Формальные методы обеспечения качества ПО

Михаил Моисеев

Методы оценки надежности ПО

2012

Постановка задачи

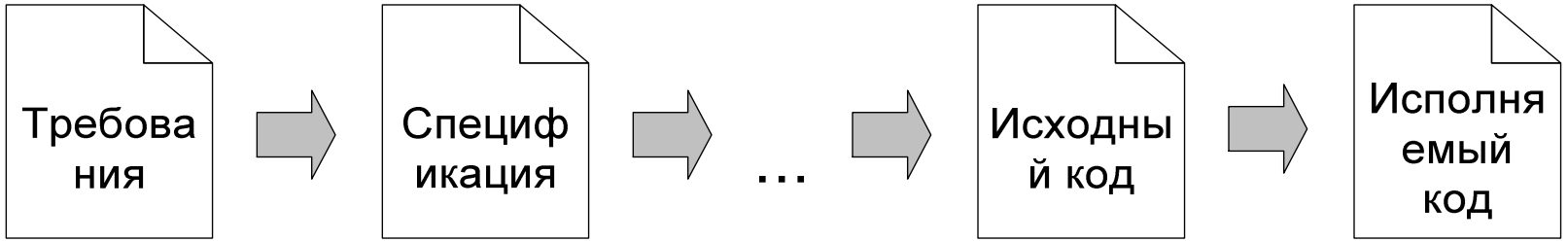
- Процесс обеспечения любой характеристики подразумевает возможность оценки полученного результата
- Измерение (оценка) надежности ПО – определение одного или нескольких показателей надежности
- История вопроса
  - Интерес к измерению надежности ПО возник одновременно с появлением программ
  - Были предприняты попытки получить традиционную вероятностную оценку надежности для ПО как для отдельной технической системы, с помощью подходов классической теории надежности
- Теория надежности ПО (***S****oftware* ***R****eliability* ***E****ngineering*)

Показатели надежности

- **Вероятность безотказной работы** – вероятность того, что в пределах заданной наработки (времени или объема работы) не возникнет отказ
- **Средняя наработка до отказа** (***M****ean* ***T****ime* ***T****o* ***F****ailure*) – мат. ожидание наработки объекта до первого отказа
- **Средняя наработка на отказ** (***M****ean* ***T****ime* ***B****etween* ***F****ailures*) – отношение суммарной наработки к мат. ожиданию числа отказов в течении этой наработки
- **Интенсивность отказов** – плотность вероятности возникновения отказов
- **Коэффициент готовности** – вероятность того, что объект окажется в работоспособном состоянии в произвольный момент времени
- **Средний срок службы**
- **Вероятность восстановления**

Ошибки в программах

- Вносят разработчики
- Процесс проектирования ПО – **преобразование** и **детализация** различных представлений программы



- Ошибки вносятся на разных стадиях и присутствуют в соответствующих представлениях программы (артефактах процесса проектирования)
- Причина появления ошибок – **высокая сложность** проектирования ПО

Классификация ошибок

- **Функциональные ошибки** – нарушения программной спецификации (несоответствие функциональным или нефункциональным требованиям). Приводят к ухудшению функциональности ПО (пригодность, точность)
- **Нефункциональные ошибки** – нарушения правил языка программирования, неправильное использование библиотечных функций и т.п. Приводят к снижению надежности (зрелости) и ухудшению функциональности (защищенности)
  - Ошибки в последовательных программах
  - Ошибки синхронизации

ЖЦ программной ошибки

- **Действие программиста приводящее к ошибке** (*Mistake*) – ошибка, опечатка, незнание или неверное понимание работы вызываемых функций
- **Программная ошибка, программный дефект** (*Error*, *Defect*) – совокупность конструкций в исходном коде программы, которые могут приводить к некорректным действиям
- **Срабатывание ошибки** (*Fault*) – наступление условий в процессе выполнения программы, при которых выполняются некорректные действия
- **Проявление ошибки, отказ** (*Failure*)– негативное влияние сработавшей ошибки на работу программы (аварийное завершение, зависание, выдача некорректных результатов)

Свойства программных ошибок

- Ошибки в программе – величина ненаблюдаемая, наблюдаются не сами ошибки, а результат их проявления – отказы
- Надежность связана с частотой проявления ошибок, но не с их количеством – разные ошибки имеют разную частоту проявления
- Отказ программы может быть следствием не одной, а сразу нескольких ошибок
- Ошибки могут компенсировать друг друга – после исправления ошибки интенсивность отказов может увеличиться
- В результате исправления ошибки или любого другого изменения получается новая программа с другими показателями надежности

Влияние внешнего окружения

- При оценке надежности аппаратуры
  - Для элементов, из которых изготавливается аппаратная система, модели надежности известны, они определяются на этапе их разработки и изготовления. Определение надежности элементов производится путем проверки достаточно большого числа этих элементов, с фиксацией сбоев и отказов
  - Внешняя среда является естественным источником ненадежности, результаты двух экспериментов получаются независимыми
  - При одинаковых условиях эксплуатации влияние внешней среды на тестируемые элементы в среднем будут одинаковым
  - Известные характеристики (модели) надежности элементов позволяют получить оценку надежности всего устройства
- Внешнее окружение вызывает сбои и отказы аппаратуры

Влияние внешнего окружения

- При оценке надежности программ
  - Для ПО обычно трудно провести разделение на части (элементы), измерение надежности каждой части в общем случае невозможно
  - Параллельное тестирование двух экземпляров ПО даст почти полностью зависимые результаты
  - Параллельное тестирование нескольких экземпляров имеет смысл только при различном внешнем окружении, разных входных данных и воздействиях, однако в этом случае трудно оценить корреляцию экспериментов и обработать полученные результаты
  - Количество всех возможных состояний внешней среды для любой нетривиальной программы слишком велико что не позволяет провести полное множество экспериментов
- Внешнее окружение увеличивает число возможных состояний программы

Изменение надежности со временем

- Надежность аппаратной системы меняется со временем
  - В начале эксплуатации она растет – фаза приработки
  - Затем некоторое время остается постоянной
  - В конце, начинает уменьшаться – фаза износа
- Надежность программы со временем не меняется
  - Для ПО отсутствует фаза приработки и износа
  - Коррекция программы, аналогична внесению изменений в конструкцию аппаратного устройства – получается новое ПО, с другими показателями надежности

Методы оценки надежности ПО

- **Динамические методы** – используют результаты выполнения программы
- **Методы на основе моделей сложности** – основаны на различных метриках сложности исходного кода программы
- **Архитектурные методы –** основаны на анализе архитектуры системы и могут использовать как динамические так и статические подходы
- **Эмпирические методы** – используют информацию о процессе проектирования
- **Методы на основе статических методов обнаружения дефектов** – основаны на обнаружении дефектов с помощью различных статических методов

Свойства методов оценки надежности

- Универсальность метода, по отношению к различным программам (архитектура программы, язык программирования, используемые библиотеки и компоненты)
- Универсальность метода, по отношению к различным условиям эксплуатации программы
- Стадии применения метода (проектирование, написание кода, тестирование, эксплуатация)
- Требуемые исходные данные
- **Точность и достоверность оценки**
- **Сложность получения оценки**

Динамические методы

- **Динамические методы** – используют результаты выполнения программы
  - Обнаруженные отказы
  - Время работы программы при каждом запуске
  - Результаты тестов и трассы выполнения программы

Модель Бернулли

- Каждый запуск программы имеет два исхода: правильный и неправильный (с вероятностями 1-*p* и *p*).
- Вероятность того, что из *n* запусков *k* приведут к неправильному результату, выражается формулой биномиального распределения

$$B(p,n,k) = C_n^k \cdot p^k \cdot (1-p)^{n-k}$$

- Вероятность *p* априорно неизвестна, но по результатам запусков известны *n* и *k*. Величина *B(p)* имеет максимум при *p* = *k/n*. В качестве оценки надежности программы принимается величина

$$R = 1 - \frac{k}{n} = \frac{n-k}{n}$$

Модель Миллса

- Используется внесение искусственных ошибок
- Пусть *N* - число ошибок в программе, *V* - число внесенных ошибок
- При тестировании было найдено *n*+*v* ошибок, *v* – число внесенных ошибок и *n* – число истинных ошибок, тогда

$$N = V \cdot \frac{n}{v}$$

- Задача внесения ошибок
  - Как обеспечить одинаковую вероятность обнаружения внесенных ошибок и истинных ошибок в программе?
  - Сколько ошибок нужно внести для получения результатов с требуемой достоверностью?

Модель Миллса

- Для оценки достоверности утверждений о числе ошибок в программе в программу также вносится *V* ошибок
- Тестирование выполняется до тех пор, пока не будут найдены все внесенные ошибки, тогда достоверность утверждения что в программе имеется *k* ошибок

$$C = 1, n > k$$

$$C = \frac{V}{V + k + 1}, n \le k$$

Пример: если мы утверждаем что в программе нет ошибок, вносим 10 ошибок, то при условии, что обнаруживались только внесенные ошибки, уровень доверия к прогнозу равен 0.91

Модель Миллса

- Эти две формулы образуют полезную модель ошибок
  - первая предсказывает их количество
  - вторая позволяет определить достоверность прогноза
- Вторую формулу можно модифицировать для случая, когда были найдены не все внесенные ошибки, пусть число обнаруженных собственных ошибок *v*

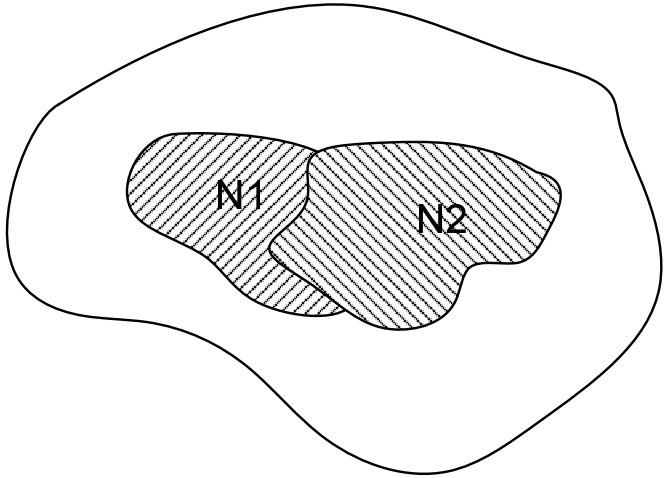
$$C = 1,\ n > k$$

$$C = \frac{C_V^{v-1}}{C_{V+k+1}^{k+v}},\ n \le k$$

Пример : посчитать число ошибок при заданных V=10, v=6, n=4 и доверительную вероятность того, что в программе осталось 1 ошибка – ***N = 6.67***, ***C = 0.32***

Простая интуитивная модель

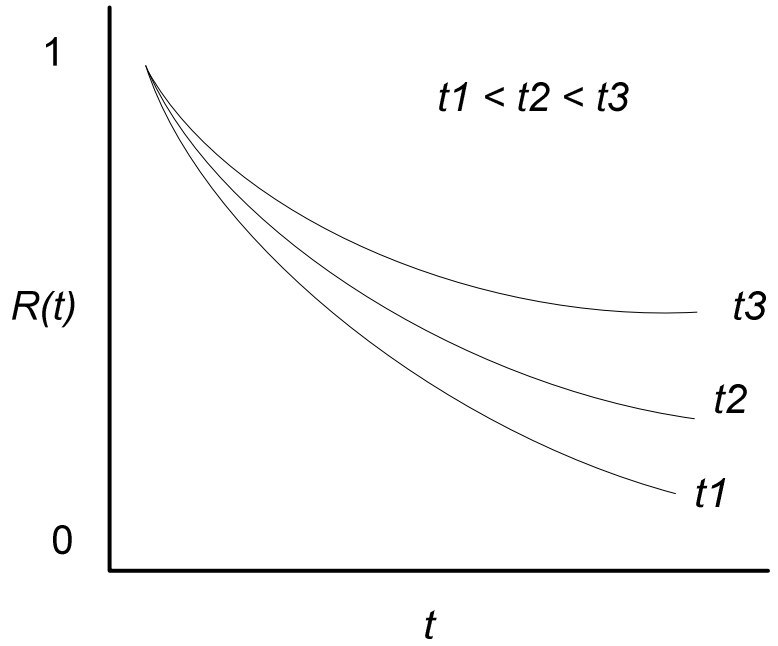
- Предполагается проведение тестирования двумя группами разработчиков независимо
- Пусть первая группа обнаружила *N1* ошибок, вторая – *N2*, *N12* - это ошибки, обнаруженные обеими группами
- Если *N* число ошибок в программе, то можно определить эффективность тестирования каждой группы: *E1=N1/N* и *E2=N2/N*



- Считая возможность обнаружения ошибок одинаковой для обеих групп, *E1=N1/N = N12/N2*, что позволяет оценить *N=N1*N2/N12*

Модели роста надежности

- Модели роста надежности (*Software reliability growth models*) – прогнозные модели, в которых предполагается, что в процессе тестирования надежность увеличивается
- Процесс тестирования делиться на этапы между отказами, при отказе обнаруживаются и устраняются ошибки, вызвавшие этот отказ и начинается следующий этап тестирования



R(t) – вероятность отсутствия отказов на интервале времени [0, t]

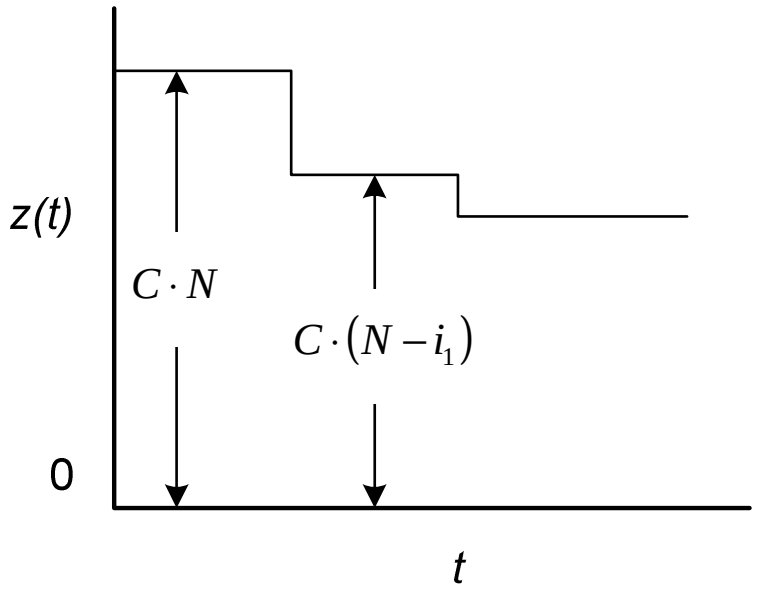
Модели роста надежности

- Исходные данными для моделей роста надежности
  - Количество отказов на этапах тестирования
  - Времена между отдельными отказами
- Выполняется экстраполяция исходных данных с учетом предположений о свойствах программы
- Примеры используемых предположений
  - при устранении ошибок новые ошибки не вносятся и надежность программы постоянно увеличивается
  - интенсивность проявления ошибок не изменяется между исправлениями ошибок и является функцией от числа оставшихся ошибок

Модель Шумана

- Предполагается что в начальный момент в программе *N* ошибок. Выполняется несколько этапов тестирования, на которых обнаруживаются ошибки, исправление ошибок выполняется в конце этапов
- Функция риска *z(t)* – условная вероятность появления ошибки на интервале *[t, t+dt]*, при условии, что до момента *t* ошибок не было
- Считается, что *z(t)* не изменяется между этапами тестирования и пропорциональна числу оставшихся ошибок

$$z(t) = C \cdot (N - i)$$



Модель Шумана

- Вероятность отсутствия отказа на интервале времени *[0, t]*, *R(t)*

$$z(t) = -\frac{dR}{dt} \cdot R(t)$$

- При *R(0)=1* решение этого уравнения

$$R(t) = e^{-\int_0^t z(x)\cdot dx}, R(t) = e^{-C(N-i)t}$$

- Среднее время между отказами $MTBF = \int_o^\infty R(t) \cdot dt$
- Неизвестные значения *C* и *N* определяются по результатам нескольких экспериментов с помощью функции максимального правдоподобия

Модель Джелинского-Моранды

- В данной модели учитывается не число отказов на интервале времени, а промежутки времени между отдельными отказами
- Предполагается что промежутки времени между обнаружением двух ошибок имеют экспоненциальное распределение, интенсивность пропорциональна числу оставшихся ошибок
- На каждом этапе тестирования обнаруживается одна ошибка, которая устраняется до следующего этапа тестирования
- Плотность распределения времени обнаружения *i*-й ошибки с момента обнаружения *(i-1)*-й ошибки

$$p(t_i) = \lambda_i \cdot e^{-\lambda_i t_i}, \lambda_i = C \cdot (N - i + 1)$$

Модель Шика-Волвертона

- Модификация модели Джелинского - Моранды для случая возникновения более одной ошибки на каждом промежутке времени
- В основе модели лежит предположение, согласно которому частота ошибок пропорциональна не только количеству ошибок в программах, но и времени тестирования – вероятность обнаружения ошибок с течением времени возрастает
- Интенсивность обнаружения ошибок

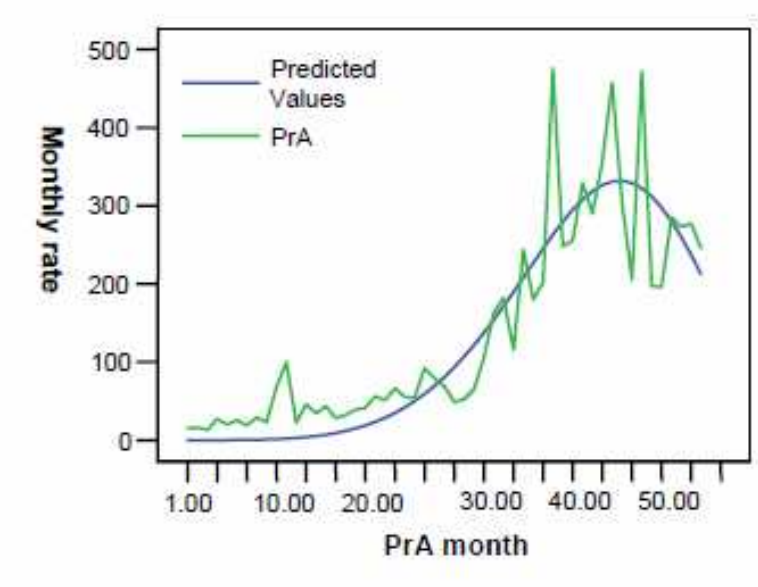
$$\lambda_i = C \cdot \left(N - n_{i-1}\right) \cdot \left(T_{i-1} + t_i / 2\right)$$

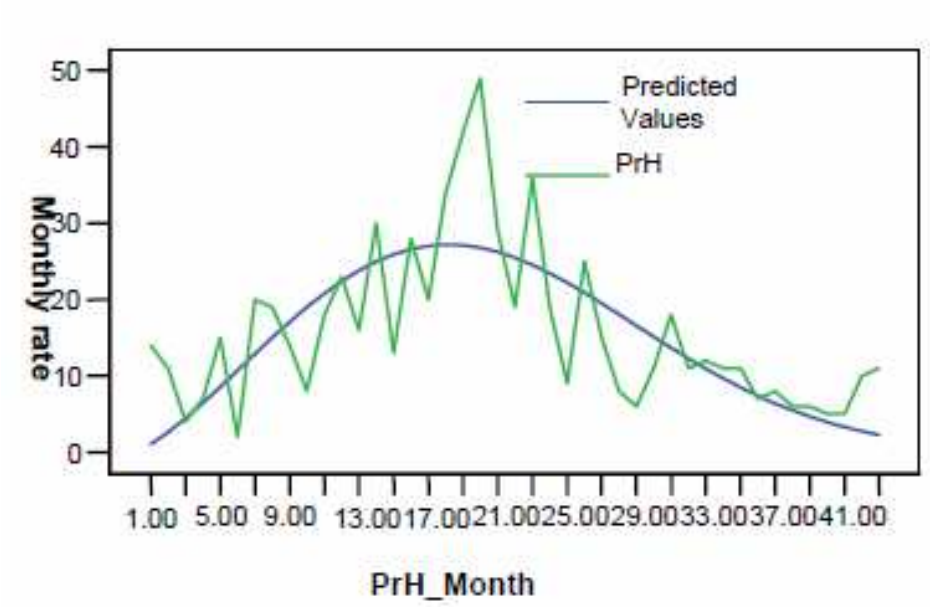
Модель Вейбулла

- Одна из используемых динамических моделей

$$f(t) = \lambda\beta(\lambda t)^{\beta-1} \cdot e^{-(\lambda t)^{\beta}}$$

- Прогноз числа обнаруживаемых ошибок в 8 проектах с открытым кодом, корреляция 0.1-0.8

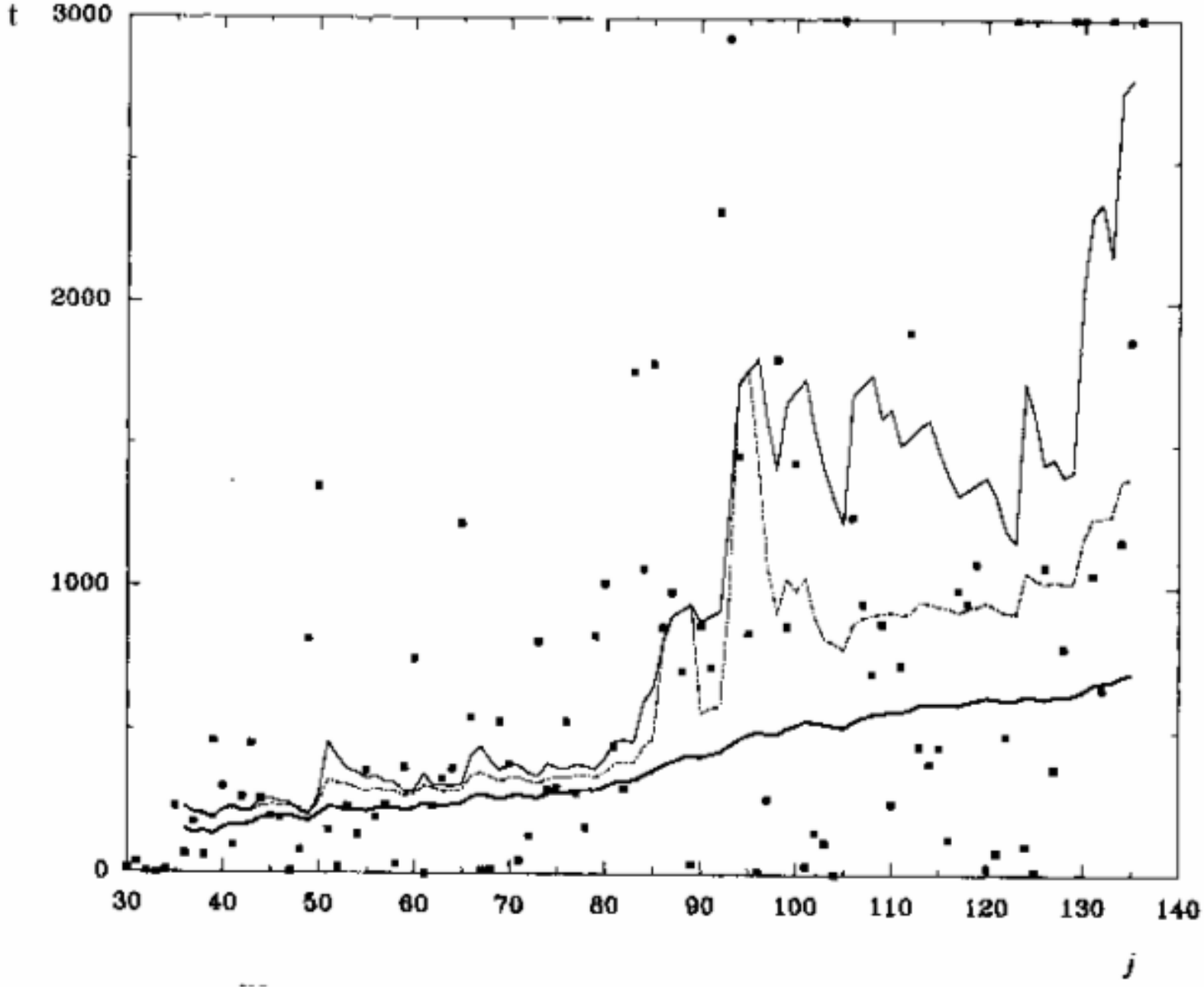




Другие динамические модели

- Модель Ла Падуа
- Модель Мусы
- Модель на основе неоднородного Пуассоновского распределения
- Модель Гоело-Окумото

Пример сравнения моделей



Свойства динамических методов

- Достоверность получаемых результатов сильно зависит от качества исходных данных. Для оценки достоверности используются метрики покрытия кода
- При использовании прогнозных моделей обычно не учитываются влияния нерегулярных флуктуаций, имеющих место в процессе разработки и отладки ПО (особенности проекта, неравномерная плотность дефектов, квалификация персонала и др.)

Свойства динамических методов

- Достоинства
  - Позволяют получать абсолютные показатели надежности
- Недостатки
  - высокая трудоемкость сбора исходной информации
  - использование упрощающих предположений о взаимных влияниях программных ошибок
  - сильная зависимость точности прогнозов от качества и объема исходной информации

Методы на основе моделей сложности

- В основе методов лежит использование взаимосвязи между сложностью и надежностью программной системы
  - Используются те или иные метрики сложности, получаемые с помощью синтаксического анализа исходного кода программы
  - Рассчитанные метрики преобразуются в оценку надежности по некоему эмпирическому правилу

Методы на основе моделей сложности

- Метрики размера программы
  - Метрика Холстеда

*Сложность V=N*log2(n), N – размер программы, n – размер словаря*)

число ошибок B = V / K, K – некоторый коэффициент (3000)

- Метрики сложности потока управления
  - Метрика Маккейба

Цикломатическая сложность Z(G)=e-v+2p, e - число вершин, v - число дуг, p – число компонентов связности ориентированного графа G)

  - Метрика Джиббла

Учитывает насыщенность программы операторами ветвления и глубину вложенности этих операторов.

Методы на основе моделей сложности

- Метрики сложности потока данных
  - Метрика использования глобальных переменных

Оценивает степень использования глобальных переменных

  - Метрика Чепина

*Сложность Q = a1*P + a2*M + a3*C + a4*T, переменные P – входные и выходные, M – модифицируемые, C – управляющие, T – неиспользуемые, ai – некоторые коэффициенты.*

Методы на основе моделей сложности

- Используя метрики сложности можно получить оценку надежности с помощью некоторых весовых коэффициентов
- Основной проблемой является получение этих коэффициентов, их рассчитывают например исходя из опыта предыдущих проектов
- При быстром изменении технологий, методов и средств проектирования опыт предыдущих проектов утрачивает свою актуальность

Методы на основе моделей сложности

- Достоинства
  - Простота получения оценок
- Недостатки
  - Использование коэффициентов преобразования сложности в оценку надежности
  - Не учитываются взаимосвязи между отдельными частями программы
  - Не учитывают семантику программы
  - Не учитываются исправленные ошибки

Архитектурные методы

- Архитектурные методы (*Architecture-based software reliability methods*) выполняют
  - декомпозицию программной системы на компоненты
  - оценку надежности каждого компонента в отдельности
  - определение взаимных влияний компонентов
  - формирование общей оценки надежности всей системы
- Используются как динамические так и статические методы
- Программная система представляется в виде Марковских цепей, стохастических сетей Петри и др.
- Попытка расширить область применения динамических методов

Архитектурные методы

- Проблема выделения компонентов
- Проблема определения вероятностей переходов
- Проблема учета взаимных влияний компонентов
  - Свойства программных компонентов, в том числе и надежность, зависят от других компонентов системы
  - Эти зависимости могут носить сложный характер и не определяться только потоками данных и потоками управления в системе

Эмпирические методы

- **Эмпирические методы** – используют информацию о процессе проектирования
  - организация процесса проектирования, сертификаты, уровень зрелости
  - опыт предыдущих проектов

Римская модель

- Учитываемые параметры
  - Тип приложения
  - Характеристики среды разработки
  - Метрики представления и дизайна (управление аномалиями, отслеживаемость, учет качества анализа ПО)
  - Метрики реализации (ЯП, размер программы, модульность программы, степень повторного использования и др.)

$$\delta_0 = A * D * (SA * ST * SQ) * (SL * SS * SM * SU * SX * SR)$$

$$\lambda_0 = F * K * (\delta_0 * \text{количество строк кода ПО}) = F * K * W_0$$

$$1.4 * 10^{-7} \le K \le 10.6 * 10^{-7}$$

Фазовая модель

- Оценка надежности с учетом фазы разработки ПО
  - Уровень надежности ПО определяется уровнем программистов
  - Плотность ошибок – количество ошибок на тысячу строк кода
- Прогнозная оценка числа обнаруженных ошибок до момента *t*

$$V_t = E \cdot \left(1 - e^{-Bt^2}\right)$$

- Коэффициент *B* рассчитывается с учетом обнаруженных ошибок и фазы каждой ошибки

Методы на основе обнаружения ошибок

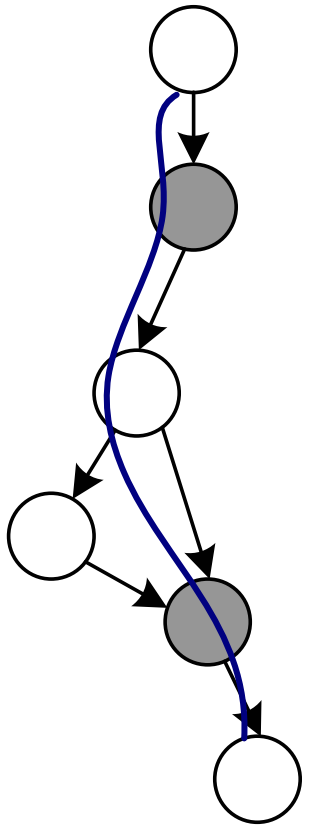
- Методы на основе статически обнаруженных программных ошибок оценивают надежность ПО рассматривая основную причину ненадежности – программные ошибки
  - Выполняется обнаружение ошибок
  - Для каждой ошибки оценивается вероятность срабатывания
  - Для каждой ошибки оценивается вероятность проявления
- Определяемые показатели надежности
  - Серверные программы – среднее число выполненных операторов (MTTN), вероятность успешного выполнения n операторов – P(n)
  - Вычислительные программы – коэффициент готовности

Методы на основе обнаружения ошибок

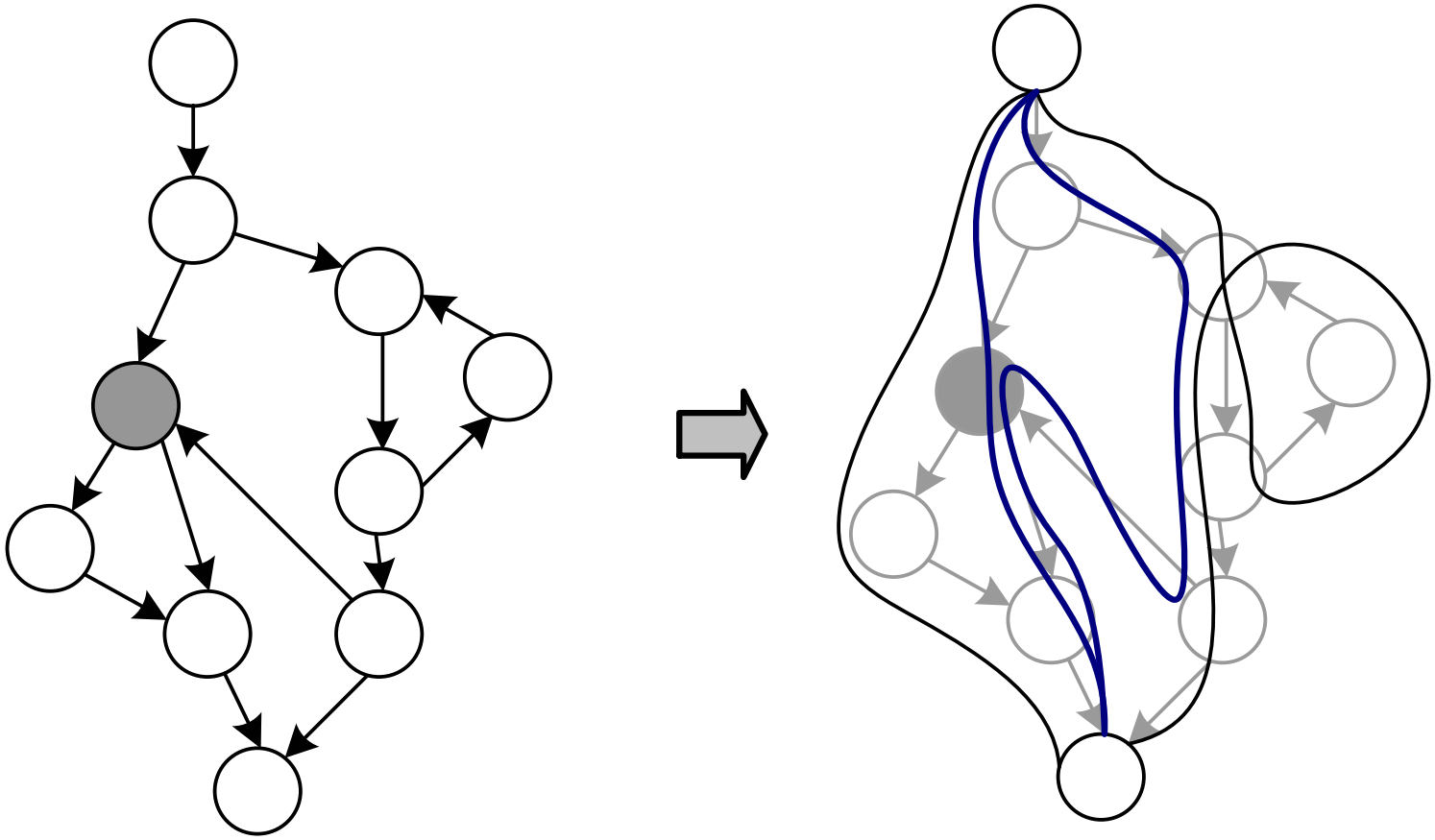
- Обнаружение ошибок выполняется с помощью рассмотренных статических методов
- Вероятности выполнения путей с помощью статического или динамического анализа
- Вероятности проявления ошибок
  - Для некоторых типов ошибок вероятность равна 1
  - Для других типов ошибок определяется эмпирически

Обнаружение ошибок

- Невозможность проверки и исправления всех ошибок
  - Большое число потенциальных дефектов
  - Внесение новых ошибок при исправлении дефектов
- Анализ множественных дефектов на пути выполнения программы
  - Ложные дефекты
  - Стохастическая природа проявления дефектов

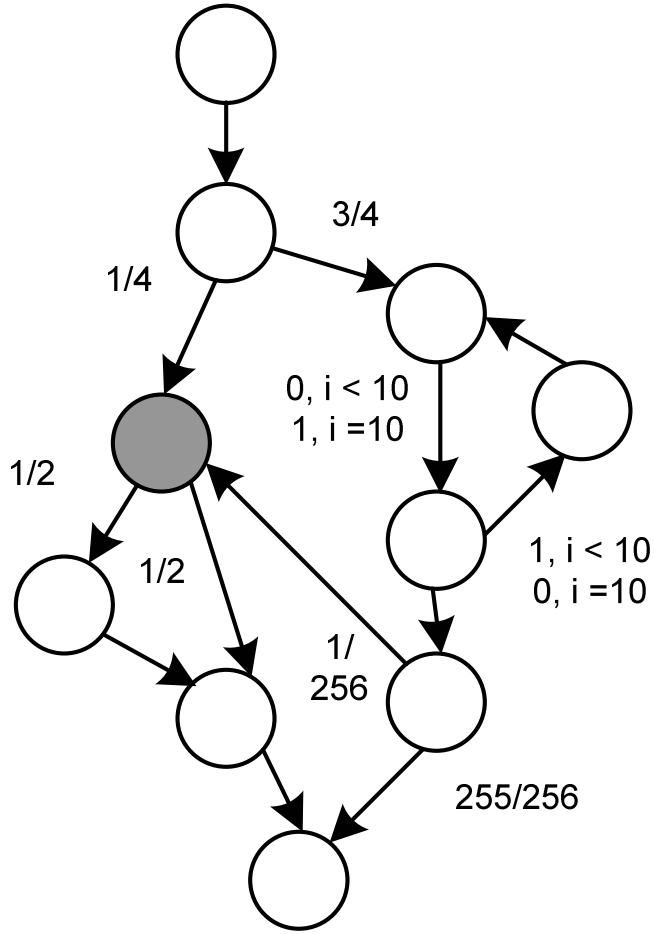


Ошибки на путях выполнения программы



Вероятности путей выполнения программы

- Вероятности путей выполнения определяются вероятностями переходов в операторах ветвления, зависит от
  - Входных данных и внешних воздействий
  - Влияния окружения



Свойства методов на основе обнаружения ошибок

- Достоинства
  - Позволяют выполнять оценку автоматически – низкая трудоемкость
- Недостатки
  - Не обеспечивают получение абсолютных показателей надежности, отсутствует количественное время
  - Сложность статического определения вероятности срабатывания и вероятности проявления ошибки
  - Высокая сложность реализации