20 февраля 2023 г.

Отчёт о проведённом аудите команды, процессов и обучении

Все имена и подробности изменены, настоящий отчёт не содержит чьих-либо персональных и иных конфиденциальных данных

В соответствии с договором на оказание услуг № 1734-02 от 10 февраля 2023 года между ООО «Группа компаний “Симбио”» и ООО «Транснефть-Цифровизация» 16 и 17 февраля в офисе компании в городе Новосибирске были проведены встречи со следующими сотрудниками отделов разработки ПО и системного администрирования:

- Фёдоров Виктор — технический директор;
- Малюков Геннадий — старший системный архитектор;
- Валиев Артур — тимлид команды разработки;
- Портнов Георгий — тимлид команды разработки;
- Сергеев Антон — ведущий разработчик;
- Бекетов Павел — разработчик;
- Самсонова Ирина — руководитель группы системных аналитиков;
- Парамонова Антонина — младший системный аналитик;
- Окунев Дмитрий — младший системный аналитик;
- Войнаревич Дмитрий — младший разработчик;
- Досымбеков Ерлан — младший разработчик;
- Сафронов Пётр — руководитель отдела системного администрирования.

В рамках проведённых встреч в целях оценки текущего состояния команды и процессов руководителям и сотрудникам задавались различные вопросы по их обязанностям, функционалу и содержанию выполняемой ими работы, по организации процессов обработки задач, процесса разработки в целом, а также по наличию у них предложений по улучшению этих процессов.

Текущее состояние команды и процессов

Продукты компании и команда разработки этих продуктов росли и развивались много лет естественным образом по мере роста количества задач и потребностей. Данный рост был ориентирован на наиболее быстрое выполнение задач и, прежде всего, задач поддержки текущих клиентов и срочной доработки продуктов. При этом рост оказался в значительной мере несистемным и хаотичным с точки зрения организации процессов разработки и архитектуры разрабатываемых продуктов. В результате накопились

перечисленные ниже проблемы системного характера, свойственные быстрорастущим компаниям, ориентированным на разработку и поддержку программных продуктов.

Первые четыре проблемы являются достаточно критичными и требующими решения (под номерами 1–4), а ещё две — обсуждаемыми и не столь критичными для компании на данном этапе развития (под номерами 5–6):

1. **Отсутствие учёта, планирования и контроля трудозатрат.** В командах отсутствуют методики и практики предварительной оценки трудозатрат на решаемые задачи, учёт уже осуществлённых трудозатрат, а также планирование трудозатрат на новые задачи и проекты. В результате работа команды недостаточно прозрачна, невозможно точно определить, на какие категории задач и на какие проекты затрачено то или иное количество ресурсов.

2. **Высокий уровень “bus factor”.** Многие сотрудники компании, особенно руководители, являются носителями уникальных знаний о продуктах, и значительное количество этой информации не задокументировано. Существующая архитектура продуктов не оценивалась, но по словесным описаниям сотрудников она представляется достаточно сложной и трудно поддерживаемой. В связи с этим, в частности, решение значительного количества задач разработчиками требует консультаций с носителями соответствующих знаний, а поиск и диагностика ряда проблем требуют значительных затрат времени. К примеру, по словам одного из разработчиков, для того, чтобы идентифицировать проблему и определить место в коде, в котором требуется внести исправления и/или доработки по той или иной задаче часто уходит до одного-двух рабочих дней.

3. **Длительный вход в продукт новых сотрудников.** Из-за недостаточно понятной архитектуры и отсутствия полной и понятной ознакомительной документации время адаптации новых сотрудников занимает часто несколько месяцев, иногда до полугода.

4. **Отсутствие понятной системы грейдирования.** В компании часто возникают вопросы относительно того, является ли уровень заработной платы того или иного сотрудника оптимальным с точки зрения его уровня, заработных плат других сотрудников и рыночных заработных плат аналогичных специалистов. Эти вопросы возникают при найме новых сотрудников, повышении существующих сотрудников, а также уходе сотрудников на работу в другие компании, и руководителям на них бывает сложно отвечать из-за отсутствия единой и понятной системы.

5. **Нетипично малое количество middle-специалистов.** В компании достаточно большое количество специалистов уровня senior и много специалистов уровня junior, при этом явно недостаточно специалистов уровня middle. Обычно в компаниях, ориентированных на разработку программного обеспечения, соотношение количества специалистов уровней junior, middle и senior составляет 20%, 60% и 20% соответственно, поскольку middle-специалисты являются основной “рабочей силой”, выполняющей большую часть работы. В то же время, существующее распределение выглядит естественным, если считать, что компания ориентирована больше на поддержку существующих продуктов, а не на их доработку, развитие и создание новых продуктов, поскольку задачи поддержки проще поручать начинающим специалистам, для которых они, с одной стороны, проще, чем задачи по созданию нового функционала, а с другой стороны, сильнее их развивают, чем плановые задачи. В связи с этим данная проблема не является критичной и не требует немедленного решения.

6. **Недостаточный уровень нематериальной мотивации.** На встречах сотрудники часто высказывали пожелания не только по улучшению процессов работы, но и относительно того, чего им не хватает в плане нематериальной мотивации. Таких пожеланий высказывалось довольно много. При этом фактический уровень мотивации сотрудников достаточно высок, как у начинающих специалистов уровня junior, которые недавно пришли в компанию, очень вовлечены в работу и вдохновлены ей, так и у сотрудников уровня senior, которые работают в компании много лет и не собираются менять работу. В связи с этим данная проблема не является критичной и не требует срочного решения.

В целом по итогу проведённых встреч все опрошенные сотрудники и руководители показали себя конструктивно настроенными и вовлечёнными, активно проявляющими инициативу, “болеющими за дело” и серьёзно нацеленными на достижение результатов.

Более того, сотрудниками и руководителями за последний год сделано множество важных действий, которые направлены на решение перечисленных выше проблем в своих группах, но данных действий явно недостаточно, принимаемые в связи с этими улучшениями решения необходимо поддерживать “сверху” и применять более системно.

В связи с вышеперечисленным ключевой проблемой отдела разработки в целом можно считать **несистемность**.

Рекомендации по решению проблем и обучение

Во второй день работы (17 февраля) по итогам встреч с сотрудниками проведено их обучение, в рамках которого рассказано об общепринятых практиках и техниках, применяемых в других компаниях, благодаря которым удаётся систематизировать их работу, решать или даже предотвращать большинство перечисленных выше проблем. Также в рамках обучения высказан ряд рекомендаций по внедрению этих техник.

Важно отметить, что многие из этих практик и техник уже используются в отдельных группах, но не применяются системно во всём отделе разработки. Ниже перечислены данные практики, техники и соответствующие рекомендации. При этом внедрение данных практик и техник является опциональным. Задачей обучения было лишь рассказать о них и предложить их начать применять.

1. Проектное управление

Причиной проблемы отсутствия системного учёта, планирования и контроля трудозатрат является отсутствие системного проектного управления.

Проектное управление имеет смысл внедрять в тот момент, когда в компании ведётся строгий учёт самих задач, что уже успешно делается на протяжении длительного времени. В настоящий момент подавляющее большинство задач, выполняемых командами, фиксируется в единой системе постановки задач Jira, что делает внедрение проектного управления не только возможным, но и достаточно лёгким.

Обычно в компаниях управление проектами осуществляет специально выделенный или нанятый менеджер (руководитель) проектов. Его задача — планировать выполнение тех или иных проектов — рассчитывать сроки и трудозатраты, разделять проекты на этапы и задачи, осуществлять промежуточный контроль выполнения проектов и добиваться того, чтобы большинство проектов выполнялись в предсказуемые сроки и в рамках определённых трудозатрат.

Одним из основных инструментов работы менеджеров (руководителей) проектов является диаграмма Ганта, позволяющая визуализировать разделение проектов на составляющие, этапы, задачи, распределение задач по ответственным и исполнителям, а также контролировать и корректировать выполняемые проекты, если такие проекты являются долгосрочными.

Для того, чтобы проектное управление стало возможным и осуществимым, сотрудники должны научиться оценивать предполагаемые трудозатраты при планировании работ. Соответствующие инициативы уже есть в ряде групп, в частности, в команде Артура Валиева есть намерение начать оценивать задачи в так называемых story point'ах, о чём на обучении было подробно рассказано одним из разработчиков его команды.

Оценка в story point'ах в компаниях обычно применяется наряду с таким инструментом, как planning poker, и осуществляется командой целиком при планировании очередного этапа работ: каждый участник команды даёт свою оценку той или иной задачи, не видя оценок остальных участников, а затем вся команда смотрит на оценки друг друга и детально обсуждает их, если они сильно отличаются, приходя, как правило, к общей наиболее точной оценке.

Отдельные сотрудники также могут и должны оценивать получаемые ими задачи. Причём важно, чтобы оценка проводилась дважды: предварительно, до начала выполнения той или иной задачи, а затем после выполнения задачи. Такая двойная оценка автоматически обучает сотрудников давать более точные предварительные оценки. Рекомендуется также регулярно (например, раз в месяц) выгружать из системы постановки задач (Jira) предварительные и фактические оценки задач для каждого сотрудника, чтобы обнаруживать систематические отклонения, например, постоянное завышение или занижение предварительных оценок отдельными сотрудниками.

При внедрении системы оценок задач первое время крайне не рекомендуется их каким-либо образом использовать для реальных расчётов трудозатрат, а также использовать какие-либо инструменты мотивации или наказания сотрудников за ошибки. Процесс внедрения такой системы и обучения навыкам оценки должен проходить максимально комфортно и мягко. Важно лишь то, чтобы он в принципе проходил, и все сотрудники приобретали привычку давать и фиксировать оценки.

Здесь также важно отметить, что не все задачи возможно оценить точно. Точная оценка возможна лишь для конкретных чётко сформулированных задач небольшого размера длительностью не более 8-16 часов. Также оценка невозможна для задач исследовательского характера.

По итогам проведённых с сотрудниками встреч и анализа в системе Jira задач, которые ставятся аналитиками команды Ирины Самсоновой, сделан вывод о том, что задачи от аналитиков поступают разработчикам в достаточно детализированном, конкретном и понятном виде. Важно продолжать придерживаться такой практики всем аналитикам, а разработчиков и других специалистов научить возвращать задачу аналитику, если она сформулирована

непонятно, недостаточно детально или слишком крупная (не разбита на подзадачи, которые возможно оценить).

Важным аспектом работы с потоком задач является разделение задач на “плановые” (“зелёные”) и “срочные” (“красные”). “Красные” задачи — это задачи, невыполнение которых обычно приводит к негативным последствиям, а “зелёные” задачи — это задачи, невыполнение которых не приводит к негативным последствиям, а выполнение приводит к позитивным изменениям. Такое разделение необходимо проводить также по сотрудникам, поскольку выполнение задач таких типов требует разных подходов и разных режимов работы. Такая практика уже используется в некоторых командах и наиболее отчётливо видна в отделе системного администрирования под руководством Петра Сафронова, который разделил сотрудников на “оперативных”, работающих посменно и круглосуточно и решающих в реальном времени “красные” задачи, а также на сотрудников, решающих задачи планового характера. Предварительная и фактическая оценка при этом полезна для обоих видов задач, но для совсем срочных задач достаточно фактической, чтобы формировать понимание того, сколько ресурсов на них тратится и какого рода задачи отнимают больше всего времени.

В контексте внедрения проектного управления также имеет смысл использовать общепринятые методологии разработки программного обеспечения или их элементы. Например, для “зелёных” плановых задач можно использовать методологию Scrum, чтобы работать итерациями, получать регулярно видимый и понятный конечным пользователям результат, а также более эффективно приоритизировать планируемые задачи. Для “красных” задач больше подходит методология Kanban, ориентированная на выполнение наибольшего количества задач и при этом максимальную концентрацию на небольшом количестве текущих задач. В компаниях также часто применяется смешанная методология, в которой работа идёт итерациями (так называемыми спринтами — как в Scrum), но задачи при этом разделены на те, которые ещё не выполнены, на текущие, которых должно быть немного, и на уже завершённые.

Помимо вышеперечисленного имеет смысл распространить практики работы в системе контроля версий Git, успешно внедрённые и используемые в команде Георгия Портнова, на остальные команды, поскольку данные практики успешно зарекомендовали себя в других компаниях и являются общепринятыми. В частности, необходимо введение системы чёткого именования и назначения веток в Git-репозиториях для отдельных релизов, проектов, продуктов и функций — так называемого Git Flow. В некоторых проектах имеет смысл также попробовать работу по системе TBD (trunk-based development), считающейся в настоящий момент более современной и удобной для работы с Git-репозиториями.

Помимо внедрения контроля сроков и планирования рекомендуется также системно усовершенствовать обработку задач:

- В различных компаниях принято собирать задачи в релизы и выпускать “в продакшен”. По некоторым продуктам данная практика уже используется. Необходимо продолжать её использовать более системно и не отказываться от неё, продолжать вести историю release notes (перечня доработок и исправлений в определённой версии программного продукта) и, возможно, даже рассылать их всем участникам процесса от аналитиков и разработчиков до сотрудников поддержки и конечных пользователей (см. также об этом ниже в блоке про нематериальную мотивацию).

- По словам некоторых сотрудников процесс тестирования выпускаемого программного обеспечения организован на данный момент плохо. При этом начаты активные изменения: уже длительное время отдельная команда тестирования разрабатывает автоматические тесты, а новые релизы основного продукта компании теперь решено выпускать только после прохождения этих тестов. Необходимо встроить этап доработки или изменения автоматических тестов в этапы, которые проходит каждая задача.

- В жизненный цикл каждой задачи также необходимо включить этап документирования, на котором отдельный специалист (обычно это технический писатель) вносит изменения или дополнения в документацию к продукту — как техническую, так и пользовательскую.

- В жизненный цикл сложных задач также рекомендуется встроить этап, на котором задача подтверждается системным архитектором (как один из вариантов, см. ниже в блоке про системную архитектуру).

В процессе встреч с системными аналитиками и разработчиками приходилось неоднократно слышать от них пожелания увеличить количество друг друга: аналитики просили увеличить количество разработчиков, а разработчики — аналитиков. После внедрения проектного управления и измерений трудозатрат можно будет принимать такие решения более обоснованно и объективно.

В качестве первых двух шагов по внедрению проектного управления рекомендуется поручить сотрудникам отмечать в Jira предполагаемое и фактически затраченное время по всем задачам, а также начать регулярно (ежемесячно) выгружать списки предварительных и фактических оценок каждому сотруднику, чтобы начинать приобретать и совершенствовать навыки

оценки трудозатрат. Здесь также имеет смысл порекомендовать всем желающим прочтение книги “Сколько стоит программный проект” Стива Макконнелла, которая в настоящий момент является фундаментальным руководством, описывающим и сравнивающим все основные методики оценки сроков в программных проектах.

На втором шаге рекомендуется нанять в команду менеджера (руководителя) проектов с опытом управления проектами по разработке и поддержке современного программного обеспечения (веб-сайтов, сервисов и мобильных приложений, кроме проектов в сфере 1С).

2. Продуктовое управление

После внедрения проектного управления в компаниях часто возникает необходимость внедрения так называемого продуктового управления. Этим видом управления обычно занимается такой специалист, как менеджер или директор по продукту.

Управление продуктом необходимо для того, чтобы все знания о продукте целиком и о его функционале в частности интегрировались в одном специалисте. Задача такого специалиста — развивать продукт с точки зрения рынка, повышать лояльной текущей целевой аудитории (пользователей продукта) и привлекать новые аудитории. Это обычно достигается работой над функционалом (исключением ненужного функционала, улучшением нужного и созданием недостающего), исходя из данных продуктовой аналитики и работы с пользователями — процесса customer development.

В качестве ранней подготовки к внедрению продуктового управления рекомендуется постепенно увеличить количество и роль системных аналитиков, а затем, на втором шаге, подчинить их директору по продукту (его можно как нанять со стороны, так и вырастить из существующего сотрудника или тимлида). Второй шаг рекомендуется делать только после внедрения проектного управления.

3. Матричная структура управления

Ещё одним способом повышения эффективности управления командой разработки программного обеспечения является внедрение матричной системы управления, предполагающей двойное подчинение всех узкоспециализированных специалистов: с одной стороны, руководителю проекта или продукта, а с другой — руководителю соответствующего узкоспециализированного отдела.

В настоящее время в команде наблюдается ситуация, при которой каждый специалист является универсальным, но работающим только на каком-то одном продукте или даже на части этого продукта. Например, в командах довольно много full-stack-разработчиков, владеющих одновременно и навыками backend-разработки на Java, и навыками frontend-разработки на фреймворках React и Vue.js. Кроме того, приходилось несколько раз слышать от разных сотрудников, что часто таким разработчикам приходится выполнять и функции аналитиков, и функции тестировщиков. Возможно, полноценная реализация матричной структуры управления не является на данном этапе целесообразной, но рекомендуется двигаться в направлении перехода от команды универсальных специалистов “по своим продуктам” к команде узкоспециализированных специалистов, владеющих конкретными технологиями.

В частности, имеет смысл разделение full-stack-разработчиков на backend и frontend, как это, в частности, сделано в команде Георгия Портнова. С полезностью реализации такого разделения также согласен Артур Валиев и некоторые другие тимлиды. Такое разделение повысит скорость и качество найма соответствующих специалистов, облегчит внедрение системы грейдирования, повысит возможности по развитию сотрудников, а также повысит качество решения соответствующих задач.

В качестве иллюстрации перечисленных преимуществ можно здесь привести пример того, что full-stack-разработчики на Java со знанием фреймворков Vue.js и React на рынке практически отсутствуют и не требуются, а отдельно разработчики Java и frontend-разработчики Vue.js есть в большом количестве, и развитие в этих областях является для большинства специалистов более понятным, логичным и интересным.

Здесь также важно отметить причины, по которым сложилась такая ситуация. Значительный поток задач в компании представляет собой задачи “красного” типа, требующие максимально быстрого решения. В таких условиях казалось более правильным “воспитать” универсальных разработчиков, способных справляться с любой задачей, будь то задача на backend, frontend или комплексная. Но такая практика обычно приводит к снижению качества разработки, особенно, на frontend.

4. Внутреннее документирование

В команде разработки программного обеспечения существует система документирования, но она, в целом, состоит лишь из пользовательской документации, составляемой аналитиками (в частности, под руководством

Ирины Самсоновой) и ознакомительной документации для новых сотрудников. При этом полноценная техническая документация, в основном, отсутствует. В некоторых командах есть инициативы по ведению именно технической документации, например, в команде Артура Валиева.

Важно отметить, что полноценное техническое документирование ведётся лишь в команде системного администрирования под руководством Петра Сафронова (по его же словам).

Рекомендуется начать вести полноценную техническую документацию по всем программным продуктам, включающую, как минимум, описание API и структуры базы данных. Описание API, в частности, также поможет чётко разделить разработку на backend- и frontend-составляющие.

Для реализации полноценной системы документирования рекомендуется нанять отдельного специалиста — технического писателя (соответствующее пожелание неоднократно озвучивалось, в том числе, Ириной Самсоновой) и внедрить использование современных систем документирования, позволяющих вести и поддерживать документацию для разных версий продуктов. Для этих целей стоит рассмотреть, например, Confluence, которая может быть легко и удобно интегрирована с используемой системой Jira.

Также рекомендуется встроить обновление документации в процесс обработки каждой задачи, чтобы технический писатель после выполнения и тестирования того или иного функционала гарантированно вносил исправления или дополнения в документацию.

5. Управление архитектурой

В команде разработки управление архитектурой осуществляется, в основном, Геннадием Малюковым, но также тимлидами, иногда разработчиками и системными аналитиками. При этом данное управление носит несистемный и несистематический характер. Рекомендуется уделять больше внимания архитектуре продуктов и делать это централизованно — одним специалистом, выполняющим роль системного архитектора. В частности, необходимо:

- выработать стандарты проектирования API и структуры базы данных, и выполнять такое проектирование в соответствии с этими стандартами;

- выполнять разработку нового функционала по принципу “API first”, то есть перед разработкой выполнять проектирование и документирования

дополнений и изменений API и структуры базы данных, а затем уже вести разработку в соответствии с данной документацией;

- планировать и контролировать проведение рефакторингов в тех или иных частях продуктов;

- установить единые стандарты написания кода и принципы проведения code review, проводить code review всего нового кода и приводить его в соответствие с этими стандартами;

- следить за выпуском новых версий используемых языков программирования, фреймворков, библиотек и технологий и обеспечивать их своевременное обновление;

- участвовать в обновлении документации, касающейся системной архитектуры;

- внедрять более современные технологии, делающие программный код более простым и поддерживаемым, например, фреймворк Spring Boot.

Важно отметить, что при решении тех или иных задач по продуктам компании разработчики тратят значительное время на поиск и точную диагностику возникающих проблем. Несколько раз приходилось слышать от разработчиков, что для нахождения той или иной возникшей проблемы требуется тратить часы или даже дни, а также прибегать к помощи старших специалистов, разбирающихся в архитектуре продуктов, в частности, Геннадия Малюкова.

Данная проблема является следствием быстрого роста системы и накапливавшегося в течение этого роста так называемого “технического долга”. При этом работа по его устранению и улучшению архитектуры в команде уже ведётся. В частности, в нескольких продуктах происходит обновление устаревшего frontend-кода на JavaScript и его структуризация путём перехода на современный фреймворк Vue.js, а также выделение некоторого функционала backend-составляющей приложений в микросервисы. Необходимо продолжать эту работу и целесообразно повысить её интенсивность вместе с остальными действиями в сфере архитектуры, описанными выше.

6. Понятная система грейдирования сотрудников

В настоящий момент в команде разработки отсутствует понятная и постоянно применяемая система грейдирования сотрудников. Рекомендуется

внедрить такую систему, которая позволила бы объективно оценивать уровень специалистов, как вновь нанимаемых в компанию, так и уже работающих и развивающихся.

По итогу проведённых встреч совместно с HR-директором Анной Сергеевой выделена следующая система соответствий уровней различных специалистов диапазонам заработных плат. Система построена на основе текущих заработных плат специалистов и рыночных заработных плат.

Разработчики Java

- Intern (менее 6 месяцев) — 45 000 – 80 000 р. в месяц.
- Junior (менее 1,5 лет) — 60 000 – 120 000 р. в месяц.
- Middle (1,5-4 года) — 120 000 – 230 000 р. в месяц.
- Senior (более 4-х лет) — 200 000 – 350 000 р. в месяц.

Разработчики frontend (Vue.js)

- Intern (менее 6 месяцев) — 45 000 – 80 000 р. в месяц.
- Junior (менее 1,5 лет) — 60 000 – 120 000 р. в месяц.
- Middle (1,5-4 года) — 120 000 – 200 000 р. в месяц.
- Senior (более 4-х лет) — 180 000 – 280 000 р. в месяц.

Системные аналитики

- Intern (менее 6 месяцев) — 30 000 – 50 000 р. в месяц.
- Junior (менее 1,5 лет) — 45 000 – 100 000 р. в месяц.
- Middle (1,5-4 года) — 90 000 – 180 000 р. в месяц.
- Senior / Lead (более 4-х лет) — 160 000 – 230 000 р. в месяц.

Системные администраторы и DevOps

- Junior (до 1 года) — 50 000 – 120 000 р. в месяц.
- Middle (от 1 до 3 лет) — 120 000 — 170 000 р. в месяц.
- Senior (от 3-х лет) — 170 000 — 300 000 р. в месяц.
- Lead — 200 000 — 350 000 р. в месяц.

Руководители разработчиков

- Системный архитектор (техлид) — 250 000 – 350 000 р. в месяц.
- Технический директор (CTO) — 300 000 – 450 000 р. в месяц.
- Тимлид команды разработчиков — 250 000 – 350 000 р. в месяц.

Также проведена приблизительная оценка уровней и рыночных заработных плат специалистов (в тысячах рублей), с которыми проводились встречи:

Специалист	Должность	Уровень	Зарплата
Фёдоров Виктор	технический директор	CTO	400–450
Малюков Геннадий	старший системный архитектор	senior	250–350
Валиев Артур	тимлид команды разработки	lead	250–350
Портнов Георгий	тимлид команды разработки	lead	250–350
Сергеев Антон	ведущий разработчик	middle	150–250
Бекетов Павел	разработчик	middle	150–250
Самсонова Ирина	руководитель группы СА	lead	180–230
Парамонова Антонина	младший системный аналитик	intern	40–70
Окунев Дмитрий	младший системный аналитик	intern	40–70
Войнаревич Дмитрий	младший разработчик	junior	60–90
Досымбеков Ерлан	младший разработчик	junior	60–90
Сафронов Пётр	руководитель отдела системного администрирования	lead	350–420

Для определения грейда того или иного специалиста необходимо использовать методику оценки уровня их самостоятельности. Все опрошенные сотрудники и тимлиды отчётливо понимают, какого уровня их коллеги и подчинённые. Примеры методик оценки тех или иных специалистов озвучены во время обучения.

Методики, используемые в настоящий момент Геннадием Малюковым для оценки уровня backend-разработков и Петром Сафроновым для оценки системных администраторов выглядят достаточно совершенными, современными и качественными. Рекомендуется их документировать и продолжать активно использовать.

В команде Петра Сафронова также практикуется промежуточная аттестация сотрудников посредством тестового задания, выполнение которого означает, что грейд данного сотрудника может быть повышен. Рекомендуется внедрить такую аттестацию на регулярной основе (каждые 3-6 месяцев) для всех сотрудников команды разработки, включая системных аналитиков.

Рекомендуется также должности всех специалистов назвать в соответствии с рыночными наименованиями соответствующих специальностей, например, “Разработчик Java”, “Frontend-разработчик (Vue.js)”, “Системный аналитик” и так далее.

7. Система нематериальной мотивации и профилактика профессионального выгорания

В процессе проведения встреч с сотрудниками отдела разработки приходилось слышать о различных проблемах в сфере нематериальной мотивации. Ниже приведён ряд рекомендаций, касающихся этой темы.

В первую очередь, хотелось бы обратить внимание на рыночный показатель баланса между текучкой сотрудников и их работой на протяжении всего времени существования компании, который составляет в среднем 5 лет. По истечении этого срока необходимо уметь “отпускать” сотрудников, причём не только с моральной точки зрения, но и с точки зрения “bus factor” — реальной готовности отпускать без ущерба для компании, её продуктов и процессов.

В случае преждевременного ухода сотрудников на другую работу рекомендуется проводить exit interview, на которых обязательно спрашивать о том, что привело к решению об уходе и что данный сотрудник предлагает улучшить в компании. Ответы уходящих сотрудников необходимо собирать и тщательно анализировать, принимая меры по снижению текучки. В настоящий момент данные вопросы уже задаются уходящим сотрудникам Артуром Валиевым, но их ответы не фиксируются и, по первому впечатлению, не анализируются достаточно тщательно. Рекомендуется поручить данную работу специалистам HR.

Рекомендуется также ввести практику построения индивидуальных планов развития (ИПР) сотрудников, в которых фиксировать структуру их мотивации и планы развития на ближайшее время. Такие планы необходимо связать с практикой регулярной аттестации сотрудников, в результате которой могут приниматься решения об изменении грейдов и, соответственно, заработных плат. Важно отметить, что изменения грейдов могут происходить не только в сторону увеличения, но и в сторону уменьшения.

В качестве дополнительной нематериальной мотивации рекомендуется внедрить практики так называемого “developer relations”, в частности:

- поддержку создаваемых в командах open-source-проектов (такие идеи и намерения уже имеются и, в частности, были озвучены Геннадием Малюковым);

- стимулирование к написанию статей на технических ресурсах, таких как, например, Хабр;

- участие в тематических и технических конференциях (не только в качестве слушателей, но и в качестве спикеров) с целью обмена опытом и самореализации сотрудников;

- организация и стимулирование регулярного и систематического обмена опытом между командами внутри компании;

- организация и стимулирование обмена опытом с другими компаниями из той же или смежных отраслей, а также имеющих схожие технические проблемы и решения;

- проведение мозговых штурмов по решению сложных технических проблем, в том числе, архитектурных;

- проведение всеобщих мероприятий, сближающих всех сотрудников компании, тимлидов и руководителей, а также анонсирующих планы и информирующих всех сотрудников о крупных достижениях компании.

Всеобщие стратегические встречи в компаниях очень сильно повышают мотивацию сотрудников, поскольку делают работу в компании более прозрачной и осмысленной. На таких встречах обычно объединяется обсуждение достижений и идей “сверху” и инициатив “снизу”. Все без исключения опрошенные сотрудники высказывались положительно о состоявшейся в декабре прошлого года стратегической сессии компании, воспоминания о которой до сих пор многих из них стимулирует в их работе. Рекомендуется продолжать эту практику.

Также рекомендуется поощрять прохождение сотрудниками курсов повышения квалификации по их специализации. В частности, такое пожелание высказывалось Ириной Самсоновой, а некоторые сотрудники отдела администрирования под руководством Петра Сафронова уже проходят такие курсы с целью повышения своего уровня.

В компаниях, занимающихся регулярным обновлением выпускаемого программного обеспечения, принято составлять и рассылать всем участникам процесса так называемые release notes — перечень сделанных в очередной версии программы доработок, исправлений и изменений. Такая информация

рассылается не только конечным пользователям, но и разработчикам, и аналитикам, и их тимлидам. Также известны случаи, когда такая рассылка делается сначала по конечным пользователям, у которых в ней же запрашиваются отзывы о новом функционале, после чего вторым этапом данная информация уже с отзывами рассылается тем, кто участвовал в выпуске этого релиза. Такой эмоциональный отклик (особенно, если он положительный) очень позитивно влияет на атмосферу в команде, на понимание и ощущение командой и отдельными сотрудниками смысла и практической пользы от своей работы.

Что касается вознаграждений сотрудников, обычно принято поощрять особые достижения и “прорывные” изменения, которые сотрудники делают помимо выполнения своей основной работы, чтобы они чувствовали и понимали, что дополнительный их вклад в компанию и в продукт видят и ценят.

8. Регулярное решение системных вопросов

Накопление в крупных компаниях системных проблем является обычным и естественным процессом, и поэтому бывает целесообразно внедрять механизмы регулярного решения таких проблем и системных вопросов. По итогам общения с командами стало понятно, что многие системные вопросы уже так или иначе решаются или даже решены в отдельных командах, но не “подняты” на уровень всей компании. Кроме того, от некоторых сотрудников поступали пожелания о более частом межкомандном взаимодействии, в рамках которого мог бы происходить обмен успешным опытом решения тех или иных проблем.

Проведение регулярных встреч тимлидов, на которых бы обсуждались и решались системные проблемы в компании и в командах, а также успешно внедрённые решения могло бы существенно повысить эффективность работы в целом, дополнительно мотивировать тимлидов и сотрудников, а также значительно снизить количество время от времени возникающих и накапливающихся проблем системного характера.

Следует также стимулировать внутреннее наставничество, при котором сотрудники и тимлиды могут помогать внедрять успешно зарекомендовавшие себя техники и практики в других командах. На текущий момент наиболее инициативными в этом отношении выглядят Артур Валиев, Георгий Портнов и Пётр Сафронов. Они могут значительно повысить эффективность (качество и скорость) работы отдела разработки, если их привлечь к активному участию в таких процессах, позволить высказывать предложения и курировать соответствующие нововведения.

9. Рекомендуемая литература

С учётом полученной в результате встреч с сотрудниками компании информации в качестве дополнения к проведённому обучению рекомендуется ознакомиться со следующей литературой:

1. “Сколько стоит программный проект”, Стив Макконнелл.
2. “Джедайские техники”, Максим Дорофеев.
3. “Управление проектом на одной странице”, Кларк А. Кемпбелл.

Заключение

По итогам встреч с сотрудниками и тимлидами отделов разработки и администрирования были сделаны выводы о высоком уровне сплочённости, вовлечённости и конструктивности как отдельных сотрудников и тимлидов, так и отделов в целом. Весь коллектив настроен на эффективную работу и достижение результатов. При этом быстрый рост компании и разрабатываемых ею продуктов привёл к появлению перечисленных в начале настоящего документа системных проблем.

Выше приведены рекомендации по решению этих проблем. Важно отметить, что все эти проблемы взаимосвязаны и влияют друг на друга, и решение какой-то одной из них неизбежно повлечёт изменения, связанные с другими проблемами. К примеру, внедрение системы проектного управления сделает работу команд и каждого отдельного сотрудника более прозрачной и понятной, что, в свою очередь, положительно скажется на качестве внедрения системы грейдирования, в рамках которой станет легче оценивать уровни сотрудников и вновь нанимаемых кандидатов, а также отслеживать их рост и развитие. В связи с этим рекомендуется проводить предлагаемые изменения постепенно и комплексно, анализируя их эффект на отдельных сотрудников, тимлидов, команды и отделы.