если инвестор разработал свой индикатор, расчеты придётся осуществлять либо вручную, либо при помощи программиста реализовать данный индикатор. Сделки же будет осуществляться вручную. Самым подходящим программным продуктом является MetaTrader4.

Для наглядности функционала и процесса осуществления торговли были разработаны информационные модели.

Информационная модель – это модель объекта, отображающаяся в виде информации, которая описывает параметры данного объекта и взаимосвязи между другими объектами. Так же отображает входящею и выходящею информацию, процессы, управляющие объектом и необходимые механизмы для осуществления работы объекта.

Для описания разрабатываемых моделей была использована методология IDEF0. Данная методология отображает иерархическую модель системы диаграмм и описание частей системы.

Изначально производится описание системы и её связи с другими объектами (контекстная диаграмма), далее производится декомпозиция системы – система разбивается на подсистемы, где отображается описание каждой подсистемы. Данный процесс повторяет необходимое количество раз, для отображения необходимой степени подробности.

На рисунке 1.1 представлена контекстная диаграмма, отображающая процесс осуществления торговли. На рисунке видно входящую информацию, механизмы, выходящую информацию и управление. Данная диаграмма предназначена для демонстрации ситуации – «КАК ЕСТЬ».

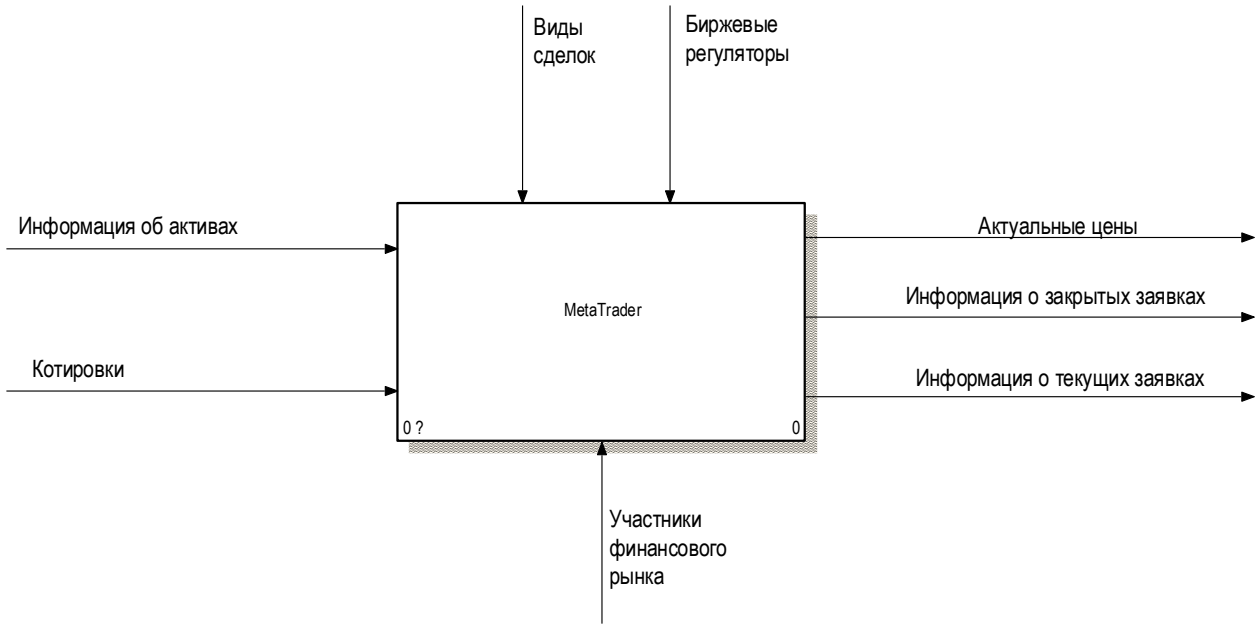


Рисунок 1.1 – Контекстная диаграмма

Как видно на рисунке 1.1, MetaTrader принимает «Котировки» и «Информацию об активах», эта информация поступает от брокера, механизмом являются участники финансового рынка, которые в свою очередь осуществляют там торговую деятельность. Управлением являются правила торговли на финансовых рынках и «Биржевые регуляторы», которые контролируют ход торгов исключая мошенничество. Выходящая информация предоставляется как пользователям, так и биржевым регуляторам, данную

информацию пользователь может получить в различных видах, таких как csv, txt, excel и т.д.

На рисунке 1.2 показана декомпозиция контекстной диаграммы, которая показывает общий процесс прохождения информации в MetaTrader.

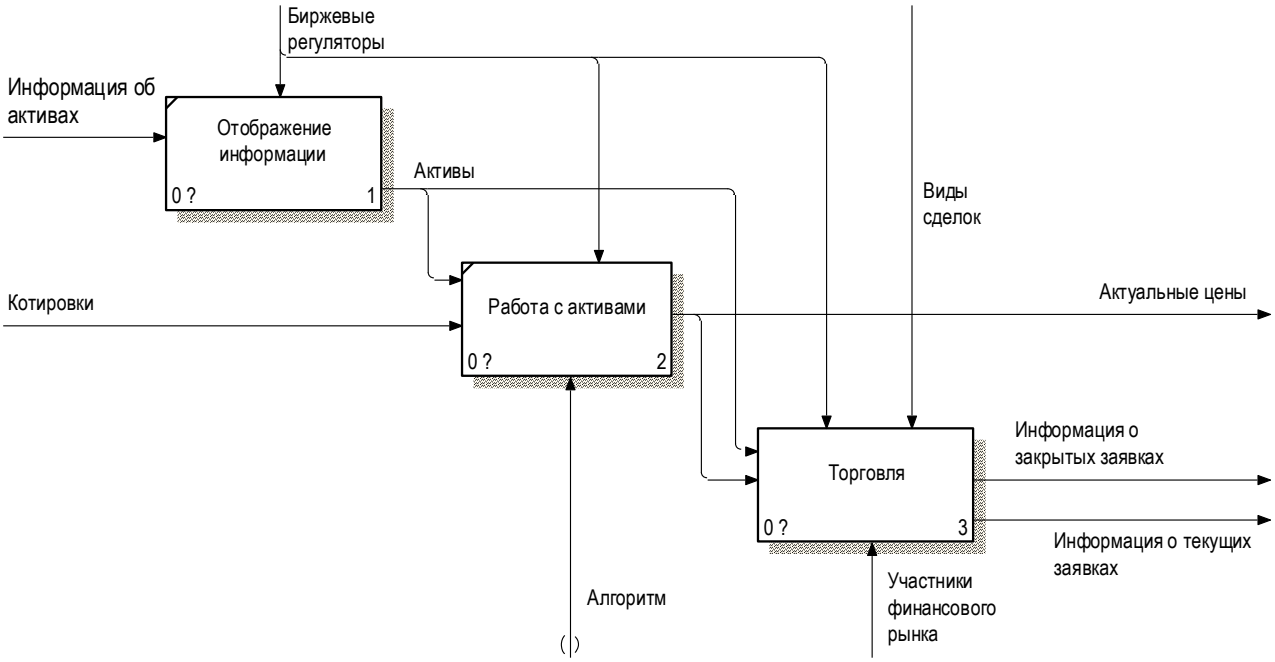


Рисунок 1.2 – Декомпозиция контекстной диаграммы

Процесс отображения информации обрабатывает входящую «Информацию об активах» и выводит её экран, после чего обработанные отправляются в процесс «Работы с активами». В нем в свою очередь алгоритм программного продукта MetaTrader принимает котировки от брокера и проверяет на наличие ошибок и отображает актуальные цены по активу пользователю. Процесс «Торговли» является более сложным, в котором принимает участие пользователи информационной системы. Декомпозиция данного процесса показана на рисунке 1.3.

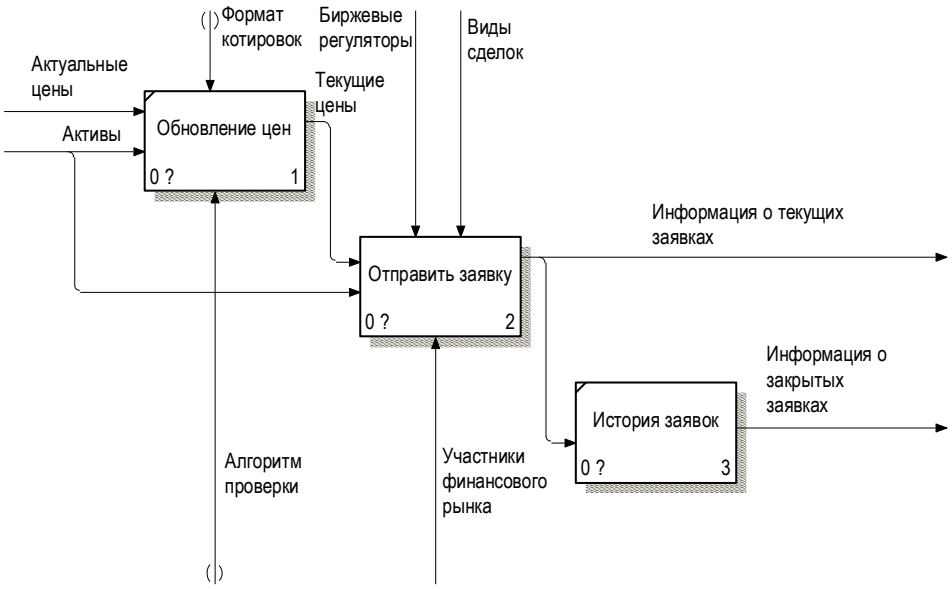


Рисунок 1.3 – Декомпозиция процесса «Торговля»

Для того чтобы не совершались несанкционированные изменения в котировках, MetaTrader осуществляет очередную проверку в процессе обновления цен, который в свою очередь принимает актуальные цены и данные об активах. Туннель «формат котировок» задается в настройках MetaTrader пользователем. После обновления цены данные отправляются в процесс отправить заявку, имеет смысл рассмотреть более подробно его, декомпозиция данного процесса отображена на рисунке 1.4. После совершения сделки, она записывается в историю заявок и отправляется брокеру и пользователю в различных форматах данных.

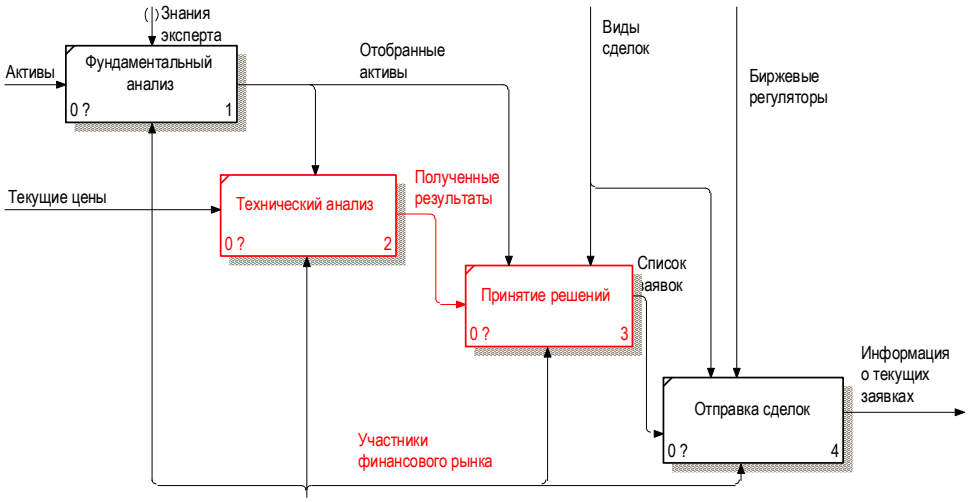


Рисунок 1.4 – Декомпозиция процесса «Отправить заявку»

На этапе, отображённом на рисунке 1.4, происходит процесс осуществления сделки участником финансового рынка. На данном этапе можно заметить, насколько данное решение является не эффективным. В процессе фундаментального анализа инвестор, имея информацию об активах, оценивает актив на возможность дальнейшего роста при помощи различной информации, такой как:

- выплата дивидендов;

- последние новости, связанные с активом;

- финансовые показатели (чистая прибыль, рентабельность, чистая стоимость компании, денежный поток и т.д.).

Данная информация помогает отфильтровать заранее убыточные активы. После чего отфильтрованная информация переходит в технический анализ, в котором на основе текущих и исторических цен, рассчитываются индикаторы. Инвестор на основе своих соображений выбирает индикаторы и устанавливает параметр, после чего проверяет на предельные значения, тем самым в очередной раз отфильтровав активы. Далее результаты переходят в процесс принятия решений, где участник финансового рынка осуществляет выбор, по каким активам открывать позиции, на основе своего баланса и в более приоритетные активы (у которых значение было более точным, как в фундаментальном, так и техническом анализе). После завершения такого трудоемкого и долгого процесса список ордеров отправляется брокеру, если все данные были правильно указаны и соответствует балансу пользователя, сделки будут успешно открыты. В ином случае пользователь на экране увидит ошибку.

Данное графическое отображение процессов показывает, что программное обеспечение более быстро позволяет принимать решение, так как участнику торгового рынка нет необходимости пытаться рассчитать значения индикаторов в голове или на бумаге.

Но данное решение не является самым оптимальным и быстрым, так как можно использовать торговых советников, которые будут действовать по определенному алгоритму, не отходя от торговой стратегии, тем самым

исключая любой человеческий фактор (страх потери, желание получить большего или просто ошибки в расчетах и при отправке заявки). В связи с этим торговый советник при помощи заданного алгоритма будет осуществлять как фундаментальный, так и технический анализ, что исключает возможности ошибки в расчетах. Инвестору просто необходимо указать параметры, по которым торговый советник будет осуществлять отбор активов.

Также торговый советник позволяет инвестору протестировать свою торговую стратегию на исторических данных. Это позволит оценить и отрегулировать любую торговую идеи и определить возможную прибыли или убытка, не допустив тем самым потерь на реальном счете.

В связи с вышеописанным процессом можно сделать вывод, что для прибыльной торговли необходимо использовать прикладные решения в виде терминала, а также торговых советников и индикаторы.

1.3 Анализ существующих торговых терминалов

Так как для реализации торгового советника можно использовать различные инструменты, следует провести анализ имеющихся аналогов.

NinjaTrader – платформа, которая имеет достаточно высокий уровень аналитический и торговых возможностей. Достаточно большой функционал, не исключает высокую эффективность и простоту в работе. Торговая платформа NinjaTrader, имеет все возможности, что и другие платформы, она может создавать торговые стратегии, тестировать их в реальном времени и торговать в автоматическом режиме. У NinjaTrade в арсенале более сотни индикаторов, разные интервалы и высокая визуализация графиков. Она имеет возможность Демо-режима, а главное, у NinjaTrader, все выше перечисленное, дает возможность подстроиться под трейдера, под необходимый стиль торговли. В прочем, как бы ее не обновляли последними ударами прогресса, как бы не

повышали эту платформу степенью разнообразия, стать исключительным лидером среди платформ, NinjaTrader пока не может [31].

Причины этого простые. При всех своих возможностях есть ряд минусов, которые срабатывают как отталкивающий фактор. Отсутствие существенной разницы как таковой между базовой и расширенной версией; Трейдерам рынок не дает расслабиться в полной мере, а тут еще и покупать платформу. Этого не должно быть. Версия должна быть только одна. Только расширенная, с максимальной возможностью адаптации к требованиям трейдера. Тем более что, как правило, платформу предоставляет брокер. А он заинтересован, чтобы трейдер получал доход. Больше доход – больше депозит. Больше депозит - больше лот, и как следствие больше доход брокера. А если брокер, у которого открыт депозит, говорит «купи», вероятно, у него проблемы с финансами. В современное время при наличии легкого доступа к интернету, а, следовательно, и к большим объёмам информации, это просто недопустимо. Существует много простых эффективных программ, которые и расширять-то уж незачем, при этом денег никто не берет. При наличии множества торговых платформ этот фактор явно отрицательно влияет в меньшую сторону использования данной платформы. Пример окна открытия сделки в NinjaTrade изображен на рисунке 1.5.

Рисунок 1.5 – Торговый терминал NinjaTrader

MetaTrader 4 является одной из самых распространенных и популярных платформ для торговли на финансовых рынках, в большей степени валютному рынк. Торговая платформа MT4 является разработкой компании «MetaQuotes Software Corp», официальный выпуск был осуществлен 1 июля 2005 года, с тех пор вышла пятая версия программы, которая до сих пор работает в тестовом режиме. Торговая платформа MT4 имеет множество плюсов, к главным из них можно отнести:

- удобный интерфейс;
- возможность доступа с мобильных устройств;
- управление несколькими счетами;
- встроенные графические и технические инструменты;
- автоматический трейдинг.

Преимуществ у платформы немало, разработчики делают все возможное, чтобы торговля через MT4 была доступна и понятна каждому желающему. MetaTrader 4 является мультиязычной платформой, через неё торгуют несколько миллионов трейдеров по всему миру. Большую доли популярности платформе приносит возможность использования торговых советников – автоматизированных систем. При данном виде торговли всем процессом, начиная от открытия и до закрытия сделки, управляет робот. Практически все форекс – брокеры или дилинговые центры, которые предоставляют услуги по торговле валютой, CFD и фьючерсами используют торговую платформу Metatrader 4, что делает ее, по сути, монополистом в данном секторе услуг [29]. Пример окна торгового терминала изображен на рисунке 1.6.

Рисунок 1.6 – Торговый терминал MetaTrader 4

MetaTrader 5 – торговая платформа, которая позволяет трейдеру участвовать в торгах на биржах, осуществлять покупку фьючерсов и заключать контракты на разницу. Обладатель MetaTrader 5 может торговать одновременно на нескольких биржах и финансовых рынках. Разработчики платформы старались придерживаться концепции «все в одном», то есть постарались вместить в нее все, что только могло понадобиться трейдеру для удачной торговли – технический анализ, система тестирования торговых систем, автоматический трейдинг и многое другое. Вот основные особенности MetaTrader 5 [32].

Проведение аналитических действий относительно будущего движения цен каких-либо финансовых инструментов является важнейшей составляющей успешной деятельности любого трейдера. Комбинация аналитических инструментов может быть самой разнообразной – объекты могут накладываться на индикаторы, одни индикаторы могут строиться от других и тому подобное. Пример окна торгового терминала изображен на рисунке 1.7.

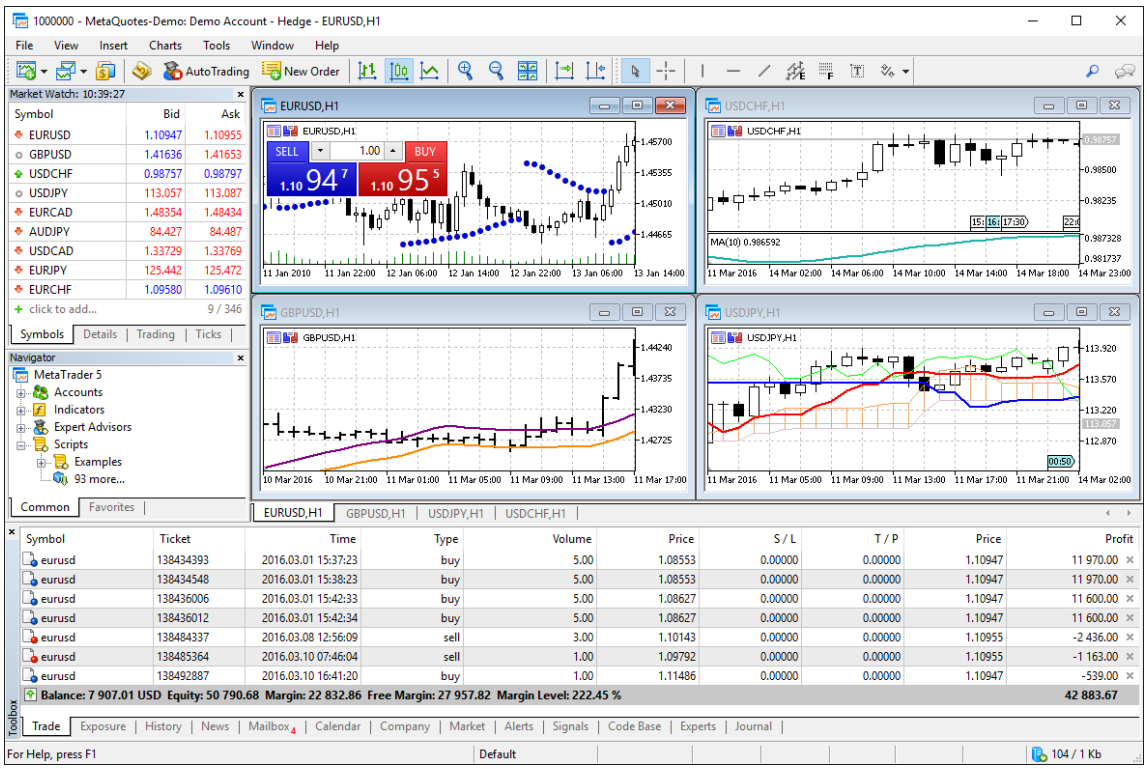


Рисунок 1.7 – Торговый терминал MetaTrader 5

QUIK — это специальная программа для обработки биржевой информации. Для передачи информации между терминалом клиента и сервером QUIK.

Программный комплекс Quik предназначен для организации доступа к биржевым рынкам. Данная программа, как программа интернет- торговли, имеет большую популярность среди трейдеров. Quik, благодаря богатой функциональности, содержит средства, позволяющие реализовывать торговые алгоритмы. Одним из таких является язык Qpile. Qpile – это встроенный скрипт язык торгового терминала Quik. Данный язык обладает небольшим набором возможностей, в отличие от языков высокого уровня, таких как C++ или C#, однако доступных возможностей языка хватает для реализации простых стратегий. Qpile позволяет создавать новые таблицы терминала Quik для расчёта каких-либо значений на основе рыночных данных, что даёт возможность создавать алгоритмы и производить вычисление параметров, изначально отсутствующих в системе.

Основные преимущества торгового терминал QUIK:

- высокая оперативность получения информации и исполнении заявок;

- оптимизированный протокол передачи данных;

- применение стойких средств защиты информации;

- полноценная поддержка торговых операций на основных биржевых площадках;

- развитый функционал работы с заявками;

- встроенный язык QPILE для создания таблиц с расчетными параметрами.

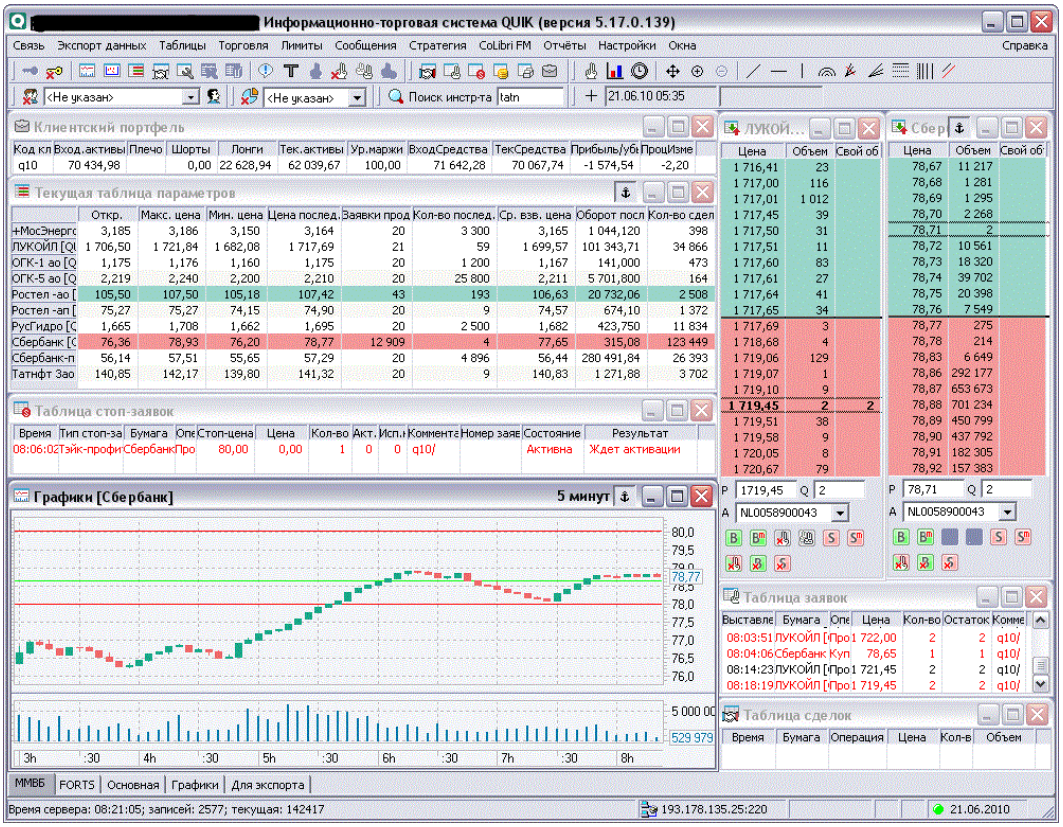


Рисунок 1.8 – Торговый терминал QUIK

Пример окна QUIK изображен на рисунке 1.8. Программный комплекс StockSharp предназначен для автоматизации биржевой торговли. S# состоит из следующих бесплатных программ: S#.API, S#.Designer, S#.Data. Остальные компоненты, облегчающие процесс разработки, такие как примеры алгоритмов, “каркасы” торговых алгоритмов, исходные коды программ S#, также подключение к Plaza II (высокоскоростной шлюз подключения к бирже) можно получить только на платной основе. S#.API является библиотекой для

профессиональной разработки торговых роботов на языке C#. Она позволяет создавать любые по сложности стратегии (как позиционные, так и высокочастотные). Её спецификой является: высокая отказоустойчивость, переносимость (робот может работать с любым подключением), скорость и производительность (высокая скорость исполнения заявок, эмуляция со скоростью более 1 млн событий в сек., поддержка многоядерной и возможность одновременного исполнения множества стратегий по любым инструментам), реалистичное тестирование (тестирование на исторических данных с максимальной точностью) [33]. На рисунке 1.9 изображен торговый терминал S#.

Рисунок 1.9 – Торговый терминал S#

В данном разделе была проанализирована предметная область, также были рассмотрены существующие разработка для расчета плановой себестоимости. Это был документ по расчету плановой себестоимости и обработка. Поскольку разрабатываемая подсистема фактически является документом, то для более детального сравнения был выбран также документ. Он стал основой для составления модели подсистемы и выявления ее недостатков, а также составления новой усовершенствованной модели,

учитывающей недоработки предыдущей. На ее основании были сформированы основные задачи и функции, которые должна выполнять разработанная подсистема.

Были проанализированы изменения, которые будут применены в отделе (подразделении) предприятия при внедрении разработанной подсистемы. В кратком содержании была описана разрабатываемая подсистема, а именно ее будущий интерфейс, состав вкладок и основные действия, которые должны быть доступны для пользователя. Помимо этого, были описаны основные принципы работы подсистемы.

Таким образом, был завершен первый этап написания выпускной квалификационной работы и собраны основные сведения, необходимые для доработки подсистемы расчета плановой себестоимости.

1.4 Постановка задачи

В данный момент существующие подходы не позволять получать высокую доходность, так как пользователь не успевает за быстро меняющимся рынком, также он не успевает принимать решения и при этом рассчитывать значения индикаторов. Поэтому необходимо разработать индикаторы, которые позволят повысить финансовую прибыльность торговой стратегии. Для этого необходимо осуществить разработку алгоритмов работы индикаторов и торговой стратегии, модифицировать торговую стратегию и провести тестирование. После чего осуществить расчет экономической эффективности при помощи различных инвестиционных показателей.

Разрабатываемый набор индикаторов является технологической (формальной) системой, а разрабатываемая торговая система является системой управления.

Технологическая система (формальная) — это совокупность операций для достижения поставленной цели. Структура такой системы определяется набором методов, регламентов, набора правил и норм. Элементами формальной системы являются операции или процессы. Под процессом понимается последовательность операций и их смены.

При разработке индикаторов задается определенная последовательность действий над котировками для получения необходимого значения в определенном диапазоне.

Реализация индикаторов должна обеспечивать следующие функциональные возможности:

- осуществлять своевременные расчеты без сдвигов на временном отрезке котировок;

- осуществлять безошибочные расчеты вне зависимости от количества пунктов в цене;

- предоставлять доступ к расчетам индикатора из торгового советника;

- давать возможность параметризировать любой из параметров.

Система управления рассматривается как действие или функция, обеспечивающая реализацию заданных целей. Система управления содержит два главных элемента:

- управляемую подсистему (объект управления);

- управляющую подсистему (осуществляющая функцию управления).

Управляемой подсистемой является торговый счет участника торгов, а управляющей подсистемой является MetaTrader.

Торговый советник должен обеспечивать реализацию своих функциональных возможностей:

- возможность параметризировать любое предельное значения индикатора для осуществления сделок;

- возможность оптимизации параметров советника и индикаторов на исторических данных;

- осуществление торговых сделок на финансовом рынке;

- бесперебойная работа алгоритма;

- четкое логирование для возможности исправления ошибок и проверки правильности работы алгоритма;

- расчет финансовых показателей при тестировании.

Для реализации всех перечисленных функциональных возможностей необходимо выполнить следующие задачи:

- осуществить реализацию индикаторов;

- осуществить внедрение в торговую стратегию индикаторов;

- провести тестирование работы советника;

- оптимизировать предельные параметры;

- провести анализ работы торгового советника на исторических данных.

После выполнения всех задач и условии положительной доходности, а также всех финансовых показателей, можно приступать к тестированию торговой стратегии на реальном торговом счете. Рекомендуется изначально использовать наименьшие объемы для снижения рисков.

В данном разделе была приведена технико-экономическая характеристика предметной области, после чего описано обоснование необходимости использования вычислительной техники для реализации торговой стратегии и индикаторов, после чего анализ существующих торговых терминалов. В конце раздела была осуществлена постановка задачи. После чего можно приступить к проектной части.

2 Проектная часть

2.1 Обоснование проектных решений

2.1.1 Обоснование проектных решений по программному обеспечению

В настоящее время имеется большое количество информационных систем, которые предоставляют доступ финансовому рынку на бирже, список данных систем указан в пункте 1.3 данной работы. В данном разделе необходимо осуществить выбор более подходящей информационной системы.

Торговый терминал – это программный комплекс, с помощью которого брокерская компания предоставляет трейдеру доступ к рынку. Именно благодаря терминалу и сети Интернет, торговля на валютном рынке является такой простой и доступной.

В данный момент на российском финансовом рынке работают десятки брокеров. В их число входят как банки, которые предлагают своим клиентам услуги предоставления доступа на биржу, так и компании, которые занимаются исключительно брокерской деятельностью. Размеры этих компаний также весьма сильно разнятся, поэтому не все из них могут позволить себе разработку собственных программных решений для своих клиентов. Поэтому доступ предоставляется к внешним информационным системам, бесплатно или платно, в зависимости от необходимых функций.

Перед выбором информационной системы, необходимо произвести анализ и определить основные преимущества каждой из систем и сделать выбор по необходимым признакам.

Признаки, которым должен соответствовать торговый терминал для возможности осуществления всех возможностей торговой стратегии:

- доступность, пользователь должен иметь постоянный доступ к торговому терминалу без какой-либо оплаты;

- популярность, торговый терминал должен быть популярным среди брокеров, чтобы в случае изменения брокера доступ к торговой стратегии оставался;

- информативность, торговый терминал должен быть информативным и иметь справочник на доступном языке;

- программируемость, терминал должен иметь возможность создания торговых советников и индикаторов;

- производительность, терминал должен быть максимально эффективным по отношению к техническим требованиям.

Далее необходимо осуществить оценку по необходимым критериям каждый торговый терминал, оценка отображена в таблице 2.1.

Таблица 2.1 – Оценка информационных систем

	NinjaTrader	MetaTrader 4	MetaTrader 5	QUIK
Доступность	-	+	+	+
Популярность	-	+	-	+
Информативность	-	+	+	-
Программируемость	+	+	+	+
Производительность	+	+	-	+

В таблице 2.1 четко видно преимущество MetaTrader 4, далее необходимо обосновать данный выбор по каждому из параметров оценивания.

- Доступность. NinjaTrader - предоставляет полный доступ ко всем возможностям терминала только после покупки премиум аккаунта. Другие же терминалы являются свободно распространяемыми, но стоит отметить что QUIK в большинстве случаев является платной функцией у брокеров.

- Популярность. NinjaTrader – очень малое количество брокеров предоставляет к данному терминалу, в большей части случаев, это зарубежные брокеры.

MetaTrader 5 – данный терминал вышел не так давно на рынок, поэтому не все брокеры осуществили переход с MetaTrader 4 на следующую версию.

Остальные терминалы являются самыми популярными и более 80% брокеров предоставляют доступ к данным терминалам.

- Информативность. NinjaTrader – существуют версии терминала на русском языке, но справки на русском – нет. При этом терминал не является легком в освоении.

QUIK – справка данного торгового терминала очень объемная и мало информативная. Стоит отметить что интерфейс крайне сложный в освоении, чем отталкивает большинство пользователей.

MetaTrader 4 и 5 – являются очень популярными среди пользователей благодаря своей информативной справки и эргономичному интерфейсу [29].

- Программируемость. Каждый терминал имеет возможность в создании торговых советников, но также каждый терминал имеет свой язык программирования.

QUIK – имеет самый сложный в освоении язык программирования, так как он не является распространенным.

- Производительность. Так как MetaTrader 5 выпущен совсем недавно, в торговом терминале могут происходить сбои. Также данный терминал является очень “громоздким” из-за добавления функции отображения стакана торгов, обновления происходят слишком часто, тем самым нагружая компьютер.

Поле проведенного анализа торговых терминалов можно заметить лидера в виде MetaTrader 4. Данный терминал идеально подходит под требуемые условия.

2.1.2 Обоснование проектных решений по информационному обеспечению

Для четкого понимания протекающих процессов в MetaTrader 4 и успешной реализации торгового советника необходимо смоделировать базу данных.

Информационное обеспечение представляет собой совокупность единой системы классификации информации, схем информационных потоков, используемых в организации, а также методология построения баз данных [1].

Одно из важнейших составляющих требований к информационному обеспечению является безопасность и доступ к базам данным. MetaTrader 4, предоставляет доступ ко всем объектам базы данных, которые хранятся на финансовом рынке. К ним доступ можно получить при помощи торговых советников, индикаторов и скриптов.

Логическая модель описывает понятия предметной области, их взаимосвязь, а также ограничения на данные, налагаемые предметной областью. Логическая модель данных является начальным прототипом будущей базы данных. Она строится в терминах информационных единиц, но без привязки к конкретной СУБД. Более того, логическая модель данных необязательно должна быть выражена средствами именно реляционной модели данных. Основным средством разработки логической модели данных в настоящий момент являются различные варианты ER-диаграмм (Entity-Relationship, диаграммы сущность-связь).

Для выделения основных процессов на финансовом рынке, была построена логическая диаграмма «сущность-связь», представленная на рисунке 2.1.

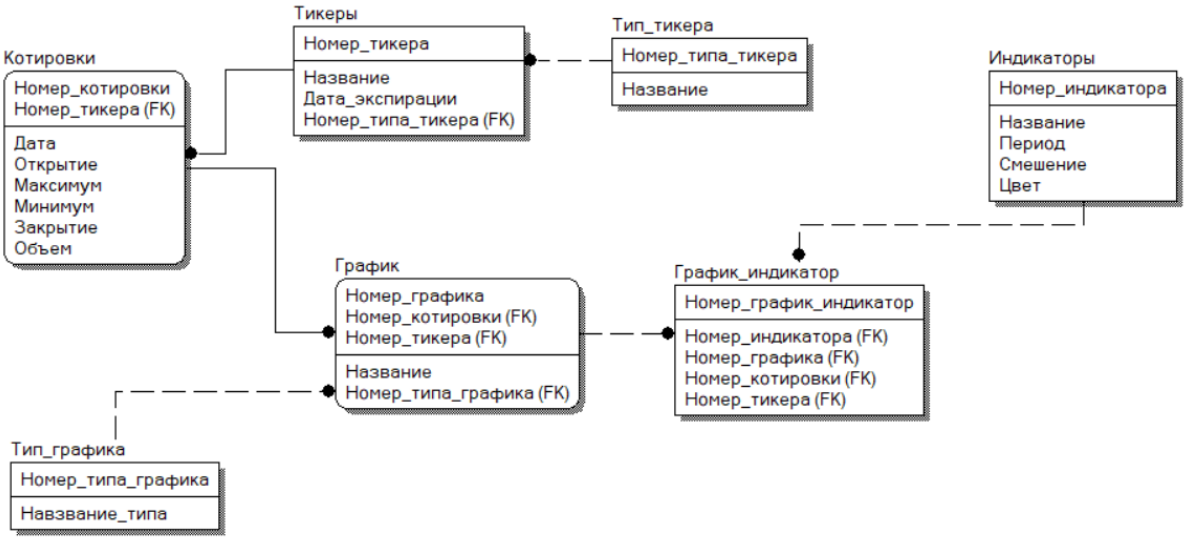


Рисунок 2.1 – Логическая модель базы данных

ER-диаграмма отражает самые главные для выбранной предметной области информационные объекты, которые на диаграмме называются сущностями, их свойства – атрибуты и связи между данными объектами [1].

В данной работе нет необходимости проектировать физическую модель базы данных, так как такое проектирование используется при разработке, что не является основной целью. А целью является изучение протекающих процессов внутри MetaTrader.

На рисунке 2.1 изображенны основные сущности:

- тикеры - это краткое название в биржевой информации котируемых инструментов;

- котировки – это цена единицы торгового инстурмента;

- график - отображает котировки пользователю, для более удобного восприятия информации;

- индикаторы – это инструмент для анализа котировок, на основе прошлых цен.

Все остальные сущности созданы для решения проблемы проектирования баз данных много-ко-многим.

2.1.3 Обоснование проектных решений по техническому обеспечению

Для эффективной работы торговых терминалов, необходимо техническую часть оснастить минимальными требованиями для торгового терминала MetaTrader 4.

Техническое обеспечение представляет собой совокупность технических средств, компьютерной техники, средств передачи информации, используемых в автоматизированных системах.

После изучения минимальных требований, было выявлено следующее. Для оптимальной работы необходимы следующие технические устройства, предоставленные в таблице 2.2.

Таблица 2.2 – Технические требования

Системный блок	
Материнская плата	GIGABYTE H370
Процессор	Intel Core i3-8300
Жесткий диск	SATA II, 500 ГБ, 7200rpm
Оперативная память	8 Гб DDR4
Сеть	10/100Мбит, Atheros AR8152
Корпус	CROWN mATX CMC-403 Black
Видеокарта	Intel UHD Graphics 630
Блок питания	450 ВТ
Устройства ввода	
Клавиатура	OKLICK 350M
Мышь оптическая	OKLICK 725G DRAGON USB
Монитор	
Название	ACER
Диагональ	18.5”
Разрешение экрана	1366x768

В таблице 2.2 описаны минимальные технические требования к ЭВМ. Любую из характеристик можно улучшить для более комфортного и быстрого использования торгового терминала.

2.2 Информационная модель и её описание

Задача выпускной квалификационной работы заключается в разработке набора индикаторов для торговой стратегии. Данная разработка поможет увеличить прибыльность торговой стратегии за счет более быстрого, точного и гибкого использования возможностей торгового терминала. После демонстрации модели «Как есть» был выявлен ряд недостатков этой модели. После этого была построена модель «Как должно быть», в которой были устранены недостатки прошлой версии.

Для моделирования диаграмм используется методология IDEF0. Имеет смысл на диаграммах отобразить изменения после прошлого анализа в первом разделе. Контекстная диаграмма изображена на рисунке 2.2.

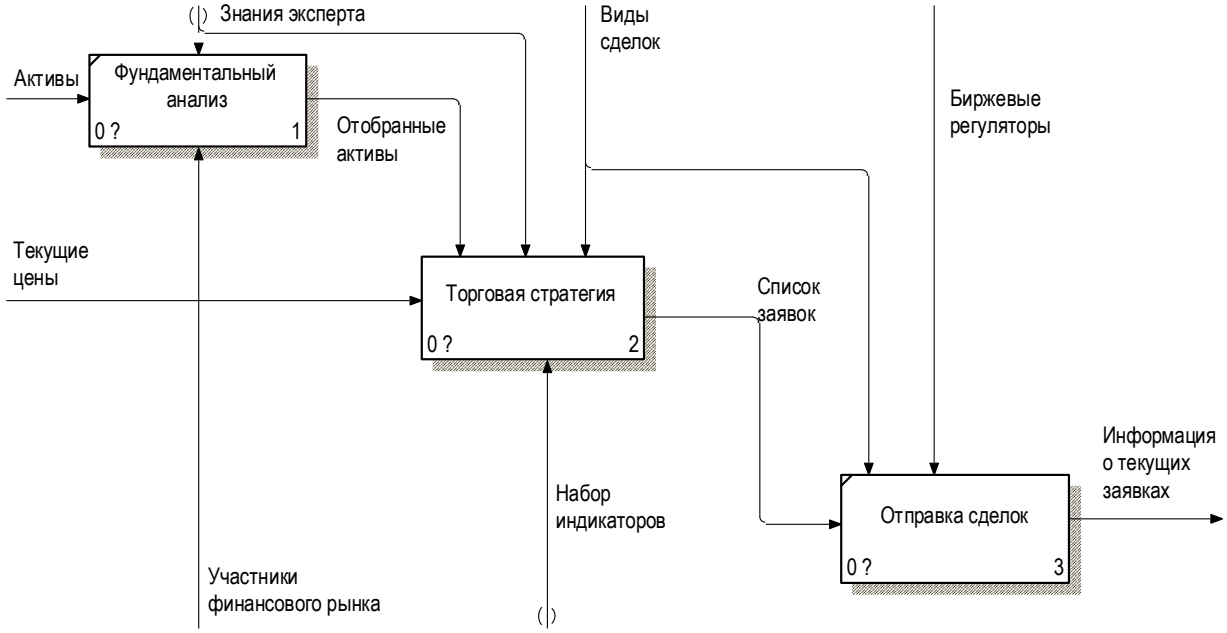


Рисунок 2.2 – Декомпозиция процесса «Отправить заявку»

На рисунке 2.2. можно заметить, что исчезло 2 процесса «Технический анализ» и «Принятие решений». Видно, что потоки данных почти не были затронуты, но исчезнувшие процессы были автоматизированы. Данные вычисления будут происходить в торговой стратегии, пользователю больше нет необходимости осуществлять все расчеты вручную.

Далее следует отобразить декомпозицию процесса «Торговая стратегия», данная декомпозиция отображена на рисунке 2.3

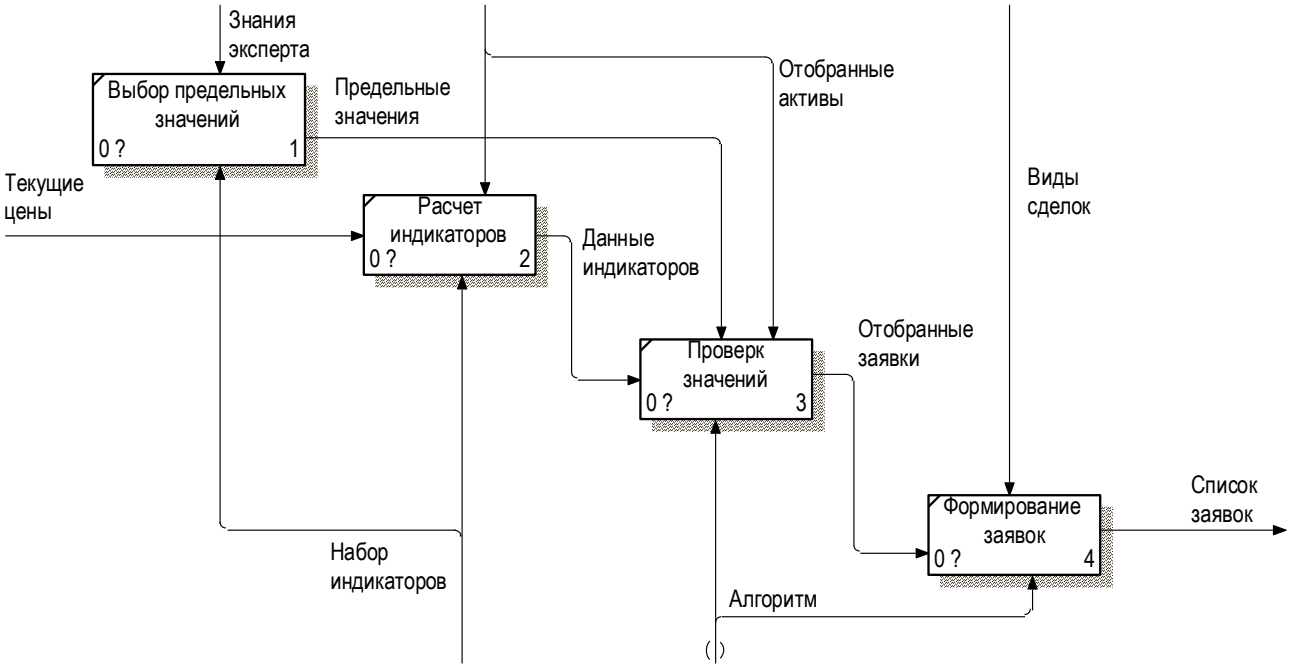


Рисунок 2.3 – Декомпозиция процесса «Торговая стратегия»

На рисунке 2.3 изображена декомпозиция, которая отражает основной принцип работы торговой стратегии. Следует дать описания работы данных процессов. После фундаментального анализа пользователем, в данный процесс приходят, только те активы, с которыми пользователь хочет торговать. Далее эксперт выбирает придельные значения, при которых торговый советник будет осуществлять торговлю. Дальше программа рассчитывает индикаторы и проверяет с предельными значениями, которые указал пользователь. Если советник замечает изменения в заданных параметрах, формируется список заявок, которые дальше отправляются на рынок.

2.3 Проектирование торгового советника

2.3.1 Проектирование технических индикаторов

Следующим этапом после изучения торговых терминалов и проектирования моделей является разработка алгоритма работы торгового советника.

За основу в торговом советнике берется индикатор AO (Awesome Oscillator) – этот индикатор помогает определить текущее поведение движущей силы рынка путем вычитания значений медленного скользящего среднего (обычно 34 периода) от типичной цены из быстрого скользящего среднего (обычно 5 периодов) от типичной цены. Типичная цена (Typical Price) представляет собой частное от деления суммы максимальной цены, минимальной цены и цены закрытия на три.

Методика расчета индикатора AO очень проста – достаточно вычислить разность экспоненциальных средних (EMA) от типичной цены (TP).

$$\mathrm{TP} = \frac{\mathrm{High} + \mathrm{Low} + \mathrm{Close}}{3}, \tag{2.1}$$

где high – максимальная цена;

low – минимальная цена;

close – цена закрытия.

$$\mathrm{AO} = \mathrm{EMA}\,(\mathrm{TP}, \mathrm{x}) - \mathrm{EMA}\,(\mathrm{TP}, \mathrm{y}), \tag{2.2}$$

где x и y – периоды индикаторов.

Следующим основополагающим индикатором являются линии Фибоначчи, которые помогают с легкостью найти уровни поддержки и сопротивления.

Линии Фибоначчи (LF) - строятся следующим образом: сначала между двумя экстремальными точками проводится линия тренда — например, от впадины до противостоящего пика. Затем проводятся девять горизонтальных линий, пересекающих линию тренда на уровнях Фибоначчи 0,0%, 23,6%, 38,2%, 50%, 61,8%, 100%, 161,8%, 261,8% и 423,6%. После сильного подъема или спада цены часто возвращаются назад, корректируя значительную долю (а иногда и полностью) своего первоначального движения. В ходе такого возвратного движения цены часто встречают поддержку/сопротивление на уровнях линий Фибоначчи или вблизи них.

$$LF = (\text{High} - \text{Low}) * \text{percent} + \text{Low}, \quad (2.3)$$

где high – максимальная цена;

low – минимальная цена;

percent – процент Фибоначчи.

2.3.2 Проектирование алгоритма работы торгового советника

После изучения информации о используемых индикаторов в торговом советнике, необходимо спроектировать алгоритм работы в котором за основу взяты перечисленные алгоритмы. Алгоритм работы торгового советника изображен на рисунке 2.4.

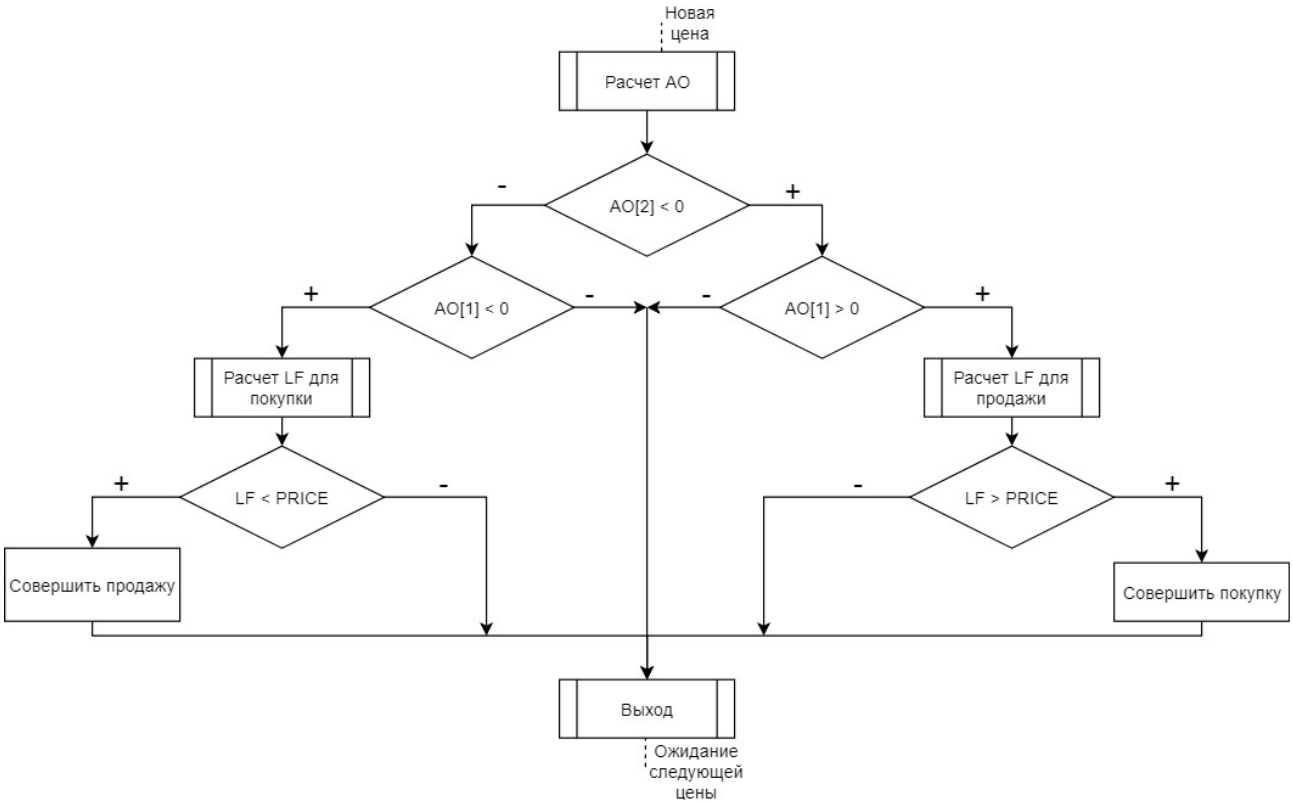


Рисунок 2.4 – Алгоритм работы торгового советника

На рисунке 2.4 изображен алгоритм работы торгового советника, который показывает поведение при поступлении новой цены. Можно заметить, что проверяется вторая свеча (элемент графика для отображения биржевых котировок за определенный период времени). Если АО преодолел значение 0 в одну из сторон (отрицательную или положительную), то проверяется, изменилось ли это движение на более поздней свече, если изменилось, то начинаются расчеты линий Фибоначчи, после которых совершается проверка на пробитие цены. В случае если цена была пробита – откроется сделка продажи или покупки.

Во втором разделе выпускной квалификационной работы, были обоснованы проектные решения по техническому, программному и информационному обеспечению. Был четко определен торговый терминал для осуществления торговли. Также определены минимальные технические требования по оборудованию. Построена логическая модель базы данных. Смоделирована диаграмма «Как должно быть».

3 Программная реализация

3.1 Разработка алгоритма работы торгового советника

3.1.1 Разработка алгоритма в MetaTrader

После построения модели следует приступить к разработке торгового советника внутри торгового терминала MetaTrader. Для начала необходимо создать нового эксперта, для этого необходимо открыть MetaEditor, который встроен в терминал. Его открыть можно при помощи кнопки клавиатуры F4 или нажать на соответствующую кнопку в верхнем меню терминала, которая изображена на рисунке 3.1.

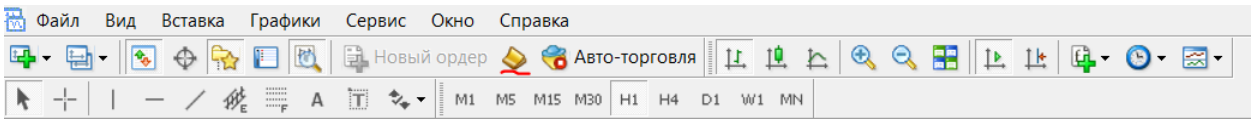


Рисунок 3.1 – Верхнее меню MetaTrader

На рисунке 3.1 подчеркнута красным маркером необходимая для открытия кнопка. После этого откроется окно MetaEditor, данное окно отображено на рисунке 3.2.

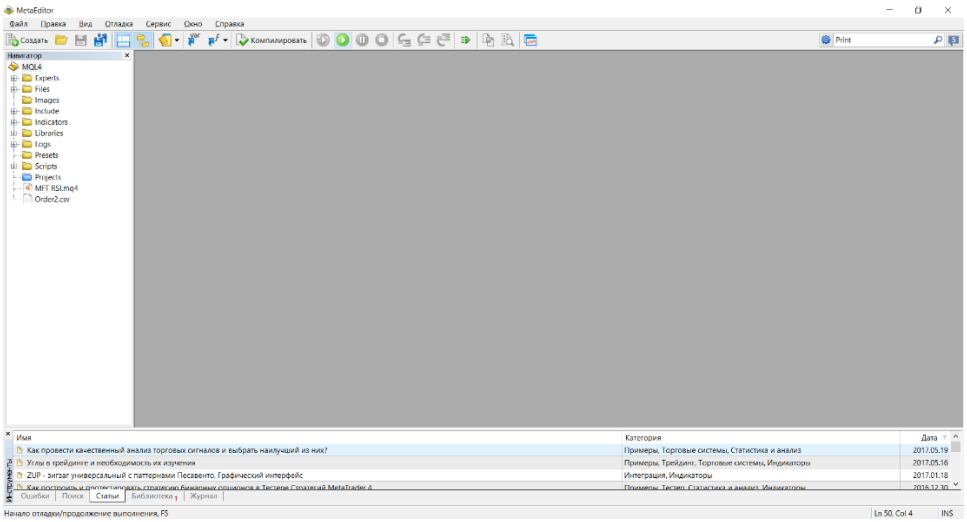


Рисунок 3.2 – Окно MetaEditor

На рисунке 3.2 видно окно редактора программного кода. Далее необходимо создать торгового советника, для этого необходимо нажать на кнопку «Создать», после чего выбрать пункт «Советник (шаблон)», окно выбора изображено на рисунке 3.3.

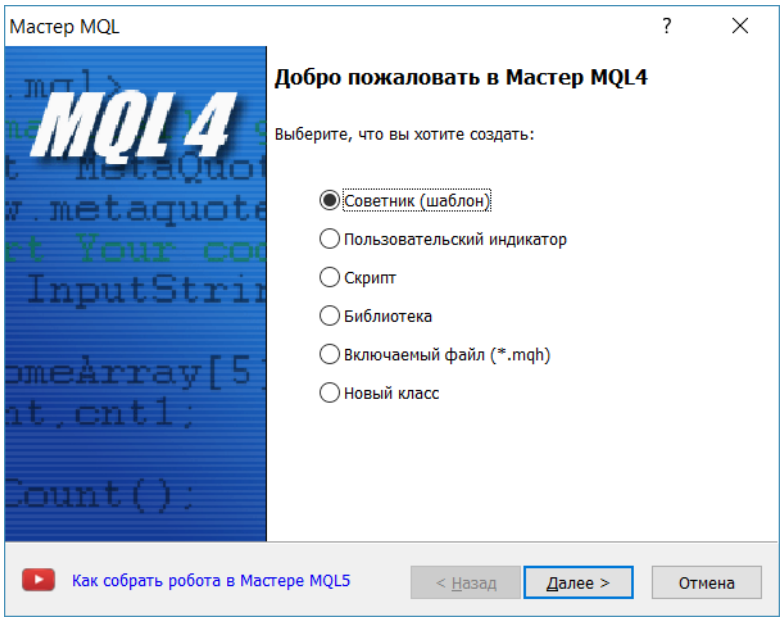


Рисунок 3.3 – Окно выбора создания советника

На рисунке 3.3 видно, что также есть возможность создать индикатор, скрипт, библиотеку и другое. После выбора советника необходимо заполнить данными окно запроса данных, изображенное на рисунке 3.4.

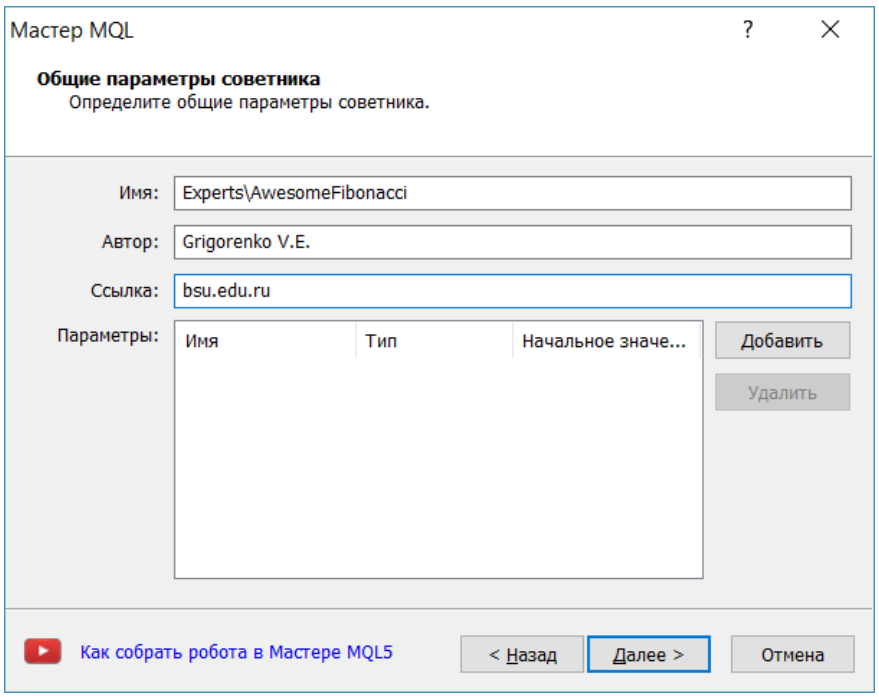


Рисунок 3.4 – Окно для заполнения данных

На рисунке 3.4 представлено окно, в котором необходимо заполнить данные о торговом советнике. После заполнения всех пунктов откроется окно для написания программного кода со стандартным шаблоном советника, результат изображен на рисунке 3.5.

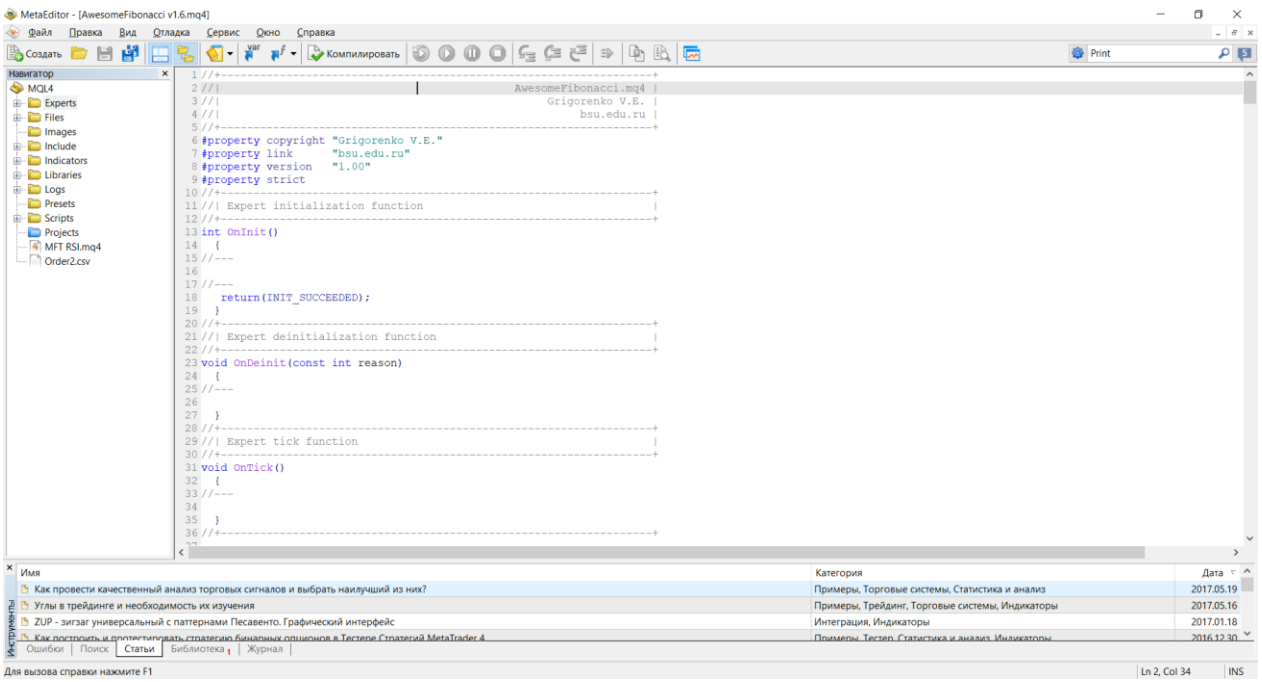


Рисунок 3.5 – Окно редактора для написания программного кода

На рисунке 3.5 изображено окно редактирования, на котором видны основные функции. Для дальнейшего написания советника необходимо рассмотреть шаблон.

Необходимо проанализировать следующие функции:

- OnInit, данная функция вызывается при первом запуске торгового советника на торговом счёте;

- OnDeinit, функция, которая вызывается при прекращении торговли при помощи советника;

- OnTick, вход в эту функцию выполняется MetaTrader при получении нового временного отрезка.

После создания шаблона торгового советника можно приступать к программированию индикаторов.

Так как АО был создан как внешний индикатор, листинг данного индикатора указан в приложении А.

Функция расчета линий Фибоначчи отображена на рисунке 3.6.

```
void CalculationFibonacci(int ticket, double up_level, double down_level)
{
   if(ticket != -1){
      DrowFibo(up_level,down_level);
      int array_size = ArraySize(orders_info);
      ArrayResize(orders_info, array_size + 1);
      orders_info[array_size].fibonacci_level[0].level = 1;
      orders_info[array_size].fibonacci_level[1].level = 0;
      orders_info[array_size].fibonacci_level[2].level = 0.236;
      orders_info[array_size].fibonacci_level[3].level = 0.382;
      orders_info[array_size].fibonacci_level[4].level = 0.50;
      orders_info[array_size].fibonacci_level[5].level = 0.618;
      orders_info[array_size].fibonacci_level[6].level = 1.618;
      orders_info[array_size].fibonacci_level[7].level = 2.618;
      orders_info[array_size].fibonacci_level[8].level = 4.236;
      orders_info[array_size].level_number = 6;
      orders_info[array_size].ticket = ticket;
      orders_info[array_size].fibonacci_level[0].price = up_level;
      orders_info[array_size].fibonacci_level[1].price = down_level;
      for(int i = 2; i < 9; i++){
         orders_info[array_size].fibonacci_level[i].price = (up_level - down_level)*orders_info[array_size].fibonacci_level[i].level + down_level;
         orders_info[array_size].fibonacci_level[i].price = NormalizeDouble(orders_info[array_size].fibonacci_level[i].price, Digits);
      }
   }
}
```

Рисунок 3.6 – Листинг расчета линий Фибоначчи

На рисунке 3.6 представлена реализация алгоритма. Можно заметить, что функция принимает номер сделки, а также High и Low цену, после чего осуществляется сохранение данных в структуру ордеров, и в цикле рассчитываются все уровни и записываются в структуру.

3.1.2 Тестирование алгоритма

Далее необходимо осуществить тестирование разработанной торговой системы. Для того этого необходимо зайти в торговый терминал MetaTrader и нажать горячие клавиши CTRL+R или в верхнем меню выбрать «Тестер стратегий», изображение данного меню подчеркнуто красной линей на изображении 3.7.

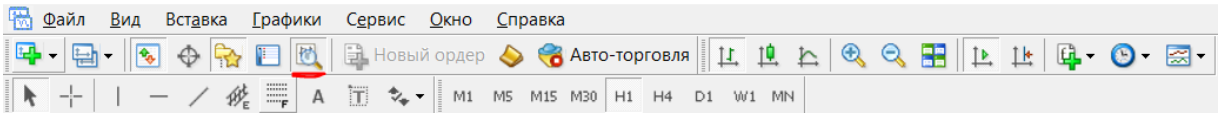


Рисунок 3.7 – Меню с тестером стратегий

На рисунке 3.7 показана кнопка с тестером стратегий, после нажатия на данную кнопку появляется специальное окно для тестирования, которое изображено на рисунке 3.8.

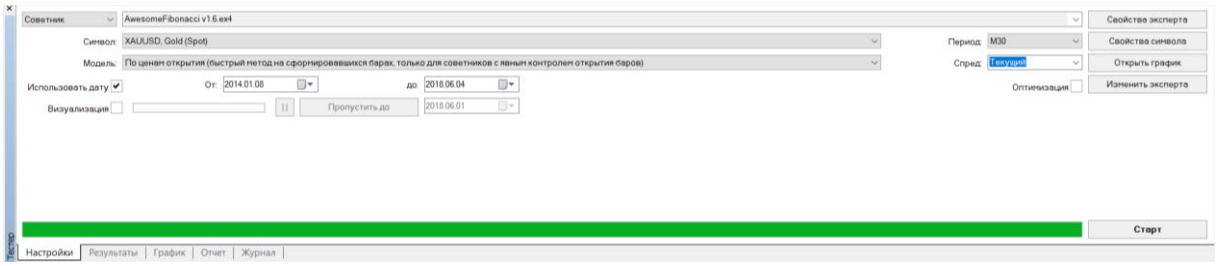


Рисунок 3.8 – Окно тестирования

На рисунке 3.8 представлено тестирование, необходимо дать краткий обзор данного окна. Слева в выпадающем окне можно выбрать, что тестировать – индикатор или советника. Далее необходимо выбрать название тестируемого советника. В пункте «Символ», необходимо выбрать торговый инструмент. В списке «Модуль» необходимо выбрать способ расчета цены для тестирования. Далее необходимо выбрать промежуток времени для тестирования торговой стратегии. Визуализация необходима для отображения сделок в «реальном

времени». В меню период выбирается, с какой частотой будут приходить свечи. Далее можно выбрать «Спред», данный пункт позволяет выбрать разницу между ценой покупки и продажи. Установив галочку в пункте «Оптимизация» при запуске будет рассчитываться указанные параметры в «Свойствах эксперта». На рисунке 3.9 отображено «Свойства эксперта».

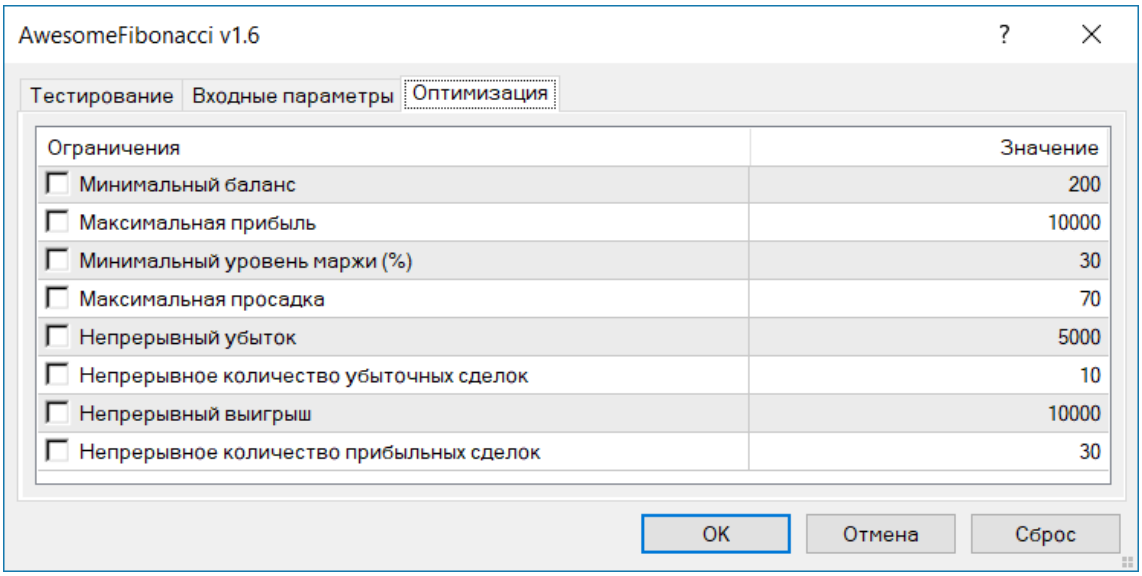


Рисунок 3.9 – Окно «Свойства эксперта»

На рисунке 3.9 видны три вкладки, на вкладке «Тестирование» можно указать валюту тестирования, начальный баланс. На вкладке «Входные параметры» можно редактировать параметры эксперта. На вкладке «Оптимизация» указываются различные пороговые значения, на которые будет реагировать оптимизатор.

Далее на рисунке 3.8 видна кнопка «Свойства символа», при нажатии которой появляется окно с информацией о финансовом инструменте. При нажатии кнопки «Открыть график» в главном окне MetaTrader появляется график со сделками, полученными в результате тестирования. Последней кнопкой является «Изменить эксперта», после нажатия кнопки открывается редактор с запущенным экспертом, в котором можно внести изменения в торгового советника.

После изучения всех функций можно перейти к тестированию. Для этого необходимо нажать кнопку «Старт». Снизу можно увидеть 4 вкладки, ниже рассмотрим их:

- результаты, отображает все исполненные сделки, их цены и причины закрытия;

- график, отображает все сделки на графике, у которого по оси x – количество сделок, а по оси y – баланс тестируемого счета;

- отчёт, отображает информацию о статистических результатах тестирования;

- журнал, в нём ведется логирование всех действий в процессе тестирования.

Теперь можно совершить анализ тестирования. График тестирования изображен на рисунке 3.10.

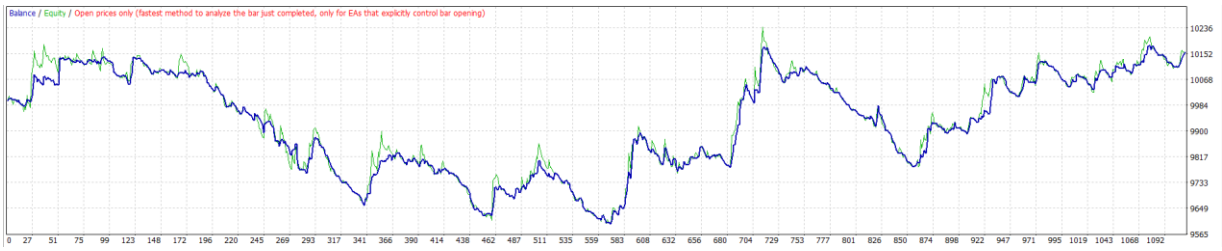


Рисунок 3.10 – График тестирования

На графике тестирования четко видно просадку баланса, так как много сделок совершенно с убытком. Несмотря на дальнейшие шаги по повышению прибыли, стабильность не будет чувствоваться, тем самым такая стратегия является не пригодной.

3.1.3 Выявление недостатков

Далее необходимо посмотреть статистику сделок, чтобы определить недостатки торговой стратегии. Для этого после тестирования необходимо перейти на вкладку «Отчёт». Статистика отображена на рисунке 3.11.

Initial deposit	10000.00			Spread	Current (518)
Total net profit	158.39	Gross profit	3709.65	Gross loss	-3551.26
Profit factor	1.04	Expected payoff	0.14		
Absolute drawdown	405.07	Maximal drawdown	604.72 (5.93%)	Relative drawdown	5.93% (604.72)
Total trades	1110	Short positions (won %)	0 (0.00%)	Long positions (won %)	1110 (48.29%)
		Profit trades (% of total)	536 (48.29%)	Loss trades (% of total)	574 (51.71%)
	Largest	profit trade	88.56	loss trade	-52.12
	Average	profit trade	6.92	loss trade	-6.19
	Maximum	consecutive wins (profit in money)	15 (271.01)	consecutive losses (loss in money)	12 (-86.86)
	Maximal	consecutive profit (count of wins)	271.01 (15)	consecutive loss (count of losses)	-86.86 (12)
	Average	consecutive wins	2	consecutive losses	3

Рисунок 3.11 – Статистика выполненных сделок

Статистика также подтверждает убыточность торговой стратегии, об этом говорят показатели «Total net profit (чистая прибыль)», «Profit factor (фактор прибыли)», «Profit trades (прибыльные сделки)». По данным показателям можно проанализировать торговую стратегию на прибыльность, при тестировании можно определить, что стратегия является непригодной.

Также в приложении Б были разработаны дополнительные показатели, которые позволяют определить прибыльность торговой стратегии. Были разработаны следующие показатели.

- Показатель Шарпа. Эффективность инвестиций часто оценивают с точки зрения дисперсии доходов. Одним из таких показателей является коэффициент Шарпа (Sharpe Ratio). Этот коэффициент показывает, как соотносятся среднее арифметическое (AHPR), уменьшенное на безрисковую ставку, и стандартное отклонение (SD) от ряда HPR. Значение без рисковой ставки RFR (Risk Free Rate) обычно принимают равным процентной ставке по доходу на депозит в банке или ставке дохода на казначейские обязательства [2].

$$SR = \frac{\text{AHPR} - (1 + \text{RFR})}{\text{SD}}, \tag{3.1}$$

где AHPR – средняя арифметическая прибыль за время удержания позиции;

RFR – без рисковая ставка;

SD – стандартное отклонение.

- Линейная регрессия и коэффициент линейной корреляции;

Можно и по-другому оценить стабильность торговых результатов. Показатель Шарпа позволяет оценить меру риска, которому подвергается торговый капитал, но можно попробовать оценить и степень гладкости кривой баланса. Если мы нанесем на график значения баланса по закрытии каждой сделки, то сможем провести ломаную линию. Можно аппроксимировать эти точки некоторой прямой линией, которая покажет нам среднее направление изменения торгового капитала [25].

Необходимо найти такие коэффициенты a и b, чтобы прямая линия проходила как можно ближе к аппроксимируемым точкам. Где x - это номер сделки, y - значение баланса по закрытии сделки.

Обычно коэффициенты аппроксимирующей прямой находят по методу наименьших квадратов (МНК). Пусть у нас есть такая прямая с известными коэффициентами, a и b. Для начала для каждой точки необходимо найти прямую по МНК и отнимать значения балансов, который вычисляется по формуле (3.5).

$$d(x) = (\mathrm{a} * \mathrm{x} + \mathrm{b}) - \mathrm{balance}(\mathrm{x}) , \quad (3.2)$$

где a и b – коэффициенты аппроксимирующей прямой;

x – индекс позиции в векторе.

Далее при помощи полученного значения можно получить сумму квадратов отклонений, которая вычисляется по формуле (3.6).

$$SD = \mathrm{d}(\mathrm{n})^2, \quad (3.3)$$

Нахождение прямой по методу наименьших квадратов означает поиск таких коэффициентов a и b, чтобы SD была минимальна. Эту прямую также называют линейной регрессией.

- Z-счет. Само предположение, что прибыль, полученная в результате серии торговых операций, является случайной, для многих трейдеров может звучать издевательски. Проведя достаточно долгое время в поисках своей торговой системы, которая на практике уже дала реальную прибыль на достаточно ограниченном промежутке времени, трейдер получает подтверждение правильности найденного подхода к рынку. И теперь допустить, что все это было случайностью? Это уже слишком, особенно для новичков. Тем не менее, необходимость объективной оценки результатов торговли очень существенна. В этом случае на помощь снова приходит нормальное распределение.

Трейдеру неизвестно, каков будет результат каждой отдельной сделки. Только можно отметить, что-либо сделка осуществляется в прибыль (+), либо в убыток (-). Чередование убытков и прибылей может происходить по-разному для каждой торговой системы. Например, если размер предполагаемой прибыли в 5 раз меньше размера предполагаемого убытка при срабатывании Stop Loss, то разумно предполагать, что прибыльных сделок (со знаком +) будет существенно больше, чем убыточных (со знаком минус). Z-счет позволяет оценить, насколько часто прибыльные сделки сменяются убыточными.

Z-счет для торговой системы вычисляется по формуле (3.4), но для этого изначально необходимо осуществить ещё один расчет, отображенный в формуле (3.4).

$$P = 2 * W * L, \tag{3.4}$$

где W- общее число прибыльных сделок;

L – общее число убыточных сделок.

После чего необходимо сделать расчет по формуле (3.5).

$$Z = \frac{N*(R-0.5)-P}{\left(\frac{P*(P-N)}{N-1}\right)^{\frac{1}{2}}}, \quad (3.5)$$

где N – общее число сделок;

R – общее число серий прибыльных сделок.

- Сортино. Показатель, позволяющий оценить доходность и риск инвестиционного инструмента, портфеля или стратегии. Коэффициент Сортино рассчитывается аналогично коэффициенту Шарпа, однако вместо волатильности портфеля используется так называемая «волатильность вниз». Основное отличие: вместо стандартного отклонения за меру риска берется отрицательное отклонение. Благодаря этому данный показатель дает инвестору более реалистичное представление о поведении управляющего во время просадки.

- Фактор восстановления. Этот показатель отображает, какой суммой рискует инвестор ради получения прибыли. Для его вычисления оценивается соотношение полученной прибыли к наибольшей просадке. Многие эксперты считают, что у эффективной торговой системы фактор восстановления должен быть не менее 3. Однако вычисление должно быть обязательно привязано к периоду торговли [3].

Для расчета фактора восстановления (Recovery Factor) необходимо осуществить расчеты по формуле (3.6).

$$RF = \frac{P}{D}, \quad (3.6)$$

где P – конечная прибыль торговой стратегии;

D – максимальная просадка с начала торговли.

После анализа всех имеющихся коэффициентов необходимо проанализировать результаты текущей торговой стратегии. На рисунке 3.12 изображены результаты торговой стратегии.

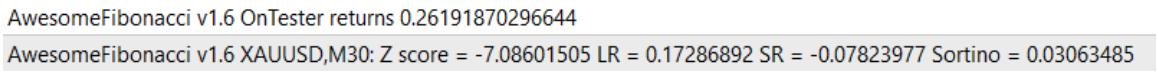


Рисунок 3.12 – Результаты торговой стратегии

На рисунке 3.12 видны все коэффициенты, а именно:

- Фактор восстановления = 0.26, приемлемым результатом является RF> 3;

- Z-счет = -7.09, приемлемым результатом является от -3 до 3;

- Линейная регрессия = 0.17, приемлемым результатом является LR> 0.98;

- Шарп = -0.08, приемлемым результатом является SR>0;

- Сортино = 0.04, приемлемым результатом является S>0.

По результатам можно определить, что все показатели кроме Сортино находятся в крайне отрицательной ситуации, но стоит отметить, что Сортино подходит лишь по минимальным требованиям. В связи с этим принято решение осуществить разработку дополнительных индикаторов, которые позволят улучить торговую стратегию.

3.2 Разработка алгоритмов работы набора индикаторов

Для начала построения моделей алгоритмов необходимо определить основной набор индикаторов, проанализировать их, после чего составить алгоритмы расчета.

В данный набор входят следующие индикаторы:

- Stochastic. Технический индикатор Стохастический Осциллятор (Stochastic Oscillator) сопоставляет текущую цену закрытия с диапазоном цен за выбранный период времени. Индикатор представлен двумя линиями. Главная линия называется %K. Вторая линия %D - это скользящее среднее линии %K. Обычно %K изображается сплошной линией, а %D – пунктирной [2]. Существует три наиболее распространенных способа интерпретации Стохастического Осциллятора:

- покупайте, когда осциллятор (%K или %D) сначала опустится ниже определенного уровня (обычно 20), а затем поднимется выше него, и продавайте, когда осциллятор сначала поднимется выше определенного уровня (обычно 80), а потом опустится ниже него;

- покупайте, если линия %K поднимается выше линии %D, и продавайте, если линия %K опускается ниже линии %D.

- следите за расхождениями, например, когда цены образуют ряд новых максимумов, а Стохастическому Осциллятору не удается подняться выше своих предыдущих максимумов.

Формула, необходимая для расчета Stochastic (3.7).

$$\%K = \frac{\left(\text{CLOSE} - \text{MIN}\left(\text{LOW}\left(P\right)\right)\right)}{\left(\text{MAX}\left(\text{HIGH}\left(P\right)\right) - \text{MIN}\left(\text{LOW}\left(P\right)\right)\right) * 100}, \quad (3.7)$$

где CLOSE – закрытие свечи;

P – период, выбранный пользователем;

MIN(LOW(P)) – наименьший минимум за число периодов P;

MAX(HIGH(P)) – наибольший максимум за число периодов P.

Далее необходимо простроить алгоритм расчета индикатора, алгоритм изображен на рисунке 3.13.

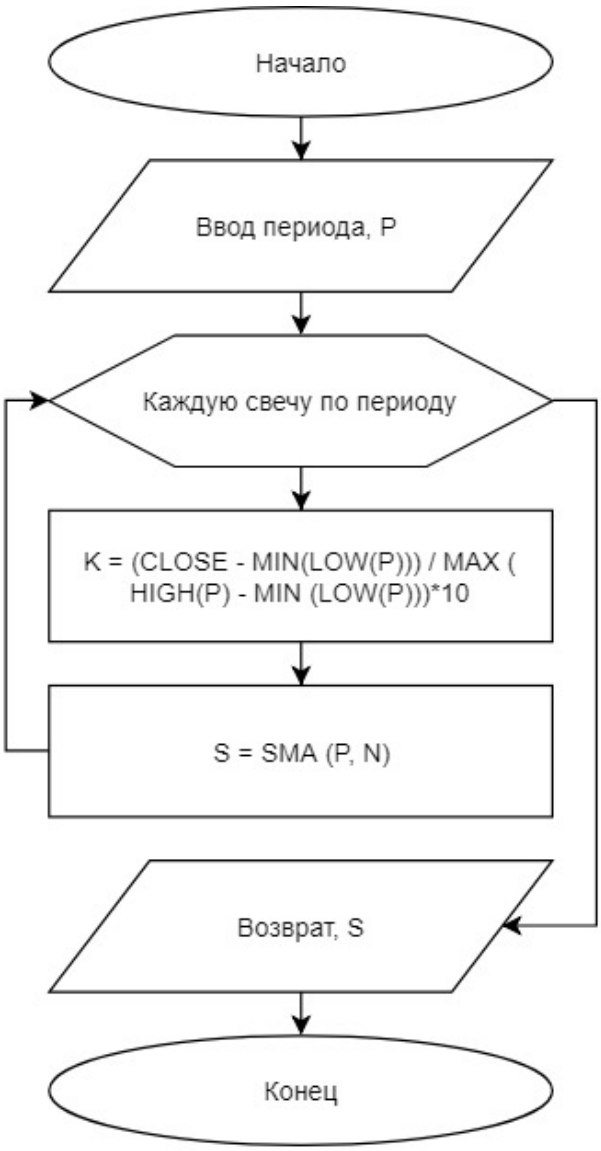


Рисунок 3.13 – Алгоритм расчета индикатора Stochastic

На рисунке 3.13 изображен алгоритм расчета индикатора, как можно заметить, пользователь указывает параметр периода. Далее при приходе новой свечи индикатор будет рассчитывать каждую свечу за указанный период, после чего возвращать значение в торговую стратегию.

- Relative Strength Index (RSI);

Технический индикатор Индекс Относительной Силы (Relative Strength Index, RSI) — это следующий за ценой осциллятор, который колеблется в диапазоне от 0 до 100. Один из распространенных методов анализа индикатора Relative Strength Index состоит в поиске расхождений, при которых цена образует новый максимум, а RSI не удается преодолеть уровень своего

предыдущего максимума. Подобное расхождение свидетельствует о вероятности разворота цен. Если затем индикатор поворачивает вниз и опускается ниже своей впадины, то он завершает так называемый «неудавшийся размах» (failure swing). Этот неудавшийся размах считается подтверждением скорого разворота цен [2].

Основная формула расчета технического индикатора Relative Strength Index (3.8).

$$\%K = 100 - \left(\frac{100}{1+\frac{U}{D}}\right), \tag{3.8}$$

где U – среднее значение положительных ценовых изменений;

D – среднее значение отрицательных ценовых изменений.

Далее необходимо построить алгоритм работы индикатора, для этого вначале необходимо разработать дополнительную функцию, которая будет считать количество положительных ценовых изменений, на рисунке 3.14 изображён алгоритм работы функции UP.

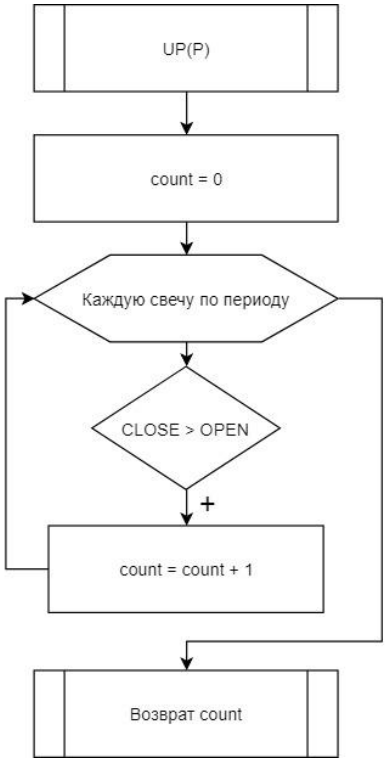


Рисунок 3.14 – Алгоритм работы дополнительной функции

На рисунке 3.14 видно, как рассчитывается количество положительных закрытий. При помощи данного показателя можно рассчитать количество отрицательных, отняв период, установленный пользователем. Далее при помощи этих значений можно рассчитать средние показатели. Алгоритм изображен на рисунке 3.15.

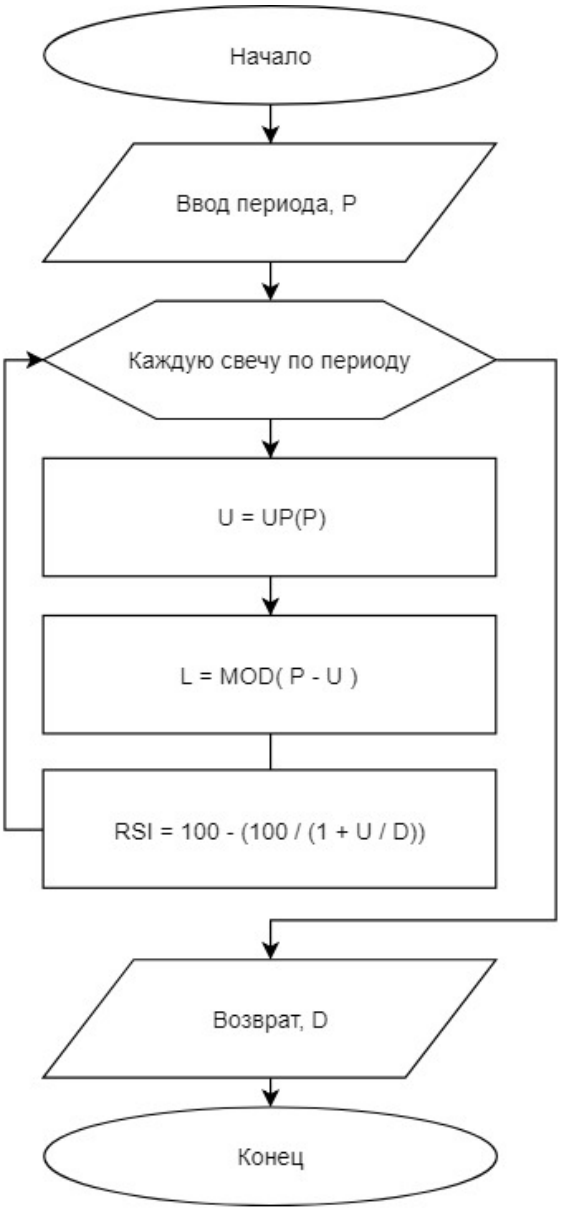


Рисунок 3.15 – Алгоритм работы индикатора RSI

На рисунке 3.15 подробно изображен алгоритм индикатора RSI, в нём можно заметить схожесть с индикатором Stochastic. Он также осуществляет

расчеты каждую новую цену. После чего остается рассчитать только формулу и вывести на экран.

- Moving average (MA). Технический индикатор Скользящее Среднее (Moving Average, MA) показывает среднее значение цены инструмента за некоторый период времени. При расчете Moving Average производится математическое усреднение цены инструмента за данный период. По мере изменения цены ее среднее значение либо растет, либо падает [2].

Самый распространенный метод интерпретации скользящего среднего цены состоит в сопоставлении его динамики с динамикой самой цены. Когда цена инструмента поднимается выше значения Moving Average, возникает сигнал к покупке, при ее падении ниже линии индикатора — сигнал к продаже.

Данная система торговли с помощью Moving Average вовсе не предназначена обеспечивать вхождение в рынок строго в его низшей точке, а выход — строго на вершине. Она позволяет действовать в соответствии с текущей тенденцией: покупать вскоре после того, как цены достигли основания, и продавать вскоре после образования вершины.

Формула расчета MA отображена в формуле (3.9).

$$MA = \frac{\mathrm{SUM(CLOSE\ (i),N)}}{\mathrm{N}}, \tag{3.9}$$

где SUM – сумма;

CLOSE (i) – цена закрытия текущего периода;

N – число периодов расчета.

На рисунке 3.16 изображен алгоритм данного индикатора, который отображает основные действия.

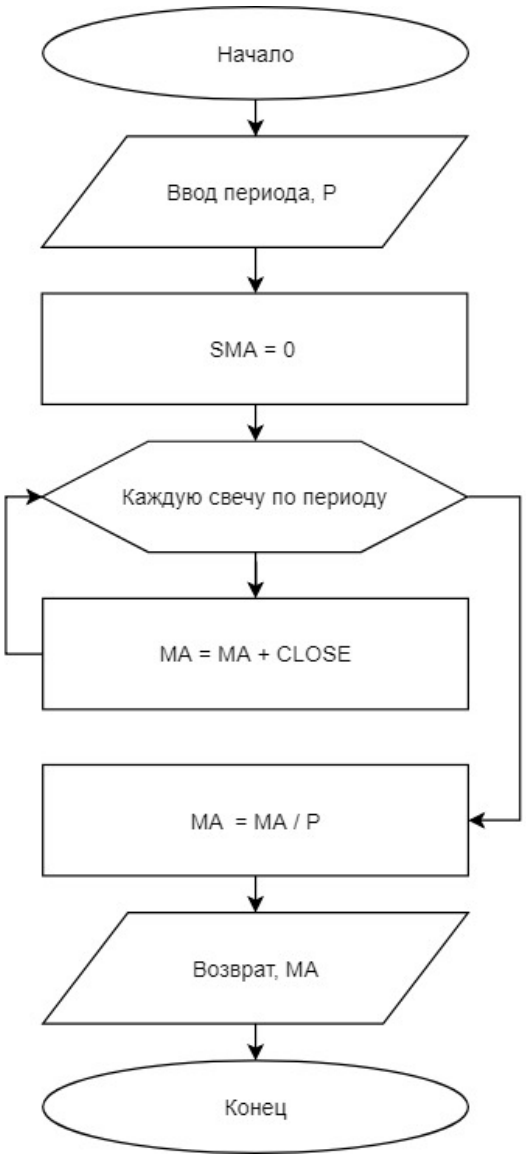


Рисунок 3.16 – Алгоритм работы индикатора MA

На рисунке 3.16 видно, что изначально пользователь также указывает параметр, после чего объявляется переменная для хранения среднего значения. Далее каждую свечу рассчитывается сумма, после чего делится на указанный период и возвращается в торгового советника.

После описания всех необходимых индикаторов перейти в MetaTrader и приступить непосредственной разработке их. Для создания нового индикатора необходимо повторить все действия, описанные в разделе 3.1.1., только вместо «Советник (Шаблон)» необходимо выбрать «Пользовательский индикатор». Результат выбора изображен на рисунке 3.17.

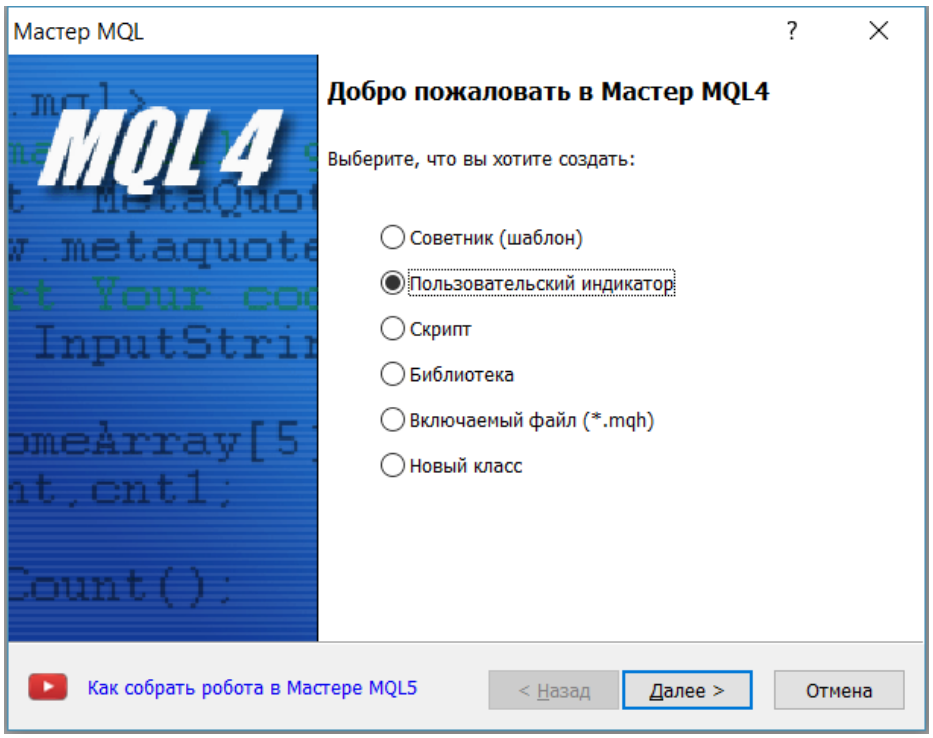


Рисунок 3.17 – Создание пользовательского индикатора

На рисунке 3.17 изображен выбор «Пользовательского индикатора», после чего необходимо указать названия индикатора и дополнительные параметры. Далее создается шаблон программного кода, с наличием главных функций в нём. Пример такого листинга изображен на рисунке 3.18.

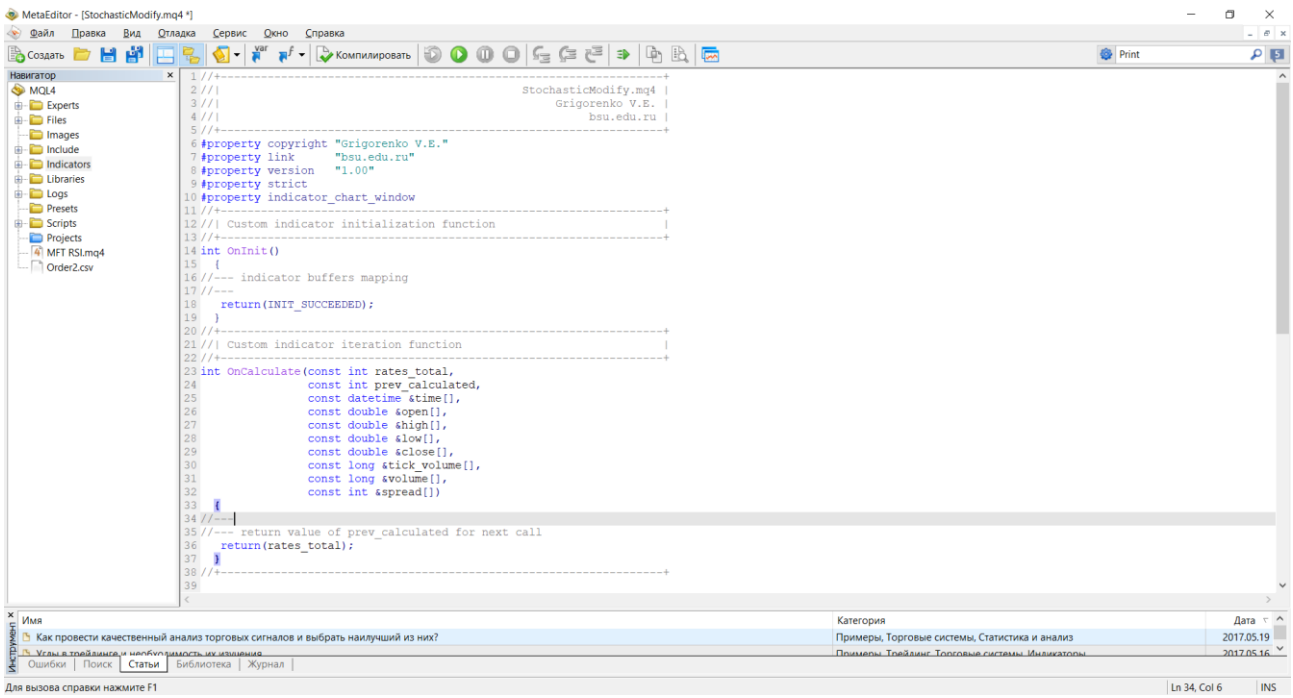


Рисунок 3.18 – Листинг шаблона «Пользовательского индикатора»

На рисунке 3.18 изображен шаблон пользовательского индикатора, на котором видно разницу с шаблоном торгового советника. Видно, что появилась новая функция OnCalculate. Данная функция вызывается каждую новую временной отрезок. Также можно заметить большое количество параметров, которые отвечают за подсчет пройдённых временных отрезков и тех, которые следует пройти, а также данные находящиеся на текущем временном отрезке.

Далее необходимо осуществить реализацию алгоритма, указанного выше в данном шаблоне индикатора. Листинги данных индикаторов указаны в приложении В.

3.3 Модификация торговой стратегии

После реализации индикаторов необходимо приступить к модификации торговой стратегии. Изначально необходимо построить модель алгоритма «Как должно быть», данная модель изображена на рисунке 3.19.

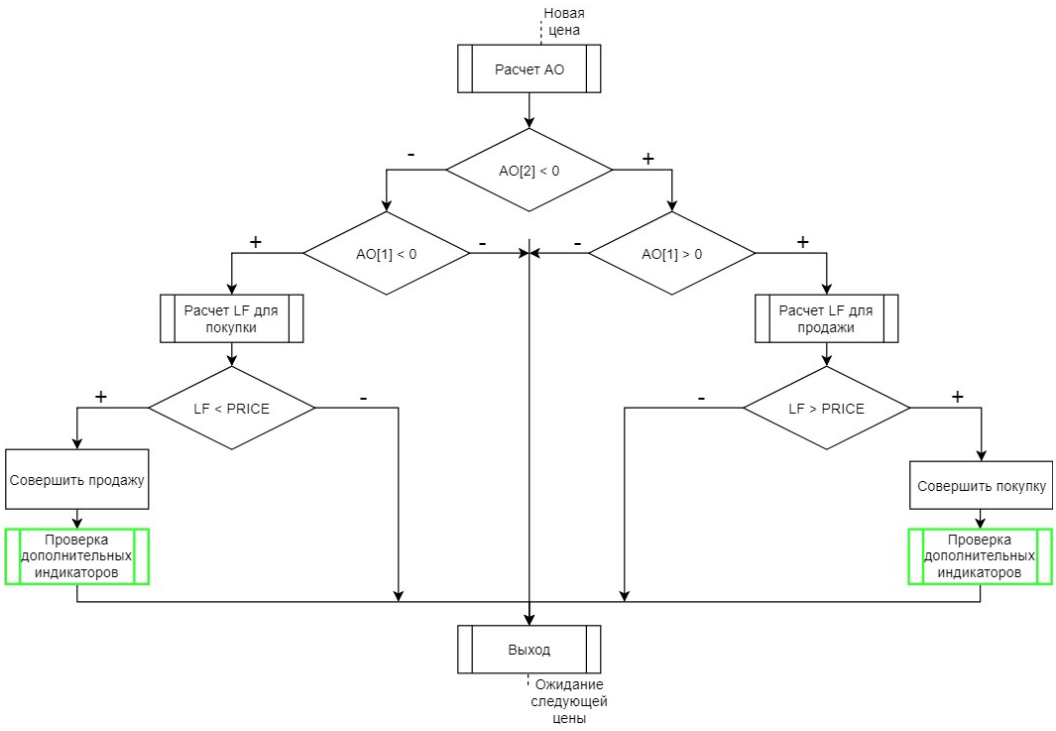


Рисунок 3.19 – Модифицированная торговая стратегия

Теперь необходимо осуществить программную реализацию модуля «Проверка дополнительных индикаторов». Чтобы получать значения от разработанных индикаторов, необходимо осуществить вызов индикаторов в торговый советник. Листинг вызова изображен на рисунке 3.20.

```
double ao_1 = iCustom(_Symbol, _Period, "AO", Fast_period_ao, Slow_period_ao, 0, 1);
double stoh = iCustom(_Symbol, _Period,"StochasticModify", 5,3,3, MODE_SMA, 0, 0, 1);
double stoh_signal = iStochastic(_Symbol, _Period, "StochasticModify", 5,3,3, MODE_SMA, 0, 1, 1);
double rsi = iCustom(_Symbol, _Period,"RSIModify" Period_rsi, PRICE_CLOSE, 1);
double ma_1 = iCustom(_Symbol, PERIOD_H1,"MAModify" Period_ma_sell, 0, MODE_SMA, PRICE_CLOSE, 1);
double ma_2 = iCustom(_Symbol, PERIOD_H4,"MAModify" Period_ma_sell, 0, MODE_SMA, PRICE_CLOSE, 1);
double ma_3 = iCustom(_Symbol, PERIOD_D1,"MAModify" Period_ma_sell, 0, MODE_SMA, PRICE_CLOSE, 1);
```

Рисунок 3.20 – Функции вызова набора индикаторов

На рисунке 3.20 видны вызовы функции ICustom, данная функция, встроенная в MetaTrader, позволяет получать данные из внешних индикаторов. После чего необходимо сверять предельные значения, указанные пользователем. Листинг изображен на рисунке 3.21.

```
if( (ao_1 < Less_ao) && (ao_1 > Stop_less_ao) && (stoh > Stoch_More) && (rsi > Rsi_More) && (FilterLastOrder(OP_SELL)) && FilterOrderModify(OP_BUY, 7) ){
   if( (ma_1 < iHigh(_Symbol, PERIOD_H1, 1)) && (ma_2 > iHigh(_Symbol, PERIOD_H4, 1)) && (ma_3 > iHigh(_Symbol, PERIOD_D1, 1)) ){
      lots = max_lots;
   }
   return true;
}
return false;
```

Рисунок 3.21 – Проверки на предельные значения

На рисунке 3.21 видно большое условие, в котором осуществляется проверка всех индикаторов по предельным значениям. Дальше на основе предельных значений открывается сделка в одну из сторон, то есть покупку или продажу. Для осуществления конфигурирования торговой стратегии инвестором можно осуществить настройку предельных значений сразу при запуске, которые выбирает пользователь в настройках торговой стратегии, выбор параметров изображен на рисунке 3.22.

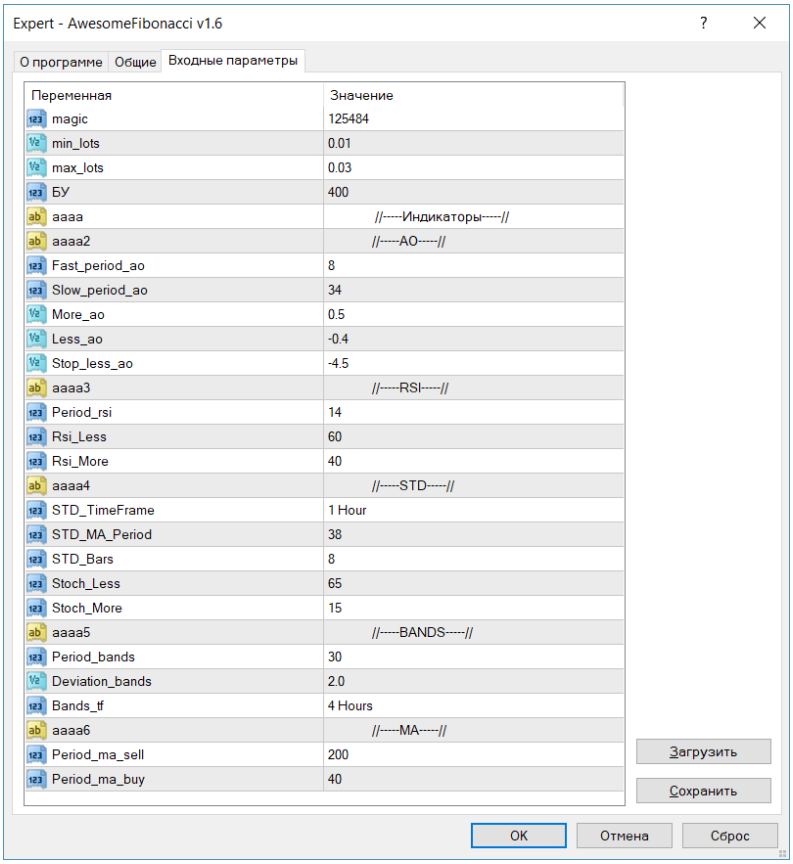


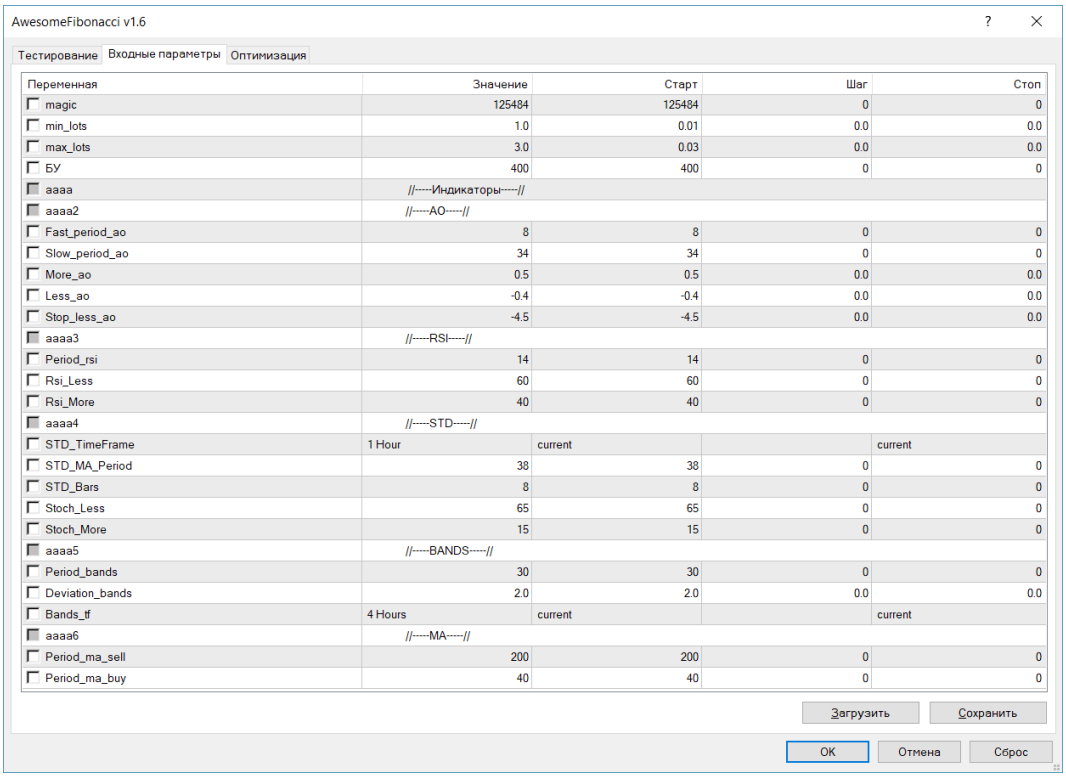
Рисунок 3.22 – Настройка параметров торговой стратегии

На рисунке 3.22 видны различные параметры, которые позволяют конфигурировать торговую стратегию. Можно выбрать минимальный и максимальный размер сделки, индикаторы (AO, RSI, Stochastic, MA, Bands), после чего запускается торговля. Полный листинг торговой стратегии можно посмотреть в приложении Г.

3.4 Описание контрольного примера реализации проекта

Для того чтобы продемонстрировать пример работы торгового советника, необходимо запустить MetaTrader и включить режим тестирования, который был рассмотрен в разделах выше. Далее в открывшемся меню снизу необходимо выбрать торговый советник и финансовый инструмент. В

Свойствах эксперта нужно указать предельные значения. Пример предельных значений изображен на рисунке 3.23.



Переменная	Значение	Старт	Шаг	Стоп
magic	125484	125484	0	0
min_lots	1.0	0.01	0.0	0.0
max_lots	3.0	0.03	0.0	0.0
БУ	400	400	0	0
aaaa	//-----Индикаторы-----//			
aaaa2	//-----AO-----//			
Fast_period_ao	8	8	0	0
Slow_period_ao	34	34	0	0
More_ao	0.5	0.5	0.0	0.0
Less_ao	-0.4	-0.4	0.0	0.0
Stop_less_ao	-4.5	-4.5	0.0	0.0
aaaa3	//-----RSI-----//			
Period_rsi	14	14	0	0
Rsi_Less	60	60	0	0
Rsi_More	40	40	0	0
aaaa4	//-----STD-----//			
STD_TimeFrame	1 Hour	current		current
STD_MA_Period	38	38	0	0
STD_Bars	8	8	0	0
Stoch_Less	65	65	0	0
Stoch_More	15	15	0	0
aaaa5	//-----BANDS-----//			
Period_bands	30	30	0	0
Deviation_bands	2.0	2.0	0.0	0.0
Bands_tf	4 Hours	current		current
aaaa6	//-----MA-----//			
Period_ma_sell	200	200	0	0
Period_ma_buy	40	40	0	0

Рисунок 3.23 – Пример заданных параметров

На рисунке 3.23 видны все выбранные параметры, которые выполняют оптимальную прибыль для торговли. После этого необходимо выбрать период тестирования торговой стратегии, в данном примере периодом является начало 2014 года по текущий года написания данной работы 05.2018 года соответственно. Результаты тестирования торговой стратегии отображены на рисунке 3.24.

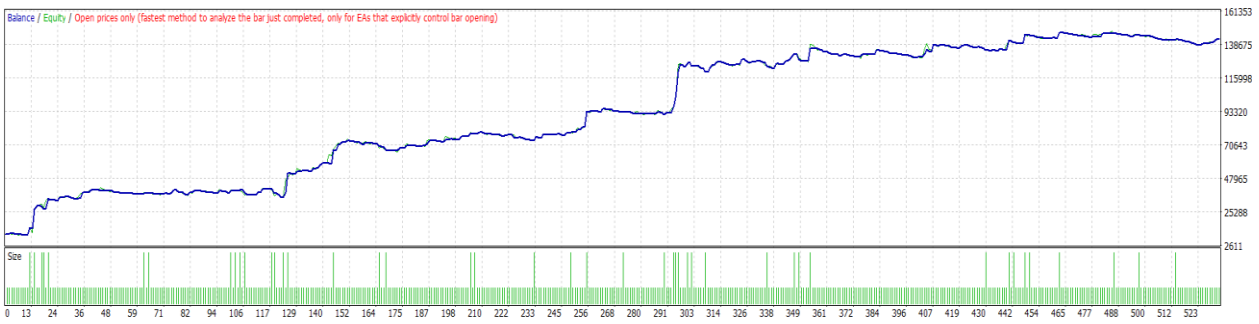


Рисунок 3.24 – График прибыли

На рисунке 3.24 изображен график прибыли после внедрения набора торговых индикаторов. Необходимо посмотреть пример сделки, чтобы убедиться в правильности работы алгоритма. Пример сделки изображен на рисунке 3.25.

Рисунок 3.25 – График с сделками

На рисунке 3.25 изображено много торговых сделок, необходимо обратить внимание на первую покупку (отображена синей линей и первые линии Фибоначчи с левой стороны). Как видно на нижнем индикаторе АО было пересечение снизу верх, и смена тенденции, также это подтверждал Stochstic и RSI тенденцией вниз. В данный момент были рассчитаны линии Фибоначчи по алгоритму и выполнена сделка. После чего был скачок цены, и открылась торговая сделка и, как видно дальше, она закрылась в прибыль. Этому поспособствовал индикатор МА, по которому осуществлялось изменение уровней убытков.

После анализа контрольного примера необходимо перейти к анализу торговых результатов и оценки экономической эффективности, по отношению к предыдущей торговой стратегии.

3.5 Оценка экономической эффективности

После реализации необходимо осуществить оценку экономической эффективности. Экономическая эффективность является показателем соотношения прибыли к результату работы торгового советника и затрат на начало инвестиционной деятельности.

Эффективность торговой стратегии можно определить при помощи тестирования её на протяжении нескольких пройденных лет, что позволяет аппроксимировать прибыль на будущих исходах торгов. Результаты торговой стратегии после внедрения индикаторов изображены на рисунке 3.26.

Initial deposit	10000.00			Spread	Current (518)
Total net profit	133161.15	Gross profit	272829.91	Gross loss	-139668.76
Profit factor	1.95	Expected payoff	250.30		
Absolute drawdown	350.18	Maximal drawdown	11146.83 (8.09%)	Relative drawdown	23.28% (10529.32)
Total trades	532	Short positions (won %)	265 (55.85%)	Long positions (won %)	267 (51.69%)
		Profit trades (% of total)	286 (53.76%)	Loss trades (% of total)	246 (46.24%)
	Largest	profit trade	17401.75	loss trade	-3563.66
	Average	profit trade	953.95	loss trade	-567.76
	Maximum	consecutive wins (profit in money)	10 (16369.68)	consecutive losses (loss in money)	7 (-2226.11)
	Maximal	consecutive profit (count of wins)	33244.62 (4)	consecutive loss (count of losses)	-5016.66 (6)
	Average	consecutive wins	2	consecutive losses	2

Рисунок 3.26 – Результаты торгов после внедрения индикаторов

На рисунке 3.26 видны значительные изменения в прибыльности торговли, далее необходимо сравнить результаты торгов начальной торговой стратегии и после внедрения дополнительных индикаторов. Изначально необходимо провести статистическое сравнение, которое изображено на таблице 3.1.

Таблица 3.1 - Статистические показатели

Показатель	До внедрения	После внедрения
RF	0.26	11.94
Z-score	-7.08	-0.87
LR	0.17	0.97
SR	-0.07	0.14
Sortino	0.03	0.51
Фактор прибыли	1.04	1.95

В таблице 3.1 видно, как торговая стратегия после внедрения индикаторов превосходит по всем показателям. В разделе 3.1.4 было подробно описано, как сравнивать статистические показатели. Далее необходимо сравнить по абсолютным показателям, анализ отображен на таблице 3.2.

Таблица 3.2 – Сравнения абсолютных показателей

Показатель	До внедрения	После внедрения	Измерение
Прибыль	158	133161	Доллары
Кол-во прибылых сделок	48.29	53.76	Проценты
Необходимая сумма для внедрения	10000	10000	Доллары
Абсолютная просадка	405	350	Долларов
Кол-во убыточных сделок	51.71	46.24	Проценты

В таблице 3.2 показаны абсолютные показатели, и в очередной раз торговая стратегия после внедрения полностью опережает в прибыли до внедрения.

По результатам проведенного анализа можно сделать вывод, что торговая стратегия после внедрения индикаторов приносит больше финансовой прибыли. Тем самым поставленная цель «повышение финансовой прибыльности торговой стратегии» является выполненной.

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы цель исследования была достигнута. Была повышена финансовая прибыльность торговой стратегии. Для достижения поставленной цели были выполнены следующие задачи:

- произведен анализ предметной области и выявлены недостатки текущей торговой системы;
- выполнена разработка алгоритмов работы индикаторов;
- осуществлена разработка набора индикаторов;
- выполнена модификация торговой стратегии;
- произведено тестирование индикаторов и торговой стратегии с использованием индикаторов;
- проанализирована экономическая эффективность усовершенствованной торговой стратегии.

Разработанная торговая стратегия доказала эффективность при помощи экономических показателей, что говорит о том, что данная стратегия имеет огромный потенциал для использования на финансовых рынках. Также в процессе подготовки ВКР был разработан набор индикаторов, который в дальнейшем может поспособствовать увеличению извлекаемой инвесторами прибыли или использования в качестве дополнительных индикаторов для имеющихся стратегий. В будущем необходимо произвести анализ рынка и оценить эффективность торгового советника спустя время.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Построение инфологической модели [Электронный ресурс] - Режим доступа: http://citforum.ru/database/dbguide/5-2.shtml, свободный.
2. Математика в трейдинге. Оценка результатов торговых сделок [Электронный ресурс] – Режим доступа https://www.mql5.com/ru/articles/1492, свободный.
3. Как использовать фактор восстановления [Электронный ресурс] – Режим доступа: https://freshforex.org/encyclopedia-forex/how_to_use_the_recovery_factor, свободный.
4. Мезенцев, К.Н. Автоматизированные информационные системы: Учебник для студентов учреждений среднего профессионального образования [Текст] / К.Н. Мезенцев. - Москва: ИЦ Академия, 2013. - 176 с.
5. Мэрфи, Д. Д. Технический анализ финансовых рынков. Полный справочник по методам и практике трейдинга. [Текст] / Д.Д. Мэрфи – Москва: Вильямс, 2016. - 496с.
6. Словарь трейдера. Основные понятия [Электронный ресурс] / Режим доступа: http://success-everywhere.ru/finansi/birzha-i-rinok/slovar_trejdera, свободный.
7. Свободная энциклопедия [Электронный ресурс] / Режим доступа: https://ru.wikipedia.org/, свободный.
8. Антамошин, А.Н. Интеллектуальные системы управления организационно-техническими системами [Текст] / А.Н. Антамошин, О.В. Близнова, А.В. Бобов, Большак. - М.: РиС, 2016. - 160 с
9. Гахов, Р.П. Компьютерное моделирование экономических процессов [Текст]: учебное пособие для студентов вузов / Р.П. Гахов, Н.В. Щербинина. - Белгород: ИД Белгород, 2014. - 88 с.

10. Гахова, Н.Н. Инструментальные средства информационных систем [Электронный ресурс] / Н.Н. Гахова - Белгород: НИУ БелГУ, 2012. - Режим доступа: http://pegas.bsu.edu.ru/course/view.php?id=5188, свободный.

11. Илюшечкин, В.М. Основы использования и проектирования баз данных [Текст] / В.М. Илюшечкин. – Москва: Юрайт, 2017. – 214 с.

12. Румбешт, В.В. Программирование информационных систем: Учебно-методическое пособие по выполнению лабораторных работ [Текст] / В.В. Румбешт, Г.Г. Банчук. – Белгород: Кооперативное образование, 2006. – 165с.

13. Чистов, Д.В. Проектирование информационных систем. Учебник и практикум [Текст] / Д.В. Чистов, П.В. Мельников. – Москва: Юрайт, 2017, – 216с.

14. Антонова, А.С. Автоматизированные информационные системы, базы и банки данных. Вводный курс: Учебное пособие [Текст] / А.С. Антонова. - М.: Гелиос АРВ, 2014. - 368 с.

15. Алиев, С.А. Быстрая разработка программного обеспечения [Текст] / С.А. Алиев. - М., 2013. - 336 с

16. Баранова, Е. Н. Основы информатики и защиты информации [текст] / Е.Н. Баранова. - М., 2013. - 192 с.

17. Дэвид Паттерсон. Архитектура компьютера и проектирование компьютерных систем [Текст] / Д. Паттерсон. - М., 2014, 784 с.

18. Документация MetaTrader 4 [Электронный ресурс] - Режим доступа: https://docs.mql4.com/ru, свободный.

19. Московская биржа [Электронный ресурс] - Режим доступа: https://www.moex.com, свободный.

20. Технические индикаторы [Электронный ресурс] - Режим доступа: https://www.metatrader4.com/ru/trading-platform/help/analytics/tech_indicators, свободный.

21. Индикаторы forex [Электронный ресурс] - Режим доступа: https://tradexperts.ru/indikatory_forex.htm, свободный.

22. Московская биржа [Электронный ресурс] - Режим доступа: https://www.moex.com, свободный.

23. Уильсам, Л. Секреты торговли на фьючерсном рынке [Текст] / Л. Уильсам – Москва: Альпина Паблишер, 2018, – 234с.

24. Голдрат М.Э. Я так и знал! Розничная торговля и Теория ограничений [Текст] / М.Э. Голдрат, И.А. Эшколи, Д.Б. Лир: Альпина Паблишер, 2018, - 168с.

25. Голдрат М.Э. Цель-2. Дело не в везении [Текст] / М.Э. Голдрат – Москва: Альбина Паблишер, 2018, - 230с.

26. Кауфман П. Системы и методы биржевой торговли [Текст] / П. Кауфман – Москва: Альбина Паблишер, 2017, - 1279с.

27. Кауфман П. Системы и методы биржевой торговли [Текст] / П. Кауфман – Москва: Альбина Паблишер, 2017, - 1279с.

28. Кац Д. О. Энциклопедия торговых стратегий [Текст] / Д.О. Кац, Д.Л. Маккормик – Москва: Альпина, 2015, - 392с.

29. Кондаков К.Г. MetaTrader 4. Учимся зарабатывать на FOREX [Текст] / К.Г. Кондаков, О.В. Бондарь – Москва: Бослен, 2012, - 152с.

30. Шилов Б. MetaTrader: пособие для «кофейников» [Текст] / Б. Шилов, Д. Раннев – Санкт-Петербург: И-Трейд, 2014, - 137с.

31. Торговый терминал NinjaTrader [Электронный ресурс] - Режим доступа: https://ninjatrader.com/ru/, свободный.

32. Торговый терминал MetaTrader 5 [Электронный ресурс] - Режим доступа: https://www.metatrader5.com/ru, свободный.

33. Торговый терминал QUIK [Электронный ресурс] - Режим доступа: https://arqatech.com/ru/products/quik/, свободный.

34. Лихтенштейн, В.Е. Стандартизация и разработка программных систем: Учебное пособие [Текст] / В.Е. Лихтенштейн. - М.: Финансы и статистика, 2010. - 288 с.

35. Мишенин, А.И. Теория экономических информационных систем [Текст]: Учеб.пособие. — М.: Финансы и статистика, 2015 -278 с.

36. Маклаков, С.В. Моделирование бизнес-процессов с BPwin 4.0 [Текст] / С.В. Маклаков. – М.: ДИАЛОГ-МИФИ, 2012. – 224 с.

37. Вендров, А.М. Проектирование программного обеспечения экономических информационных систем: Учебник [Текст]/ А.М. Вендров. - Москва: Финансы и статистика, 2013 – 522с.

38. Назарова, С.В. Компьютерные технологии обработки информации [Текст] / С.В. Назарова - Москва: Финансы и статистика, 2013. – 248 с.

39. Емельянова, Н.З. Проектирование информационных систем [текст]/ Н.З. Емельянова, Партыка Т. Л., Попов И. И – М., 2014. - 432 с.

40. 18. Репин, В.В, Елиферов В.Г. Процессный подход к управлению. Моделирование бизнес-процессов: [Текст]/ В.В Репин, В.Г. Елиферов. – Москва: Манн, Иванов и Фербер, 2013 – 524 с.

41. Маторин, С.И. Теория систем и системный анализ [Текст] / С.И. Маторин, О.А. Зимовец. – Белгород. ИД Белгород, 2012 – 154 с.

42. Коноплева, И.А. Информационные технологии [Текст] / И.А. Коноплева, О.А. Хохлова, А.В. Денисов. - Москва: Проспект, 2015. - 328 с.

43. Каймин, В.А. Бизнес- процессы: Учебник [Текст] / В.А. Каймин - М.: ИНФРА-М, 2013 – 179 с.

44. Голицына, О. Л. Информационные технологии [Текст] / О.Л. Голицина, Попов И. И., Максимов Н. В., Партыка Т. Л., - М., 2014. - 608 с

ПРИЛОЖЕНИЕ А

Индикатор AO

```
//+------------------------------------------------------------------+
//|                                                      Awesome.mq4 |
//|                 Copyright 2005-2014, MetaQuotes Software Corp. |
//|                                              https://www.mql5.com |
//+------------------------------------------------------------------+
#property copyright   "2005-2014, MetaQuotes Software Corp."
#property link        "https://www.mql5.com"
#property description "Awesome Oscillator"
#property strict
//--- indicator settings
#property  indicator_separate_window
#property  indicator_buffers 3
#property  indicator_color1  Black
#property  indicator_color2  Green
#property  indicator_color3  Red
//--- buffers
double     ExtAOBuffer[];
double     ExtUpBuffer[];
double     ExtDnBuffer[];
//---
extern int PERIOD_FAST = 5;
extern int PERIOD_SLOW = 34;
//--- bars minimum for calculation
int DATA_LIMIT  = PERIOD_SLOW;
//+------------------------------------------------------------------+
//| Custom indicator initialization function                         |
//+------------------------------------------------------------------+
void OnInit(void)
  {
//--- drawing settings
   SetIndexStyle(0,DRAW_NONE);
   SetIndexStyle(1,DRAW_HISTOGRAM);
   SetIndexStyle(2,DRAW_HISTOGRAM);
   IndicatorDigits(Digits+1);
   SetIndexDrawBegin(0,DATA_LIMIT);
   SetIndexDrawBegin(1,DATA_LIMIT);
   SetIndexDrawBegin(2,DATA_LIMIT);
//--- 3 indicator buffers mapping
   SetIndexBuffer(0,ExtAOBuffer);
   SetIndexBuffer(1,ExtUpBuffer);
   SetIndexBuffer(2,ExtDnBuffer);
//--- name for DataWindow and indicator subwindow label
   IndicatorShortName("AO");
   SetIndexLabel(1,NULL);
   SetIndexLabel(2,NULL);
  }
//+------------------------------------------------------------------+
//| Awesome Oscillator                                               |
//+------------------------------------------------------------------+
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
  {
   int    i,limit=rates_total-prev_calculated;
   double prev=0.0,current;
//--- check for rates total
   if(rates_total<=DATA_LIMIT)
      return(0);
//--- last counted bar will be recounted
   if(prev_calculated>0)
     {
      limit++;
```

```
      prev=ExtAOBuffer[limit];
     }
//--- macd
   for(i=0; i<limit; i++)
      ExtAOBuffer[i]=iMA(NULL,0,PERIOD_FAST,0,MODE_SMA,PRICE_MEDIAN,i)-
                iMA(NULL,0,PERIOD_SLOW,0,MODE_SMA,PRICE_MEDIAN,i);
//--- dispatch values between 2 buffers
   bool up=true;
   for(i=limit-1; i>=0; i--)
     {
      current=ExtAOBuffer[i];
      if(current>prev)
        up=true;
      if(current<prev)
        up=false;
      if(!up)
        {
         ExtDnBuffer[i]=current;
         ExtUpBuffer[i]=0.0;
        }
      else
        {
         ExtUpBuffer[i]=current;
         ExtDnBuffer[i]=0.0;
        }
      prev=current;
     }
//--- done
   return(rates_total);
  }
//+------------------------------------------------------------------+
```

ПРИЛОЖЕНИЕ Б

```
double OnTester()
{
  Print("Z score = " + DoubleToStr(Z_Score()) + " LR = " + DoubleToStr(Regress()) + " SR = " + DoubleToStr(Sharpe()) + " Sortino = " +
DoubleToStr(Sortino()));
  return  TesterStatistics(STAT_PROFIT) / TesterStatistics(STAT_EQUITY_DD);
}
double Z_Score()
{
  double r = 1, w = 0, l = 0;
  bool last_up = true, f = true;

  int j = 0 , n = 1;
  while(f && j < OrdersHistoryTotal()-1){
    if(OrderSelect(j, SELECT_BY_POS, MODE_HISTORY) && (OrderType() == OP_BUY || OrderType() == OP_SELL)){
      if(OrderProfit() > 0) {
        w++;
      } else {
        l++;
        last_up = false;
      }
      f = false;
    }
    j++;
  }

  for(int i = j; i < OrdersHistoryTotal(); i++) {
    if(OrderSelect(i, SELECT_BY_POS, MODE_HISTORY) && (OrderType() == OP_BUY || OrderType() == OP_SELL)) {
      if(OrderProfit() > 0) {
        w++;
        if(!last_up) {
          r++;
          last_up = true;
        }
      } else {
        l++;
        if(last_up) {
          r++;
          last_up = false;
        }
      }
      n++;
    }
  }
  double p = 2 * w * l;

  double a = n * (r - 0.5) - p;
  double b = pow((p * (p - n)) / (n - 1), 1./2);

  return a / b;
}
double Regress()
{
  int x = 0;
  double balance = TesterStatistics(STAT_INITIAL_DEPOSIT);
  double s_x = 0, s_xy = 0,sum_y = 0, s_x2 = 0;
  for(int i = 0; i < OrdersHistoryTotal(); ++i) {
    if(OrderSelect(i, SELECT_BY_POS, MODE_HISTORY) && (OrderType() == OP_BUY || OrderType() == OP_SELL)) {
      x++;
      s_x += x;

      balance += OrderProfit();
      sum_y += balance;
      s_x2 += x*x;
      s_xy += x*balance;
    }
  }
  double avg_x = s_x / x;
  double avg_y = sum_y / x;
  double b = (x * s_xy - s_x * sum_y) / (x * s_x2 - (pow(s_x, 2)));
  double a =  avg_y - b * avg_x;
  double y_line = a + b * x;
```

```
  double sy_line = 0;

  x = 0;
  for(int i = 0; i < OrdersHistoryTotal(); ++i) {
    if(OrderSelect(i, SELECT_BY_POS, MODE_HISTORY) && (OrderType() == OP_BUY || OrderType() == OP_SELL)) {
      x++;
      sy_line += a + b * x;
    }
  }
  double avg_y_line = sy_line / x;

  double sum_t = 0, sum_sx = 0, sum_sy = 0;

  x = 0;
  sy_line = 0;
  balance = TesterStatistics(STAT_INITIAL_DEPOSIT);

  for(int i = 0; i < OrdersHistoryTotal(); ++i) {
    if(OrderSelect(i, SELECT_BY_POS, MODE_HISTORY) && (OrderType() == OP_BUY || OrderType() == OP_SELL)) {
      x++;
      balance += OrderProfit();
      sy_line = a + b * x;
      sum_t += (balance - avg_y) * (sy_line - avg_y_line);
      sum_sx += pow(balance - avg_y,2);
      sum_sy += pow(sy_line - avg_y_line,2);
    }
  }
  double t = sum_t / x - 1;
  double sx = sqrt(sum_sx/ x -1);
  double sy = sqrt(sum_sy / x -1);

  return t/(sx*sy);
}

double Sharpe()
{
  double sum_hpr = 0;
  int counter = 0;
  for(int i = 0; i < OrdersHistoryTotal(); i++) {
    if(OrderSelect(i, SELECT_BY_POS, MODE_HISTORY) && (OrderType() == OP_BUY || OrderType() == OP_SELL)) {
      counter++;
      sum_hpr += OrderProfit();
    }
  }

  double ahpr = sum_hpr / counter;
  double sum_sqr_hpr = 0;

  for(int i = 0; i < OrdersHistoryTotal(); i++) {
    if(OrderSelect(i, SELECT_BY_POS, MODE_HISTORY)) {
      sum_sqr_hpr += pow((OrderProfit() - ahpr), 2);
    }
 }

 double sd = sqrt(sum_sqr_hpr/(counter-1));

 return (ahpr - 1) / sd;
}

double Sortino()
{
  double sum_hpr = 0, sum_less_profit = 0;
  int counter = 0;

  for(int i = 0; i < OrdersHistoryTotal(); i++) {
    if(OrderSelect(i, SELECT_BY_POS, MODE_HISTORY) && (OrderType() == OP_BUY || OrderType() == OP_SELL)) {
      counter++;
      sum_hpr += OrderProfit();
      if(OrderProfit() < 0){
        sum_less_profit += pow(OrderProfit(),2);
      }
    }
  }

  double ahpr = sum_hpr / counter;

  double sd = sqrt(sum_less_profit / counter);
  return ahpr/sd;
}
```

ПРИЛОЖЕНИЕ В

Stochastic

```
//+------------------------------------------------------------------+
//|                                        StochasticModify.mq4 |
//|                                            Grigorenko V.E. |
//|                                                bsu.edu.ru |
//+------------------------------------------------------------------+
#property strict
#property indicator_separate_window
#property indicator_minimum    0
#property indicator_maximum    100
#property indicator_buffers    2
#property indicator_color2     Red
#property indicator_level1     20.0
#property indicator_level2     80.0
input int KPer=5;
input int DPer=3;
input int Slow=3;
 double CentralBuffer[];
double FirstSingnalbuffer[];
double HBuffer[];
double LBuffer[];
int draw1=0;
int draw2=0;
int OnInit(void)
  {
   string short_name;
   IndicatorBuffers(4);
   SetIndexBuffer(2, HBuffer);
   SetIndexBuffer(3, LBuffer);
   SetIndexStyle(0,DRAW_LINE);
   SetIndexBuffer(0, CentralBuffer);
   SetIndexStyle(1,DRAW_LINE);
   SetIndexBuffer(1, FirstSingnalbuffer);
   IndicatorShortName(short_name);
   SetIndexLabel(0,short_name);
   SetIndexLabel(1,"Signal");
   draw1=KPer+Slow;
   draw2=draw1+DPer;
   SetIndexDrawBegin(0,draw1);
   SetIndexDrawBegin(1,draw2);
   return(INIT_SUCCEEDED);
  }
int OnCalculate(const int rates_total,
                const int prev_calculated,
                const datetime &time[],
                const double &open[],
                const double &high[],
                const double &low[],
                const double &close[],
                const long &tick_volume[],
                const long &volume[],
                const int &spread[])
 {
  int    i,k,p;
  if(rates_total<=KPer+DPer+Slow)
     return(0);
  ArraySetAsSeries(CentralBuffer,false);
  ArraySetAsSeries(FirstSingnalbuffer,false);
  ArraySetAsSeries(HBuffer,false);
  ArraySetAsSeries(LBuffer,false);
  ArraySetAsSeries(low,false);
  ArraySetAsSeries(high,false);
  ArraySetAsSeries(close,false);
  p=KPer-1;
  if(p+1<prev_calculated)
     p=prev_calculated-2;
  else
    {
     for(i=0; i<p; i++)
       {
        LBuffer[i]=0.0;
```

```
      HBuffer[i]=0.0;
     }
   }
  for(i=p; i<rates_total && !IsStopped(); i++)
   {
    double devmin=1000000;
    double devmax=-1000000;
    for(k=i-KPer+1; k<=i; k++)
     {
      if(devmin>low[k])
        devmin=low[k];
      if(devmax<high[k])
        devmax=high[k];
     }
    LBuffer[i]=devmin;
    HBuffer[i]=devmax;
   }
  p=KPer-1+Slow-1;
  if(p+1<prev_calculated)
    p=prev_calculated-2;
  else
   {
    for(i=0; i<p; i++){
      CentralBuffer[i]=0.0;
    }
   }
  for(i=p; i<rates_total && !IsStopped(); i++)
   {
    double sumlow=0.0;
    double sumhigh=0.0;
    for(k=(i-Slow+1); k<=i; k++)
     {
      sumlow +=(close[k]-LBuffer[k]);
      sumhigh+=(HBuffer[k]-LBuffer[k]);
     }
    if(sh==0.0)
      CentralBuffer[i]=100.0;
    else
      CentralBuffer[i]=sumlow/sh*100.0;
   }
  p=DPer-1;
  if(p+1<prev_calculated)
    p=prev_calculated-2;
  else
   {
    for(i=0; i<p; i++)
      FirstSingnalbuffer[i]=0.0;
   }
  for(i=p; i<rates_total && !IsStopped(); i++)
   {
    double sum=0.0;
    for(k=0; k<DPer; k++)
      sum+=CentralBuffer[i-k];
    FirstSingnalbuffer[i]=sum/DPer;
   }
  return(rates_total);
 }
```

RSI

```
//+------------------------------------------------------------------+
//|                                                RSIModify.mq4 |
//|                                               Grigorenko V. |
//|                                                 bsu.edu.ru |
//+------------------------------------------------------------------+
#property copyright "Grigorenko V."
#property strict
#property indicator_separate_window
#property indicator_minimum 0
#property indicator_maximum 100
#property indicator_color1 DarkGreen
#property indicator_color2 Crimson
#property indicator_color3 Green
//---- input parameters
extern bool ManyIndicators = true;
extern int BarsForSearch = 1000000;
extern string ss = "//--------------------PJ--------------------//";
extern int OverboughtLevel=99;
extern int OversoldLevel=1;
extern int PeriodPJ = 2;
```

```
extern ENUM_TIMEFRAMES TimeFramePJ = PERIOD_CURRENT;
extern string sss = "//-------------------RSI--------------------//";
extern int PeriodRSI=14;
extern ENUM_TIMEFRAMES TimeframesRSI = PERIOD_M5;
extern bool OffDiv = false;
extern string s = "//-------------------Colors--------------------//";
extern color ColorSeparator = clrGreen;
extern color ColorTrendLine = clrRed;
extern color ColorForCheckTrendLne = clrBlue;
extern string ssss = "//-------------------Lines--------------------//";
extern bool DeleteLine = true;
extern int MissBlock = 2;
extern int CountBarsStopLine = 1000;
extern bool ClearNonCheckLine = false;
extern bool OnSeparators = true;
//extern bool ShiftLeft = false;
extern string sssssss = "//-------------------Arrow--------------------//";
extern color ColorArrow = clrRed;
extern int DistanceArrows  = 15;
extern int DistanceCheckArrows = 20;
extern string sssss = "//-------------------ColorRsi--------------------//";
extern color M1 = clrAqua;
extern color M5 = clrBlue;
extern color M15 = clrAzure;
extern color M30 = clrAquamarine;
extern color H1 = clrCoral;
extern color H4 = clrBlue;
extern color D1 = clrBlue;
extern color W1 = clrBlue;
extern color MN = clrBlue;

int cntBars = 0;
int numberIndicator = 0;
//---- indicator buffers
double Buffer1[];
double Buffer2[];
double Buffer3[];
datetime TimeArrayRSI[];
datetime TimeArrayPJ[];
int counted_bars=0;
int last_y = 0;
bool RayCheckTrandLine = false;
bool IsCurrentDay = false;
double rsi_main = 0;
int numberChart = 0;
string name_arrow = "trendArrow = ";
string name_trendLine = "trendLine = ";
string name_check_arrow = "checkArrow =";
int init()
{
  string indi_name ="MFT(" + PeriodRSI + ")";
  SetIndexStyle(0,DRAW_LINE,STYLE_SOLID,2);
  SetIndexStyle(1,DRAW_HISTOGRAM,STYLE_SOLID,2);
  SetIndexStyle(2,DRAW_HISTOGRAM,STYLE_SOLID,2);
  SetIndexBuffer(0,Buffer1);
  SetIndexBuffer(1,Buffer2);
  SetIndexBuffer(2,Buffer3);
  IndicatorShortName(indi_name);
  SetIndexLabel(0,"Rsi " + TimeframesRSI );
  SetIndexLabel(1,"PJDown " + PeriodPJ);
  SetIndexLabel(2,"PJUp " + PeriodPJ);
  SetIndexDrawBegin(0,PeriodRSI);
  SetIndexDrawBegin(1,PeriodPJ);
  SetIndexDrawBegin(2,PeriodPJ);
  if(ManyIndicators){
    numberChart = rand();
  }else{numberChart = 0;}
  if(DeleteLine){
    for(int i = 0; i < 1000000; i++)
      ObjectDelete(numberChart + "Trend = " + i);
    for(int i = 0; i < 1000000; i++)
      ObjectDelete(numberChart + name_trendLine + i);
    for(int i = 0; i < 1000000; i++)
      ObjectDelete(numberChart + name_arrow + i);
    numberIndicator = 0;
  }  return(0);
}
int start()
{
```

```
  ArrayCopySeries(TimeArrayRSI,MODE_TIME,Symbol(),TimeframesRSI);
  ArrayCopySeries(TimeArrayPJ,MODE_TIME,Symbol(), TimeFramePJ);
  int limit=Bars-counted_bars - PeriodRSI;
  if(limit > BarsForSearch){
    limit = BarsForSearch;
  }
  int limit_y = iBars(_Symbol, TimeframesRSI) - PeriodRSI;
  int limit_pj = iBars(_Symbol, TimeFramePJ) - PeriodPJ;
  for(int i=0,y=0,pj;i<limit;i++)
  {
    if(Time[i] == TimeArrayRSI[y]){
      if(OnSeparators){
        string temp_name = numberChart + "Trend = " + numberIndicator;
        ObjectCreate(0, temp_name,OBJ_VLINE, 0, Time[i], 0);
        ObjectSet(temp_name, OBJPROP_STYLE, STYLE_DOT);
        ObjectSet(temp_name, OBJPROP_COLOR, ColorSeparator);
        ObjectSet(temp_name, OBJPROP_BACK, true);
      }
      numberIndicator++;
    }
    if(pj < limit_pj && Time[i] < TimeArrayPJ[pj]) pj++;
    double rsi = iRSI(NULL, TimeFramePJ,PeriodPJ,PRICE_CLOSE, pj);
    if(rsi > OverboughtLevel){
      Buffer3[i+1] = 21;
      Buffer2[i+1] = -10;
    }else if(rsi < OversoldLevel){
      Buffer2[i+1] = 21;
      Buffer3[i+1] = -10;
    }else{
      Buffer3[i+1] = -10;
      Buffer2[i+1] = -10;
    }     if (y < limit_y && Time[i] < TimeArrayRSI[y]) y++;
    Buffer1[i] = iRSI(NULL,TimeframesRSI,PeriodRSI,PRICE_CLOSE,y);
    counted_bars++;  }
  IsCurrentDay = false;
  return(0);
}void ChekLastPoint(int x){
  double price = Close[x+1];
  double rsi = Buffer1[x+1];
  int index_y = last_y - MissBlock - 1;
  int stop_line = 0;

  if( x - CountBarsStopLine >= 0){
    stop_line = x - CountBarsStopLine;
  }
  for(int index=x;index>= stop_line;index--)
  {
    if(index == stop_line){
      if(!ClearNonCheckLine){
        ObjectSet(numberChart + name_trendLine + numberIndicator, OBJPROP_TIME2, Time[index+1]);
        ObjectSet(numberChart + name_trendLine + numberIndicator, OBJPROP_RAY, RayCheckTrandLine);
      }else{
        ObjectDelete(numberChart + name_trendLine + numberIndicator);}
      break;}
    if(index_y >= 0 && Time[index] == TimeArrayRSI[index_y]){
      double info_price = price - Close[index+1];
      double info_buffer2 = rsi - Buffer1[index+1];

      if((info_price > 0 && info_buffer2 < 0) || (info_price < 0 && info_buffer2 > 0)){

        ObjectSet(numberChart + name_trendLine + numberIndicator, OBJPROP_TIME2, Time[index+1]);
        ObjectSet(numberChart + name_trendLine + numberIndicator, OBJPROP_PRICE2, Close[index+1]);
        ObjectCreate("sdasdas" + numberIndicator, OBJ_TEXT, 0, Time[index+1], Close[index+1]);
        ObjectSet("sdasdas" + numberIndicator, OBJPROP_TEXT, "AΓA");
        ObjectCreate(0, numberChart + name_check_arrow + numberIndicator, OBJ_ARROW,0, Time[index+1], Close[index+1] -
DistanceCheckArrows * Point);
        ObjectSet(numberChart + name_check_arrow + numberIndicator, OBJPROP_COLOR,CheckTimeFrame());
        ObjectSet(numberChart + name_trendLine + numberIndicator, OBJPROP_RAY, RayCheckTrandLine);
        ObjectSet(numberChart + name_trendLine + numberIndicator, OBJPROP_COLOR, CheckTimeFrame());
        break;
      } else{
        index_y--;
      }
    }else if(index_y < 0){
      if(ClearNonCheckLine)
        ObjectDelete(numberChart + name_trendLine + numberIndicator);
      break;}}}
```

MA

```
#property copyright "Copyright © 2011 Best-metatrader-indicators.com."
#property link      "http://www.best-metatrader-indicators.com"#property indicator_chart_window
#property indicator_buffers 1
#property indicator_color1 DodgerBlue
//---- input parameters

extern string TimeFrameNote="TimeFrame =0 - Current Timeframe, =1 - 1MIN, =5 - 5MIN, =15 - 15MIN, =30 - 30MIN, =60 - 1H, =240 - 4H,
=1440 - D1, =10080 - W1, =43200 - MN1";
extern int TimeFrame=0;
extern int MAPeriod=13;
extern int ma_shift=0;
extern int ma_method=MODE_SMA;
extern int applied_price=PRICE_CLOSE;
//----
double ExtMapBuffer1[];
string Copyright="\xA9 WWW.BEST-METATRADER-INDICATORS.COM";
string MPrefix="FI";
int init()
  {
   string short_name;
//---- indicator line
   SetIndexBuffer(0,ExtMapBuffer1);
   SetIndexStyle(0,DRAW_LINE);
//---- name for DataWindow and indicator subwindow label
   switch(ma_method)
     {
      case 1 : short_name="MTF_EMA("; break;
      case 2 : short_name="MTF_SMMA("; break;
      case 3 : short_name="MTF_LWMA("; break;
      default : short_name="MTF_SMA(";
     }
   switch(TimeFrame)
     {
      case 1 : string TimeFrameStr="Period_M1"; break;
      case 5 : TimeFrameStr="Period_M5"; break;
      case 15 : TimeFrameStr="Period_M15"; break;
      case 30 : TimeFrameStr="Period_M30"; break;
      case 60 : TimeFrameStr="Period_H1"; break;
      case 240 : TimeFrameStr="Period_H4"; break;
      case 1440 : TimeFrameStr="Period_D1"; break;
      case 10080 : TimeFrameStr="Period_W1"; break;
      case 43200 : TimeFrameStr="Period_MN1"; break;
      default : TimeFrameStr="Current Timeframe";
     }
   IndicatorShortName(short_name+MAPeriod+") "+TimeFrameStr);
   DL("001", Copyright, 5, 20,Gold,"Arial",10,0);
   return(0);
  }int deinit()
  {   ClearObjects();
   return(0);
  }int start()
  {
   datetime TimeArray[];
   int        i,shift,limit,y=0,counted_bars=IndicatorCounted();      ArrayCopySeries(TimeArray,MODE_TIME,Symbol(),TimeFrame);      limit=Bars-
counted_bars;
   for(i=0,y=0;i<limit;i++)
     {
      if (Time[i]<TimeArray[y]) y++;      ExtMapBuffer1[i]=iMA(NULL,TimeFrame,MAPeriod,ma_shift,ma_method,applied_price,y);
     }   return(0);
  }void DL(string label, string text, int x, int y, color clr, string FontName = "Arial",int FontSize = 12, int typeCorner = 1)

{
   string labelIndicator = MPrefix + label;
   if (ObjectFind(labelIndicator) == -1)
   {
      ObjectCreate(labelIndicator, OBJ_LABEL, 0, 0, 0);
  }
   ObjectSet(labelIndicator, OBJPROP_CORNER, typeCorner);
   ObjectSet(labelIndicator, OBJPROP_XDISTANCE, x);
   ObjectSet(labelIndicator, OBJPROP_YDISTANCE, y);
   ObjectSetText(labelIndicator, text, FontSize, FontName, clr);
}
void ClearObjects()
{
  for(int i=0;i<ObjectsTotal();i++)
  if(StringFind(ObjectName(i),MPrefix)==0) { ObjectDelete(ObjectName(i)); i--; }
}
//+------------------------------------------------------------------+
```

ПРИЛОЖЕНИЕ Г

```
#property strict
int cntBars;
int result;
extern int magic = 125484;
double lots = 0.01;
extern double min_lots = 0.01;
extern double max_lots = 0.03;
extern int offset = 400; //БУ
//---Индикаторы
input string aaaa = "               //-----Индикаторы-----//         ";
input string aaaa2 = "              //-----AO-----//         ";
extern int Fast_period_ao           = 8;
extern int Slow_period_ao           = 34;
extern double More_ao               = 0.5;
//extern double Stop_more_ao          = 12;
extern double Less_ao               = -0.4;
extern double Stop_less_ao          = -4.5;
input string aaaa3 = "              //-----RSI-----//         ";
extern int Period_rsi               = 14;
extern int Rsi_Less                 = 60;
extern int Rsi_More                 = 40;
input string aaaa4 = "              //-----STD-----//         ";
extern ENUM_TIMEFRAMES STD_TimeFrame = 60;
extern int STD_MA_Period            = 38;
extern int STD_Bars                 = 8;
extern int Stoch_Less               = 65;
extern int Stoch_More               = 15;
input string aaaa5 = "              //-----BANDS-----//         ";
extern int Period_bands             = 30;
extern double Deviation_bands       = 2;
extern ENUM_TIMEFRAMES Bands_tf     = PERIOD_H4;
input string aaaa6 = "              //-----MA-----//         ";
extern int  Period_ma_sell          = 200;
extern int  Period_ma_buy           = 40;

/*
input string aaaa7 = "              //-----ATR-----//         ";
extern ENUM_TIMEFRAMES Atr_tf       = PERIOD_D1;
extern int Period_atr               = 14;
*/
//---Индикаторы

struct Fibonacci_Levels
{
  double level;
  double price;
};

struct TradeInfo
{
  int ticket;
  Fibonacci_Levels fibonacci_level[9];
  int level_number;

} orders_info[];



double high_level;
double low_level;


int OnInit()
{

return(INIT_SUCCEEDED);
}
void OnTick()
{
  Tralling();
  OrderClear();

  if(cntBars == Bars){
```

```
    return;
  }
  cntBars = Bars;

  if(SerchHighAndLow()){
    if(FilterForBuy()){
      if(Ask < high_level ){
        result = OrderSend(_Symbol, OP_BUYSTOP, lots, high_level, 0, low_level, 0, "", magic);
        CalculationFibonacci(result, high_level, low_level);
      } else {
        if(StdFilter()) {
          result = OrderSend(_Symbol, OP_BUY, lots, Ask, 0, low_level, 0, "", magic);
          CalculationFibonacci(result, high_level, low_level);
        }
      }
    } else if(FilterForSell()){
      if(Bid > low_level ){
        result = OrderSend(_Symbol, OP_SELLSTOP, lots, low_level, 0, high_level, 0, "", magic);
        CalculationFibonacci(result, low_level, high_level);
      } else {
        if(StdFilter()) {
          result = OrderSend(_Symbol, OP_SELL, lots, Bid, 0, high_level, 0, "", magic);
          CalculationFibonacci(result, low_level, high_level);
        }}}  }
  CheckRange();
  if(DayOfWeek() == 1 && Hour() <= 1){
    CloseOldOrder();
  }
}bool FilterLastOrder(ENUM_ORDER_TYPE order_type)
{
  // Если прошлая сделка такая же и она закрылась в минус, не открываемся
  int i = OrdersHistoryTotal()-1;
  while(OrderSelect(i, SELECT_BY_POS, MODE_HISTORY)){
    if( OrderMagicNumber() == magic ){
      if( (OrderType() == order_type) && (OrderProfit() < 0) ){
        return false;
      }else {
        return true;}}
    i--;}
  return true;}
bool FilterForBuy()
{
  double ao_1 = iCustom(_Symbol, _Period, "AO", Fast_period_ao, Slow_period_ao, 0, 1);
  double stoh = iStochastic(_Symbol, _Period, 5,3,3, MODE_SMA, 0, 0, 1);
  double stoh_signal = iStochastic(_Symbol, _Period, 5,3,3, MODE_SMA, 0, 1, 1);
  double rsi = iRSI(_Symbol, _Period, Period_rsi, PRICE_CLOSE, 1);
  double ma_1 = iMA(_Symbol, PERIOD_H1, Period_ma_buy, 0, MODE_SMA, PRICE_CLOSE, 1);
  double ma_2 = iMA(_Symbol, PERIOD_H4, Period_ma_buy, 0, MODE_SMA, PRICE_CLOSE, 1);
  double ma_3 = iMA(_Symbol, PERIOD_D1, Period_ma_buy, 0, MODE_SMA, PRICE_CLOSE, 1);

  lots = min_lots;

  if( (ao_1 > More_ao) /*&& (ao_1 < Stop_more_ao)*/ && (stoh < Stoch_Less) && (rsi < Rsi_Less) && (FilterLastOrder(OP_BUY)) &&
FilterOrderModify(OP_SELL, -1)){
    if( (ma_1 > iLow(_Symbol, PERIOD_H1, 1)) && (ma_2 < iLow(_Symbol, PERIOD_H4, 1)) && (ma_3 < iLow(_Symbol, PERIOD_D1, 1)) ){
      lots = max_lots;
    }
    return true;}
  return false;}

bool FilterOrderModify(ENUM_ORDER_TYPE order_type, int lvl_number)
{
  // Если противоположная сделка сделала Modify, тогда не открываемся
  int array_size = ArraySize(orders_info) - 1;
  for(int i = array_size; i >= 0 ; i--){
    if(OrderSelect(orders_info[i].ticket, SELECT_BY_TICKET) && OrderMagicNumber() == magic){
      if( OrderType() == order_type && lvl_number == orders_info[i].level_number){
        return false;
      }}}
  return true;
}
bool FilterForSell()
{
  double ao_1 = iCustom(_Symbol, _Period, "AO", Fast_period_ao, Slow_period_ao, 0, 1);
  double stoh = iStochastic(_Symbol, _Period, 5,3,3, MODE_SMA, 0, 0, 1);
  double stoh_signal = iStochastic(_Symbol, _Period, 5,3,3, MODE_SMA, 0, 1, 1);
  double rsi = iRSI(_Symbol, _Period, Period_rsi, PRICE_CLOSE, 1);
  double ma_1 = iMA(_Symbol, PERIOD_H1, Period_ma_sell, 0, MODE_SMA, PRICE_CLOSE, 1);
  double ma_2 = iMA(_Symbol, PERIOD_H4, Period_ma_sell, 0, MODE_SMA, PRICE_CLOSE, 1);
```

```
  double ma_3 = iMA(_Symbol, PERIOD_D1, Period_ma_sell, 0, MODE_SMA, PRICE_CLOSE, 1);
  lots = min_lots;

  if( (ao_1 < Less_ao) && (ao_1 > Stop_less_ao) && (stoh > Stoch_More) && (rsi > Rsi_More) && (FilterLastOrder(OP_SELL)) &&
FilterOrderModify(OP_BUY, 7) ){
    if( (ma_1 < iHigh(_Symbol, PERIOD_H1, 1)) && (ma_2 > iHigh(_Symbol, PERIOD_H4, 1)) && (ma_3 > iHigh(_Symbol, PERIOD_D1, 1)) ){
      lots = max_lots;
    }
    return true;}
  return false;}
void DrowFibo(double up_level, double down_level)
{
  string a = "fibo" + IntegerToString(rand() + Hour() + rand() + Minute() + Seconds() + rand());
  if(ObjectCreate(a, OBJ_FIBO, 0, Time[3], up_level,Time[2],down_level)){
    result = ObjectSetInteger(0,a,OBJPROP_RAY_RIGHT,0);
  }
}
void CheckRange()
{
  double bands_up = iBands(_Symbol, Bands_tf, Period_bands, Deviation_bands, 0, PRICE_CLOSE, 1, 1);
  double bands_down = iBands(_Symbol, Bands_tf, Period_bands, Deviation_bands, 0, PRICE_CLOSE, 2, 1);

  for(int i = OrdersTotal()-1; i>=0; i--){
    if(OrderSelect(i, SELECT_BY_POS) && (OrderType() == OP_BUYSTOP) && (OrderMagicNumber() == magic)){
      if( ((OrderOpenPrice() - Ask) > (bands_up - bands_down))){
        string name = "vline" + IntegerToString(rand() + rand() + rand());
        ObjectCreate(0, name, OBJ_VLINE, 0, Time[0], 0);
        result = OrderDelete(OrderTicket(), clrBlack);
      }
    }else if(OrderSelect(i, SELECT_BY_POS) && (OrderType() == OP_SELLSTOP) && (OrderMagicNumber() == magic)){
      if((Bid - OrderOpenPrice()) > (bands_up - bands_down) ){
        string name = "vline" + IntegerToString(rand() + rand() + rand());
        ObjectCreate(0, name, OBJ_VLINE, 0, Time[0], 0);
        result = OrderDelete(OrderTicket(), clrBlack);}}}}

bool StdFilter()
{
  double std = iStdDev(_Symbol, STD_TimeFrame, STD_MA_Period, 0, MODE_SMA, PRICE_CLOSE, 1);
  for(int i = 2; i <= STD_Bars; i++) {
    std += iStdDev(_Symbol, STD_TimeFrame, STD_MA_Period, 0, MODE_SMA, PRICE_CLOSE, i);
  }
  double avg = std/STD_Bars;
  std = iStdDev(_Symbol, STD_TimeFrame, STD_MA_Period, 0, MODE_SMA, PRICE_CLOSE, 1);
  if(std > avg) {
    return true;
  }
  return false;
}
void Tralling()
{
  int array_size = ArraySize(orders_info) - 1;
  for(int i = array_size; i >= 0 ; i--){
    if(OrderSelect(orders_info[i].ticket, SELECT_BY_TICKET) && (OrderMagicNumber() == magic)){
      if(OrderType() == OP_BUY){
        if(orders_info[i].level_number == 6){
          RefreshRates();
          if(orders_info[i].fibonacci_level[6].price < Bid){
            if(OrderModify(OrderTicket(), OrderOpenPrice(), orders_info[i].fibonacci_level[0].price + offset * Point, 0,0,clrAqua)){
              orders_info[i].level_number = 7;
            }
          }
        }else if(orders_info[i].level_number == 7){
          RefreshRates();
          if(orders_info[i].fibonacci_level[7].price < Bid){
            RefreshRates();
            if( orders_info[i].fibonacci_level[8].price > Bid){
              if(OrderModify(OrderTicket(),                    OrderOpenPrice(),                    orders_info[i].fibonacci_level[6].price,
orders_info[i].fibonacci_level[8].price,0,clrBeige)){
                orders_info[i].level_number = -1;
              }
            }else{
              result = OrderClose(orders_info[i].ticket, lots,Bid,0);
              orders_info[i].level_number  = -1;
            }
          }
        }
      } else if(OrderType() == OP_SELL){
        if(orders_info[i].level_number == 6){
          RefreshRates();
```

```
            if(orders_info[i].fibonacci_level[6].price > Ask){
              if(OrderModify(OrderTicket(), OrderOpenPrice(), orders_info[i].fibonacci_level[0].price - offset * Point, 0,0,clrAqua)){
                orders_info[i].level_number = 7;
              }
            }
          }else if(orders_info[i].level_number == 7){
            RefreshRates();
            if(orders_info[i].fibonacci_level[7].price > Ask){
              RefreshRates();
              if(orders_info[i].fibonacci_level[8].price < Ask){
                if(OrderModify(OrderTicket(),                OrderOpenPrice(),                orders_info[i].fibonacci_level[6].price,
orders_info[i].fibonacci_level[8].price,0,clrBeige)){
                  orders_info[i].level_number  = -1;
                }
              }else{
                result = OrderClose(orders_info[i].ticket,lots, Ask, 0);
                orders_info[i].level_number  = -1;
}}}
void OrderClear()
{
  int array_size = ArraySize(orders_info) - 1;

  for(int i = array_size; i >= 0 ; i--){
    if(OrderSelect(orders_info[i].ticket, SELECT_BY_TICKET) && (OrderMagicNumber() == magic)){
      if(OrderCloseTime() != 0){
        if(i != array_size) {
          orders_info[i].ticket = orders_info[array_size].ticket;
          orders_info[i].fibonacci_level[0].price = orders_info[array_size].fibonacci_level[0].price;
          orders_info[i].fibonacci_level[1].price = orders_info[array_size].fibonacci_level[1].price;
          orders_info[i].fibonacci_level[2].price = orders_info[array_size].fibonacci_level[2].price;
          orders_info[i].fibonacci_level[3].price = orders_info[array_size].fibonacci_level[3].price;
          orders_info[i].fibonacci_level[4].price = orders_info[array_size].fibonacci_level[4].price;
          orders_info[i].fibonacci_level[5].price = orders_info[array_size].fibonacci_level[5].price;
          orders_info[i].fibonacci_level[6].price = orders_info[array_size].fibonacci_level[6].price;
          orders_info[i].fibonacci_level[7].price = orders_info[array_size].fibonacci_level[7].price;
          orders_info[i].fibonacci_level[8].price = orders_info[array_size].fibonacci_level[8].price;
          orders_info[i].level_number = orders_info[array_size].level_number;
        }
        ArrayResize(orders_info, array_size);
        array_size = ArraySize(orders_info) - 1;
      }}}  }
void CalculationFibonacci(int ticket, double up_level, double down_level)
{
  if(ticket != -1){
    DrowFibo(up_level,down_level);
    int array_size = ArraySize(orders_info);
    ArrayResize(orders_info, array_size + 1);
    orders_info[array_size].fibonacci_level[0].level = 1;
    orders_info[array_size].fibonacci_level[1].level = 0;
    orders_info[array_size].fibonacci_level[2].level = 0.236;
    orders_info[array_size].fibonacci_level[3].level = 0.382;
    orders_info[array_size].fibonacci_level[4].level = 0.50;
    orders_info[array_size].fibonacci_level[5].level = 0.618;
    orders_info[array_size].fibonacci_level[6].level = 1.618;
    orders_info[array_size].fibonacci_level[7].level = 2.618;
    orders_info[array_size].fibonacci_level[8].level = 4.236;
    orders_info[array_size].level_number = 6;
    orders_info[array_size].ticket = ticket;
    orders_info[array_size].fibonacci_level[0].price = up_level;
    orders_info[array_size].fibonacci_level[1].price = down_level;
    for(int i = 2; i < 9; i++){
      orders_info[array_size].fibonacci_level[i].price = (up_level - down_level)*orders_info[array_size].fibonacci_level[i].level + down_level;
      orders_info[array_size].fibonacci_level[i].price = NormalizeDouble(orders_info[array_size].fibonacci_level[i].price, Digits);
   }}}
bool SerchHighAndLow()
{
  static datetime start_time, end_time;
  static bool first = false;
  double ao_1 = iCustom(_Symbol, _Period, "AO", Fast_period_ao, Slow_period_ao, 0, 1);
  static int k = 0;
  if( (StartTimeDown() != WRONG_VALUE) || (StartTimeUp() != WRONG_VALUE)){
    if(StartTimeDown() != WRONG_VALUE){
      start_time = StartTimeDown();
      first = true;
      end_time = -1;
    } else {
      start_time = StartTimeUp();
      first = true;
      end_time = -1;
```

```
      }
    } else if( ((EndTimeDown() != WRONG_VALUE) || (EndTimeUp() != WRONG_VALUE)) && first ){
      if(EndTimeDown() != WRONG_VALUE){
        end_time = EndTimeDown();
        first = false;
        high_level = iHigh(_Symbol, _Period, iHighest(_Symbol, _Period, MODE_HIGH, iBarShift(_Symbol, _Period,start_time), iBarShift(_Symbol,
_Period,end_time)));
        low_level = iLow(_Symbol, _Period, iLowest(_Symbol, _Period, MODE_LOW, iBarShift(_Symbol, _Period,start_time), iBarShift(_Symbol,
_Period,end_time)));
        return true;
      } else {
        end_time = EndTimeUp();
        first = false;
        high_level = iHigh(_Symbol, _Period, iHighest(_Symbol, _Period, MODE_HIGH, iBarShift(_Symbol, _Period,start_time), iBarShift(_Symbol,
_Period,end_time)));
        low_level = iLow(_Symbol, _Period, iLowest(_Symbol, _Period, MODE_LOW, iBarShift(_Symbol, _Period,start_time), iBarShift(_Symbol,
_Period,end_time)));
        return true;
      }
    }
    return false;
}
datetime EndTimeDown()
{
    datetime end_time = -1;
    double ao_1 = iCustom(_Symbol, _Period, "AO", Fast_period_ao,Slow_period_ao, 0, 1);
    double ao_2 = iCustom(_Symbol, _Period, "AO", Fast_period_ao, Slow_period_ao, 0, 2);
    double ao_3 = iCustom(_Symbol, _Period, "AO", Fast_period_ao, Slow_period_ao, 0, 3);
    double ao_4 = iCustom(_Symbol, _Period, "AO", Fast_period_ao, Slow_period_ao, 0, 4);
    if( (ao_1 < 0) && (ao_3 < 0) ){
      if( (ao_3 < ao_2) && (ao_2 < ao_1) ){
        end_time = Time[1];
      } }
    return end_time;}
datetime EndTimeUp()
{
    datetime end_time = -1;
    double ao_1 = iCustom(_Symbol, _Period, "AO", Fast_period_ao,Slow_period_ao, 0, 1);
    double ao_2 = iCustom(_Symbol, _Period, "AO", Fast_period_ao, Slow_period_ao, 0, 2);
    double ao_3 = iCustom(_Symbol, _Period, "AO", Fast_period_ao, Slow_period_ao, 0, 3);
    double ao_4 = iCustom(_Symbol, _Period, "AO", Fast_period_ao, Slow_period_ao, 0, 4);
    if( (ao_1 > 0) && (ao_3 > 0) ){
      if( (ao_3 > ao_2) && (ao_2 > ao_1) ){
        end_time = Time[1];}}
    return end_time;
}
datetime StartTimeDown()
{
    datetime start_time = -1;
    double ao_1 = iCustom(_Symbol, _Period, "AO", Fast_period_ao,Slow_period_ao, 0, 1);
    double ao_2 = iCustom(_Symbol, _Period, "AO", Fast_period_ao, Slow_period_ao, 0, 2);
    if( (ao_2 >= 0) && (ao_1 < 0) ){
      start_time = Time[1];
    }
    return start_time;
}
datetime StartTimeUp()
{
    datetime start_time = -1;

    double ao_1 = iCustom(_Symbol, _Period, "AO", Fast_period_ao,Slow_period_ao, 0, 1);
    double ao_2 = iCustom(_Symbol, _Period, "AO", Fast_period_ao, Slow_period_ao, 0, 2);

    if( (ao_2 <= 0) && (ao_1 > 0) ){
      start_time = Time[1];
    }
    return start_time;
}
void CloseOldOrder()
{
    for (int i=OrdersTotal()-1;i>=0;i--){
      if(OrderSelect(i, SELECT_BY_POS) && (OrderType()==OP_SELLSTOP) && (OrderMagicNumber() == magic)){
        result = OrderDelete(OrderTicket(),clrIndigo);
      }
      else if(OrderType()==OP_BUYSTOP && (OrderMagicNumber() == magic)){
        result = OrderDelete(OrderTicket(),clrIndigo);
      }
    }
}
```