УДК 004.7
DOI: 10.26102/2310-6018/2023.40.1.013

Решение системных проблем при разработке и развитии WEB-приложений с помощью MODx-концепции

Д.И. Гутовский, В.Н. Добрынин, А.С. Минзов, С.А. Подгорный✉

Государственный университет «Дубна», Дубна, Российская Федерация
podgornyi.s.a@uni-dubna.ru✉

Резюме. Существующие CMS, вне зависимости от их сложности и направленности, можно разделить на две группы: одна придерживается MODx-концепции, другая – нет. В статье описаны проблемы разработки и развития WEB-приложений, характерные для стандартных CMS. Рассмотренные в статье проблемы затрагивают всех участников WEB-разработки. Проблема кардинальной смены интерфейса заключается в необходимости постоянного изменения интерфейса при изменении какого-либо функционала системы, даже не связанного с изменяемой частью интерфейса. Такая смена требует существенных затрат, как со стороны WEB-разработчиков, так и со стороны пользователей. Проблема отсутствия преемственности API в стандартных CMS связана с отсутствием принципа атомарности сущностей. Отсутствие преемственности API приводит к существенным затратам, связанным с развитием CMS. Проблема стыковки различных приложений возникает из-за сильной связности компонентов CMS, что, в свою очередь, не дает возможности гибко настроить систему для генерации содержимого с необходимыми для стыкуемых приложений параметрами. Проблема конфликта модулей WEB-приложений связана с несистематической зависимостью компонентов CMS. В проблеме безопасности WEB-приложений существенную роль играет невозможность переопределения клиентской части и негибкая файловая иерархия CMS. Проблема масштабируемости WEB-приложений прежде всего связана с отсутствием независимости от адресов файловой системы у стандартных CMS. В статье авторы дают обоснования для применения MODx-концепции коллективом разработчиков на стадиях проектирования и эксплуатации WEB-приложений, что позволяет решить вышеперечисленные проблемы.

Ключевые слова: CMS, MODx, шаблон, верстка, установка, MODx-концепция, сайт, содержимое, WEB-сайт, WEB, управление, API, HTML, CSS, JS, PHP, автоматизация.

Для цитирования: Гутовский Д.И., Добрынин В.Н., Минзов А.С., Подгорный С.А. Системные проблемы разработки и развития web-приложений. *Моделирование, оптимизация и информационные технологии.* 2023;11(1). URL: https://moitvivt.ru/ru/journal/pdf?id=1257 DOI: 10.26102/2310-6018/2023.40.1.013

Solving system-problems of WEB application development by means of the MODx-concept

D.I. Gutovskii, V.N. Dobrynin, A.S. Minzov, S.A. Podgorny✉

Dubna State University, Dubna, Russian Federation
podgornyi.s.a@uni-dubna.ru✉

Abstract. Current CMS, regardless of their complexity and orientation, can be divided into two groups: one follows the MODx-concept, the other does not. The article describes the problems of building and development of WEB applications that are typical for standard CMS. The problems regarded in this article are relevant for all participants of WEB development. The problem of drastic changes to the interface is that it has to be constantly edited when the functionality is modified even if it is not related to this part of interface. This changing requires significant costs on the part of WEB developers and users. The problem of API continuity absence in standard systems is associated with the absence of

atomic entity principle. The absence of API continuity leads to significant costs for CMS developing. The problem of connection between the different applications is due to the strong relationship between CMS components, which, in turn, does not make it possible to flexibly adjust the system to generate the content with the specific properties which are needed for joint applications. The problem of WEB application module conflicts is connected with unsystematic dependencies between CMS components. The inability to redefine the client part and the inflexible file hierarchy of standard CMS play an essential role in the problem of WEB application security. The scalability problem is primarily associated with the nonindependence of addresses on the site filesystem for standard CMS. The authors provide justifications for the application of the MODx-concept by a team of developers at the stages of WEB application design and operation to solve these problems.

Keywords: CMS, MODx, template, layout, setup, MODx-concept, site, content, WEB-site, WEB, management, API, HTML, CSS, JS, PHP, automation.

For citation: Gutovskii D.I., Dobrynin V.N., Minzov A.S., Podgorny S.A. Solving system-problems of WEB application development by means of the MODx-concept. *Modeling, Optimization and Information Technology*. 2023;11(1). URL: https://moitvivt.ru/ru/journal/pdf?id=1257 DOI: 10.26102/2310-6018/2023.40.1.013 (In Russ.).

Введение

WEB-разработка осуществляется коллективом, члены которого реализуют отдельные компоненты проекта. На стадии эксплуатации, включающей поддержку, модификацию и развитие проекта, важную роль играет потребительская аудитория (пользователи). Именно пользователи при эксплуатации WEB-приложения выявляют слабые места проекта, обнаруживают ошибки, согласовывают и оценивают (на начальной стадии эксплуатации) ожидания возможностей проекта, формируют реальные потребности и предпочтения при решении своих проблем и задач. Тем самым коллектив WEB-разработчиков и пользователи представляют собой динамическую систему, адаптация которой к сфере использования и времени эксплуатации WEB-приложения существенно зависит от гибкости и «ремонтопригодности» приложения. Использование коллективом разработчиков принципов MODx-концепции комплексно решает типовые проблемы разработки и эксплуатации WEB-приложения. На Рисунке 1 представлена схема коллектива, участвующего в разработке и дальнейшем развитии WEB-приложения.

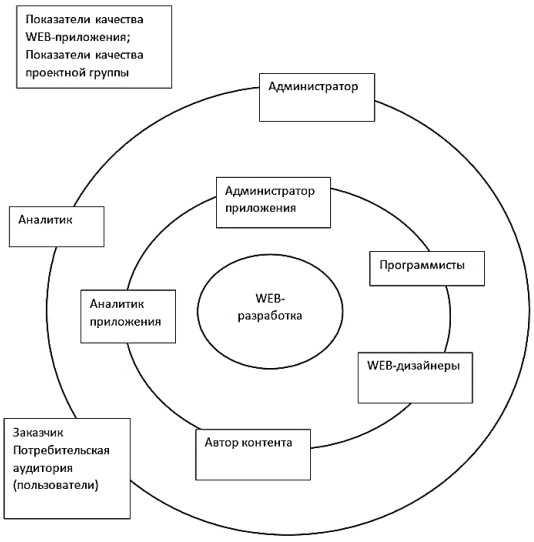

Рисунок 1 – Схема коллектива участников разработки WEB-приложения
Figure 1 – Scheme of the team developing a WEB application

Подробную информацию о MODx-концепции можно найти в статьях [1-6].

MODx-концепция

Анализ литературных источников [7-11] позволил сформулировать принципы MODx-концепции. MODx-концепция включает следующие принципы:

1. **Принцип атомарности сущностей вывода.** MODx-подобные системы работают с примитивами, из которых состоят любые сложносоставные сущности, работающие с любыми типами данных на сайте. Примером сложносоставной сущности может быть раздел информации о товаре в Интернет-магазине. Атомарные сущности, из которых состоит этот раздел – поле для ввода текста с названием товара, поле для ввода текста с описанием, поле для выбора изображения с фотографией товара и т. д. Однако, атомарные сущности, например, поля для ввода текста, ничем не отличаются при их использовании в составе разных сложносоставных сущностей. Текстовое поле в названии статьи и текстовое поле в названии товара – одна и та же атомарная сущность. Именно с такими примитивами работают MODx-подобные системы, а разработчик сам решает, какие сложносоставные сущности собирать под конкретную задачу.
2. **Принцип атомарности сущностей политики безопасности.** Данный принцип является частным случаем принципа атомарности сущностей. Как и в случае сущностей, отвечающих за вывод информации, сущности политики безопасности также должны быть примитивны, позволяя создавать любые наборы прав доступа и ограничений для любых групп пользователей, файлов и других частей сайта.
3. **Принцип преемственности.** Данный принцип требует, чтобы любые шаблоны и модули, написанные под MODx-подобную CMS, могли быть использованы без изменения их кода, при обновлении самой CMS.
4. **Принцип адаптивности**. Данный принцип обеспечивает возможность добавления на сайт «чужеродных» модулей с минимальной их переделкой.
5. **Принцип независимости от адресов файловой системы сайта.** Этот принцип дает возможность располагать различные файлы сайта без жесткой привязки к определенным местам (вплоть до распределения сайта на несколько отдельных физических серверов).
6. **Принцип равносильности панели управления и пользовательских страниц.** Данный принцип позволяет использовать единый инструментарий для реализации функционала как для страниц конечного пользователя, так и для секций панели управления.
7. **Принцип независимости функционала от интерфейса**. Этот принцип требует разделения серверной логики и пользовательских представлений (интерфейса пользователя).
8. **Принцип функциональной расширяемости.** Этот принцип обеспечивает включение новых модулей в систему с минимальными рисками для работы существующих компонентов. Это касается модулей, которые специально адаптированы под MODx-подобные CMS и модулей, написанных для стандартных CMS.
9. **Принцип декомпозиции контекстов.** Этот принцип дает возможность изолировать различные группы ресурсов, пользователей, файлов и других элементов сайта в любой степени изолированности.

Все вышеописанные принципы работают только в MODx-подобных системах. В не MODx-подобных (стандартных) CMS данные принципы работать не будут. В качестве аргументов можно привести следующее:

1. Принцип атомарности сущностей политики безопасности не может быть реализован без атомарных сущностей вывода, так как в сложносоставной сущности

невозможно выделить ее отдельные компоненты, соответственно невозможно задать индивидуальные права на них.

2. Принцип преемственности следует из принципа атомарности сущностей. Если бы атомарности не было, приходилось бы изменять API для работы с постоянно появляющимися новыми сложносоставными сущностями. Новые сущности и их параметры будут постоянно дополняться. Без примитивов невозможно предусмотреть полноценно-универсальную систему. Это можно описать формулой $N = p^q$ (N – количество сложносоставных сущностей, p – количество различных типов атомарных сущностей, q – количество параметров, включенных в сложносоставную сущность). Количество типов атомарных сущностей конечно, а лимита на количество различных полей (параметров) внутри одной сложносоставной сущности не существует. Например, если в качестве сложносоставной сущности рассмотреть товар в Интернет-магазине, то в раздел товара будут постоянно вноситься новые характеристики (например, в дополнение к фото, названию и описанию появились 3D-модели, видео-обзоры и т. д.). В итоге количество сложносоставных сущностей никогда не будет ограничено, так как не ограничено количество атомарных параметров внутри сложносоставной сущности. Таким образом, приходится постоянно изменять правила работы с имеющимися сущностями и добавлять новые правила для новых сущностей, что ведет к изменению API и к несовместимости старых модулей с новыми версиями CMS.

3. Принцип адаптивности также следует из принципа атомарности сущностей. Так как сложносоставные сущности не MODx-подобных CMS связаны, то модули, отвечающие за работу с ними, полностью зависят друг от друга.

4. Принцип независимости от адресов файловой системы сайта также невозможен без принципа атомарности сущностей. Иначе, пришлось бы предусмотреть все возможные варианты расположения всех групп компонентов и сущностей. Однако, если эти сущности не являются атомарными, то их может быть сколь-угодно много. Соответственно, функций, отвечающих за работу с сущностями, может быть также много. Следовательно, нельзя заранее предсказать, от каких параметров будет зависеть работа той или иной функции.

5. Принцип равносильности панели управления и пользовательских страниц также использует принцип атомарности сущностей. На основе атомарных сущностей строятся, как шаблоны интерфейса пользователей, так и панели управления. Они примитивны и не зависят от области их применения.

6. Принцип независимости функционала от интерфейса следует из принципа атомарности сущностей. При отсутствии атомарности сущностей за работу с разными частями интерфейса отвечали бы разные функции. В общем случае, функциональные модули, предназначенные для решения различных задач, могут иметь разные представления (интерфейсы пользователя). При работе со сложносоставными сущностями часто приходится использовать различные модули не по прямому назначению. Это вызвано сложностью API у не MODx-подобных CMS. Например, в CMS WordPress есть боковая панель. Такая панель имеет ряд ограничений, в частности таких, как повторение содержимого на всех страницах, которые ее поддерживают. Если понадобится добавить параметр для снятия этого ограничения, то наиболее частым решением является создание произвольного типа записей. В терминах WordPress – это уже не боковая панель, а записи. Интерфейсы для работы с записями в штатной панели управления WordPress различны. Таким образом, при добавлении одной функции изменился интерфейс. Следовательно, без принципа атомарности, интерфейс пользователя будет зависеть от используемых сущностей и функций CMS для работы с ними.

7. Принцип функциональной расширяемости следует из принципа атомарности сущностей. API не MODx-подобных CMS имеют сильную связность. Тем самым, возрастает риск вывода из строя функционала при добавлении нового или изменении существующего функционала.

8. Принцип декомпозиции контекстов основывается на принципе атомарности сущностей. Если сущности неатомарные, разделение их групп вызвано ограничениями API. Ограничения API не позволяют разделить группы сущностей по произвольному набору параметров.

Основные проблемы стандартных WEB-приложений

Одной из наиболее распространенных проблем стандартных CMS является проблема кардинальной смены интерфейса у разных версий системы. Эта проблема может быть выражена в разной степени в зависимости от конкретной CMS, но полностью избежать вынужденной смены интерфейса не получится, если не соблюдать MODx-концепцию. Данная проблема усложняет эксплуатацию системы, заставляя пользователей (не обязанных иметь специальную подготовку) переучиваться на использование новой версии системы.

Проблема смены интерфейса, главным образом, связана с отсутствием принципа атомарности сущностей у стандартных CMS, из-за чего компоненты системы сильно связаны между собой, и изменения одних модулей системы часто затрагивают и другие. Вместе с тем интерфейс системы приходится менять из-за особенностей API используемой CMS. В стандартных CMS часто происходит смешение серверной логики и пользовательских интерфейсов. Для реализации тех или иных задач используют конкретные модули, которые (в стандартных CMS) часто несут на себе и интерфейс пользователя. Однако API конкретного модуля может не иметь функционала, необходимого для решения обновленных задач. Таким образом, используемые ранее модули становятся не актуальными для реализации добавленного функционала. Приходится использовать другие модули, влекущие и другой пользовательский интерфейс.

Проблема преемственности API также перекликается с проблемой смены интерфейса. Если проблема смены интерфейса, по большей части, касалась пользователей системы, то проблема постоянного изменения API усложняет разработку приложения. Вынужденной смены API также не избежать в стандартных CMS из-за отсутствия атомарности сущностей. Различные сущности системы, отвечающие за работу с разными типами содержимого, необходимо постоянно обновлять, добавляя новые параметры. В итоге сохранить API, отвечающее за работу с предыдущими сущностями не получится, так как не удастся избежать их постоянного изменения.

В MODx-подобных системах нет проблемы смены интерфейса пользователя и API системы. Так как MODx-подобные CMS работают с атомарными сущностями, которые базовые и неизменные. Тем самым нет необходимости менять API, отвечающее за работу с ними. Разработчик добавляет новый функционал, используя тот же API, что и до этого, а добавление интерфейса возникнет только для работы с новыми функциями.

Также, при разработке и развитии стандартных WEB-приложений, возникает проблема их стыковки с другими приложениями. Это связано не только с вышеупомянутыми изменениями API, но и с необходимостью использования модулей системы не по прямому назначению. Например, в какой-нибудь CMS есть опции для работы с элементами боковой панели и для создания произвольных типов записей. Допустим, что функционала боковой панели недостаточно, а в модуле для произвольных типов записей нужный функционал присутствует. В итоге использование модуля

произвольных типов записей для реализации боковых панелей может оказаться целесообразнее, чем использование модуля, предназначенного специально для боковых панелей. Однако, внешние приложения и дополнения для системы могут иметь четкую привязку к API конкретных модулей и при получении неожидаемых ответов от других модулей, они работать не будут.

Проблема стыковки приложений напрямую связана с проблемой переопределения, которая также характерна для стандартных CMS. Из-за отсутствия разделения серверной логики и пользовательских интерфейсов, часто невозможно четко отделить модули ядра системы от конечных модулей. Модули стандартных систем сильно связаны, что не позволяет сделать API достаточно гибким, учитывая все параметры, необходимые разработчику для корректной генерации конкретного участка клиентской части. Например, в стандартной CMS может быть модуль, отвечающий за генерацию меню сайта. Данный модуль генерирует не только названия пунктов меню и их ссылки, но и всю клиентскую «оснастку» вместе с тэгами и их атрибутами (классами, ID и т. д.). Естественно, в созданном шаблоне могут не учитываться особенности CMS, под которую предполагается адаптировать шаблон. В этом случае модуль CMS должен предусматривать функционал, позволяющий переопределить фрагменты генерируемой клиентской части, для единой стилистики кода верстки и корректной семантики. Например, роботы-сборщики информации с сайта могут ориентироваться на конкретные классы, поисковые системы и могут некорректно интерпретировать информацию с сайта, если сайт использует семантически-неверные тэги и т. д. Все это приводит к проблемам, связанным с необходимостью переделки модуля. Однако из-за «неповоротливости» API стандартных CMS такая переделка часто бывает нецелесообразной.

Также при несоблюдении MODx-концепции часто возникают конфликты между различными модулями CMS, ее дополнениями и плагинами как на стороне сервера, так и в клиентской части. Простейшим примером может стать необходимость подключения разных версий библиотеки jQuery на странице, генерируемой разными дополнениями для CMS, например, плагином для создания слайдера и модулем стилизации элементов формы. Из-за проблем переопределения, далеко не всегда можно «заставить» модуль использовать нужную версию библиотеки и придется изменять код модуля. Однако с изменяемым модулем могут быть связаны другие части приложения, что вызовет их некорректную работу. Естественно, что характерная для стандартных CMS сильная связность компонентов может возникать как на стороне клиента, так и на стороне сервера. Понятно, что проблемы конфликта модулей не ограничиваются проблемой переопределения клиентской части.

Проблемы безопасности тоже часто возникают при использовании стандартных CMS. Во-первых, из-за отсутствия полноценной преемственности версий не MODx-подобных CMS приходится несвоевременно выполнять их обновление. Количество и сложность модифицируемого функционала могут быть настолько высоки, что принимается решение об использовании старой версии CMS, несмотря на то что ее уязвимости известны. Однако дело не только в отсутствии преемственности.

Проблема переопределения клиентской части также несет потенциальную угрозу безопасности WEB-приложения. В общем случае, нельзя однозначно сказать об использовании конкретных технологий серверной части без получения доступа к серверу. Однако, по характерным признакам, оставляемым CMS при генерации WEB-страниц, можно предположить, какая CMS и другие компоненты серверной части WEB-приложения используются. Например, можно просмотреть генерируемые в определенных местах, особенно в тех, в которых больше произвольного содержимого и переопределение не было проведено заранее под конкретный вариант HTML-верстки.

Таким образом, даже если не касаться произвольного содержимого, очевидно, что проблемы переопределения часто влекут за собой выдачу излишней информации на сторону клиента.

Также, важную роль в обеспечении безопасности WEB-приложения играет его файловая иерархия. Во-первых, по характерным местам расположения определенных файлов тоже можно попытаться вычислить модель используемой CMS, а во-вторых, настройка аспектов, связанных с безопасностью приложения, может быть возложена и на окружение этого приложения, например серверы, на которых работает WEB-приложение. Серверы можно настроить отдельно от приложения, возложив на них часть функций по регулировки прав доступа на уровне ОС, не CMS, автоматической проверке на вирусы загружаемого содержимого без вмешательства в само WEB-приложение и т. д. Естественно, это возможно только в рамках MODx-концепции, так как в ней присутствует принцип независимости от адресов файловой системы сайта: можно распределять файлы, в том числе системные, без четкой привязки к конкретному местоположению, вплоть до разнесения приложения на несколько физических серверов.

Кроме того, проблемы безопасности в стандартных WEB-приложениях возникают из-за отсутствия атомарности на уровне сущностей политики безопасности. Нельзя произвольно создать и настроить любую группу пользователей, любые права доступа на различные участки сайта (на любые файлы, части кода, адреса и т. д.). Гибкость политики безопасности в стандартных CMS ограничивается набором готовых ролей и групп пользователей, с заранее предусмотренным набором прав. Однако сущности политики безопасности в стандартных CMS не могут быть атомарными, так как сущности вывода сложносоставные. Если есть тип содержимого, например, для работы с профилем товара в Интернет-магазине, то нельзя отсоединить его отдельную любую часть, задав ей независимые права, расположив ее в отдельной секции сайта, изолировав в отдельный контекст и т. д. В итоге сильная связность компонентов в стандартных WEB-приложениях наносит вред и их безопасности.

Для стандартных WEB-приложений характерна также проблема масштабируемости. Это явно видно хотя бы из отсутствия принципа независимости от адресов файловой системы сайта. Как было отмечено, при соблюдении MODx-концепции можно распределять сайт по разным площадкам различной степени автономности и изолированности. Эти площадки, представляющие в сумме окружение WEB-приложения, могут настраиваться отдельно, независимо от самого приложения. Не требуется устанавливать дополнения или писать какой-то отдельный функционал для CMS при автоматизации и распределении задач обслуживания создаваемого приложения. Пусть, например, требуется организовать автоматическое копирование особо важных частей сайта. Конечно, это можно сделать на уровне CMS, но есть возможность, как и в случае с автоматической проверкой на вирусы, произвести эти настройки на отдельной площадке, а на уровне WEB-приложения просто задать адрес этой площадки для хранения содержимого нужной секции сайта. Таким образом, WEB-приложение никак не будет связано с задачей резервного копирования и вообще не будет знать о его ходе, ведь оно будет осуществляться уровнем ниже, за счет настроек сервера. Таким способом можно настраивать WEB-приложение под площадки, на которых оно размещается, а можно и сами площадки, независимо от WEB-приложения. Безусловно, это не только облегчит масштабируемость и переносимость разрабатываемого приложения, но и даст возможность сэкономить время на разработку и отладку, за счет распараллеливания задач и грамотного распределения обязанностей между специалистами.

В результате анализа вышеописанных проблем предложен подход преобразования стандартных WEB-приложений в MODx подобные [2, 3].

Решение основных проблем разработки WEB-приложений

Из описания основных проблем стандартных WEB-приложений становится понятным, что их перевод на MODx-концепцию решил бы эти проблемы. Для этого требуется выполнить перевод на MODx как можно более плавно, во избежание проблем работы инфраструктуры, зависящей от модифицируемых приложений. Плавный переход могла бы обеспечить автоматизированная система трансформации шаблонов из специфических API стандартных CMS в шаблоны для MODx-подобных систем. Такая система трансформации должна иметь 2 основных модуля: первый отвечает за перевод шаблона из специфического API конкретной стандартный CMS в унифицированный MODx-подобный язык, второй модуль позволяет автоматически установить MODx-шаблон на MODx-подобную CMS.

Модуль перевода из специфических API утратит свою актуальность после того, как MODx-стандарт станет общепринятым для всех CMS, а модуль автоустановки останется актуальным, позволяя любому пользователю, не имеющему навыков программирования, устанавливать готовый шаблоны на MODx. Появление автоустановщика шаблонов для MODx окончательно лишает смысла не MODx-подобный подход, однако, речь не идет о попытке искоренения каких-либо CMS, языков программирования или других технологий. Речь идет только о стандартизации API CMS, беря за основу API MODx. В то же время это не затрагивает ядро CMS, или язык программирования, на котором она написана. Каждая CMS по-прежнему будет иметь свои сильные стороны при решении конкретных групп задач, а переход с одной CMS на другую станет легким.

Далее приведен алгоритм автоматического установщика MODx-шаблонов:

1. **Подключение к БД сайта.** Происходит соединение с БД сайта, на который идет установка шаблона. Если соединение не удалось, то происходит вывод ошибки и выход из системы. Если соединение успешно, то выполняется переход к следующему шагу.

2. **Определение диалекта CMS.** Для определения диалекта языка MODx-подобной CMS из указанного места расположения шаблона происходит линейное сканирование всех его файлов.

3. **Занесение шаблона в БД сайта.**

a. Происходит линейное сканирование каждого файла шаблона, кроме файла зависимостей.

b. Из таблицы сайта вычисляется максимальный ID компонента, который относится к той же группе, что и добавляемые из файла компоненты. То есть, если файл шаблона содержит информацию о добавлении, например, дополнительных полей, то перед добавлением самих полей из этого файла сначала требуется узнать, какой максимальный номер дополнительного поля (в терминах MODx) есть на сайте. Групповая принадлежность определяется по корневому тэгу в файле шаблона.

c. Поля добавляются, начиная с максимального ID плюс 1. Аналогичная ситуация и с другими группами компонентов: чанками, сниппетами, и т. д.

4. **Выявление конфликтов компонентов разных шаблонов.** Отслеживание конфликтов, например, совпадение имен дополнительных полей, происходит по ответам от сервера БД при добавлении компонента. В БД MODx-подобных CMS указаны ограничения на то, какие имена и у каких компонентов могут совпадать. Если происходит конфликт, сервер БД выдает ошибку при добавлении соответствующих записей. При возникновении конфликтов имена конфликтующих компонентов и их групповая принадлежность запоминаются во временный файл (файл конфликтов).

5. **Контроль конфликтов.** Каждый добавляемый компонент сверяется со временным файлом конфликтов. Если имя компонента и его групповая принадлежность совпадают, то данный компонент пропускается.

6. **Модификация имени компонента шаблона.**

a. Когда все компоненты кроме конфликтующих добавлены, происходит попытка модификации имени компонента путем увеличения порядкового номера.

b. Повторяется попытка адаптации данного компонента. Если конфликт возникает снова, происходит повторение модификации параметров и попытка повторного добавления. Это повторяется до тех пор, пока компонент не будет добавлен успешно.

c. После успешного добавления компонента в файл конфликтов вместо старого имени конфликтующего компонента записывается новое имя. После этого происходит дальнейшее сканирование и добавление компонентов из файла конфликтов (с их модификацией).

7. **Удаление из БД сайта.** Когда все конфликты разрешены и шаблон добавлен полностью, он удаляется из БД сайта. При возникновении хотя бы одного конфликта вся установка будет произведена повторно.

8. **Устранение конфликтов в коде.** Повторное линейное сканирование каждого компонента из файла шаблона и сравнение его кода с файлом конфликтов. Если будет обнаружен вызов компонента, присутствующего в файле конфликтов, то его вызов заменяется на модифицированный код, с подменой имени на новое из файла конфликтов. Если в коде шаблона происходит создание конфликтующего компонента, то его имя заменяется на новое аналогичным образом. В итоге все конструкции, в которых происходит создание или вызов конфликтующего компонента, заменяются на бесконфликтные.

9. **Повторная установка шаблона.** После полной модификации всех данных, полученных из файлов шаблона, происходит повторная установка шаблона с использованием исправленных данных.

10. **Сбор информации о необходимых дополнениях.** После успешной установки шаблонов происходит сбор информации об установленных дополнениях для CMS.

11. **Разрешение зависимостей.** Линейно сканируется файл зависимостей шаблона. Если в списке установленных дополнений, полученном на шаге 10, нет какого-либо дополнения или его версия ниже, чем указано в файле зависимостей, то пользователь получает рекомендацию о необходимости доустановить требуемое дополнение. Если версия дополнения неважна, то она не указывается разработчиком в файле зависимостей шаблона. В этом случае проверяется только имя дополнения, и если оно есть в списке установленных, то, вне зависимости от версии, пользователь не получает рекомендацию о доустановке.

Приведенный алгоритм поясняет только модуль автоустановки шаблонов. Модули, отвечающие за перевод из специфических API, разрабатываются отдельными группами разработчиков, имеющих отношение к конкретной переводимой CMS, и база этих модулей постоянно пополняется новыми вариантами стандартных CMS. Постепенно, все WEB-приложения, даже построенные на индивидуальных для конкретного проекта CMS, станут MODx-подобными, и каждое WEB-приложение будет иметь все преимущества MODx-концепции, а WEB-разработка перейдет на качественно-новый уровень, без необходимости когда-либо возвращаться к не MODx-подходу работы CMS.

Заключение

Описаны проблемы, характерные для стандартных WEB-приложений построенных на основе не MODx-подобных CMS. Описанные проблемы в разной степени проявляются в стандартных CMS. Подавляющее большинство проблем современных WEB-приложений вытекают из приведенных выше и решаются путем использования MODx-подобного подходах. Полноценное решение рассмотренных проблем невозможно вне рамок MODx-концепции [1-7].

Применение MODx-концепции не только повысит эффективность разработки WEB-приложения, но и упростит его поддержку, дальнейшее развитие, обслуживание и эксплуатацию. Отметим, что все преимущества MODx-концепции характерны как для самой CMS MODx, так и для всех MODx-подобных CMS. Благодаря гибкости API таких систем, переход между их различными моделями не представляет особых сложностей, что позволит, при необходимости, сменить CMS, заменив ядро, или даже язык программирования серверной части, сохраняя функционал всех конечных модулей, интерфейс пользователя и стыковку с другими приложениями. Тем самым MODx-концепция является решением системных проблем WEB-разработки.

СПИСОК ИСТОЧНИКОВ

1. Гутовский Д.И., Филозова И.А. Общие рекомендации для проектирования и реализации WEB-сайтов. *SCI-Article*. 2016;34(6). Доступно по: https://sci-article.ru/stat.php?i=1466521208 (дата обращения: 27.10.2022).
2. Гутовский Д.И., Добрынин В.Н. Определение основных концепций CMS. *Системный анализ в науке и образовании.* 2019;4. Доступно по: http://sanse.ru/download/371 (дата обращения: 27.10.2022).
3. Гутовский Д.И., Добрынин В.Н. Перспективы развития CMS. *Системный анализ в науке и образовании.* 2020;4. Доступно по: https://sanse.ru/index.php/sanse/article/view/209 (дата обращения: 27.10.2022).
4. Гутовский Д.И., Добрынин В.Н. Задачи оптимизации при разработке WEB-приложений, в рамках MODx-концепции. *Системный анализ в науке и образовании.* 2022;1. Доступно по: https://sanse.ru/index.php/sanse/article/view/524 (дата обращения: 27.10.2022).
5. Шпак Ю. *Web-разработка средствами MODx*. М.: МК – Пресс; 2012. 400 с.
6. Tanjung H., Gestel B. van, Andjarwirawan J. Migrating autolux website from MODX to phalconphp with further developments (Search Engine Optimization and Mobile Application Support Implementation). *Informatika*. 2017;14(1):42–46.
7. Игнатов П.И., Ефромеева Е.В. Анализ систем управления контентом. *Проблемы современных интеграционных процессов и пути их решения*. 2019:82–85.
8. Bob Ray. *MODX: The Official Guide*. MODX Press; 2011. 757 p.
9. Shawn Wilkerson W. *MODX Revolution – Building the Web Your Way*. Sanity Press; 2012. 622 p.
10. John, Antano Solar. *MODx web development: building dynamic web sites with the PHP application framework and CMS*. Birmingham, U.K.: Packt Pub; 2009. 257 p.
11. John, Antano Solar. *Modx 2.0 Web Development*. Packt Publishing Ltd; 2011. 288 p.

REFERENCES

1. Gutovskij D.I., Filozova I.A. Obshchie rekomendacii dlya proektirovaniya i realizacii WEB-sajtov. *SCI-Article.* 2016;34(6). (In Russ.). Available from: https://sci-article.ru/stat.php?i=1466521208 (accessed on 27.10.2022).
2. Gutovskij D.I., Dobrynin V.N. Opredelenie osnovnyh koncepcij CMS. *Sistemnyj analiz v nauke i obrazovanii = System Analysis in Science and Education*. 2019;4. Available at: http://sanse.ru/download/371 (accessed on 27.10.2022). (In Russ.)
3. Gutovskij D.I., Dobrynin V.N. Perspektivy razvitiya CMS. *Sistemnyj analiz v nauke i obrazovanii = System Analysis in Science and Education*. 2020;4. (In Russ.). Available from: https://sanse.ru/index.php/sanse/article/view/209 (accessed on 27.10.2022).
4. Gutovskij D.I., Dobrynin V.N. Zadachi optimizacii pri razrabotke WEB-prilozhenij, v ramkah MODx-koncepcii. *Sistemnyj analiz v nauke i obrazovanii = System Analysis in Science and Education*. 2022;1. (In Russ.). Available from: https://sanse.ru/index.php/sanse/article/view/524 (accessed on 27.10.2022).
5. Shpak Yu. *Web-razrabotka sredstvami MODx*. M.: MK – Press; 2012. 400 p. (In Russ.)
6. Tanjung H., Gestel B. van, Andjarwirawan J. Migrating autolux website from MODX to phalconphp with further developments (Search Engine Optimization and Mobile Application Support Implementation). *Informatika*. 2017;14(1):42–46.
7. Ignatov P.I., Efromeeva E. V. Analiz sistem upravleniya kontentom. *Problemy sovremennykh integratsionnykh protsessov i puti ikh resheniya*. 2019:82–85. (In Russ.).
8. Bob Ray. *MODX: The Official Guide*. MODX Press; 2011. 757 p.
9. Shawn Wilkerson W. *MODX Revolution – Building the Web Your Way*. Sanity Press; 2012. 622 p.
10. John, Antano Solar. *MODx web development: building dynamic web sites with the PHP application framework and CMS*. Birmingham, U.K.: Packt Pub; 2009. 257 p.
11. John, Antano Solar. *Modx 2.0 Web Development*. Packt Publishing Ltd; 2011. 288 p.

ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

Дмитрий Игоревич Гутовский, аспирант кафедры системного анализа и управления, Государственный университет «Дубна», Институт системного анализа и управления, Дубна, Российская Федерация.
e-mail: dimgut@uni-dubna.ru

Dmitrii Igorevich Gutovskii, Postgraduate Student, the Department of System Analysis and Management, Dubna State University, Institute of System Analysis and Management, Dubna, Russian Federation.

Владимир Николаевич Добрынин, кандидат технических наук, старший научный сотрудник, профессор кафедры системного анализа и управления, Государственный университет «Дубна», Институт системного анализа и управления, Дубна, Российская Федерация.
e-mail: arbatsolo@uni-dubna.ru

Vladimir Nikolaevich Dobrynin, Candidate of Technical Sciences, Senior Researcher, Professor at the Department of System Analysis and Management, Dubna State University, Institute of System Analysis and Management, Dubna, Russian Federation.

Анатолий Степанович Минзов, доктор технических наук, профессор кафедры информационных технологий, Государственный университет «Дубна», Институт системного анализа и управления, Дубна, Российская Федерация.

Anatolii Stepanovich Minzov, Doctor of Technical Sciences, Professor at the Department of Information Technology, Dubna State University, Institute of System Analysis and Management, Dubna, Russian Federation.

e-mail: minzovas@uni-dubna.ru

Сергей Александрович Подгорный, доктор технических наук, проректор по цифровому развитию, Государственный университет «Дубна», Дубна, Российская Федерация.
e-mail: podgornyi.s.a@uni-dubna.ru

Sergey Aleksandrovich Podgorny, Doctor of Technical Sciences, Vice-Rector for Digital Development, Dubna State University, Dubna, Russian Federation.

Статья поступила в редакцию 11.11.2022; одобрена после рецензирования 21.01.2023; принята к публикации 16.02.2023.

The article was submitted 11.11.2022; approved after reviewing 21.01.2023; accepted for publication 16.02.2023.