## Тема 1.1. Основы и принципы тестирования

### История

9-го сентября 1946-го года ГрейсХоппер работала в Гарвардском университете с вычислительной машиной Harvard Mark II. Проследив возникшую ошибку в работе программы до электромеханического реле машины, она нашла между замкнувшими контактами сгоревшего мотылька. Извлечённое насекомое было вклеено скотчем в технический дневник с сопроводительной ироничной надписью: «Первый реальный случай обнаружения жучка»

### 1985-1987: Шесть передозировок радиацией

С распространением программного обеспечения люди узнали о возможной цене ошибок. Так, при использовании аппарата THERAC 25, шесть человек получили серьёзную дозу радиации, а двое из них умерли от последствий облучения. Причина была в том, что устройство могло работать в двух режимах: облучение электронным пучком (безвредное) и облучение рентгеновскими лучами. Как оказалось, переключение на режим облучения электронным пучком срабатывал не всегда и пациенты получали смертельную дозу радиации.

“Медсестра вспомнила, что в тот день она меняла режим с «Х» на «Е». Выяснилось, что если нажать на кнопку достаточно быстро, переоблучение случалось практически со 100-процентной вероятностью.”

Так разработчики начали осознавать важность качества кода и начало зарождаться тестирование,

### Тестирование, тест и тестировщик

Тестирование – это проверка соответствия между *реальным* поведением программы и её *ожидаемым* поведением на *конечном наборе тестов*, выбранном определенным образом;

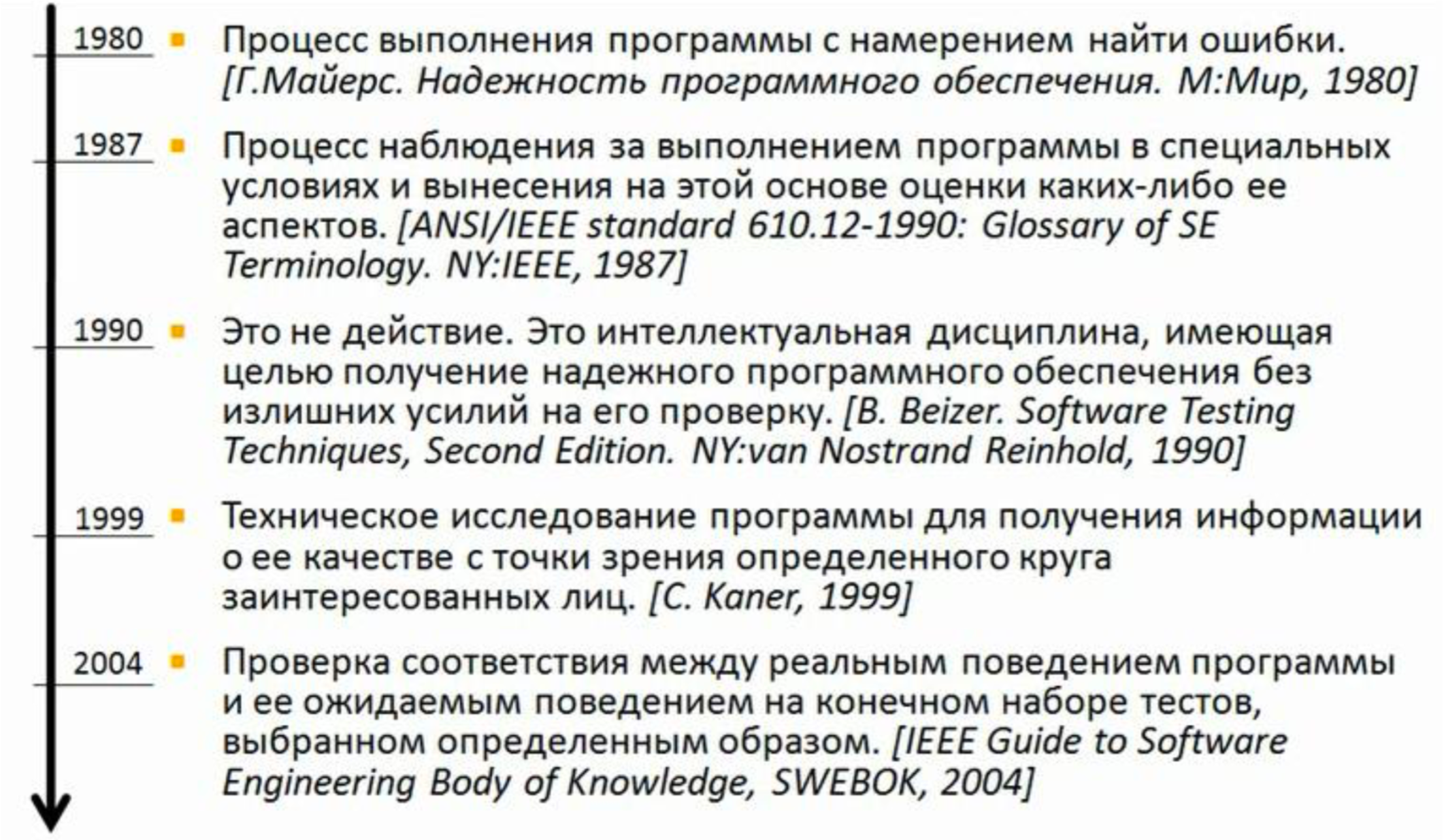


Рисунок 1. Эволюция определений тестирования

Тест – это специальная, искусственно созданная ситуация, выбранная определенным образом, и описание того, какие наблюдения за работой программы нужно сделать для проверки её соответствия некоторому требованию;

Тестировщик – это человек, управляющий выполнением программы и создающий *искусственные ситуации*; человек, наблюдающий за поведением программы и сравнивающий *наблюдаемое* поведение с *ожидаемым*

### Цели тестирования

Тестирование ставит перед собой несколько целей: в первую очередь, это обнаружение дефектов, которые могут привести к нежелательным последствиям. Устраняя эти дефекты, разработчик может быть уверен в том, что на выходе получится более качественный продукт. Для менеджмента информация, получаемая после тестирования, может служить индикатором для принятия решений о выпуске продукта или его очередной версии на рынок. Некоторые методики, такие как Test-Driven Development (TDD) позволяют не искать дефекты в уже написанном коде, а заранее предотвращать их появление. В конце концов, если продукт изменяется, нужно каким-то образом сделать так, чтобы изменения не повлияли не работу уже готовых частей – этому тоже помогает тестирование.

### Семь принципов тестирования

#### 1. Тестирование демонстрирует наличие дефектов

Тестирование может показать, что дефекты присутствуют, но не может дать гарантии их отсутствия. Другими словами, тестирование снижает вероятность наличия дефектов, находящихся в программном обеспечении, но, даже если дефекты не были обнаружены, это не доказывает корректности работы ПО. Причины объясняются следующим принципом.

#### 2. Исчерпывающее тестирование недостижимо

Полное тестирование программы с использованием всех комбинаций вводов и предусловий физически невыполнимо, за исключением тривиальных случаев, Вместо исчерпывающего тестирования должны использоваться анализ рисков и расстановка приоритетов, чтобы более точно сфокусировать тестировочные работы.

#### 3. Раннее тестирование

Чтобы найти дефекты как можно раньше, тестирование должно быть внедрено в жизненный цикле разработки программного обеспечения или системы на как можно более ранних стадиях. Активность должна быть сфокусирована на определенных целях.

#### 4. Скопления дефектов

Усилия, затраченные на тестирование, должны быть сосредоточены пропорционально ожидаемой, а позже реальной плотности дефектов по модулям.

Как правило, большая часть дефектов, обнаруженных при тестировании или повлекших за собой основное количество сбоев системы, содержится в небольшом количестве модулей.

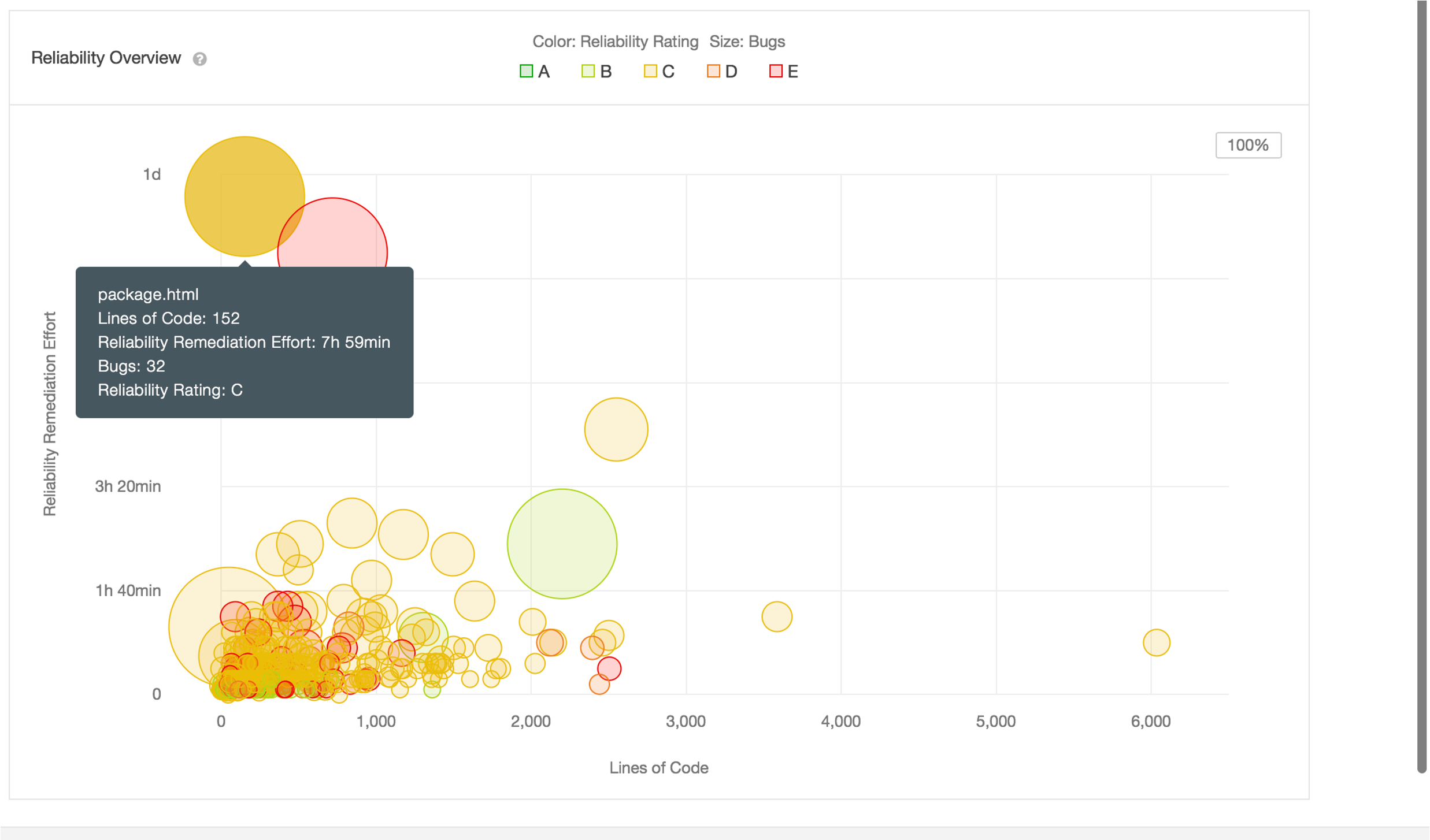


Рисунок 2. 32 дефекта на 152 строки кода считается скоплением дефектов. Такой модуль лучше переписать полностью

#### 

#### 5. Парадокс пестицида

Если одни и те же тесты будут запускаться много раз, в конечном счёте этот набор тестовых сценариев больше не будет находить новых дефектов. Чтобы преодолеть этот “парадокс пестицида”, тестовые сценарии должны регулярно пересматриваться и корректироваться, новые тесты должны быть разносторонними, чтобы охватить все компоненты программного обеспечения или системы и найти как можно больше дефектов.

#### 6. Тестирование зависит от контекста

Тестирование выполняется по-разному в зависимости от контекста. Например, программное обеспечение, в котором критически важна безопасность, тестируется иначе, чем сайт электронной коммерции.

#### 7. Заблуждение об отсутствии ошибок

Всестороннее тестирование, обнаружение и исправление дефектов не помогут, если созданная система не подходит пользователю и не удовлетворяет его ожиданиям и потребностям