УДК 004.4'2 + 004.032.26

КОНЦЕПЦИЯ И АРХИТЕКТУРА ПРОГРАММНОГО НЕЙРОКОМПЬЮТЕРА SNC

Гриценко В.И., Мисуно И.С., Рачковский Д.А., Ревунова Е.Г., Слипченко С.В., Соколов А.М.

Проанализирована современная ситуация в области нейрокомпьютинга. Сформированы требования к архитектуре и реализации программного нейрокомпьютера. Приведено описание разработанного согласно этим требованиям программного нейрокомпьютера SNC (Software NeuroComputer), обеспечивающего пользователю создание произвольных алгоритмов путем их визуального конфигурирования.

THE CONCEPT AND ARCHITECTURE OF SOFTWARE NEUROCOMPUTER SNC

GritsenkoV.I., Misuno I.S., Rachkovskij D.A., Revunova E.G., Slipchenko S.V., Sokolov A.M.

Current situation in neurocomputing is investigated. Architectural and implementation requirements to modern software neurocomputers are formulated. A description of Software Neurocomputer SNC, implemented accordingly to the formulated requirements is presented. The neurocomputer described allows users to build arbitrary complex data-processing algorithms in visual configuration mode.

КОНЦЕПЦІЯ І АРХІТЕКТУРА ПРОГРАМНОГО НЕЙРОКОМП'ЮТЕРА SNC

Гріценко В.І., Місуно І.С., Рачковський Д.А., Ревунова Е.Г., Сліпченко С.В., Соколов А.М.

Проаналізована сучасна ситуація в області нейрокомп'ютинга. Сформульовані вимоги до архітектури і реалізації програмного нейрокомп'ютера. Наведений опис програмного нейрокомп'ютера SNC (Software NeuroComputer), що був розроблений згідно цих вимог, і який забезпечує користувачу створення довільних алгоритмів шляхом їх візуального конфігурування.

КОНЦЕПЦИЯ И АРХИТЕКТУРА ПРОГРАММНОГО НЕЙРОКОМПЬЮТЕРА SNC

Гриценко В.И., Мисуно И.С., Рачковский Д.А., Ревунова Е.Г., Слипченко С.В., Соколов А.М.

ВВЕДЕНИЕ

С конца 80-х годов наблюдается динамичный рост объема разработок и практических систем, использующих нейронные сети для обработки информации. Различного рода реализации нейросетевых моделей получили название *нейрокомпьютеров* (НК) [1], см также раздел 1.

Применение нейроподобных (или просто нейронных) сетей (НС) в разных задачах обработки информации позволяет перейти от поиска правил решения задач к обучению сети на примерах. НС используются при решении задач классификации, распознавания, поиска нужной информации (например, изображения, акустических сигналов, или текстов) [2]. Таким образом, *нейросетевые информационные технологии* (технологии обработки информации с использованием нейроподобных сетей*)* заняли важное место в спектре современных информационных технологий.

Средством разработки, реализации, и применения нейросетевых информационных технологий служат нейрокомпьютеры. Современный нейрокомпьютер (НК) – это пакет программ, реализующий нейросетевые алгоритмы. В качестве платформы для реализации нейрокомпьютера используются обычные компьютеры, чаще всего персональные, в состав которых могут быть включены специальные аппаратные средства для ускорения работы.

При разработке и реализации описываемого НК были проанализированы аналогичные системы, учтены их достоинства и недостатки. В результате создан нейрокомпьютер SNC (Software NeuroComputer), который является программной системой с модульной архитектурой и графическим интерфейсом пользователя. Система реализована на базе современных технологий программирования и позволяет конструировать произвольные конфигурации обработки данных, используя расширяемый набор обрабатывающих блоков.

Нейрокомпьютер SNC предназначен для разработки и исследования новых алгоритмов и технологий обработки информации, а также для создания и использования приложений для решения широкого спектра задач. Визуальное конфигурирование позволяет пользователям с минимальными навыками программирования применять SNC для работы с существующими и создания собственных алгоритмов обработки данных. Гибкость и модульность программной архитектуры SNC позволяет программистам наращивать набор программных модулей, непосредственно работающих с данными, без изменения системной архитектуры SNC. Кроме того, SNC может использоваться для ознакомления неспециалистов с подходами и алгоритмами интеллектуальной обработки данных.

1 КОНЦЕПЦИЯ СОВРЕМЕННОГО НЕЙРОКОМПЬЮТЕРА

В зависимости от способа реализации нейронных сетей, выделяют 4 уровня нейрокомпьютеров [3]:

- Уровень 0: Теоретико-алгоритмический. Описание различных нейросетевых моделей.
- Уровень 1: Программный. Программная реализация различных нейросетевых моделей на обычных компьютерах.
- Уровень 2: Программно-аппаратный. Сопроцессоры для ускорения моделирования нейросетей.
- Уровень 3: Аппаратный. Аппаратные реализации моделей нейронных сетей.

Специфика нейросетевых операций, а также параллелизм моделей нейросетей и соответствующих алгоритмов, подталкивают к созданию специализированных вычислительных устройств для их реализации. Таким образом строились «традиционные» нейрокомпьютеры [4], соответствующие уровням 2 и 3. Аппаратные и программно-аппаратные устройства широко разрабатывались в конце 80-х и начале 90-х. Их разработка в то время была достаточно обоснованной, так как персональные компьютеры не обеспечивали адекватной вычислительной мощности и объема памяти для реализации нейросетевых алгоритмов.

В основном, нейрокомпьютеры создавались как сопроцессоры с SIMD архитектурой для выполнения векторных операций, например, ANZA Plus (фирма HNC) или DELTA II (фирма SAIC). Ряд нейрокомпьютеров был разработан в это время и в институтах бывшего СССР, в том числе в Кибцентре им. В.М.Глушкова. В начале 90-х годов совместно с японской фирмой WACOM в Кибцентре были созданы многоплатные нейрокомпьютеры В-512 и С-2048 [5] [6] [7]. Несколько позже был разработан одноплатный нейрокомпьютер [8], [9]. Такие нейрокомпьютеры представляли собой модули, подключаемые к персональному компьютеру (ПК) с целью повышения скорости и предоставления памяти для моделирования нейронных сетей.

Параллельно с созданием нейрокомпьютеров, соответствующих уровням 1 и 2, начался выпуск нейрочипов (нейрокомпьютеры уровня 3), в которых нейросети реализованы аппаратно с помощью аналоговых, цифровых, или гибридных технологий. Некоторые из созданных нейрочипов оказались удобными или были разработаны для специфических массовых приложений, таких как распознавание штрих-кодов, управление пылесосом или стиральной машиной. В таких приложениях решаемые задачи достаточно просты и не требуют больших нейронных сетей, а использование универсальных персональных компьютеров недопустимо из-за их стоимости и габаритов.

Что касается программно-аппаратных нейрокомпьютеров (нейрокомпьютеры уровня 2), практика показала, что они оказались недостаточно эффективными для целей разработки и исследований нейросетевых алгоритмов, а также для применения в конечных продуктах. Малые объемы производства не позволили снизить цену НК. Кроме того, огромные инвестиции в прогресс ПК не позволили нейрокомпьютерам угнаться за темпами роста их производительности и объема памяти, и через несколько лет характеристики большинства нейрокомпьютеров стали доступны пользователям персональных компьютеров. Например, в 1992 году объем памяти ПК в Украине не превышал 1 Мб, в Японии - 4 Мб, при наличии отдельных ПК для спецприменений с объемом памяти 12Мб. Американские нейрокомпьютеры ANZA Plus и DELTA II, а также НК, созданные в Киевском Кибцентре, имели объем памяти до 12 Мб уже в 1988-1989 году [1]. НК В-512 (1992 год) и С-2048 (1992-1993 годы) имели объемы памяти 64 и 256МБ. Такой объем памяти стал обычным для ПК в 1998-2000 годах.

Недостатки традиционных нейрокомпьютеров, по сравнению с ПК, связаны также с особенностями их использования в процессе научных исследований, а также для разработки и использования нейросетевых информационных технологий. Практика использования НК в этих областях показывает, что скорость выполнения нейросетевых операций, как таковая, не определяет эффективность применения НК, а также временные и стоимостные затраты на разработку и исследование нейросетевых алгоритмов. Более важными факторами являются вычислительная сложность применяемых алгоритмов, эффективность представления данных, детали реализации алгоритмов, простота и удобство процесса разработки новых приложений и взаимодействия с пользователем, эффективность адаптации новых информационных технологий к конкретным приложениям, и др. По этим причинам, активность на рынке традиционных нейрокомпьютеров значительно сократилась.

В последние годы получили широкое распространение специализированные пакеты нейросетевого моделирования, которые позволяют использовать различные

нейросетевые алгоритмы как для из изучения, так и для работы с приложениями. Такие пакеты представляют собой нейрокомпьютеры уровня 1 [3], и известны как «программные нейрокомпьютеры», «программы-нейроимитаторы», «нейропакеты», «программы моделирования нейронных сетей». К наиболее распространенным нейропакетам (по данным журнала «Нейрокомпьютер», книги «Искусственные нейронные сети» [10]) относятся NeuroSolutions, NeuralWorks Professional, Process Advisor, NeuroShell32, BrainMaker, MPIL, Braincel, Excel Neural Package, Fuzzy Logic Toolbox [10], и др. В Институте математических машин и систем разработаны модульный нейрокомпьютер MNN-CAD [11] и многофункциональный нейрокомпьютер NeuroLand [12]. Кроме того, нейросетевые модули включены в широко известные математические пакеты общего назначения (MatLab [13], Statistica [14], Maple [15], Mathematica [16]). Эти разработки опираются на максимальное использование возможностей современных ПК.

Итак, современный нейрокомпьютер представляет собой средство для эффективной разработки, реализации и применения нейросетевых информационных технологий. Очевидно, цели обработки информации с помощью нейросетевых информационных технологий могут быть различными – от разработки и исследования нейросетевых архитектур до разработки и применения их в приложениях. Приведенное определение нейрокомпьютера делает упор не на способе его реализации, а на предоставлении пользователю эффективной среды для разработки нейросетевых информационных технологий, их эффективной реализации и применения.

Модульная архитектура и интерфейс SNC предоставляют возможности для упрощения процесса разработки и применения нейросетевых информационных технологий и алгоритмов, а также оптимизации на алгоритмическом и программном уровнях.

Упрощение процесса разработки и использования достигается за счет стандартизации программных модулей (технология COM [17]); модульной компонентно-ориентированной программной архитектуры [18]; предоставления удобных средств создания и включения в систему новых модулей; стандартизации внутренних форматов данных; обширной библиотеки реализованных модулей; удобного графического интерфейса пользователя; наличия дополнительных средств тестирования, статистической обработки, создания отчетов, и т.п.

Оптимизация достигается путем использования алгоритмов с достаточно низкой вычислительной сложностью, "быстрыми" процедурами обучения и функционирования; оптимизации потоков данных; эффективной реализации алгоритмов, включая оптимизацию использования памяти, программных циклов и т.п.; учета особенностей архитектуры и использования современных ПК и операционных систем; использования в качестве языка программирования C++ и возможностей оптимизирующего компилятора VC 6.0.

Нейрокомпьютер SNC предназначен для разработки и исследования новых алгоритмов и технологий обработки информации, а также для создания и использования приложений для решения широкого спектра задач. Визуальное конфигурирование позволяет пользователям с минимальными навыками программирования применять SNC для работы с существующими и создания собственных алгоритмов обработки данных. Гибкость и модульность программной архитектуры SNC позволяет программистам наращивать набор программных модулей, непосредственно работающих с данными, без изменения системной архитектуры SNC.

2 АРХИТЕКТУРА НЕЙРОКОМПЬЮТЕРА SNC

2.1 Общая архитектура SNC

В основу системной архитектуры SNC положены следующие принципы.

- Унификация представления и обработки информации:
- обработка информации осуществляются с помощью унифицированных обрабатывающих блоков, являющихся COM-объектами;
- сохранение и обработка данных проектов и построение отчетов осуществляется унифицированными средствами;
- выполнение серийных запусков осуществляется унифицированными, в том числе визуальными, средствами.
- поддерживается возможность прерывания и возобновления выполнения проекта.
- Модульность архитектуры
- возможные алгоритмы обработки информации создаются комбинированием обрабатывающих блоков в конфигурации (проекты);
- взаимодействие частей системы осуществляется посредством COM-интерфейсов;
- Визуальное конфигурирование:
- задание конфигурации и параметров проекта унифицировано и осуществляются с помощью визуальных средств;

Общий вид программной архитектуры SNC представлен на Рис. 1.

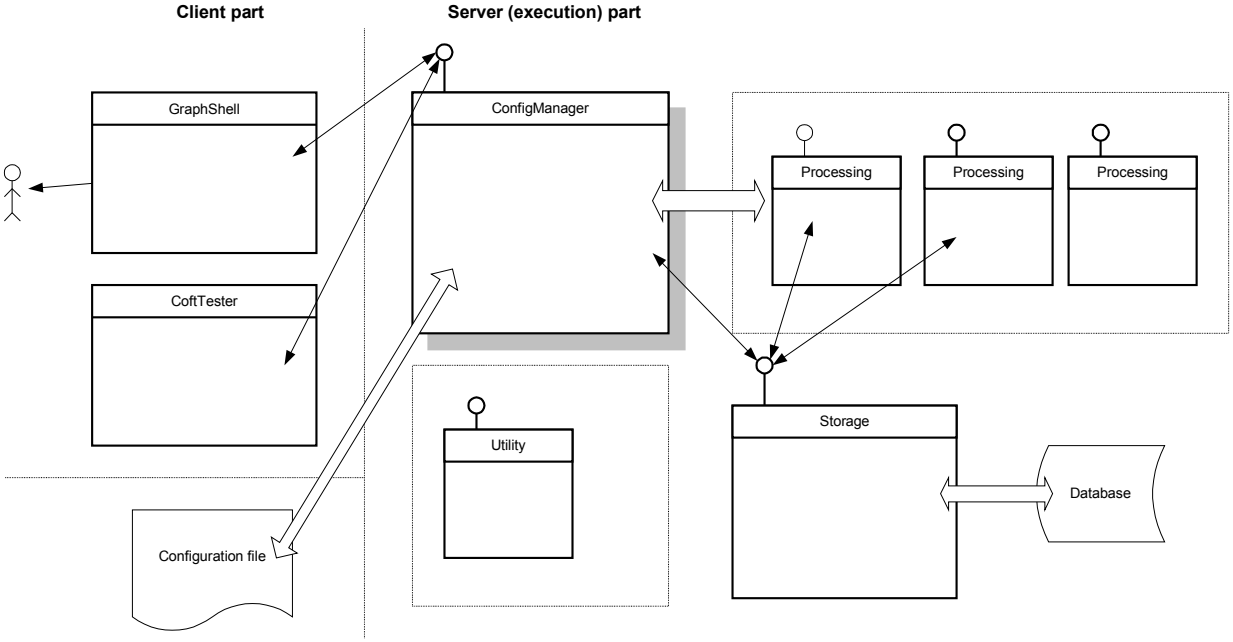

Рис. 1. Общая архитектура SNC

SNC состоит из следующих основных модулей:

- ***ConfigManager*** – управляющая часть (объект) системы. Непосредственно взаимодействует с COM-объектами, реализующими обрабатывающие блоки SNC и осуществляет выполнение конфигурации.
- ***Обрабатывающие блоки – объекты Processings*** – осуществляют обработку данных, предусмотренную конфигурацией проекта.
- ***Storage*** – библиотека объектов работы с базой данных. Позволяет во время выполнения проекта записывать произвольно организованные иерархические данные в реляционную базу данных. Позволяет работать с любыми типами реляционных баз данных, поддерживающих технологию OLEDB [19].
- ***Utility*** – различные служебные утилиты. Классы контейнеров параметров, парсеры вспомогательные классы, и т.д. Организованы в виде COM или C++ классов.
- ***Configuration file*** – файл, в который записывается созданная пользователем конфигурация проекта, включая типы и расположение обрабатывающих и управляющих блоков и связей между ними, параметры блоков и т.д.

- ***GraphShell***, ***ConfTester*** – клиентские приложения, загружающие в ConfigManager конфигурацию и вызывающие её выполнение.
- ***GraphShell*** – графическая оболочка SNC. Это оконное приложение позволяет пользователю создавать и редактировать конфигурацию в режиме САПР, а также выполнять конфигурацию.
- ***ConfTester*** – консольное приложение. Предназначено для выполнения конфигураций в консольном или пакетном режиме. Оба приложения поддерживают загрузку проектов, их трансляцию, запуск, прерывание, вывод информации в файл протокола.

В состав SNC также входят не указанные на этой схеме средства (отдельные приложения) для построения отчетов и для запуска серий проектов, которые являются автономными Windows приложениями.

2.2 Объекты SNC

ConfigManager – это центральный объект системной архитектуры SNC. Через его интерфейсы, реализованные в нем классы (Рис. 2) и методы производятся основные операции SNC:

- создание конфигурации проекта, включая задание параметров и свойств блоков;
- чтение и запись файла конфигурации;
- взаимодействие элементов конфигурации друг с другом и с системными объектами, такими как ConfigManager, Storage, Utility, и т.д.;
- трансляция, выполнение, и прерывание работы конфигурации;
- запись данных в БД.

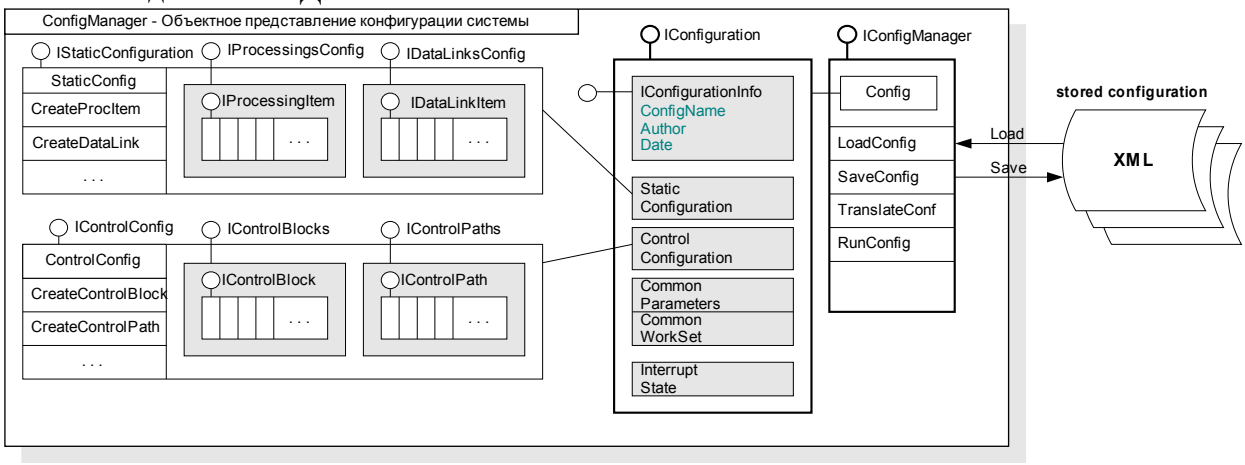

Рис. 2. Структура ConfigManager

На Рис. 2 показана внутренняя структура ConfigManager и реализованные в нем объекты, с помощью которых происходит взаимодействие с программами GraphShell, ConfTester. Доступ к внутренним объектам ConfigManager происходит через COM-интерфейсы.

Обрабатывающие блоки – объекты **Processings** являются обрабатывающими блоками (процессингами и форматами) в визуальной конфигурации SNC. Объекты Processings являются COM-объектами и поддерживают следующие группы интерфейсов SNC:

- Визуального конфигурирования: ISupportsInput, ISupportsOutput, IProcessingInfoEx;
- Состояния: IStateful, коллекции параметров IParametersAware, IWorkingSetAware;
- Обработки данных: IRunnable;
- Поддержки прерывания: IInterruptable, IPersistStream;

Добавление в систему SNC новых обрабатывающих блоков происходит с помощью регистрации COM-класса объекта в соответствующей COM Component Category. Удаление блока из системы происходит путем удаления зарегистрированного класса из Component Category.

Функциональность объектов как обрабатывающих блоков определяется логикой C++ класса, реализующего интерфейс IRunnable. Взаимодействие с объектами системы SNC происходит посредством вызовов их методов через реализованные в них COM-интерфейсы. Кроме того, объекты могут получать извещения о системных событиях.

Передача исходных и обработанных данных между объектами текущей конфигурации происходит через *буфера данных*. Блок может поддерживать несколько буферов данных, при этом различают входные и выходные буфера объекта.

Буфера данных – это внутренние объекты обрабатывающих блоков. Они наследуют базовый COM-интерфейс IDataBuffer и один из нескольких производных от него COM-интерфейсов представления данных. Базовый интерфейс служит для идентификации буфера данных. Интерфейсы представления данных осуществляют взаимодействие и передачу данных между обрабатывающими блоками конфигурации. Набор дополнительных интерфейсов, поддерживаемых буфером данных, определяется спецификой объекта.

Обычно, группы объектов (обрабатывающих блоков), предназначенных для решения одного класса задач, поддерживают общий набор интерфейсов представления данных. Общие интерфейсы представления позволяют эффективно реализовывать различные схемы передачи данных и сохранить возможность проверки типов данных. Последнее невозможно при передаче данных в виде бинарных областей памяти, что является потенциальным источником ошибок.

Файл конфигурации проекта – это файл в формате XML, в который записывается информация о созданной пользователем конфигурации:

- типы и расположение обрабатывающих и управляющих блоков и связей между ними;
- параметры и переменные обрабатывающих блоков (Processing Parameters, WorkSet);
- выражения JScript в управляющих блоках;
- параметры и переменные конфигурации проекта (Common Parameters, Common WorkSet);
- свойства конфигурации ConfigurationInfo: поля Author (автор конфигурации), ConfigurationName (внутреннее имя конфигурации), Date (дата создания), LogFileName имя файла протокола, а также настройки параметров записи в БД (путь к файлу .mdb и значение параметра EXPERIMENT_TYPE).

Совокупность этих данных является полным описанием созданной пользователем конфигурации проекта и позволяет выполнять эту конфигурацию как в графической оболочке GraphShell, так и из консольного приложения ConfTester.

Сохранение параметров и результатов выполнения проектов обеспечивается объектами **Storage**. В качестве хранилища может использоваться любая реляционная база данных, предоставляющая OLEDB-провайдер. При первом запуске проекта в базе данных создается иерархическая схема организации данных проекта. В неё помещаются схема данных конфигурации проекта и схемы данных, поддерживаемых обрабатывающими блоками проекта. Схемы данных, создаваемые в БД, специфичны для каждой конфигурации проекта.

Созданная схема используется при запусках проектов для записи данных в БД. Такими данными являются: сам файл конфигурации, его начальные параметры, параметры обрабатывающих блоков, и данные (результаты работы), получаемые обрабатывающими блоками во время работы.

Логика сохранения результатов и параметров блоков обработки определяется самим блоком. Блоки обработки могут сохранять как финальные, так и промежуточные результаты работы. В дальнейшем сохраненные данные используются при построении отчетов. Извлечение данных, как правило, осуществляется средствами JScript и объектов Storage.

В структуру SNC входит группа вспомогательных COM и C++ классов, под общим названием **Utility**. В состав этих классов входят:

- классы работы с векторными представлениями данных (VectorLib). Реализуют основные типы векторных представлений (LongVector, GradualVector<int|float|double>, SparseVector<int|float|double>) и операции с ними. Это, в основном, побитовые логические операции для бинарных векторов (конъюнкция, дизъюнкция, подсчет единиц, и т.д.), и поэлементные арифметические – для векторов с компонентами - вещественными числами (сложение, умножение, скалярное произведение, и т.д.);
- классы генераторов случайных чисел. Реализуют несколько типов системных и арифметических генераторов случайных чисел (system_default, rnd16, rnd32, MersenneTwister [20]);
- классы ввода/вывода. Реализуют работу со строками, потоками, и т.д.;
- библиотеки работы с параметрами (VarParamParser). Реализуют работу с коллекциями параметров на объектном уровне и в виде текстовых представлений;
- классы работы с XML. Предоставляют удобные классы-оболочки для упрощения работы с COM-объектами модели MSXML;
- библиотека для работы с Windows Script Host. Предоставляет классы-оболочки для работы с механизмами выполнения сценариев;

Все эти классы позволяют использовать библиотеки низкоуровневых функций для упрощения и ускорения разработки новых блоков-обработчиков.

Пользовательский интерфейс и графическая оболочка системы SNC реализованы в виде приложения **GraphShell**, которое предназначено для визуального создания и редактирования проектов. С помощью этого приложения можно производить манипуляции с элементами конфигурации в режиме САПР. Оболочка позволяет размещать на рабочем поле блоки-процессинги, управляющие блоки, задавать связи между ними, задавать параметры и блоков, производить другие действия по визуальному конфигурированию, а также запускать созданные проекты на выполнение и визуализировать результаты работы в графическом и текстовом виде. По сохраненным в БД результатам GraphShell обеспечивает формирование отчетов различной степени детализации.

3 ПОЛЬЗОВАТЕЛЬСКИЕ КОНФИГУРАЦИИ НЕЙРОКОМПЬЮТЕРА

Для решения определенной задачи пользователь должен сконфигурировать нейрокомпьютер – выбрать соответствующие задаче блоки обработки данных и установить функциональные связи между ними. Такой набор взаимосвязанных блоков назван **конфигурацией проекта**.

Создание конфигурации проекта осуществляется с помощью графической оболочки GraphShell, которая работает в режиме САПР, называемом также режимом визуального конфигурирования (Рис. 3). В этом режиме пользователю доступны (в виде списков на панели в левой части окна программы) имеющиеся в системе элементы конфигурации. Это обрабатывающие блоки – "процессинги" (Processings) и "форматы" (Formats), а также управляющие блоки (Control Blocks). В центральной части экрана расположено рабочее поле проекта, на которое пользователь мышкой помещает нужные блоки, устанавливает связи между ними и задает параметры блоков. Общие параметры и переменные проекта (Common Parameters и Common WorkSet) задаются на панелях в правой части экрана. Созданную конфигурацию проекта пользователь записывает в файл конфигурации.

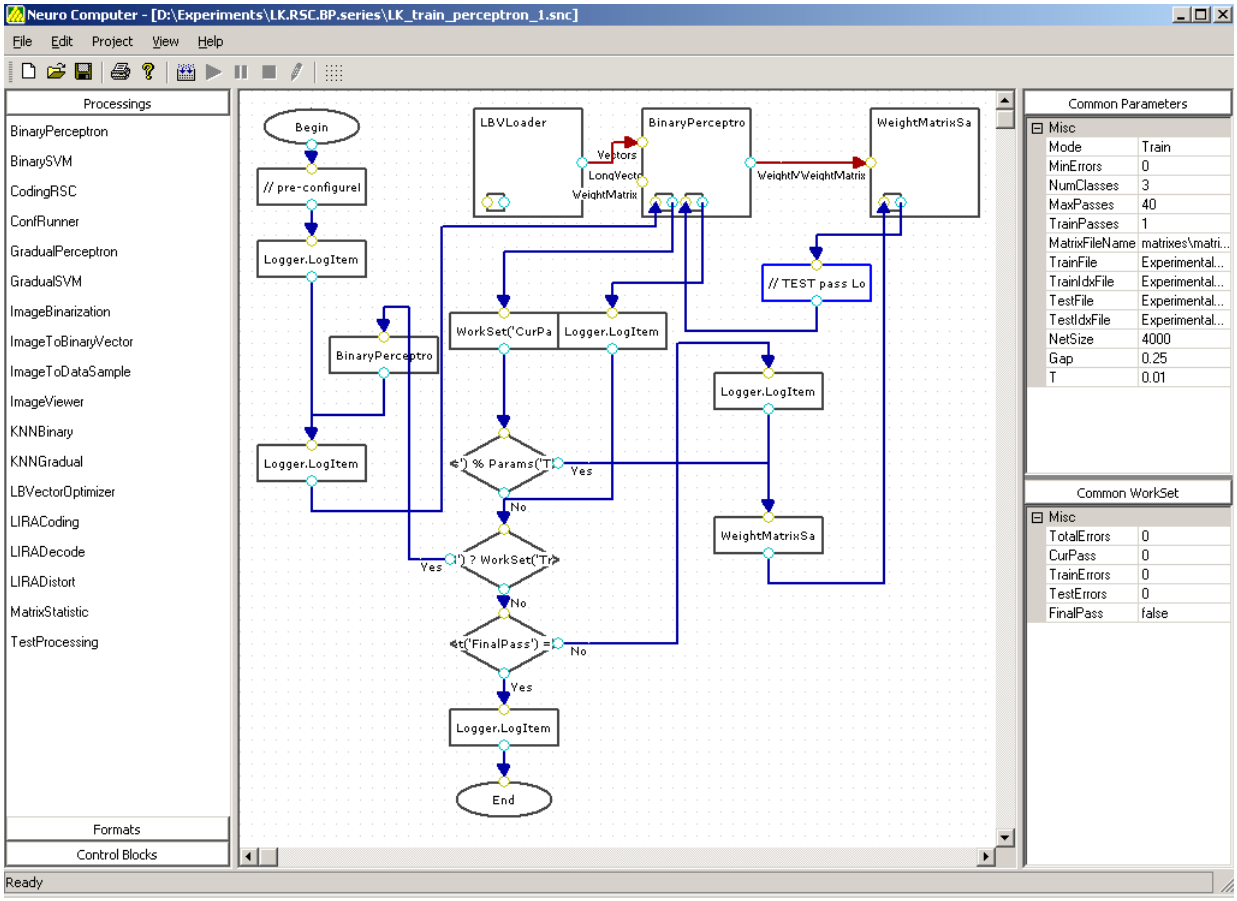

Рис. 3. Графическая оболочка SNC в режиме САПР.

Сконфигурированный проект может быть запущен пользователем на выполнение, во время которого графическая оболочка переключается в режим RunTime (Рис. 4). В процессе выполнения проекта формируются данные, в том числе результаты работы, которые могут выводиться на экран в виде текстовых сообщений, графиков и т.д., и запоминаться на диске. В верхней части экрана сохраняется рабочая область проекта, в которую в режиме выполнения пользователь не может вносить изменения. В центральной и нижней части экрана могут появляться графические и текстовые консоли.

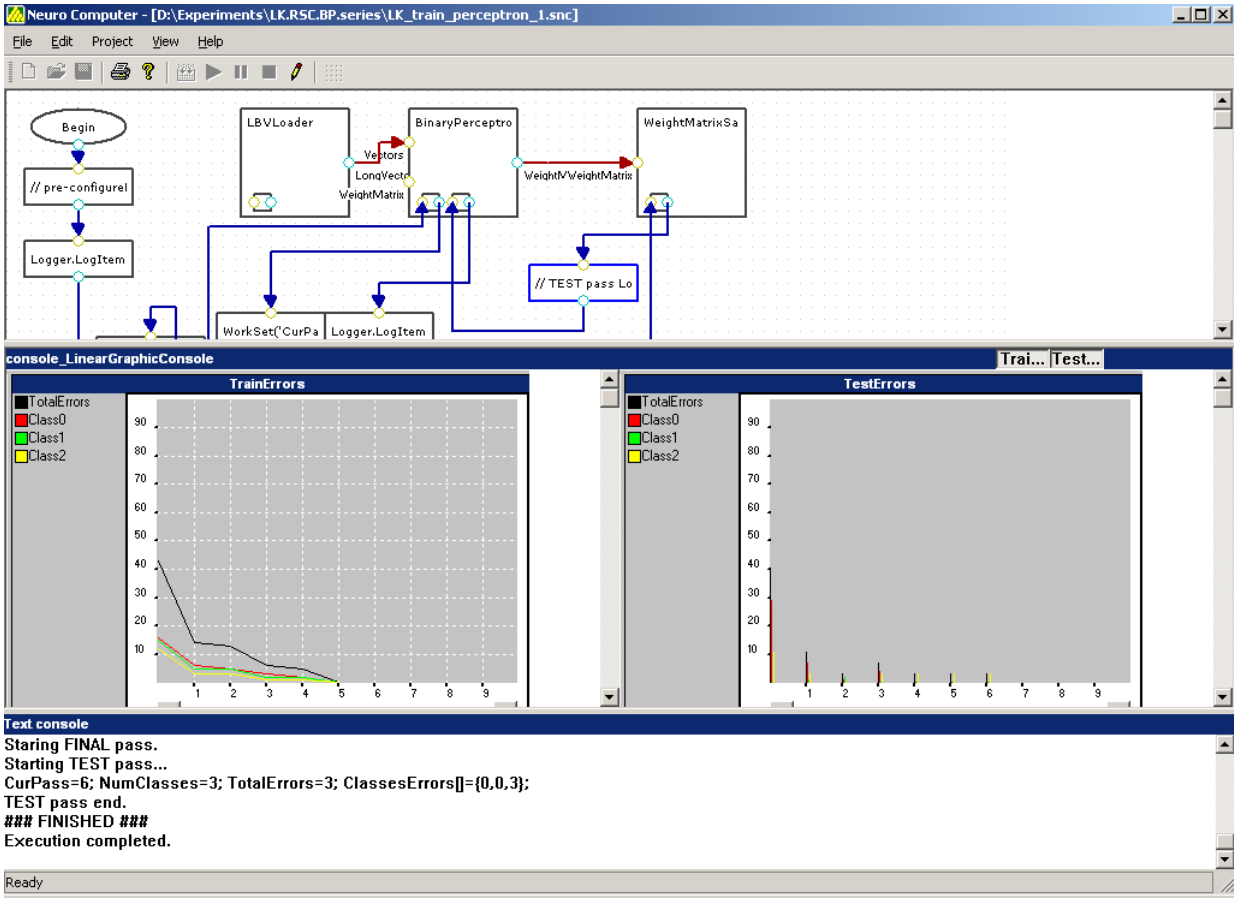

Рис. 4. Графическая оболочка SNC в режиме выполнения проекта (RunTime)

При конфигурировании пользователь представляет алгоритм решаемой задачи с помощью следующих элементов конфигурации: *обрабатывающие блоки, управляющие блоки, связи по данным* и *управляющие связи*.

Обрабатывающие блоки выполняют обработку данных, предусмотренную алгоритмом работы блока и конфигурацией проекта. Блоки имеют внутренние параметры и переменные - Params и WorkSet. При выполнении проекта обрабатывающие блоки также имеют доступ и используют общие параметры и переменные проекта (CommonParameters и CommonWorkSet). Блоки соединяются друг с другом с помощью *связей по данным* – DataLink. Такие связи соединяют входные и выходные буферы данных и не изменяются в процессе выполнения проекта. Каждый выходной буфер может быть соединен связью по данным только с одним входным буфером.

Алгоритм (последовательность) выполнения конфигурации задается пользователем с помощью **управляющих блоков** и связей между ними (ControlBlocks, ControlPaths). Управляющие блоки могут быть самостоятельными элементами конфигурирования (control_Begin, control_End, control_Evaluate, control_If) или принадлежать блоку обработки (control_Processing). В последнем случае вызывается метод соответствующего объекта, реализующего блок обработки.

Управляющие блоки связаны друг с другом с помощью *управляющих связей* – ControlPath, определяющих алгоритм выполнения конфигурации. В отличие от DataLink, управляющие связи могут входить одна в другую и стыковаться в точках входа в управляющие блоки, позволяя задавать алгоритмы произвольной сложности.

Таким образом, *конфигурация проекта* - это совокупность обрабатывающих и управляющих блоков, определенным образом связанных друг с другом для реализации определенного алгоритма обработки данных (информации).

3.1 Свойства и параметры проекта

При конфигурировании пользователь задает общие параметры проекта (Common Parameters, Common WorkSet) на панели справа от рабочей области графической оболочки. Внешний вид панелей ввода общих параметров представлен на Рис. 5. Введенные значения параметров используются в дальнейшей работе конфигурации, при вычислении выражений в управляющих блоках, а также для управления работой участвующих в конфигурации обрабатывающих блоков. Смысл этих параметров определяется алгоритмом выполнения конфигурации, заданном с помощью управляющих блоков и выражений на языке сценариев.

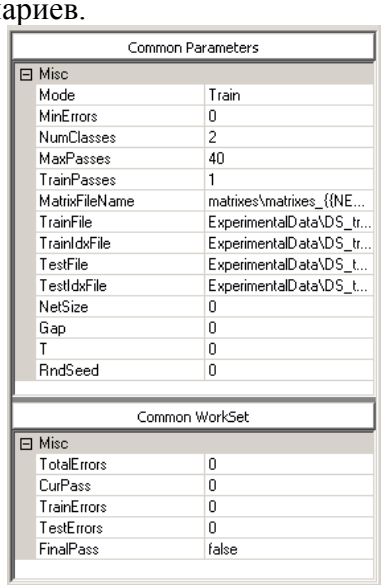

Рис. 5. Фрагмент окна графической оболочки – Common Parameters, Common WorkSet

Созданный проект записывается в файл конфигурации. В окне диалога Project Settings (Рис. 6) пользователь вводит следующие данные - свойства конфигурации: имя автора (Author); название конфигурации (ConfigurationName); дата создания конфигурации (Date) – вводится автоматически; имя файла протокола (Name of LogFile). Вводится также строка соединения с базой данных (Database Conneciton) и заполняется поле «Тип эксперимента» (Experiment Type), используемое при записи данных в БД.

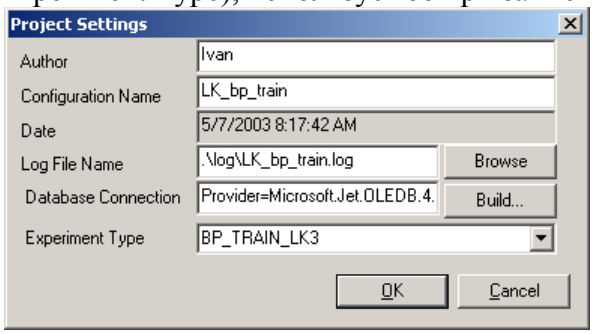

Рис. 6. Окно диалога Project Settings

3.2 Обрабатывающие блоки

Обрабатывающие блоки – это основные вычислительные элементы конфигурации, из которых строится схема (алгоритм) обработки данных. Обрабатывающие блоки делятся на блоки-форматы и блоки-процессинги, отличающиеся выполняемыми функциями.

На Рис. 7 приведено графическое изображение одного из типов обрабатывающих блоков. Показаны входные и выходные буферы данных, два блока control_Processing, линии связи по данным (DataLink), и линии управляющих связей (ControlPath). Связи разных типов на рабочем поле отображаются линиями разного цвета. Такие же обозначения применяются при описании других элементов конфигурации.

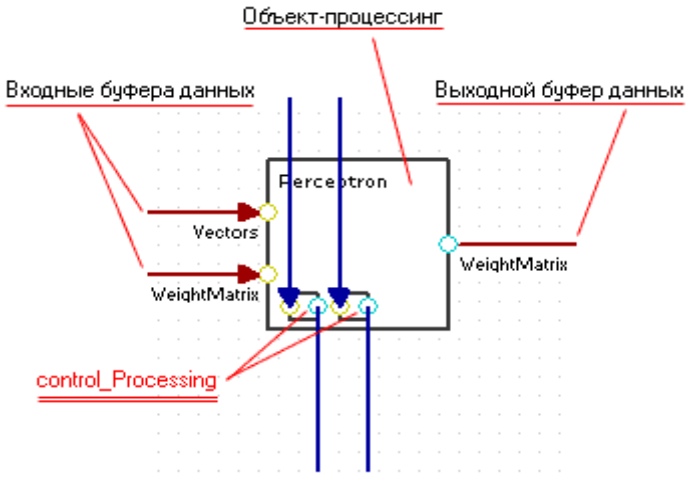

Рис. 7. Элементы графического изображения обрабатывающего блока

Буфера данных и связи обрабатывающих блоков. Передача данных между обрабатывающими блоками осуществляется посредством *буферов данных*. Блок может поддерживать несколько буферов данных, при этом различают входные и выходные буфера. Схематически буфер данных изображается как кружок на левой (входные буферы) или правой (выходные буферы) границе блока. Пользователь, при конфигурировании проекта, может с помощью мыши соединить между собой выход одного блока с входом другого с помощью связи по данным.

Связи по данным обрабатывающих блоков (DataLink) – это элементы конфигурации проекта, показывающие направление передачи данных обрабатывающих блоков. Пользователь задает начальную и конечную точку передачи данных с помощью перетягивания мышкой линии связи от выходного буфера (DataBuffer) одного обрабатывающего блока к входному буферу другого блока. Можно соединить один выходной буфер только с одним входным буфером, и наоборот.

Связь по данным не является эквивалентом буфера данных обрабатывающего блока. Буфер данных – это внутренний объект блока, в то время как связь по данным – это элемент визуального представления конфигурации, показывающий, какие блоки соединены буфером данных.

Параметры и переменные обрабатывающих блоков. Обрабатывающие блоки содержат коллекции внутренних параметров – начальных параметров (Parameters) и рабочих переменных (WorkSet). Параметры служат для инициализации блоков (например, для первоначального задания размера нейронной сети, путей к файлам данных) и управления их изменением во время выполнения проекта. Рабочие переменные служат для доступа к внутреннему состоянию блоков из кода в управляющих блоках и управляют работой алгоритма (например, определение признака окончания обучения по числу ошибок распознавания, получения номера текущей итерации, и т.п.).

Задание параметров обрабатывающих блоков производится пользователем при конфигурировании проекта в диалоговом окне (Рис. 8) графической оболочки GraphShell. Количество и типы параметров определяются обрабатывающим блоком.

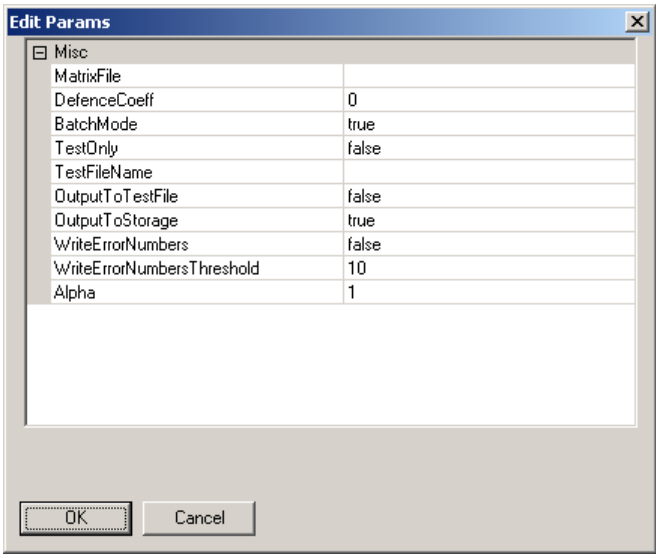

Рис. 8. Диалоговое окно задания начальных параметров обрабатывающих блоков

Блоки-форматы предназначены для работы с внешними данными. Блок-формат выполняет чтение или запись данных в файлы по запросам подключенных к ним других обрабатывающих блоков. Блоки-форматы могут поддерживать одиночный и блочный режим работы с данными. В первом случае чтение или запись происходит по одной записи за раз, во втором – по нескольким записям, что позволяет оптимизировать работу с большими массивами данных. Режим определяется как параметрами самого формата, так и способом вызова его методов другими объектами. Некоторые блоки-форматы поддерживают также специальные методы работы с памятью, такие как кэширование прочитанных записей, контроль объема используемой памяти, и т.п.

Графическое изображение блоков-форматов на рабочем поле графической оболочки представлено на Рис. 9.

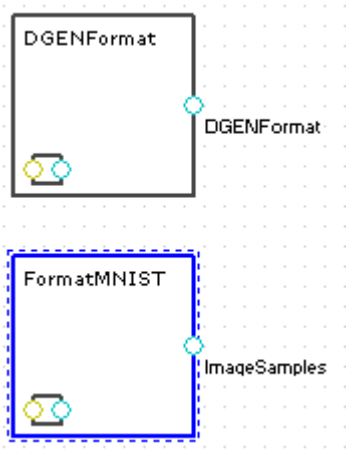

Рис. 9. Графическое изображение блоков-форматов

Блоки-процессинги – это обрабатывающие блоки, выполняющие основную обработку информации. Они обладают большей функциональностью, чем блоки-форматы. Графическое изображение блоков-процессингов на рабочем поле графической оболочки представлено на Рис. 10:

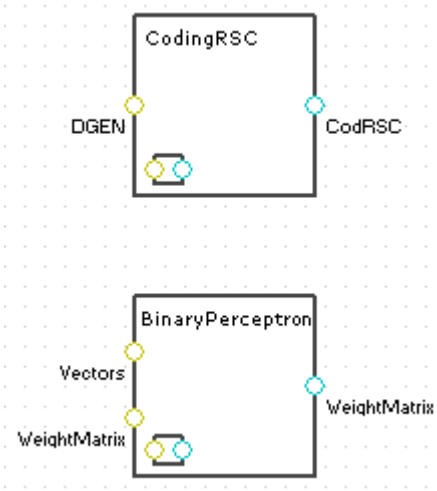

Рис. 10. Графическое изображение блоков-процессингов

3.3 Управляющие блоки и связи

С помощью управляющих блоков и связей (графическое изображение показано на Рис. 11) задается алгоритм выполнения конфигурации.

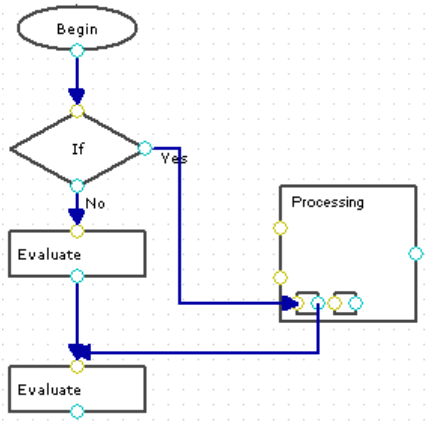

Рис. 11. Графическое изображение управляющих блоков и связей

Управляющие блоки содержат точки входа и выхода (на схеме изображаются кружками), которые используются для расстановки управляющих связей. Надписи в управляющих блоках If, Evaluate отражают первые строки находящихся в них выражений на языке JScript (Рис. 3). Если выражения в блоках не заданы (при помещении новых блоков на рабочее поле), то блоки подписываются по умолчанию (Рис. 11). Задание и редактирование выражений в управляющих блоках производит пользователь при конфигурировании проекта в окнах редактирования выражений (Рис. 12).

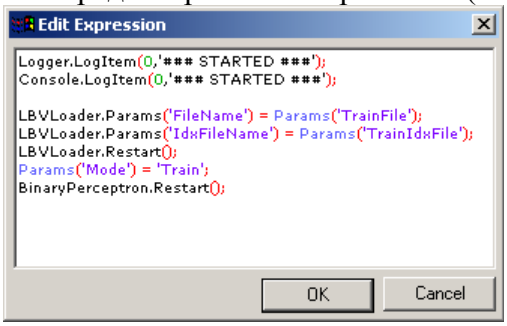

Рис. 12. Окно редактирования выражений в управляющих блоках

Пути перехода (передачи управления) от одного управляющего блока к другому определяются с помощью направленных *управляющих связей* (ControlPath). Они задаются пользователем в графической оболочке путем перетягивания мышью линии от выхода

одного управляющего блока к входу другого. Управляющие связи могут входить одна в другую и стыковаться в точке входа в управляющий блок, позволяя задавать произвольный алгоритм выполнения конфигурации.

3.4 Конфигурирование и работа проекта

Как упоминалось выше, пользователь создает конфигурацию проекта в режиме САПР с помощью оболочки GraphShell (Рис. 3). При этом на рабочее поле помещаются обрабатывающие и управляющие блоки, задаются связи между ними, устанавливаются общие параметры конфигурации проекта и параметры обрабатывающих блоков. Созданная конфигурация проекта сохраняется в файле.

Созданная и сохраненная конфигурация проекта может быть запущена пользователем на выполнение. Запуск в графическом режиме осуществляется из оболочки GraphShell путем переключения графической оболочки из режима САПР в режим выполнения. Запуск в консольном режиме осуществляется с помощью консольного приложения ConfTester. В обоих случаях при запуске автоматически выполняется трансляция проекта. При этом осуществляется проверка согласованности расставленных связей данных, корректности установленных параметров блоков и доступности внешних файлов данных, инициализируется система поддержки скриптовых языков ScriptingEngine.

При работе с системой SNC в графической оболочке GraphShell пользователь имеет возможность скорректировать проект при возникновении ошибок трансляции. При запуске проекта в консольном режиме, возникновение ошибки при трансляции приводит к прерыванию работы. В обоих случаях, в текстовую консоль и файл протокола выводятся сообщения об ошибках.

Выполнение проекта и вывод данных. После успешной трансляции проекта дальнейшее выполнение конфигурации осуществляется средствами ConfigManager. Выполнение конфигурации происходит путем последовательного вычисления выражений, указанных пользователем в управляющих блоках конфигурации, и вызовов методов обрабатывающих блоков. Переходы между управляющими блоками осуществляются по управляющим связям. Выполнение проекта прекращается по достижении управляющего блока End, или при прерывании конфигурации пользователем.

Во время выполнения проекта обрабатывающие блоки могут производить вывод разнообразной информации. Для этого в графическом и консольном режиме доступны следующие средства: единая текстовая консоль SNC, файл протокола, база данных. В графическом режиме доступны также специализированные графические консоли.

Прерывание и возобновление работы проекта. Выполнение проекта может быть приостановлено пользователем. Возможность прерывания работы конфигурации заложена в системную архитектуру SNC и все обрабатывающие блоки обеспечивают её поддержку. При этом сохраняется вся необходимая информация для продолжения работы проекта с прерванного момента. Структура и характер сохраняемой информации определяется соответствующими обрабатывающими блоками. Средствами системы автоматически сохраняется информация о контексте прерванной конфигурации, и другая системная информация.

Информация о прерванном состоянии записывается в файлы на диске компьютера. В дальнейшем имеется возможность возобновления выполнения проекта с момента прерывания.

Запись данных в БД и формирование отчетов. Если при конфигурировании проекта установлена связь с базой данных, обрабатывающие блоки во время выполнения проекта могут производить запись данных в БД. Структура базы позволяет хранить произвольно организованные иерархические данные, обеспечивая возможность их последующего анализа.

Анализ данных, записанных в БД, производится с помощью отдельных приложений на скриптовых или других языках программирования, входящих в состав SNC или разрабатываемых пользователем. Они извлекают данные из БД с помощью объектов Storage, производят необходимые вычисления, и формируют отчеты.

Средства серийного запуска проектов. SNC предоставляет возможность серийного запуска проектов в пакетном режиме. Эта возможность реализована двумя способами. Первый позволяет генерировать файлы конфигураций с помощью специальной программы, подставляющей различные параметры запуска, и их последующего выполнения из командных файлов. Второй способ позволяет программно управлять процессом запуска конфигураций, позволяя реализовывать запуски в нужной последовательности и варьировать параметры запуска конфигураций непосредственно во время работы

Использование такого подхода позволяет реализовать все возможности, которые предоставляет SNC обрабатывающим блокам, при работе с сериями. Это – визуальное конфигурирование, возможность задания параметров, возможность применения языка сценариев в управляющих блоках, возможность прерывания всей серии проектов с автоматическим сохранением данных, необходимых для продолжения работы.

ЗАКЛЮЧЕНИЕ: СРАВНИТЕЛЬНЫЙ АНАЛИЗ ПРОГРАММНЫХ НК

При проведении сравнительного анализа были исследованы следующие пакеты:

- MNN-CAD [11], NeuroLand [12];
- Neural10, QwikNet32, BrainMaker, MPIL, Braincel, Excel Neural Package, Fuzzy Logic Toolbox [10];
- MATLAB [13];
- NeuroSolutions [21];
- NeuralWorks [22];

При сравнении учитывались такие возможности пакетов как задание произвольного алгоритма выполнения нейросетевых вычислений; визуальное конфигурирование нейросетевых вычислений; наращивание и расширения функциональности пакета; сохранение результатов в базе данных для последующей обработки; проведение серийных экспериментов.

В результате проведенного сравнительного анализа установлено следующее:

(1) SNC предоставляет пользователю возможность наращивать функциональность пакета путем реализации новых обрабатывающих блоков – форматов и процессингов. Подобную возможность предоставляют только пакеты: MatLab, MNN-CAD, NeuroLand, NeuroSolutions, NeuralWorks, Maple, Mathematica.

(2) SNC предоставляет возможность реализации произвольных алгоритмов обработки информации. Это обеспечивается визуальным заданием алгоритма выполнения конфигурации проекта и поддержкой выполнения выражений на языке JScript в управляющих блоках во время работы конфигурации. Подобную возможность предоставляют только пакеты: MatLab, Maple, Mathematica, MNN-CAD (задание структуры обрабатывающих блоков и порядка выполнения), NeuroSolutions, NeuralWorks. MatLab, Maple, Mathematica позволяют задавать алгоритм только на встроенном языке без возможности визуального конфигурирования. MNN-CAD позволяет выбирать один из нескольких предопределенных алгоритмов. NeuroSolutions и NeuralWorks также имеют встроенный язык конфигурирования.

(3) SNC предлагает разработчикам и пользователям системы универсальный механизм сохранения результатов в БД. Это позволяет сохранять в удобном виде все необходимые данные выполняемого проекта: параметры конфигурации, результаты работы, и т.д. Информация может использоваться для дальнейшей обработки и формирования отчетов. Из всех рассмотренных пакетов возможность сохранения данных

в БД предоставляет только MatLab и Mathematica средствами встроенного языка программирования. При этом поддерживаются только вызовы методов ODBC/OLEDB, что требует дополнительного программирования. Для создания отчетов в MatLab предусмотрен соответствующий инструмент Report Generator.

(4) SNC предоставляет возможность серийного запуска проектов в пакетном режиме. Такую возможность предоставляют пакеты MNN-CAD, NeuroLand (средствами системы) и MatLab, Maple, Mathematica (средствами встроенного языка программирования).

Результаты проведенного сравнения дают основания предполагать, что программный нейрокомпьютер SNC окажется весьма полезным средством создания новых интеллектуальных информационных технологий.

БЛАГОДАРНОСТИ

Авторы выражают благодарности Л.М.Касаткиной и А.М.Касаткину за обсуждения рукописи этой статьи, Александру Тетерюку за участие в реализации разработки, Михаилу Куссулю – за консультации. Работа частично поддерживалась за счет финансирования по договору с МПП № 92289/48-2001 и № 92211/07-42-00.

1. *Нейрокомпьютеры* и интеллектуальные роботы / Амосов Н.М., Байдык Т.Н., Гольцев А.Д., Касаткин А.М., Касаткина Л.М., Куссуль Э.М., Рачковский Д.А. – Киев: Наук. думка, 1991. – 272 с.
2. *Wunsch D., Hasselmo M., Venayagamoorthy K., Wang D.* Advances in Neural Networks Research – IJCNN 2003 // Neural Networks. – 2003. – 16, N 5-6. – P. 521-945.
3. *Stubbs D.* Neurocomputers // M.D. Computing – 1988. – 5, N 3. – P. 14-24.
4. *Hecht-Nielsen R.* Performance Limits of Optical, Electro-Optical and Electronic Artificial Neural System Processors // Soc. Photo-Opt. Instrum. – 1986. – 634. – P. 277.
5. *Куссуль Е.М., Лукович В.В., Луценко В.Н.* Мультипроцессорное вычислительные устройства для управления транспортными роботами в естественной среде // УСИМ – 1988. – N 5. – С. 102-105.
6. *Kussul E.M., Rachkovskij D.A., Baidyk T.N.* Associative-Projective Neural Networks: architecture, implementation, applications // 4th Intern. Conference "Neural Networks & their Applications", 1991. – P. 463-476.
7. *Куссуль Э.М.* Ассоциативные нейроподобные структуры. – Киев: Наук. думка, 1992. – 144 с.
8. *Куссуль М., Забара С., Комухаев Э., Сахарин, В.* Проектирование логических схем нейрокомпьютера в элементном базисе ПЛИС XC2000 // УСиМ – 1993. – N 6.
9. *Куссуль М.Э., Симоров С.Н.* Одноплатный нейроклассификатор // *IV Всероссийская конф. "Нейрокомпьютеры и их применение",* 1998.
10. *Круглов В.В., Борисов В.В.* Искусственные нейронные сети. Теория и практика. – М: Горячая линия – Телеком, 2002. – 382 с.
11. *Kussul M., Riznyk A., Sadovaya E., Sitchov A., Chen T.-Q.* A visual solution to modular neural network system development // *International Joint Conference on Neural Networks IJCNN'02,* 2002. – 1.
12. *Резник А.М., Калина Е.А., Садовая Е.Г., Дехтяренко А.К., Сичев А.С., Галинская А.А.* Многофункциональный нейрокомпьютер NeuroLand // Математичні машини і системи. – 2003. – N 1. – С. 36-45.
13. *Пакет* программ математических вычислений MatLab. http://www.mathworks.com
14. *Пакет* программ математических вычислений Statistica. http://www.statsoft.ru
15. *Пакет* программ математических вычислений Maple. http://www.maple.com
16. *Пакет* программ математических вычислений *Mathematica*. http://www.wolfram.com

17. *Технология* программирования COM. Microsoft. http://msdn.microsoft.com/
18. *Грищенко В.Н., Лаврищева Е.М.* Компонентно-ориентированное программирование. Состояние, направления и перспективы развития // Проблемы программирования. – 2002. – N 1-2. – С. 80-90.
19. *Технология* доступа к данным Microsoft Data Access. http://www.microsoft.com/data
20. *Matsumoto M., Nishimura T.* Algoritmic Conjurings – A Psuedo-Random Number Generator. – 1997. – http://www.coyotegulch.com
21. *Пакет* программ нейросетевого моделирования NeuroSolutions. http://www.nd.com/
22. *Пакет* программ нейросетевого моделирования NeuralWorks. http://www.neuralware.com/