Лабораторная работа №4

Реинжиниринг – процесс привнесения новой функциональности либо устранения ошибок, путем внедрения серьезных изменений в существующее программное обеспечение. Процесс реинжиниринга по своей сложности превосходит рефакторинг и бытует мнение, что когда система подходит к порогу реинжиниринга, проще стереть все и написать с нуля. Рассмотрим основные предпосылки к реинжинирингу.

Первой и основной причиной для проведения реинжиниринга является необходимость добавления функционала в систему. Например, при распространении способов безналичной оплаты, потребовалось добавить функционал взаимодействия кассового аппарата с терминалом. Затем внедрить протоколы считывания терминалом данных карты с помощью технологий PayPass и подобных. Все это примеры расширения функционала, который не предусматривался в системе изначально, но без которого можно попасть в аутсайдеры рынка.

Второй возможной причиной является необходимость оптимизации работы системы. Это может быть актуально для крупных предприятий с самописным программным обеспечением. Писалось оно, к примеру, под Windows 98 и работало на ней вполне себе неплохо. Затем программа пережила переезд на новую систему, потом еще на одну и где-то в районе третьей миграции возникла проблема невозможности запуска данного ПО на современных машинах, из-за разницы архитектур. Какой наиболее простой выход из этой ситуации? Установить эмулятор старой системы и продолжить запускать программу из-под нее. Но каждая подобная надстройка снижает эффективность всего процесса в целом, так как эмуляторы достаточно затратны по ресурсам, из чего возникает необходимость адаптации программы под реалии нового окружения.

При проведении реинжиниринга необходимо выполнить два ключевых этапа: выделение актеров, т.е. основных действующих сущностей в системе, и построения общей функциональной схемы работы нашей системы.

У каждого актера есть набор действий и свойств в системе. После ее преобразования часть функций может уйти или видоизмениться, часть функций – добавиться. Все эти преобразования должны быть четко зафиксированы в процессе, так как потеря функционала в процессе недопустима, за исключением случаев, когда добавляемый функционал напрямую противоречит старому поведению. Помимо этого, возможно добавление новых актеров, чье взаимодействие с существующими сущностями должно быть строго задекларировано на этапе проектирования.

Общая функциональная схема несет в себе помимо связи актеров друг с другом также особенности реализации внутренней логики. За ее основу обычно берется модель взаимодействия исполнителей и строится она на основании результатов изучения алгоритмов поведения действующей системы.

Задание:

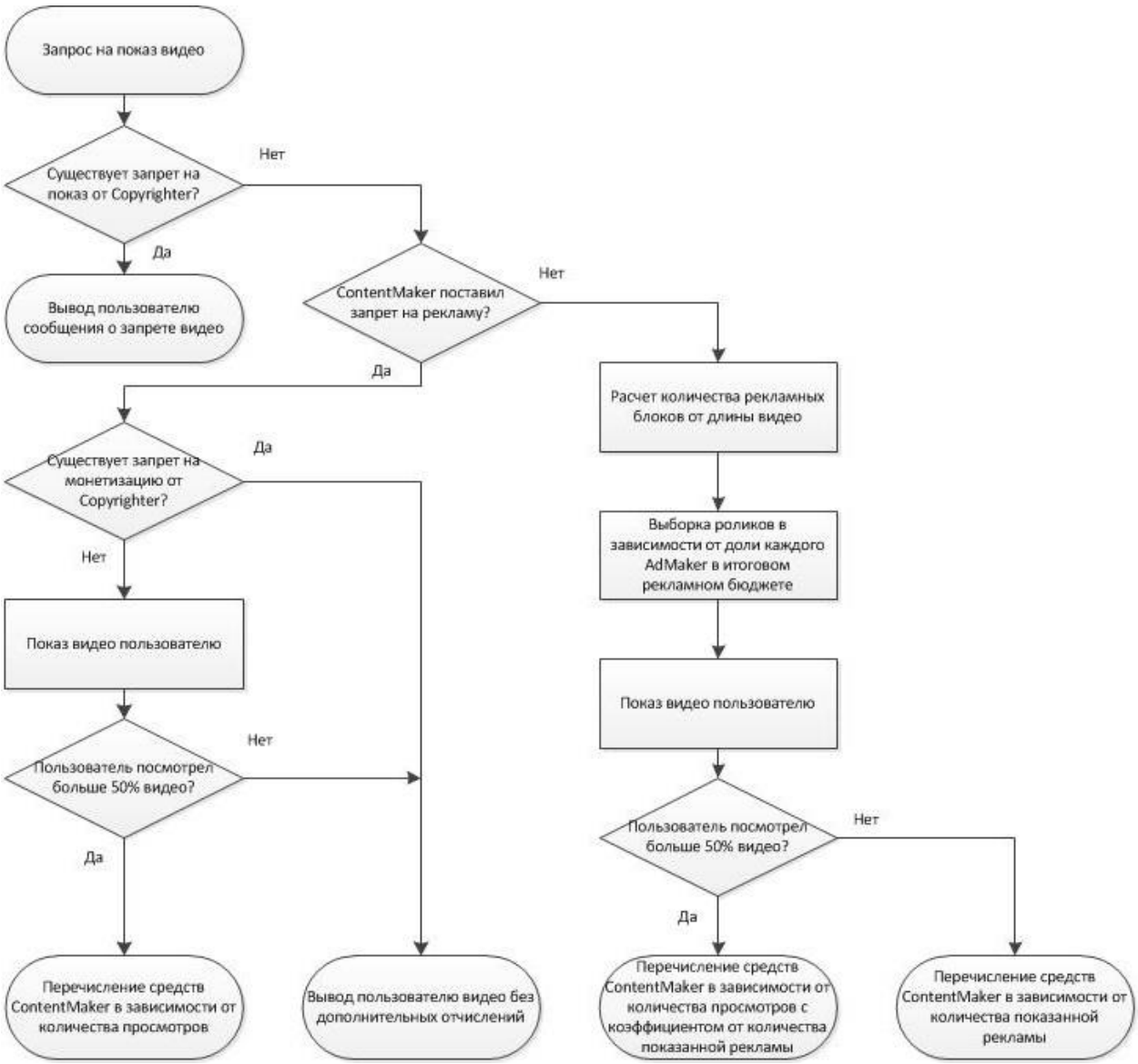
Вводная часть:

Вы разработчик видеохостинга*. У Вас есть несколько основных сущностей:

- AdMaker – актер, производящий рекламу и платящий за возможность и частоту ее показа на хостинге. От внесенной суммы в данном месяце и ее отношения к общему рекламному бюджету будет определяться частота показа рекламного ролика каждого объекта.
- Copyrighter – актер, обладающий авторским правом на определенный контент. Может запросить снятие монетизации либо полный запрет видео, содержащего его контент.
- ContentMaker – актер, производящий содержимое Вашего хостинга. Может поставить запрет на трансляцию рекламы в своем контенте (платно). Может подвергаться запретам со стороны Copyrighter. При оформлении договора на показ рекламы, имеет доход с каждого показанного ролика. Имеет безусловный доход, за исключением запрета монетизации, за каждый просмотр своего видео более чем на 50%.

Общая схема функционирования Вашего хостинга приведена ниже:

* - все события и действующие лица вымышлены. Любые совпадения случайны.



Описание дальнейших событий:

С приходом сущности «эффективный менеджер» в Вашу компанию, в головах руководства созревает мысль, что для такого популярного хостинга, как Ваш, текущие доходы слишком малы и возникает идея добавить следующий функционал, для внедрения которого необходимо провести реинжиниринг:

- Ограничить у ContentMaker возможность исключать рекламу из своих видео, оставив возможность сократить количество рекламных вставок до двух (в начале и в конце видео);
- Предоставить пользователю хостинга возможность оформления премиальной подписки, исключающей рекламу из видео;
- Предоставить возможность AdMaker оплачивать строго заданное количество показов рекламного контента в рамках месяца, с оставлением механики процента от общего рекламного бюджета;

- Предоставить Copyrighter право давать пользователю временный или неограниченный по времени доступ к своему контенту за соответствующую плату.

Выходные данные:

Итогом лабораторной работы должен стать список сущностей с привязанными к ним функциями и общая схема работы описанного проекта, выполненная сходным образом с тем, что демонстрировалось выше.