От Алексея Абашева

Отрицание. Основные концепции объектно-ориентированного подхода (ООП) зародились в 60-х годах прошлого века и уже тогда подобный путь разработки программного обеспечения казался естественным и не требующим какого-либо дополнительного пояснения. Казалось бы, что может быть проще? Достаточно взглянуть на мир вокруг нас, посмотреть, какие сущности взаимодействуют друг с другом и описать его с помощью алгоритмического языка. За 50 лет сделано множество объектно-ориентированных языков, выпущены сотни книг, появились шаблоны проектирования, им стали противопоставляться анти-шаблоны, сотни тысяч студентов изучали ООП в университетах, миллионные армии программистов по всему миру пишут программные продукты, используя ООП. Спроси любого разработчика: “Появилось ли что-то новое в концепции ООП за последнее время?”, и он ответит—“Нет, как оно было сформулировано изначально, так все и осталось”. Зачем еще дополнительно пояснять и разжевывать очевидные вещи?

Гнев. Однако несмотря на всю очевидность концепции находятся люди, которые считают, что все вокруг понимают неправильно, а только они одни обладают сокровенным знанием. Несмотря на то, что Егор занимается разработкой программных продуктов довольно долгое время, вы не найдете его фамилии в списках авторов каких-либо языков программирования, где можно было реализовать ООП “правильно”. Нет его и в списках авторов каких-либо значимых Java проектов, которые реально изменили саму концепцию разработки, как это сделал, например, Maven в

деле управления зависимостями, или Spring как универсальная платформа для разработки приложений, или Hibernate как слой взаимодействия с базой данных. И опять же—практика это самое лучшее подтверждение теории, и любые изыски в разработке программного обеспечения имеют смысл только, если они помогают удешевить разработку, упростить поддержку, облегчить развертывание. Но я как-то не слышал, чтобы teamed.io захватили весь рынок разработки за счет минимальных счетов за поддержку или были самой быстрой разработкой на рынке.

Торг. Но что нам мешает взглянуть под другим углом на, казалось бы, очевидные истины? Очень многие известные разработчики считают, что зацикливаться на одном языке программирования не стоит и надо добавлять в свою палитру знания и умения из разных областей - кто-то изучает новые языки, кто-то движется в сторону DevOps и администрирования, кто-то участвует в open-source проектах, которые показались интересными. Не будет же ничего плохого, если вы прочитаете новую книгу о программировании? Вы можете соглашаться или не соглашаться с автором, но почему бы не расширить горизонты своих знаний и не добавить к себе в копилку опыт другого разработчика - ведь любая дополнительная информация сделает вас лучше как профессионала и даст дополнительные поводы задуматься, ну или блеснуть на митинге широтой кругозора.

Депрессия. Вообще, только читая подобные книги или, например, Clean Code, понимаешь насколько чудовищна наша повседневная работа и насколько бы она стала продуктивнее и интереснее, если бы мы могли всё

изначально сделать правильно. Чем лучше продумана система управления, тем более сложные конструкции с её помощью можно создавать и тем более эффективно управлять. Шаблоны проектирования дали нам язык общения, показали что-то большее, чем просто код, рассортированный по классам. Но насколько красива была идея, настолько печально выглядят воздушные замки, построенные на её основе: насколько широко описана идея—настолько велико количество различных толкований и толкователей. Вечные споры, что такое proxy-adapter-decorator-facade и где какой начинается. Чудовищные кодогенераторы от IBM-Rational, использующие UML, и шаблоны как единицы взаимодействия. Нет выхода из этого тупика, и даже не видно где он может быть.

Принятие. Однако, чем дальше я продвигался в чтении книги, тем менее различимы в моей голове становились нотки несогласия, и тем больше я понимал, что точка зрения Егора, которую он описал в своей книге, заслуживает право на жизнь и даже больше—достойна того, чтобы с ней ознакомился каждый, кто хочет стать чуточку лучше как разработчик. Я не могу согласиться со всем, что он написал, но я уверен, что не будет среди читателей и того, кто согласен на 100%, ну может кроме самого Егора. Единственное предостережение—рекомендовать эту книгу начинающему программисту я бы не стал, потому что то, что в ней описано, должно быть выстрадано годами. Бессонные ночи над чужим кодом, попытки разобраться в архитектурных фантазиях, которым бы позавидовал Гауди, пропихивание immutable объектов в concurrent код, утечки памяти из-за не закрытия ресурсов—только тогда вы

сможете понять всю необходимость этой книги, и почему вам стоит дочитать её до конца. Она займет достойное место на полке с книгами, к которым стоит возвращаться снова и снова.

`@abashev`, 19 сентября 2016, Россия