Предисловие

Понятие рефакторинга возникло в кругах, близких к Smalltalk, но очень быстро проложило дорогу к другим языкам программирования. Поскольку оптимизация является неотъемлемой частью развития программного обеспечения, этот термин появляется, как только проектировщики начинают вести профессиональные беседы. Он приходит и когда речь идет об уточнении иерархии классов, и когда обсуждается, сколько строк кода можно удалить. Программисты знают, что с первого раза хорошо работающую программу не получить — она должна развиваться постепенно, по мере накопления опыта. Они также знают, что код будет куда чаще читаться и изменяться, чем писаться "с нуля". Рефакторинг является ключом к поддержке удобочитаемости и изменяемости кода — как для всего программного обеспечения в целом, так и для конкретных программ.

В чем же заключается проблема? Ответ прост: рефакторинг — это риск. Он требует изменения рабочего кода, в ходе которого могут возникнуть не только улучшения, но и новые ошибки. Небрежно выполненный рефакторинг может отбросить вас назад на дни и даже недели. Еще более рискованным оказывается рефакторинг, выполняемый неофициально или от случая к случаю. Вы начинаете копаться в коде. Вскоре вы обнаруживаете новые возможности для внесения изменений и закапываетесь глубже. Чем глубже вы копаете, тем больше возможностей для изменений обнаруживаете. В конце концов вы выкапываете яму, из которой уже не в состоянии выбраться. Чтобы эта яма не превратилась в могилу, рефакторинг необходимо выполнять систематически. В нашей книге "Design Patterns"[1] было упомянуто, что проектные шаблоны обеспечивают цели для рефакторинга. Однако определение цели является лишь частью проблемы; другой задачей является преобразование кода, позволяющее достичь поставленной цели.

Мартин Фаулер и его соавторы внесли неоценимый вклад в развитие объектно-ориентированного программного обеспечения, пролив свет на процесс рефакторинга. В этой книге описаны принципы и передовой опыт рефакторинга и указано, когда и где следует начинать работу с кодом для его улучшения. В основе книги находится подробный перечень рефакторингов. Каждый рефакторинг описывает мотивацию и технологию преобразования кода. Некоторые из разновидностей рефакторинга, такие как, например, "Извлечение метода" или

[1] Имеется русский перевод: Гамма Э., Хелм Р., Джонсон Р., Влиссидес Д. *Приемы объектно-ориентированного проектирования. Паттерны проектирования*. — С-Пб, "Питер", 2000.

"Перемещение поля", могут показаться очевидными. Но не обманывайтесь — понимание технологии таких рефакторингов является ключом к выполнению организованного рефакторинга. Описания рефакторингов в этой книге помогут вам изменять свой код небольшими порциями за один раз, снижая тем самым риск резкого изменения самих принципов вашего проекта. Названия этих рефакторингов очень быстро займут свое место в вашем словарном запасе.

Мой первый опыт "пошагового" рефакторинга я получил, работая на пару с Кентом Беком на высоте 9 км. Он сумел обеспечить пошаговое применение описанных в этой книге рефакторингов, поразив меня тем, насколько хорошо все это работало. Я не только получил более надежный код, в котором был более уверен, но и почувствовал себя менее напряженным. Я настоятельно рекомендую вам испытать эти рефакторинги: и ваш код, и вы сами почувствуете себя намного лучше.

— Эрих Гамма (Erich Gamma)
Object Technology International, Inc.