4. Elze, K. Der Kalberdurchfall [Text] / K. Elze // Milchpraxis. — 2006. — № 4. — P. 178–182.
5. Белановский, А. С. Основы биофизики в ветеринарии [Текст] / А. С. Белановский. — М.: Дрофа, 2007. — 332 с.
6. Сасимова, И. А. Обоснование биофизического действия информационных электромагнитных излучений на микробиологические объекты животноводства [Текст] / И. А. Сасимова, Л. Ф. Кучин // Восточно-Европейский журнал передовых технологий. — 2008. — № 4/2(34). — С. 27–29.
7. Шестопалов, В. П. Спектральная теория и возбуждение открытых структур [Текст] / В. П. Шестопалов. — Киев: Наукова думка, 1987. — 276 с.
8. Бреховских, Л. М. Волны в слоистых средах [Текст] / Л. М. Бреховских. — М.: Наука, 1973. — 343 с.
9. Корн, Г. Справочник по математике [Текст] / Г. Корн. — М.: Наука, 1970. — 720 с.
10. Крылов, В. И. Вычислительные методы [Текст] / В. И. Крылов, В. В. Бобков, П. И. Монастырский. — М.: Наука, 1976. — 302 с.
11. Анго, А. Математика для электро- и радиоинженеров [Текст] / А. Анго. — М.: Наука, 1965. — 776 с.

ТЕОРЕТИЧНИЙ АНАЛІЗ РОЗПОДІЛУ ВІДЕОІМПУЛЬСІВ В МОЛОЧНІЙ ЗАЛОЗІ КОРІВ

Проведено теоретичний аналіз взаємодії імпульсного електричного поля з молочною залозою новотільних корів. Отримані результати дозволяють досліджувати розподіл імпульсного електричного поля в обсязі молочної залози та визначити параметри імпульсного випромінювання (амплітуда імпульсів; частота і період проходження імпульсів; величина експозиції) для збільшення імуноглобулінів класу LgG і LgM.

Ключові слова: молочна залоза корів, імпульси електричного поля, молозиво та молоко корів, імуноглобуліни.

Торчук Михаил Васильевич, *ассистент, кафедра технотроники и теоретической электротехники, Харьковский национальный технический университет сельского хозяйства им. П. Василенко, Украина,* ***e-mail: tte_nniekt@ukr.net****.*

Торчук Михайло Васильович, *асистент, кафедра технотроніки і теоретичної електротехніки, Харківський національний технічний університет сільського господарства ім. П. Василенка, Україна.*

Torchuk Mihail, *Kharkiv Petro Vasylenko National Technical University of Agriculture, Ukraine,* ***e-mail: tte_nniekt@ukr.net***

УДК 681.3:621.3(62-52)

Шемсединов Т. Г.,
Маленко Н. В.,
Мороз А. И.,
Карасюк П. В.

ИСПОЛЬЗОВАНИЕ ИНТРОСПЕКТИВНЫХ ИНТЕРФЕЙСОВ В ПРОТОКОЛАХ ПРИКЛАДНОГО УРОВНЯ

В статье предлагается подход к решению задачи динамического связывания прикладных программных интерфейсов (API) в распределенных информационных системах класса SaaS (Software as a Service), построенных в сервисно-ориентированной архитектуре (SOA) и web-сервисов с применением метапрограммирования, и его техник: интроспекции, динамической модификации структуры и функций программных модулей и динамической интерпретации метамоделей.

Ключевые слова: *сервисная архитектура, метапрограммирование, интроспекция, динамическая интерпретация, метамодель, метаданные, связывание, интерфейсы.*

1. Введение

Одна из основных задач при разработке современных приложений в сервисной архитектуре (SOA) — это создание и связывание прикладных программных интерфейсов (API) двух типов: без состояния (STATEless или REST серверов), и с состоянием (STATEful) [1, 2]. Для создания таких API существует множество технологических стеков, однако, связывание программных интерфейсов происходит, чаще всего, вручную, с помощью программирования соответствующих вызовов или выделения в вызывающей системе специализированного слоя доступа, т. е. «обертки», которая получает запросы от бизнес-логики приложения, совершает асинхронные сетевые вызовы к удаленному серверу приложений, получает ответы и передает их в функции обратного вызова приложения. Модификация структуры и функций как самих информационных систем (ИС), так и их компонентов, приводит к частым изменениям в их API и, как следствие, к необходимости постоянного переписывания «обертки» или слоя доступа. В статье предлагается применение методов метапрограммирования, интроспекции и динамического связывания, позволяющие решить данную проблему.

2. Анализ литературных данных и постановка задачи исследования

Чтобы выявить проблемы статического связывания интерфейсов приложений и моделей предметной области, мы рассмотрим существующие подходы и основные их особенности. В работах [3, 4] описано, как построение модели предметной области с помощью структур данных и программного кода является основой разработки информационных систем. Структуры данных включают в себя структуры в оперативной памяти и в протоколах передачи, в файлах на диске и в базах данных, а программный код — это активная (императивная, событийная или функциональная) модель решаемой задачи над моделируемыми данными.

Существует множество подходов и технологий, для которых модель является статической, т. е. зафиксиро-

ванной на продолжительный срок (период жизни одной версии программного обеспечения). В таких статических технологиях применяются те или иные средства синхронизации структур данных постоянного хранения и структуры объектов в программном коде. Но есть класс задач, и он становится все актуальнее, для которого подобные технологии не являются приемлемым решением, т. к. требуют вмешательства разработчиков и выпуска новой версии системы при любом изменении в модели предметной области, а в конечном итоге, требуют пересборки всех компонентов информационной системы [5–7]. При таких изменениях требуется повторное развертывание этих компонент на рабочих местах пользователей и серверах, а для веб-приложений процесс этот более мягкий, т. к. развертывание касается только серверов. Однако, в любом случае, изменение модели связано со следующими необходимыми действиями:

— прерывание штатной работы системы на определенное время при развертывании (прерывание сессий пользователей, временная недоступность сервиса);
— конвертация данных в постоянном хранилище из старых форматов в новые;
— внесение изменений в программный код (что предусматривает выделение времени на разработку, отладку, тестирование);
— ревизия всех связей с другими подсистемами (как на уровне данных, так и на уровне программных вызовов).

Ограничим класс задач, которые требуют принципиально другого подхода:

— задачи с динамической предметной областью, где изменение структуры и параметров модели является нормальным штатным режимом функционирования;
— задачи обработки слабо-связанных данных или данных с непостоянными структурой, параметрами или логикой обработки;
— задачи, в которых количество классов обрабатываемых объектов сравнимо с количеством их экземпляров или количество экземпляров всего на один-два порядка выше;
— задачи, в которых от скорости интеграции обновленной модели с другими подсистемами зависит успешность модели бизнеса (в идеале требуется интеграция в реальном времени или приближенная к реальному времени);
— задачи межкорпоративного обмена данными, межсистемной интеграции (включая интеграцию гетерогенных распределенных приложений);
— прикладные задачи, не связанные с массовым пользователем.

Такие задачи традиционно считаются слабо автоматизируемыми [3, 8] или ограниченно автоматизируемыми. Но если искать путь их решения, то естественно предположить, что повышение уровня абстракции программного решения позволит покрыть более широкий круг задач, в который войдут и смежные задачи, с измененными параметрами и структурой [9, 10]. Назовем такую модель белее общей задачи, метамоделью и предложим способ ее динамического отображения (проецирования) на предметную область, при котором мы сможем динамически получить несколько моделей предметной области из одной метамодели в зависимости от подаваемых в систему метаданных. Такие метаданные должны дополнять и конкретизировать метамодель (дописывать структуру и параметры) [11–13].

Прикладные задачи требуют все более быстрого изменения бизнес-логики, а значит и API интерфейсов. Предложенный в статье подход привлекает для решения задачи методологию метапрограммирования, а именно: интроспекцию и динамическую интерпретацию метамоделей.

Метамодель — это информационная модель более высокого уровня абстракции, чем конкретная модель предметной области одного приложения. Метамодель описывает и покрывает функционалом не отдельную задачу, а широкий круг задач с выделением в этих задачах общих абстракций, правил обработки данных и управления бизнес-процессами. К необходимой конкретике и специфике метамодель адаптируется уже в момент исполнения, преобразовывая метаданные в динамический код и обеспечивая их динамическое связывание со средой запуска (виртуальной машиной, операционной системой или облачной инфраструктурой). Интроспекция же является механизмом получения структуры удаленного сетевого интерфейса и автоматического построения «обертки» со стороны вызывающей системы [14, 15].

Целью статьи является описание подхода с применением динамического связывания компонентов программных систем при помощи интроспекции и динамической интерпретации метамодели. Для этого, необходимо решить следующие задачи: построение классификации метаданных, выделение которых из метамодели позволяет повысить уровень абстракции; описание принципа динамического связывания; обоснование преимуществ применения интроспекции и динамической интерпретации для прикладных программных систем.

3. Динамическое связывание компонентов

Рассмотрим модель ИС (или компонента ИС), учитывая лишь аспект ее (или его) взаимодействия с другими ИС (или компонентами ИС). Такой компонент имеет бизнес-логику и интерфейс и осуществляет вызовы с нижним слоем, получая ответы, и сам отвечает на вызовы верхнего слоя (рис. 1).

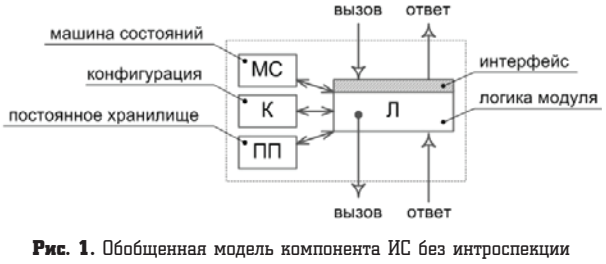

Рис. 1. Обобщенная модель компонента ИС без интроспекции

Вызовы в нижние слои осуществляются ИС (или компонентом ИС) на основании «известных» ему интерфейсов нижнего слоя, то есть, на основании информации про идентификаторы методов API, их параметры и типы. Информация эта «зашита» в бизнес-логике и любые изменения интерфейсов приводят к модификации бизнес-логики, чего и нужно избежать.

Первый шаг к этому — добавление прослойки или «обертки» (рис. 2), которая представляет вызываемую систему внутри вызываемой и концентрирует в себе все внешние вызовы. Прослойка и интерфейс оборачивают

ISSN 2226-3780

слой логики с двух сторон, что позволяет повысить его внутреннюю связность, и уменьшить внешнюю связанность кода.

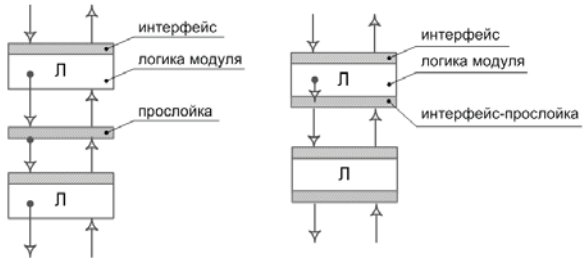

Рис. 2. Введение интерфейса-прослойки

Стыковка таких модулей на порядок проще, но трудоемкость их разработки и сборки существенно повышается. Далее, естественно принять решение о выносе информации о вызовах (структуре, идентификаторах методов, параметров и типов) в особый класс данных — метаданные. Бизнес-логика, после выноса метаданных из нее, оставшаяся в слое логики приложения будет называться уже не моделью, а метамоделью. Прослойка же (или «обертка») уже может быть построена автоматически, при попадании метаданных в метамодель в процессе обмена данными с верхним и нижним слоями (рис. 3).

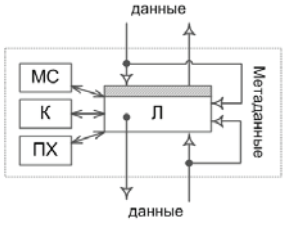

Рис. 3. Введение динамической интерпретации метаданных: МС — машина состояний; К — конфигурация; ПХ — постоянное хранилище

Для доступа к удаленным компонентам ИС в системах с динамическим связыванием и метамоделями, предлагается использовать открытые протоколы передачи данных, совместимые с требованиями корпоративных правил безопасности. Межсетевые экраны и брандмауэры должны быть прозрачны для таких протоколов, однако они имеют ограниченный набор функций, специфицированный по умолчанию, как например, набор методов HTTP 1.0 и HTTP 1.1 (а так же, совместимого с 1.1 протокола SSE). Традиционно, для расширения набора методов HTTP (отличных от GET, POST, PUT и т. д.) применяется кодирование идентификаторов методов в URL запросов, реже — в параметры запросов, но передача идентификаторов в теле пакета в формате XML (так называемый envelope/конверт) применяется все реже. Таким образом, интроспекция для самого протокола HTTP бессмысленна, т. к. он имеет фиксированный набор команд (аналогичный CRUD), а вот для URL интроспекция применима двумя способами:

1. Получение списка каталога в стиле отдачи индекса файловой системы. То есть, энумерация ресурсов для корня дает список программных интерфейсов, а конкатенируя их идентификаторы с базовым URL метод GET возвращает набор методов. Дальнейшая же конкатенация и применение метода GET к полученному URL отдает параметры вызова, типы и модификаторы. Приведенный способ предполагает парсинг (полный синтаксический разбор) текстового формата вывода для получения идентификаторов из тела отдаваемой HTML страницы.
2. Второй способ состоит в предоставлении программного интерфейса интроспекции, то есть, в доступе к специальному системному интерфейсу, структура и методы которого фиксированы и «зашиты» в компонент при компиляции, а интерфейс этот содержит методы для получения метаданных, описывающих другие интерфейсы — динамически прикладные (т. е. привязанные к идентификаторам и специфике модели или метамодели). Этот метод так же предполагает парсинг, однако, формат сообщения можно ограничить одним из удобных форматов сериализации, например: XML, JSON, CLEAR и т. д. Эти форматы выбираются таким образом, чтобы обеспечить их автоматическую или естественную обработку со стороны клиентского приложения. Например, парсинг JSON встроен в веб-браузер, т. к. JSON является родным форматом описания объектов в JavaScript.
3. Возможен и гибридный вариант двух, описанных выше, — это отдавать JSON в ответ на GET запросы к индексам каталогов, но в данной работе он не будет рассматриваться по причине недостаточной безопасности такого способа. Навигация по такому веб-сервису была бы возможна еще до вызова методов программного интерфейса аутентификации или пришлось бы реализовывать плагин для веб-сервера, проверяющий наличие сессии еще до энумерации индекса каталога по шаблону вывода.

4. Применение интроспекции для веб-сервисов

Итак, очевидны преимущества второго метода интроспекции, который создает слой представления и соответствующий протокол для реализации сервисной архитектуры с инкапсуляцией в HTTP и получая от него услуги транспортного слоя. Таким образом, есть возможность построения универсальной клиентской консоли доступа к веб-сервису, которая базируется на программной интроспекции, но позволяет осуществить ручную навигацию по всем энумераторам веб-сервиса.

Уточним обязательные (необходимые для работы динамического связывания) энумераторы:

1. Энумеротор программных интерфейсов.
2. Энумератор методов.
3. Энумератор параметров.

Однако, могут быть полезны и дополнительные приемы сервисной интроспекции:

4. Энумератор типов данных.
5. Энумератор классов.
6. Энумератор событий.
7. Энумератор баз данных.
8. Энумератор форматов сериализации.

Рассмотрим их подробнее:

Энумеротор программных интерфейсов — позволяет получить список доступных интерфейсов веб-сервиса, этот список может быть статическим или динамическим, например, для систем с динамической интерпретацией метамоделей достаточно естественно использовать динамическое построение части интерфейсов при старте веб-сервиса. Это происходит следующим образом: при предыдущих запусках системы была сформирована стартовая конфигурация, которая содержит метаданные, необходимые для последующего старта: подключения к системе управления базами данных (СУБД), идентификаторы для доступа к метаданным в БД и на диске,

идентификаторы других компонентов, с которыми необходимо осуществить связь при старте (IP-адреса, номера портов, URL, пути и запросы). Стартовая конфигурация нужна только для того, чтобы осуществить доступ к метаданным, а сам программный компонент имеет в себе метамодель, которая при наполнении метаданными порождает программные интерфейсы, доступные в последствии с помощью механизма интроспекции и энумератора программных интерфейсов, как его составной части.

Энумератор методов — каждый программный интерфейс экспортирует наружу набор методов. Методы имеют, как минимум, названия и набор параметров, однако могут иметь еще и дополнительные метаданные: описание, владелец, время последнего вызова, ссылка на документацию или сам контент документации, перечисление зависимостей вызова. Если метод имеет дочерние или родительские зависимости, то это должно быть дополнительно описано в его метаданных. Метод может иметь метаданные и о своем применении в последовательностях вызовов или транзакциях, переводящих систему в определенное состояние или когда порядок вызовов не произвольный или данные, полученные от одних вызовов предполагается использовать в качестве параметров других вызовов.

Энумератор параметров — позволяет получить список, типы данных и модификаторы параметров вызова. Список параметров — это список идентификаторов, определяющих их семантическое наполнение, однако, это достаточно условные идентификаторы и метаданные о параметрах, получаемые через программный интерфейс интроспекции может содержать еще и ссылку на документацию или непосредственно фрагмент этой документации. Типы параметров необходимы для дополнительной верификации при преобразовании абстрактных вызовов метамодели в физические запросы к веб-сервису, по типам происходит проверка совместимости интерфейсов. Дополнительные модификаторы позволяют определить такие атрибуты параметров вызова: значение по умолчанию, диапазон значений (для перечисляемых типов) или источник диапазона (например, в форме ссылки на вызов другого метода или идентификатор справочника), флаг разрешения или запрета NULL значений для данного параметра.

Энумератор типов данных — позволяет получить список пользовательских или прикладных типов, которые, однако, должны быть сконструированы на базе системных типов. Именно поэтому, такой энумератор редко используется и имеет скорее концептуальное значение для систем и динамической интерпретации метамоделей, чем какое-либо полезное применение. Типы данных сводятся к числовым, строковым, логическим, перечислимым, структурам, множествам и т. д., и все эти возможности завязаны на математическом аппарате, предоставляемом платформой запуска (виртуальной машиной, операционной системой), а в конечном итоге, определяемого аппаратной архитектурой вычислительной машины.

Энумератор классов — если метамодель, которая используется в системе для решения задач предметной области, создана с использованием объектно-ориентированного подхода (имеется в виду OOD и OOA, а не OOP), то в системе необходим энумератор классов предметной области. Такой энумератор не обязательно связан один-к-одному с таблицами БД, т. к. не все таблицы представляют собой отображение сущностный предметной области. То есть энумератор классов невозможно свести напрямую к технологии скафолдинга, — это тип энумераторов, применяемых для интроспекции веб-интерфейсов в системах с динамическим связыванием.

Энумератор событий — позволяет получить список доступных на сервере событий и/или каналов событий (доступных для подписки удаленными пользователями или программными компонентами систем). Список событий так же имеет параметры и типы данных, получаемые или передаваемые событиями, поэтому, такой энумератор может быть разделен на несколько, в зависимости от специфики решаемой задачи и модели обмена событиями.

Энумератор баз данных — такой энумератор используется чаще всего в Data-aware и Data-centric приложениях и технологиях скафолдинга, позволяет осуществлять доступ к метаданным СУБД, описывающим структуры хранения данных, такие как: базы данных, таблицы, схемы данных, связи, триггера, хранимые процедуры, поля, типы данных, домены.

Энумератор форматов сериализации — способ получения списка сериализаторов (синтаксических форматов, как то JSON, XML и т. д.)

Из перечисленных энумераторов удаленная сторона, делающая вызовы веб-сервиса, может получать метаданные о структуре программного компонента. Процесс же динамического связывания состоит в установлении соответствия методов и событий, происходящих в прикладной модели одного программного компонента, с событиями и методами, находящимися в модели другого компонента и осуществлении сквозной передачи вызовов между ними. Это есть способ межпроцессового взаимодействия, допустимый в рамках одного сервера, группы серверов, целой корпоративной инфраструктуры, распределенной системы удаленных офисов и даже межкорпоративной среды взаимодействия.

5. Классификация метаданных

Метаданные описания информационных объектов обработки трех типов:

1. Системные: заголовки протоколов (HTTP, HTTPS, SSE и т. д.); параметры оптимизации; метаданные обращения к ресурсам (тип браузера и другие параметры браузера, версия, технологии, которые поддерживает браузер); тип, название и версия ОС клиента; язык интерфейса ОС и браузера, настройки (серверные); размер экрана клиента и возможности ввода; метаданные межпроцессового взаимодействия; параметры кэширования ресурсов (как на клиентской стороне, в браузере, так и на сервере); даты и версии ресурсов (для синхронизации); параметры манифеста offline WebApplication.
2. Прикладные метаданные: параметры приходящие к клиенту как метаданные спецификации (схемы) предметной области; метаданные интерпретаторов (языка трансляции и преобразования декларативных форматов в т. ч. используемые в браузере или другой среде запуска); параметры компонентов, расшифровывающих метаданные (параметры алгоритмов кодирования и т. д.); обрабатывающие параметры (операции мультиплексирования, конвертации, преобразования в машинный код, развертывания и др.); параметры компонентов, формирующих запросы к web-сервисам (типы данных, названия методов, параметры вызовов); описания структуры

и схемы данных; параметры компонентов, выполняющих анализ метаданных (валидацию параметров, проверяют согласованность); параметры инициализации процессов обработки.

3. Интеграционно-коммуникационные: параметры мижсерверного взаимодействия и трансляции сообщений, событий; параметры связывания объектов и процессов; параметры распространения (трансляции и ретрансляции) широковещательных сообщений об изменениях в метамодели в облачной среде, предусматривающий синхронизацию версий метамоделей в оперативной мяты распределенных приложений; параметры быстрого обмена сообщениями (межпроцессового взаимодействия IPC — interprocess communication); описания ресурсов облака провайдера облачных вычислений или корпоративного уровня.

6. Выводы

В статье показано, как при взаимодействии двух и более систем через сетевые интерфейсы с интроспекцией и интерпретацией метамодели, происходит динамическое связывание их интерфейсов. Оно позволяет модифицировать функциональность прикладных информационных систем без специализированной адаптации вызовов и изменения программного кода, а также связывать системы, взаимодействие которых первоначально не предполагалось. Передавая метамодель при помощи протоколов транспортного уровня, стороны не знают заранее структуру и параметры информационных объектов, и не привязаны жестко к именам функций и наборам параметров, при совершении межсистемных вызовов. Вместо этого, стороны знают язык метаописания, позволяющий динамически интерпретировать данные и совершать вызовы, формируя параметры и интерпретируя ответы удаленной стороны. В статье дана классификация метаданных, выделение которых из метамодели, повышает ее абстрактный уровень. Показано, что данный метод повышает гибкость ИС класса SaaS и позволяет сократить количество промежуточных уровней абстракции, повысить повторное использование кода и эффективность процессов разработки и эксплуатации ИС в целом.

Литература

1. Richardson, L. Sam Ruby RESTful Web Services [Text] / L. Richardson, S. Ruby. — O'Reilly Media, Inc., 2008. — 454 p.
2. Стенін, О. А. Розробка фізичних і логічних метрик в задачі багатокритеріальної оптимізації інформаційного навантаження при структуризації корпоративного центру даних. Адаптивні системи автоматичного управління [Текст] / О. А. Стенін, Ю. А. Тимошин, Т. Г. Шемсединов, С. О. Шуст // ДНВП Системні технології. — 2009. — Вип. 12(32). — С. 86–91.
3. Elmagarmid, A. K. Duplicate Record Detection: A Survey [Text] / A. K. Elmagarmid, P. G. Ipeirotis, V. S. Verykios // IEEE Transactions on Knowledge and Data Engineering archive. — January 2007. — Vol. 19, Iss. 1. — P. 1–16. — Available at: \www/URL: doi: 10.1109/TKDE.2007.9.
4. Narendula, R. A Decentralized Online Social Network with Efficient User-Driven Replication [Text] / R. Narendula, A. Papaioannou, K. Aberer // IEEE International conference on Social Computing (SocialCom 2012). — Amsterdam, 3–5 Sept. 2012. — P. 166–175. — Available at: \www/URL: doi: 10.1109/SocialCom-PASSAT.2012.127.
5. Giannikis, G. Workload optimization using SharedDB [Text] / G. Giannikis, D. Makreshanski, G. Alonso, D. Kossmann // SIGMOD'13 Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. — NY, USA, 2013. — P. 1045–1048. — Available at: \www/URL: doi: 10.1145/2463676.2463678.
6. Alonso, G. SWissBox: An Architecture for Data Processing Appliances [Text] / G. Alonso, D. Kossmann, T. Roscoe // 5th Biennial Conference on Innovative Data Systems Research (CIDR'11). — Asilomar, California, USA, January 9–12, 2011. — P. 32–37. — Available at: \www/URL: doi: 10.1.1.229.2866.
7. Ahrens, M. Relational databases, virtualization, and the cloud [Text] / M. Ahrens, G. Alonso // IEEE 27th International Conference on Data Engineering (ICDE), 2011. — Hannover, Germany, 11–16 April 2011. — P. 1254. — Available at: \www/URL: doi: 10.1109/ICDE.2011.5767966.
8. Subasu, I. E. Architectural Concerns for Flexible Data Management [Text] / I. E. Subasu, P. Ziegler, K. R. Dittrich, H. Gall // SETMDM'08 Proceedings of the 2008 EDBT workshop on Software engineering for tailor-made data management. — Nantes, FrancePages, 2008. — P. 35–40. — Available at: \www/URL: doi: 10.1145/1385486.1385497.
9. Тимошин, Ю. А. Технология распределенной обработки данных и приложений с использованием динамически интерпретируемых метамоделей [Текст] / Ю. А. Тимошин, Т. Г. Шемсединов, В. П. Ярченко, А. И. Мороз // Адаптивные системы автоматического управления. — 2014. — № 1(24). — С. 120–133.
10. Боркус, В. Методы и инструменты интеграции корпоративных приложений [Текст]: отчет / В. Боркус. — М.: RC Group, 2006. — 13 с.
11. Аткин, А. Интеграция ИТ: основные понятия и технологии [Электронный ресурс] / А. Аткин. — 2010. — С. 284–289. — Режим доступа: \www/URL: tvvlibrary.narod.ru/papers/2010/37.pdf.
12. Шемсединов, Т. Г. Динамическая интерпретация метамоделей [Электронный ресурс] / Т. Г. Шемсединов. — 2012. — Режим доступа: \www/URL: http://habrahabr.ru/post/154891/.
13. Шемсединов, Т. Г. Метопрограммироваие [Электронный ресурс] / Т. Г. Шемсединов. — 2012. — Режим доступа: \www/URL: http://habrahabr.ru/post/137446/.
14. Bernus, P. Enterprise Architecture, Integration and Interoperability [Text] / Eds. P. Bernus, G. Doumeingts, M. Fox // IFIP TC 5 International Conference, EAI2N 2010, Held as Part of WCC 2010, Brisbane, Australia, September 20–23, 2010, Proceedings Series: IFIP Advances in Information and Communication Technology, Vol. 326. — Springer, Berlin, 2010. — 177 p.
15. Зауфер, Г. Шаблоны для информационного сервиса [Электронный ресурс] / Г. Зауфер, М. Сельваж, Э. Лейн, Б. Мэтьюс. — 2007. — Режим доступа: \www/URL: http://www.ibm.com/developerworks/ru/library/ws-soa-infoserv1/.

ВИКОРИСТАННЯ ІНТРОСПЕКТИВНИХ ІНТЕРФЕЙСІВ В ПРОТОКОЛАХ ПРИКЛАДНОГО РІВНЯ

У статті пропонується підхід до вирішення задачі динамічного зв'язування прикладних програмних інтерфейсів (API) у розподілених інформаційних системах класу SaaS (Software as a Service), побудованих в сервісно-орієнтованій архітектурі (SOA) та web-сервісів із застосуванням метапрограмування і його технік: інтроспекції, динамічної модифікації структури та функцій програмних модулів і динамічної інтерпретації метамоделей.

Ключові слова: сервісна архітектура, метапрограмування, інтроспекція, динамічна інтерпретація, метамодель, метадані, зв'язування, інтерфейси.

Шемсединов Тимур Гафарович, *научный сотрудник, Научно-исследовательский институт системных технологий, Национальный технический университет Украины «Киевский политехнический институт», Украина,* ***e-mail: timur.shemsedinov@gmail.com.***

Маленко Николай Васильевич, *кафедра технической кибернетики, Национальный технический университет Украины «Киевский политехнический институт», Украина,*
e-mail: nikolay.malenko@gmail.com.

Мороз Алексей Игоревич, *кафедра технической кибернетики, Национальный технический университет Украины «Киевский политехнический институт», Украина,*
e-mail: alex.frost.ua@gmail.com.

Карасюк Павел Валериевич, *кафедра технической кибернетики, Национальный технический университет Украины «Киевский политехнический институт», Украина,* ***e-mail: karac38@gmail.com****.*

Шемсединов Тимур Гафарович, *науковий співробітник, Науково-дослідний інститут системних технологій, Національний технічний університет України «Київський політехнічний інститут», Україна.*

Маленко Микола Васильович, *кафедра технічної кібернетики, Національний технічний університет України «Київський політехнічний інститут», Україна.*

Мороз Олексій Ігоревич, *кафедра технічної кібернетики, Національний технічний університет України «Київський політехнічний інститут», Україна.*

Карасюк Павло Валерійович, *кафедра технічної кібернетики, Національний технічний університет України «Київський політехнічний інститут», Україна.*

Shemsedinov Timur, *National Technical University of Ukraine «Kyiv Polytechnic Institute», Ukraine,* ***e-mail: timur.shemsedinov@gmail.com****.*

Malenko Nikolay, *National Technical University of Ukraine «Kyiv Polytechnic Institute», Ukraine,*

e-mail: nikolay.malenko@gmail.com*.*

Moroz Oleksii, *National Technical University of Ukraine «Kyiv Polytechnic Institute», Ukraine,* ***e-mail: alex.frost.ua@gmail.com****.*

Karasiuk Pavel, *National Technical University of Ukraine «Kyiv Polytechnic Institute», Ukraine,* ***e-mail: karac38@gmail.com***

УДК 621.374

Чёрная М. А.

ОПРЕДЕЛЕНИЕ ПАРАМЕТРОВ ЭЛЕКТРОМАГНИТНОГО ПОЛЯ ДЛЯ ПРЕДПОСЕВНОЙ ОБРАБОТКИ СЕМЯН ПОДСОЛНУХА

В работе, на основе разработанной модели семян подсолнуха, проведены теоретические исследования по определению биотропных параметров низкоэнергетического (информационного) электромагнитного поля миллиметрового диапазона длин волн (частота, плотность потока мощности, экспозиция, амплитудная модуляция), которые могут оказывать воздействие на биофизические процессы в семенах. Применение электромагнитного поля, с установленными биотропными параметрами для предпосевной обработки семян, повысит урожайность и масляничность семян нового урожая.

Ключевые слова: *семена подсолнуха, частота электромагнитного поля, модуляционные параметры, мощность ЭМП, предпосевная обработка семян.*

1. Введение

Экономический анализ показывает, что в настоящее время в Украине средняя урожайность сельскохозяйственных культур снизилась на 20–25 %, в том числе и подсолнуха, из-за высокой стоимости и недостатка минеральных удобрений и средств защиты растений от вредителей [1, 2]. Поэтому актуальной задачей является разработка новых экономичных, эффективных и экологически безопасных технологий, направленных на повышение урожайности и качества семян подсолнечника. Одним из путей решения данной задачи является использование информационного ЭМП КВЧ (крайне высокие частоты) диапазона [2]. Главным достоинством электромагнитной технологии по предпосевной обработке семян подсолнуха низкоэнергетическим излучением КВЧ диапазона заключается в возможности улучшения их роста и развития за счет мобилизации внутренних резервов самих семян без химических препаратов или методов генной инженерии [3].

2. Анализ литературных данных и постановка проблемы

В последние годы для интенсификации растениеводства в практику сельского хозяйства стали активно внедрять электрофизические методы воздействия на растения и семена зерновых и овощных культур с целью повышения урожайности и улучшения качества получаемой продукции [4, 5]. К электрофизическим способам предпосевной обработки относятся: воздействие постоянного электрического поля, постоянного магнитного поля, инфракрасных лучей, электрического поля переменного тока высокого напряжения, сильного электростатического поля, электрическое поле коронного разряда [6]. Общим недостатком всех существующих технологий с использованием предпосевной обработки семян электрофизическими способами является низкая повторяемость результатов обработки, и невысокой прибавкой к урожаю — 10...12 %. Это можно объяснить несовершенством существующих технических средств и методик исследования, отсутствием экспресс-методов диагностики, а также отсутствием достаточно глубоких теоретических и экспериментальных исследований механизма действия различных физических факторов на посевной материал [7].

Наиболее эффективным, энергосберегающим и рентабельным является способ обработки семян информационным электромагнитным полем [4].

Исходя из этого, можно предположить, что применение электромагнитных технологий может повысить урожайность и качество семян подсолнуха, а также их