ИНТУИТ
НАЦИОНАЛЬНЫЙ ОТКРЫТЫЙ УНИВЕРСИТЕТ

Национальный Открытый Университет "ИНТУИТ": www.intuit.ru
Екатерина Лаврищева, Владимир Петрухин
Лекция 11. Модели качества и надежности в программной инженерии

Разработка ПС достигла такого уровня развития, что стало необходимо использовать инженерные методы, в том числе для оценивания результатов проектирования на этапах ЖЦ, контроля достижения показателей качества и метрического их анализа, оценки риска и степени использования готовых компонентов для снижения стоимости разработки нового проекта. Основу инженерных методов в программировании составляет повышение качества, для достижения которого сформировались методы определения требований к качеству, подходы к выбору и усовершенствованию моделей метрического анализа показателей качества, методы количественного измерения показателей качества на этапах ЖЦ.

Главная составляющая качества – *надежность*, которой уделяется большое внимание в области надежности технических средств и тех критических систем (реального времени, радарные системы, системы безопасности и др.), для которых *надежность* – главная целевая *функция* оценки их реализации. Как следствие, в проблематике надежности разработано более сотни математических моделей надежности, являющихся функциями от ошибок, оставшихся в ПС, от *интенсивности отказов* или частоты появления дефектов в ПС. На их основе производится оценка надежности программной системы.

Качество *ПО* – предмет стандартизации. Стандарт ГОСТ 2844-94 дает *определение* качества *ПО* как совокупность свойств (показателей качества) *ПО*, которые обеспечивают его способность удовлетворять потребности заказчика в соответствии с назначением. Этот стандарт регламентирует базовую модель качества и показатели, главным среди них – *надежность*. Стандарт ISO/IEC 12207 определил не только основные процессы ЖЦ разработки ПС, но и организационные и дополнительные процессы, которые регламентируют инженерию, планирования и управления качеством ПС.

Согласно стандарту на этапах ЖЦ должен проводиться *контроль* качества *ПО*:

- проверка соответствия требований проектируемому продукту и критериев их достижения;
- верификация и аттестация (валидация) промежуточных результатов ПО на этапах ЖЦ и измерение степени удовлетворения достигаемых отдельных показателей;тестирование готовой ПС, сбор данных об отказах, дефектах и других ошибках, обнаруженных в системе;
- подбор моделей надежности для оценивания надежности по полученным результатам тестирования (дефекты, отказы и др.);
- оценка показателей качества, заданных в требованиях на разработку ПС.

Далее излагаются модели качества и надежности, а также способы их применения.

10.1. Модель качества ПО

Качество *ПО* – это относительное понятие, которое имеет смысл только при учете реальных условий его применения, поэтому требования, предъявляемые к качеству, ставятся в соответствии с условиями и конкретной областью их применения. Оно характеризуется тремя аспектами: *качество программного продукта*, качество процессов ЖЦ и качество сопровождения или внедрения (рис. 10.1).

Рис. 10.1. Основные аспекты качества ПО

Аспект, связанный с процессами ЖЦ, определяет степень формализации, достоверности самих процессов ЖЦ разработки *ПО*, а также верификацию и валидацию промежуточных результатов на этих процессах. *Поиск* и устранение ошибок в готовом *ПО* проводится методами тестирования, которые снижают количество ошибок и повышают качество этого продукта.

Качество продукта достигается процедурами контроля промежуточных продуктов на процессах ЖЦ, проверкой их на достижение необходимого качества, а также методами сопровождения продукта. Эффект от внедрения ПС в значительной степени зависит от знаний обслуживающего персонала функций продукта и правил их выполнения.Модель качества *ПО* имеет следующие четыре уровня представления.

Первый уровень соответствует определению характеристик (показателей) качества *ПО*, каждая из которых отражает отдельную точку зрения пользователя на качество. Согласно стандарту [10.1–10.4] в модель качества входит шесть характеристик или шесть показателей качества:

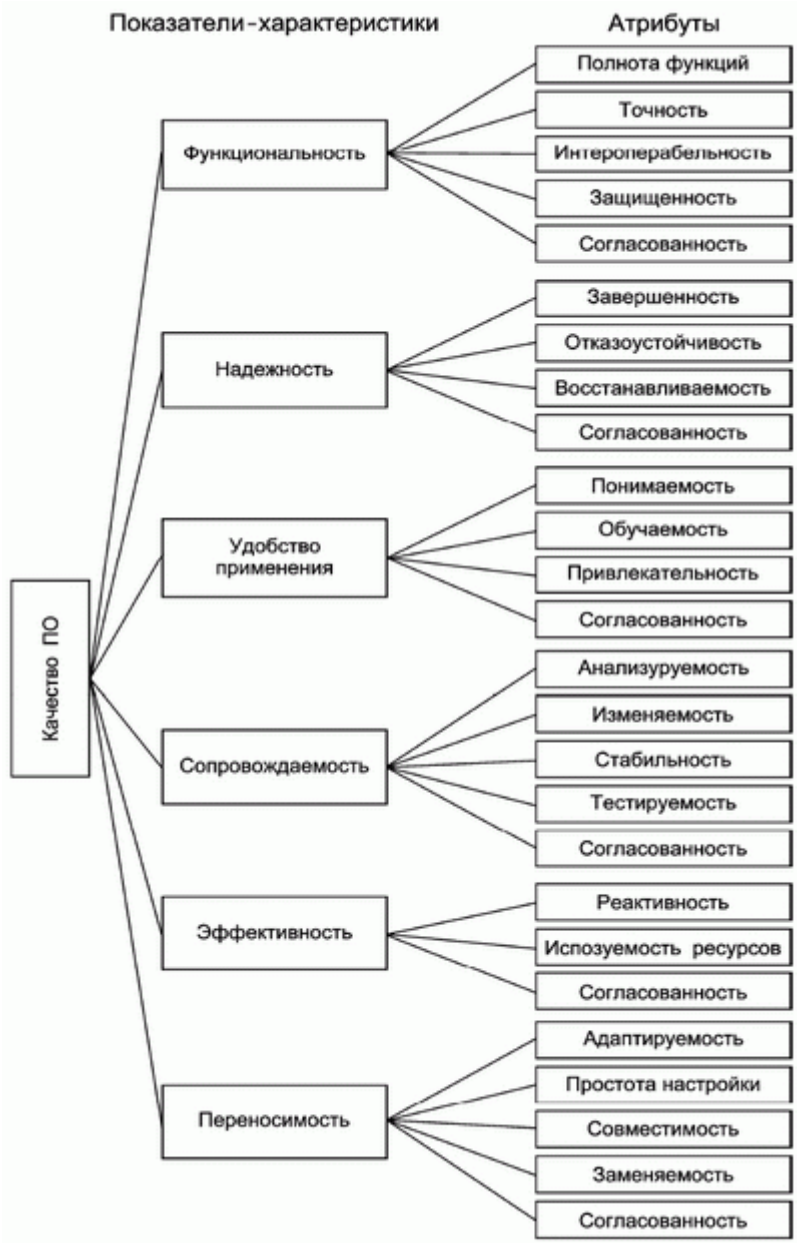
1. функциональность (functionality);
2. надежность (realibility);
3. удобство (*usability*);
4. эффективность (*efficiency*);
5. сопровождаемость (maitainnability);
6. переносимость (portability).

Второму уровню соответствуют атрибуты для каждой характеристики качества, которые детализируют разные аспекты конкретной характеристики. Набор атрибутов характеристик качества используется при оценке качества.

Третий уровень предназначен для измерения качества с помощью метрик, каждая из них согласно стандарту [10.1] определяется как комбинация метода измерения атрибута и шкалы измерения значений атрибутов. Для оценки атрибутов качества на этапах ЖЦ (при просмотре документации, программ и результатов тестирования программ) используются метрики с заданным оценочным весом для нивелирования результатов метрического анализа совокупных атрибутов конкретного показателя и качества в целом. *Атрибут* качества определяется с помощью одной или нескольких методик оценки на этапах ЖЦ и на завершающем этапе разработки *ПО*.

Четвертый уровень – это оценочный элемент метрики (*вес*), который используется для оценки количественного или качественного значения отдельного атрибута показателя *ПО*. В зависимости от назначения, особенностей и условий сопровождения *ПО* выбираются наиболее важные характеристики качества и их атрибуты (рис. 10.2).

Выбранные атрибуты и их приоритеты отражаются в требованиях на разработку систем либо используется соответствующие приоритеты эталона класса *ПО*, к которому это *ПО* относится.



увеличить изображение

Рис. 10.2. Модель характеристик качества

10.1.1. Характеристика показателей качества

Краткое описание семантики характеристик модели качества приведены в табл. 10.1, а их содержательное описание – ниже.

Таблица 10.1. Краткая характеристика показателей качества

Показатель	Описание свойств показателя
Функциональность	Группа свойств ПО, обуславливающая его способность выполнять определенный перечень функций, которые удовлетворяют потребностям в соответствии с назначением
Надежность	Группа свойств, обусловливающая способность ПО сохранять работоспособность и преобразовывать исходные данные в результат за установленный период времени, характер отказов которого является следствием внутренних дефектов и условий его применения
Удобство применения	Совокупность свойств ПО для предполагаемого круга пользователей и отражающих легкость его освоения и адаптации к изменяющимся условиям эксплуатации,

	стабильность работы и подготовки данных, понимаемость результатов, удобства внесения изменений в программную документацию и в программы
Сопровождаемость	Группа свойств, определяющая усилия, необходимые для выполнения, приспособленность к диагностике отказов и последствий внесения изменений, модификации и аттестации модифицируемого ПО
Рациональность	Группа свойств, характеризующаяся степенью соответствия используемых ресурсов среды функционирования уровню качества (надежности) функционирования ПО при заданных условиях применения
Переносимость	Группа свойств ПО, обеспечивающая его приспособленность для переноса из одной среды функционирования в другие, усилия для переноса и адаптацию ПО к новой среде функционирования

1. **Функциональность** – совокупность свойств, определяющих способность ПО выполнять перечень функций в заданной среде и в соответствии с требованиями к обработке и общесистемным средствам. Под *функцией* понимается некоторая упорядоченная последовательность действий для удовлетворения потребительских свойств. Функции бывают целевые (основные) и вспомогательные.

   К атрибутам функциональности относятся:

   функциональная полнота – свойство компонента, которое показывает степень достаточности основных функций для решения задач в соответствии с назначением ПО;

   правильность (точность) – атрибут, который показывает степень достижения правильных результатов;

   интероперабельность – атрибут, который показывает возможность взаимодействовать на ПО специальными системами и средами (ОС, сеть);

   защищенность – атрибут, который показывает на способность ПО предотвращать несанкционированный доступ (случайный или умышленный) к программам и данным.

2. **Надежность** – совокупность атрибутов, которые определяют способность ПО преобразовывать исходные данные в результаты при условиях, зависящих от периода времени жизни (износ и его старение не учитываются). Снижение надежности ПО происходит из-за ошибок в требованиях, проектировании и выполнении. Отказы и ошибки в программах появляются на заданном промежутке времени [10.5–10.10].

   К подхарактеристикам надежности ПО относятся:

   безотказность – атрибут, который определяет способность ПО функционировать без отказов (как программы, так и оборудования);

   устойчивость к ошибкам – атрибут, который показывает на способность ПО выполнять функции при аномальных условиях (сбой аппаратуры, ошибки в данных и интерфейсах, нарушение в действиях оператора и др.);

   восстанавливаемость – атрибут, который показывает на способность программы к перезапуску для повторного выполнения и восстановления данных после отказов.

   К некоторым типам систем (реального времени, радарных, систем безопасности, коммуникация и др.) предъявляются требования для обеспечения высокой надежности (недопустимость ошибок, точность, достоверность, удобство применения и др.). Таким образом, надежность ПО в значительной степени зависит от числа оставшихся и не устраненных ошибок в процессе разработки на этапах ЖЦ. В ходе эксплуатации ошибки обнаруживаются и устраняются.

   Если при исправлении ошибок не вносятся новые или, по крайней мере, новых ошибок вносится меньше, чем устраняется, то в ходе эксплуатации надежность ПО непрерывно возрастает. Чем интенсивнее проводится эксплуатация, тем интенсивнее выявляются ошибки и быстрее растет надежность ПО.

   К факторам, влияющим на надежность ПО, относятся:

   совокупность угроз, приводящих к неблагоприятным последствиям и к ущербу системы или среды ее функционирования;

   угроза как проявление нарушения безопасности системы;

   целостность как способность системы сохранять устойчивость работы и не иметь риска.

Обнаруженные ошибки могут быть результатом угрозы извне или отказов, они повышают риск и уменьшают некоторые свойства надежности системы.

Надежность – одна из ключевых проблем современных программных систем, и ее роль будет постоянно возрастать, поскольку постоянно повышаются требования к качеству компьютерных систем. Новое направление – инженерия программной надежности (*Software reliability* engineering) – ориентировано на количественное изучение операционного поведения компонентов системы по отношению к пользователю, ожидающему надежную работу системы [10.7], и включает:

- измерение надежности, т.е. проведение ее количественной оценки с помощью предсказаний, сбора данных о поведении системы в процессе эксплуатации и современных моделей надежности;
- стратегии и метрики конструирования и выбора готовых компонентов, процесс разработки компонентной системы, а также среда функционирования, влияющая на надежность работы системы;
- применение современных методов инспектирования, верификации, валидации и тестирования при разработке систем, а также при эксплуатации.

Верификация применяется для определения соответствия готового ПО установленным спецификациям, а валидация – для установления соответствия системы требованиям пользователя, которые были предъявлены заказчиком.

В инженерии надежности термин dependability (пригодноспособность) обозначает способность системы иметь свойства, желательные для пользователя, который уверен в качественном выполнении функций ПС, заданных в требованиях. Данный термин определяется дополнительным количеством атрибутов, которыми должна обладать система, а именно:

- готовность к использованию (availability);
- готовностью к непрерывному функционированию (*reliability*);
- безопасность для окружающей среды, т.е. способность системы не вызывать катастрофических последствий в случае отказа (*safety*);
- секретность и сохранность информации (confidential);
- способность к сохранению системы и устойчивости к самопроизвольному ее изменению (integrity);
- способность к эксплуатации ПО, простота выполнения операций обслуживания, а также устранения ошибок, восстановление системы после их устранения и т.п. (*maintainability*);
- готовность и сохранность информации (security) и др.

Достижение надежности системы обеспечивается предотвращением отказа (fault prevention), его устранением (removal fault), а также оценкой возможности появления новых отказов и мер борьбы с ними с применением методов теории вероятности.

Каждый программный компонент, его операции и данные обрабатываются в дискретные моменты времени, например, $\delta, 2\delta, \ldots n\delta$. Пусть за время T после первого неудачно обработанного компонента системы появился отказ, q_b – вероятность этой неудачи, тогда $P\{T > n\delta\} = (1 - q_s)^n$ и среднее время ожидания $T = \delta / q_s$.

Положим, что момент времени δ убывает, а время T остается фиксированным, тогда имеем $P\{T > t\} = (1 - \frac{\delta}{T})\frac{t}{\delta} = e^{-\frac{t}{T}}$, время до отказа в данном случае – непрерывная величина $c \ldots l$, распределенная экспоненциально с параметром $T, \ldots, 1/T$.

Таким образом, оценка надежности ПО – это трудоемкий процесс, требующий создания устойчивой работы системы по отношению к отказам ПО, т.е. вероятности того, что система восстановится самопроизвольно в некоторой точке после возникновения в ней отказа (fault).

3. **Удобство применения** характеризуется множеством атрибутов, которые показывают на необходимые и пригодные условия использования (диалоговое или не диалоговое) ПО заданным кругом пользователей для получения соответствующих результатов. В стандарте [10.3] удобство применения определено как специфическое множество атрибутов программного продукта, характеризующих его эргономичность.

   К подхарактеристиками удобства применения относятся:

   - понимаемость – атрибут, который определяет усилия, затрачиваемые на распознавание логических концепций и условий применения ПО;

изучаемость (легкость изучения) – атрибут, который определяет усилия пользователей на определение применимости ПО путем использования операционного контроля, диагностики, а также процедур, правил и документации;

оперативность – атрибут, который показывает на реакцию системы при выполнении операций и операционного контроля;

согласованность – атрибут, который показывает соответствие разработки требованиям стандартов, соглашений, правил, законов и предписаний.

4. **Эффективность** – множество атрибутов, которые определяют взаимосвязь уровней выполнения ПО, использования ресурсов (средства, аппаратура, материалы – бумага для печатающего устройства и др.) и услуг, выполняемых штатным обслуживающим персоналом и др.

   К подхарактеристикам эффективности ПО относятся:

   реактивность – атрибут, который показывает время отклика, обработки и выполнения функций;

   эффективность ресурсов – атрибут, показывающий количество и продолжительность используемых ресурсов при выполнении функций ПО;

   согласованность – атрибут, который показывает соответствие данного атрибута с заданными стандартами, правилами и предписаниями.

5. **Сопровождаемость** – множество свойств, которые показывают на усилия, которые надо затратить на проведение модификаций, включающих корректировку, усовершенствование и адаптацию ПО при изменении среды, требований или функциональных спецификаций.

   Сопровождаемость включает подхарактеристики:

   анализируемость – атрибут, определяющий необходимые усилия для диагностики отказов или идентификации частей, которые будут модифицироваться;изменяемость – атрибут, который определяет удаление ошибок в ПО или внесение изменений для их устранения, а также введение новых возможностей в ПО или в среду функционирования;

   стабильность – атрибут, указывающий на постоянство структуры и риск ее модификации;

   тестируемость – атрибут, показывающий на усилия при проведении валидации и верификации с целью обнаружения несоответствий требованиям, а также на необходимость проведения модификации ПО и сертификации;

   согласованность – атрибут, который показывает соответствие данного атрибута соглашениям, правилам и предписаниям стандарта.

6. **Переносимость** – множество показателей, указывающих на способность ПО адаптироваться к работе в новых условиях среды выполнения. Среда может быть организационной, аппаратной и программной. Поэтому перенос ПО в новую среду выполнения может быть связан с совокупностью действий, направленных на обеспечение его функционирования в среде, отличной от той среды, в которой оно создавалось с учетом новых программных, организационных и технических возможностей.

   Переносимость включает подхарактеристики:

   адаптивность – атрибут, определяющий усилия, затрачиваемые на адаптацию к различным средам;

   настраиваемость (простота инсталляции) – атрибут, который определяет необходимые усилия для запуска данного ПО в специальной среде;

   сосуществование – атрибут, который определяет возможность использования специального ПО в среде действующей системы;

   заменяемость – атрибут, который обеспечивают возможность *интероперабельности* при совместной работе с другими программами с необходимой инсталляцией или адаптацией ПО;

   согласованность – атрибут, который показывает на соответствие стандартам или соглашениями по обеспечению переноса ПО.

10.1.2. Метрики качества программного обеспечения

В настоящее время в программной инженерии еще не сформировалась окончательно система метрик. Действуют разные подходы к определению их набора и методов измерения [10.11–10.13].

Система измерения включает метрики и модели измерений, которые используются для количественной оценки качества ПО.

При определении требований к ПО задаются соответствующие им внешние характеристики и их атрибуты (подхарактеристики), определяющие разные стороны управления продуктом в заданной среде. Для набора

характеристик качества ПО, приведенных в требованиях, определяются соответствующие метрики, модели их оценки и диапазон значений мер для измерения отдельных атрибутов качества.

Согласно стандарту [1.16] метрики определяются по модели измерения атрибутов ПО на всех этапах ЖЦ (промежуточная, внутренняя метрика) и особенно на этапе тестирования или функционирования (внешние метрики) продукта.

Остановимся на классификации метрик ПО, правилах для проведения метрического анализа и процесса их измерения.

Типы метрик.

Существует три типа метрик:

- метрики программного продукта, которые используются при измерении его характеристик – свойств;
- метрики процесса, которые используются при измерении свойства процесса ЖЦ создания продукта.
- метрики использования.

Метрики программного продукта включают:

- внешние метрики, обозначающие свойства продукта, видимые пользователю;
- внутренние метрики, обозначающие свойства, видимые только команде разработчиков.

Внешние метрики продукта – это метрики:

- надежности продукта, которые служат для определения числа дефектов;
- функциональности, с помощью которых устанавливаются наличие и правильность реализации функций в продукте;
- сопровождения, с помощью которых измеряются ресурсы продукта (скорость, память, среда);применимости продукта, которые способствуют определению степени доступности для изучения и использования;
- стоимости, которыми определяется стоимость созданного продукта.

Внутренние метрики продукта включают:

- метрики размера, необходимые для измерения продукта с помощью его внутренних характеристик;
- метрики сложности, необходимые для определения сложности продукта;
- метрики стиля, которые служат для определения подходов и технологий создания отдельных компонентов продукта и его документов.

Внутренние метрики позволяют определить производительность продукта и являются релевантными по отношению к внешним метрикам.

Внешние и внутренние метрики задаются на этапе формирования требований к ПО и являются предметом планирования и управления достижением качества конечного программного продукта.

Метрики продукта часто описываются комплексом моделей для установки различных свойств, значений модели качества или прогнозирования. Измерения проводятся, как правило, после калибровки метрик на ранних этапах проекта. Общая мера – степень трассируемости, которая определяется числом трасс, прослеживаемых с помощью моделей сценариев типа UML и оценкой количества:

- требований;
- сценариев и действующих лиц;
- объектов, включенных в сценарий, и локализация требований к каждому сценарию;
- параметров и операций объекта и др.

Стандарт ISO/IEC 9126-2 определяет следующие типы мер:

- мера размера ПО в разных единицах измерения (число функций, строк в программе, размер дисковой памяти и др.);
- мера времени (функционирования системы, выполнения компонента и др.);
- мера усилий (производительность труда, трудоемкость и др.);
- мера учета (количество ошибок, число отказов, ответов системы и др.).

Специальной мерой может служить уровень использования повторных компонентов и измеряется как отношение размера продукта, изготовленного из готовых компонентов, к размеру системы в целом. Данная мера используется также при определении стоимости и качества ПО. Примеры метрик:

- общее число объектов и число повторно используемых;
- общее число операций, повторно используемых и новых операций;

число классов, наследующих специфические операции;

число классов, от которых зависит данный класс;

число пользователей класса или операций и др.

При оценке общего количества некоторых величин часто используются среднестатистические метрики (среднее число операций в классе, наследников класса или операций класса и др.).

Как правило, меры в значительной степени являются субъективными и зависят от знаний экспертов, производящих количественные оценки атрибутов компонентов программного продукта.

Примером широко используемых внешних метрик программ являются метрики Холстеда – это характеристики программ, выявляемые на основе статической структуры программы на конкретном языке программирования: число вхождений наиболее часто встречающихся операндов и операторов; длина описания программы как сумма числа вхождений всех операндов и операторов и др.

На основе этих атрибутов можно вычислить время программирования, уровень программы (структурированность и качество) и языка программирования (абстракции средств языка и ориентация на проблему) и др.

В качестве метрик процесса могут быть время разработки, число ошибок, найденных на этапе тестирования и др. Практически используются следующие метрики процесса:

общее время разработки и отдельно время для каждой стадии;

время модификации моделей;

время выполнения работ на процессе;

число найденных ошибок при инспектировании;

стоимость проверки качества;

стоимость процесса разработки.

Метрики использования служат для измерения степени удовлетворения потребностей пользователя при решении его задач. Они помогают оценить не свойства самой программы, а результаты ее эксплуатации – эксплуатационное качество. Примером может служить – точность и полнота реализации задач пользователя, а такжезатраченные ресурсы (трудозатраты, производительность и др.) на эффективное решение задач пользователя. *Оценка требований* пользователя проводится с помощью внешних метрик.

10.1.3. Стандартная оценка значений показателей качества

Оценка качества ПО согласно четырехуровневой модели качества начинается с нижнего уровня иерархии, т.е. с самого элементарного свойства оцениваемого атрибута показателя качества согласно установленных мер. На этапе проектирования устанавливают значения оценочных элементов для каждого атрибута показателя анализируемого ПО, включенного в требования.

По определению стандарта ISO/IES 9126-2 метрика качества ПО представляет собой "модель измерения атрибута, связываемого с показателем его качества". При измерении показателей качества данный стандарт позволяет определять следующие типы мер:

меры размера в разных единицах измерения (количество функций, размер программы, объем ресурсов и др.);

меры времени – периоды реального, процессорного или календарного времени (время функционирования системы, время выполнения компонента, время использования и др.);

меры усилий – продуктивное время, затраченное на реализацию проекта (производительность труда отдельных участников проекта, коллективная трудоемкость и др.);

меры интервалов между событиями, например, время между последовательными отказами;

счетные меры – счетчики для определения количества обнаруженных ошибок, структурной сложности программы, числа несовместимых элементов, числа изменений (например, число обнаруженных отказов и др.).

Метрики качества используются при оценке степени тестируемости с помощью данных (*безотказная* работа, выполнимость функций, удобство применения интерфейсов пользователей, БД и т.п.) после проведения испытаний ПО на множестве тестов.

Наработка на отказ как атрибут надежности определяет среднее время между появлением угроз, нарушающих безопасность, и обеспечивает трудноизмеримую оценку ущерба, которая наносится соответствующими угрозами.Очень часто оценка программы проводится по числу строк. При сопоставлении двух программ, реализующих одну прикладную задачу, предпочтение отдается короткой программе, так как её создает более квалифицированный персонал и в ней меньше скрытых ошибок и легче модифицировать. По стоимости она дороже, хотя времени на отладку и модификацию уходит больше. Т.е. длину программы можно использовать в качестве вспомогательного свойства для сравнения программ с учетом одинаковой квалификации разработчиков, единого стиля разработки и общей среды.

Если в требованиях к ПО было указано получить несколько показателей, то просчитанный после сбора данных показатель умножается на соответствующий весовой коэффициент, а затем суммируются все показатели для получения комплексной оценки уровня качества ПО.

На основе измерения количественных характеристик и проведения экспертизы качественных показателей с применением весовых коэффициентов, нивелирующих разные показатели, вычисляется итоговая оценка качества продукта путем суммирования результатов по отдельным показателям и сравнения их с эталонными показателями ПО (стоимость, время, ресурсы и др.).

1. Т.е. при проведении оценки отдельного показателя с помощью оценочных элементов просчитывается весомый коэффициент k -метрика, j -показатель, i -атрибут. Например, в качестве j -показателя возьмем переносимость. Этот показатель будет вычисляться по пяти атрибутам ($i = 1, \ldots, 5$), причем каждый из них будет умножаться на соответствующий коэффициент k_i .

Все метрики j -атрибута суммируются и образуют i -показатель качества. Когда все атрибуты оценены по каждому из показателей качества, производится суммарная оценка отдельного показателя, а потом и интегральная оценка качества с учетом весовых коэффициентов всех показателей ПО.

В конечном итоге результат оценки качества является критерием эффективности и целесообразности применения методов проектирования, инструментальных средств и методик оценивания результатов создания программного продукта на стадиях ЖЦ.

Для изложения оценки значений показателей качества используется стандарт [10.4], в котором представлены следующие методы: измерительный, регистрационный, расчетный и экспертный (а также комбинации этих методов). *Измерительный метод* базируется на использовании измерительных и специальных программных средств для получения информации о характеристиках ПО, например, определение объема, числа строк кода, операторов, количества ветвей в программе, число точек входа (выхода), реактивность и др.

Регистрационный метод используется при подсчете времени, числа сбоев или отказов, начала и конца работы ПО в процессе его выполнения.

Расчетный метод базируется на статистических данных, собранных при проведении испытаний, эксплуатации и сопровождении ПО. Расчетными методами оцениваются *показатели надежности*, точности, устойчивости, реактивности и др.

Экспертный метод осуществляется группой экспертов – специалистов, компетентных в решении данной задачи или типа ПО. Их оценка базируется на опыте и интуиции, а не на непосредственных результатах расчетов или экспериментов. Этот метод проводится путем просмотра программ, кодов, сопроводительных документов и способствует качественной оценки созданного продукта. Для этого устанавливаются контролируемые признаки, которые коррелированны с одним или несколькими показателями качества и включены в опросные карты экспертов. Метод применяется при оценке таких показателей, как анализируемость, документируемость, структурированность ПО и др.

Для оценки значений показателей качества в зависимости от особенностей используемых ими свойств, назначения, способов их определения используются:

шкала метрическая (1.1 – абсолютная, 1.2 – относительная, 1.3 – интегральная);

шкала порядковая (ранговая), позволяющая ранжировать характеристики путем сравнения с опорными;

классификационная шкала, характеризующая наличие или отсутствие рассматриваемого свойства у оцениваемого программного обеспечения.

Показатели, которые вычисляются с помощью метрических шкал, называются количественные, а определяемые с помощью порядковых и классификационных шкал – качественные.

Атрибуты программной системы, характеризующие ее качество, измеряются с использованием метрик качества. Метрика определяет меру атрибута, т.е. переменную, которой присваивается значение в результате измерения.Для правильного использования результатов измерений каждая мера идентифицируется шкалой измерений.

Стандарт ISO/IES 9126-2 рекомендует применять 5 видов шкал измерения значений, которые упорядочены от менее строгой к более строгой:

номинальная шкала отражает категории свойств оцениваемого объекта без их упорядочения;

порядковая шкала служит для упорядочивания характеристики по возрастанию или убыванию путем сравнения их с базовыми значениями;

интервальная шкала задает существенные свойства объекта (например, календарная дата);

относительная шкала задает некоторое значение относительно выбранной единицы;

абсолютная шкала указывает на фактическое значение величины (например, число ошибок в программе равно 10).

10.1.4. Управление качеством ПС

Под *управлением качества* понимается совокупность организационной структуры и ответственных лиц, а также процедур, процессов и ресурсов для планирования и управления достижением качества ПС. Управление качеством – SQM (Software *Quality Management*) базируется на применении стандартных положений по гарантии качества – *SQA* (Software *Quality Assurance*) [10.4, 10.15].

Цель процесса *SQA* состоит в гарантировании того, что продукты и процессы согласуются с требованиями, соответствуют планам и включают следующие виды деятельности:

внедрение стандартов и соответствующих процедур разработки ПС на этапах ЖЦ;

оценка соблюдения положений этих стандартов и процедур. Гарантия качества состоит в следующем:

проверка непротиворечивости и выполнимости планов;

согласование промежуточных *рабочих продуктов* с плановыми показателями;

проверка изготовленных продуктов заданным требованиям;

анализ применяемых процессов на соответствие договору и планам;согласование с заказчиком среды и методов разработки продукта;

проверка принятых метрик продуктов, процессов и приемов их измерения в соответствии с утвержденным стандартом и процедурами измерения.

Цель процесса управления SQM – мониторинг (систематический контроль) качества для гарантии того, что продукт будет удовлетворять потребителю и предполагает выполнение следующего:

определение количественных свойств качества, основанных на выявленных и предусмотренных потребностях пользователей;

управление реализацией поставленных целей для достижения качества.

Процесс SQM основывается на гарантии того, что:

цели достижения требуемого качества установлены для всех *рабочих продуктов* в контрольных точках продукта;

определена стратегия достижения качества, метрики, критерии, приемы, требования к процессу измерения и др.;

определены и выполняются действия, связанные с предоставлением продуктам свойств качества;

проводится контроль качества (*SQA*, верификация и валидация) и целей;

выполняются процессы измерения и оценивания конечного продукта на достижение требуемого качества.

Основные стандартные положения [10.1[10.2–10.4, 10.15] по созданию качественного продукта и оценки достигнутого его уровня позволяют выделить два процесса обеспечения качества на этапах ЖЦ:

гарантия (подтверждение) качества ПС как результат определенной деятельности на каждом этапе ЖЦ с проверкой соответствия системы стандартам и процедурам, ориентированным на достижении качества;

инженерия качества как процесс предоставления продуктам ПО свойств функциональности, надежности, сопровождения и других характеристик качества.

Процессы достижения качества предназначены для:

управления, разработки и обеспечения гарантий в соответствии с указанными стандартами и процедурами;

управления конфигурацией (идентификация, учет состояния и действий по аутентификации), риском и проектом в соответствии со стандартами и процедурами;

контроль базовой версии ПС и реализованных в ней характеристик качества.

Выполнение указанных процессов включает такие действия:

оценка стандартов и процедур, которые выполняются при разработке программ;

ревизия управления, разработки и обеспечение гарантии качества ПО, а также проектной документации (отчеты, графики разработки, сообщения и др.);

контроль проведения формальных инспекций и просмотров;

анализ и контроль проведения приемочного тестирования (испытания) ПС.

Для организации, которая занимается разработкой ПС, в том числе из компонентов, инженерия качества ПС должна поддерживаться системой управлением качеством (планирование, учет и контроль).

Инженерия качества включает набор методов и мероприятий, с помощью которых программные продукты проверяются на выполнение требований к качеству и снабжаются характеристиками, предусмотренными в требованиях на ПО.

Система качества (Quality systems - QS) [10.15] - это набор организационных структур, методик, мероприятий, процессов и ресурсов для осуществления управления качеством. Для обеспечения требуемого уровня качества ПО применяются два подхода. Один из них ориентирован на конечный программный продукт, а второй - на процесс создания продукта.

При подходе, ориентированном на продукт, оценка качества проводится после испытания ПС. Этот подход базируется на предположении, что чем больше обнаружено и устранено ошибок в продукте при испытаниях, тем выше его качество.

При втором подходе предусматриваются и принимаются меры по предотвращению, оперативному выявлению и устранению ошибок, начиная с начальных этапов ЖЦ в соответствии с планом и процедурами обеспечения качества разрабатываемой ПС. Этот подход представлен в серии стандартов *ISO 9000* и 9000-1,2,3, который дает рекомендации организациямразработчикам создавать систему качества согласно схемы, приведенной на рис. 10.3.

Важное место в инженерии качества отводится процессу измерения характеристик процессов ЖЦ, его ресурсов и создаваемых на них *рабочих продуктов*. Этот процесс реализуется группой качества,верификации и тестирования. В ее функции входит планирование, оперативное управление и обеспечение качества.

Рис. 10.3. Требования стандарта к организации системы качества

Планирование качества представляет собою деятельность, направленную на определение целей и требований к качеству. Оно охватывает идентификацию, установление целей, требований к качеству, классификацию и оценку качества. Составляется календарный планграфик для проведения анализа состояния разработки и последовательного измерения спланированных показателей и критериев на этапах ЖЦ.

Оперативное управление включает методы и виды деятельности оперативного характера для текущего управления процессом проектирования и устранения причин плохого или неудовлетворительного функционирования ПС.

Обеспечение качества заключается в выполнении и проверки того, что объект разработки выполняет указанные требования к качеству. Цели обеспечения качества могут быть внутренние и внешние. Внутренние цели – создание уверенности у руководителя проекта, что качество обеспечивается. Внешние цели – это создание уверенности у пользователя, что требуемое качество достигнуто и получено качественное программное обеспечение.

Как показывает опыт, ряд фирм, выпускающих программную продукцию, имеют системы качества, что обеспечивает им производство конкурентоспособной продукции. Система качества включает мониторинг спроса на выпускаемый новый вид продукции, контроль всех звеньев его производства, включая подбор и поставку готовых компонентов для системы.

При отсутствии соответствующих служб качества разработчики ПО должны применять собственные нормативные и методические документы, регламентирующие процесс управления качеством ПО для всех категорий разработчиков и пользователей программной продукции.

10.2. Модели оценки надежности

Из всех областей программной инженерии *надежность* ПС является самой исследованной областью. Ей предшествовала разработка теории надежности технических средств, оказавшая влияние на развитие надежности ПС. Вопросами надежности ПС занимались разработчики ПС, пытаясь разными системными средствами обеспечить *надежность*, удовлетворяющую заказчика, а также теоретики, которые, изучая природу функционирования ПС, создали математические модели надежности, учитывающие разные аспекты работы ПС (возникновение ошибок, сбоев, отказов и др.) и позволяющие оценить реальную *надежность*. В результате *надежность* ПС сформировалась как самостоятельная теоретическая и прикладная наука [10.5–10.10, 10.16–10.24].

Надежность сложных ПС существенным образом отличается от надежности аппаратуры. Носители данных (файлы, *сервер* и т.п.) обладают высокой надежностью, записи на них могут храниться длительное время без разрушения, поскольку физическому разрушению они не подвергаются.

С точки зрения прикладной науки *надежность* – это способность ПС сохранять свои свойства (*безотказность*, *устойчивость* и др.), преобразовывать исходные данные в результаты в течение определенного промежутка времени при определенных условиях эксплуатации. Снижение надежности ПС происходит из-за ошибок в требованиях, проектировании и выполнении. Отказы и ошибки зависят от способа производства продукта и появляются в программах при их исполнении на некотором промежутке времени.

Для многих систем (программ и данных) *надежность* – главная целевая *функция* реализации. К некоторым типам систем (реального времени, радарные системы, системы безопасности, медицинскоеоборудование со встроенными программами и др.) предъявляются высокие требования к надежности, такие, как отсутствие ошибок, *достоверность*, *безопасность* и др.

Таким образом, оценка надежности ПС зависит от числа оставшихся и не устраненных ошибок в программах. В ходе эксплуатации ПС ошибки обнаруживаются и устраняются. Если при исправлении ошибок не вносятся новые или, *по* крайней мере, новых ошибок вносится меньше, чем устраняется, то в ходе эксплуатации *надежность* ПС непрерывно возрастает. Чем интенсивнее проводится *эксплуатация*, тем интенсивнее выявляются ошибки и быстрее растет *надежность* системы и соответственно ее качество.

Надежность является функцией от ошибок, оставшихся в ПС после ввода его в эксплуатацию. ПС без ошибок является абсолютно надежным. Но для больших программ абсолютная *надежность* практически недостижима. Оставшиеся необнаруженные ошибки проявляют себя время от времени при определенных условиях (например, при некоторой совокупности исходных данных) сопровождения и эксплуатации системы.

Для оценки надежности ПС используются такие статистические показатели, как *вероятность* и время *безотказной* работы, возможность отказа и частота (*интенсивность) отказов*. Поскольку в качестве причин отказов рассматриваются только ошибки в программе, которые не могут самоустраниться, то ПС следует относить к классу невосстанавливаемых систем.

При каждом проявлении новой ошибки, как правило, проводится ее *локализация* и исправление. Строго говоря, набранная до этого *статистика* об отказах теряет свое *значение*, так как после внесения изменений *программа*, *по* существу, является новой программой в отличие от той, которая до этого испытывалась.

В связи с исправлением ошибок в ПС *надежность*, т.е. ее отдельные атрибуты, будут все время изменяться, как правило, в сторону улучшения. Следовательно, их оценка будет носить временный и приближенный характер. Поэтому возникает необходимость в использовании новых свойств, адекватных реальному процессу измерения надежности, таких, как зависимость интенсивности обнаруженных ошибок от числа прогонов программы и зависимость отказов от времени функционирования ПС и т.п.

К факторам гарантии надежности относятся:

- риск как совокупность угроз, приводящих к неблагоприятным последствиям и ущербу системы или среды;

угроза как проявление неустойчивости, нарушающей безопасность системы;

анализ риска – изучение угрозы или риска, их частота и последствия;

целостность – способность системы сохранять устойчивость работы и не иметь риска;

Риск преобразует и уменьшает свойства надежности, так как обнаруженные ошибки могут привести к угрозе, если отказы носят частотный характер.

10.2.1. Основные понятия в проблематике надежности ПС

Формально модели оценки надежности ПС базируются на теории надежности и математическом аппарате с допущением некоторых ограничений, влияющих на эту оценку. Главным источником информации, используемой в моделях надежности, является процесс тестирования, эксплуатации ПС и разного вида ситуации, возникающие в них. Ситуации порождаются возникновением ошибок в ПС, требуют их устранения для продолжения тестирования.

Базовыми понятиями, которые используются в моделях надежности ПС, являются [10.5–10.10].

Отказ ПС (failure) – это переход ПС из работающего состояния в нерабочее или когда получаются результаты, которые не соответствуют заданным допустимым значениям. Отказ может быть вызван внешними факторами (изменениями элементов среды эксплуатации) и внутренними – дефектами в самой ПС.

Дефект (fault) в ПС – это последствие использования элемента программы, который может привести к некоторому событию, например, в результате неверной интерпретации этого элемента компьютером (как ошибка (fault) в программе) или человеком (ошибка (error) исполнителя). Дефект является следствием ошибок разработчика на любом из процессов разработки – в описании спецификаций требований, начальных или проектных спецификациях, эксплуатационной документации и т.п. Дефекты в программе, не выявленные в результате проверок, являются источником потенциальных ошибок и отказов ПС. Проявление дефекта в виде отказа зависит от того, какой путь будет выполнять специалист, чтобы найти ошибку в коде или во входных данных. Однако не каждый дефект ПС может вызвать отказ или может быть связан с дефектом в ПС или среды. Любой отказ может вызвать аномалию от проявления внешних ошибок и дефектов.

Ошибка (error) может быть следствием недостатка в одном из процессов разработки ПС, который приводит к неправильной интерпретации промежуточной информации, заданной разработчиком или при принятии им неверных решений.

Интенсивность отказов – это частота появления отказов или дефектов в ПС при ее тестировании или эксплуатации.

При выявлении отклонения результатов выполнения от ожидаемых во время тестирования или сопровождения осуществляется поиск, выяснение причин отклонений и исправление связанных с этим ошибок.

Модели оценки надежности ПС в качестве входных параметров используют сведения об ошибках, отказах, их интенсивности, собранных в процессе тестирования и эксплуатации.

10.2.2. Классификация моделей надежности

Как известно, на данный момент времени разработано большое количество моделей надежности ПС и их модификаций. Каждая из этих моделей определяет функцию надежности, которую можно вычислить при задании ей соответствующих данных, собранных во время функционирования ПС. Основными данными являются отказы и время. Другие дополнительные параметры связаны с типом ПС, условиями среды и данных.

Ввиду большого разнообразия моделей надежности разработано несколько подходов к классификации этих моделей. Такие подходы в целом основываются на истории ошибок в проверяемой и тестируемой ПС на этапах ЖЦ. Одной из классификаций моделей надежности ПО является классификация Хетча [10.10]. В ней предлагается разделение моделей на прогнозирующие, измерительные и оценочные (рис. 10.4).

Прогнозирующие модели надежности основаны на измерении технических характеристик создаваемой программы: длина, сложность, число циклов и степень их вложенности, количество ошибок на страницу операторов программы и др.

Например, модель Мотли–Брукса основывается на длине и сложности структуры программы (количество ветвей, циклов, вложенность циклов), количестве и типах переменных, а также интерфейсов. В этих моделях длина программы служит для прогнозирования количества ошибок, например, для 100 операторов программы можно смоделировать *интенсивность отказов*.

Рис. 10.4. Классификация моделей надежности

Модель Холстеда прогнозирует количество ошибок в программе в зависимости от ее объема и таких данных, как число операций (n_1) и операндов (n_2), а также их общее число (N_1, N_2).

Время программирования программы предлагается вычислять по следующей формуле:

$$T = \frac{n_1 N_2 (n_1 \log_2 n_1 + n_2 \log_2 n_2) \log_2 n_1}{2 n_2 S},$$

где S – число Страуда (Холстед принял равным 18 – число умственных операций в единицу времени).

Объем вычисляется по формуле:

$$V = (2 + n_2^*) \log_2 (2 + n_2^*),$$

где n_2^* – максимальное число различных операций.

Измерительные модели предназначены для измерения надежности программного обеспечения, работающего с заданной внешней средой. Они имеют следующие ограничения:

- программное обеспечение не модифицируется во время периода измерений свойств надежности;
- обнаруженные ошибки не исправляются;
- измерение надежности проводится для зафиксированной конфигурации программного обеспечения.

Типичным примером таких моделей являются модели Нельсона и РамамуртиБастани и др.Модель оценки надежности Нельсона основывается на выполнении k-прогонов программы при тестировании и позволяет определить надежность

$$R(k) = exp[-\sum \nabla t_j \lambda(t)],$$

где t_j – время выполнения j -прогона, $\lambda(t) = -[\ln(1 - q_i)]$ и при $q_i \leq 1$ она интерпретируется как *интенсивность отказов*.

В процессе испытаний программы на тестовых n_l прогонах оценка надежности вычисляется по формуле

$$R(l) = \frac{1 - n_l}{k},$$

где k – число прогонов программы.

Таким образом, данная модель рассматривает полученные количественные данные о проведенных прогонах.

Оценочные модели основываются на серии тестовых прогонов и проводятся на этапах тестирования ПС. В тестовой среде определяется *вероятность отказа* программы при ее выполнении или тестировании.

Эти типы моделей могут применяться на этапах ЖЦ. Кроме того, результаты прогнозирующих моделей могут использоваться как входные данные для оценочной модели. Имеются модели (например, модель Муссы), которые можно рассматривать как оценочную и в то же время как измерительную модель [10.16, 10.17].

Другой вид классификации моделей предложил Гоэл [10.18, 10.19], согласно которой модели надежности базируются на отказах и разбиваются на четыре класса моделей:

- без подсчета ошибок;
- с подсчетом отказов;
- с подсевом ошибок;
- модели с выбором областей входных значений.

Модели без подсчета ошибок основаны на измерении интервала времени между отказами и позволяют спрогнозировать количество ошибок, оставшихся в программе. После каждого отказа оценивается надежность и определяется среднее время до следующего отказа. К таким моделям относятся модели Джелински и Моранды, Шика Вулвертона и Литвуда-Верала [10.20, 10.21].

Модели с подсчетом отказов базируются на количестве ошибок, обнаруженных на заданных интервалах времени. Возникновение отказов в зависимости от времени является стохастическим процессом с непрерывной интенсивностью, а количество отказов является случайной величиной. Обнаруженные ошибки, как правило, устраняются и поэтому количество ошибок в единицу времени уменьшается. К этому классу моделей относятся модели Шумана, Шика- Вулвертона, Пуассоновская модель и др. [10.21-10.24].

Модели с подсевом ошибок основаны на количестве устраненных ошибок и подсеве, внесенном в программу искусственных ошибок, тип и количество которых заранее известны. Затем определяется соотношение числа оставшихся прогнозируемых ошибок к числу искусственных ошибок, которое сравнивается с соотношением числа обнаруженных действительных ошибок к числу обнаруженных искусственных ошибок. Результат сравнения используется для оценки надежности и качества программы. При внесении изменений в программу проводится повторное тестирование и оценка надежности. Этот подход к организации тестирования отличается громоздкостью и редко используется из-за дополнительного объема работ, связанных с подбором, выполнением и устранением искусственных ошибок.

Модели с выбором области входных значений основываются на генерации множества тестовых выборок из входного распределения, и оценка надежности проводится по полученным отказам на основе тестовых выборок из входной области. К этому типу моделей относится модель Нельсона и др.

Таким образом, классификация моделей роста надежности относительно процесса выявления отказов, фактически разделена на две группы:

- модели, которые рассматривают количество отказов как марковский процесс;
- модели, которые рассматривают *интенсивность отказов* как пуассоновский процесс.

Фактор распределения *интенсивности отказов* разделяет модели на экспоненциальные, логарифмические, геометрические, байесовские и др.

10.2.3. Марковские и пуассоновские модели надежности

Марковский процесс характеризуется дискретным временем и конечным множеством состояний. Временной параметр пробегает неотрицательные числовые значения, а процесс (цепочка) определяется набором *вероятностей перехода* $p_{ij}(n)$, т.е. вероятностью перейти на n -шаге из состояния i в состояние j. Процесс называется однородным, если он не зависит от n. В моделях, базирующихся на процессе Маркова, предполагается, что количество дефектов, обнаруженных в ПС, в любой момент времени зависит от поведения системы и представляется в виде стационарной цепи Маркова [10.5, 10.7, 10.10]. При этом количество дефектов конечное, но является неизвестной величиной, которая задается для модели в виде константы. *Интенсивность отказов* в ПС или скорость прохода по цепи зависит *лишь от количества дефектов*, которые остались в ПС. К этой группе моделей относятся: Джелински- Моранды [10.20], Шика-Вулвертона, Шантикумера [10.21] и др.

Ниже рассматриваются некоторые модели надежности, которые обеспечивают рост надежности ПО (модели роста надежности [10.7, 10.10]), находят широкое применение на этапе тестирования и описывают процесс обнаружения отказов при следующих предположениях:

- все ошибки в ПС не зависят друг от друга с точки зрения локализации отказов;
- *интенсивность отказов* пропорциональна текущему числу ошибок в ПС (убывает при тестировании программного обеспечения);
- вероятность локализации отказов остается постоянной;
- локализованные ошибки устраняются до того, как тестирование будет продолжено;
- при устранении ошибок новые ошибки не вносятся.

Приведем основные обозначения величин при описании моделей роста надежности:

- m - число обнаруженных отказов ПО за время тестирования;

X_i – интервалы времени между отказами $i-1$ и i, при $i=1,\ldots,m$;

S_i – моменты времени отказов (длительность тестирования до i -отказа), $S_i = X_k$ при $i=1,\ldots,m$;

T – продолжительность тестирования ПО (время, для которого определяется надежность);

N – оценка числа ошибок в ПО в начале тестирования;

M – оценка числа прогнозированных ошибок;

MT – оценка среднего времени до следующего отказа;

$E(T_p)$ – оценка среднего времени до завершения тестирования;

$Var(T_p)$ – оценка дисперсии;

$R(t)$ – функция надежности ПО;

$Z_i(t)$ – функция риска в момент времени t между $i-1$ и i -отказами;

c – коэффициент пропорциональности;

b – частота обнаружения ошибок.

Далее рассматриваются несколько моделей роста надежности, основанные на этих предположениях и использовании результатов тестирования программ в части отказов, времени между ними и др.

Модель Джелинского-Моранды. В этой модели используются исходные данные, приведенные выше, а также:

m – число обнаруженных отказов за время тестирования;

X_i – интервалы времени между отказами;

T – продолжительность тестирования.

Функция риска $Z_i(t)$ в момент времени t расположена между $i-1$ и i имеет вид:

$$Z_i(t) = c(N - n_{i-1}),$$

где $i=1,\ldots,m$; $T_{i-1} < t < T_i$.

Эта функция считается ступенчатой кусочнопостоянной функцией с постоянным коэффициентом пропорциональности и величиной ступени – c. Оценка параметров c и N производится с помощью системы уравнений:

$$\sum_{i=1}^{m} \frac{1}{N - N_{i-1}} - \sum_{i=1}^{m} cX_i = 0,$$
$$\frac{n}{c} - NT - \sum_{i=1}^{m} X_i n_{i-1} = 0$$

При этом суммарное время тестирования вычисляется так: $T = \sum_{i=1}^{m} X_i$

Выходные показатели для оценки надежности относительно указанного времени T включают:

число оставшихся ошибок $M_m = N - m$;

среднее время до текущего отказа $MT_m = 1/(N-m)c$;

среднее время до завершения тестирования и его дисперсию

$$E(T_p) = \sum_{i=1}^{N-n} \frac{1}{ic},$$

$$Var(T_p) = \sum_{i=1}^{N-n\}} \frac{1}{(ic)^2}.$$

При этом функция надежности вычисляется по формуле:

$$Rm(t) = exp(-(N-m)ct),$$

при $t > 0$ и числе ошибок, найденных и исправленных на каждом интервале тестирования, равным единице.

Модель Шика-Вулвертона. Модель используется тогда, когда *интенсивность отказов* пропорциональна не только текущему числу ошибок, но и времени, прошедшему с момента последнего отказа. Исходные данные для этой модели аналогичны выше рассмотренной модели Джелински-Моранды:

m - число обнаруженных отказов за время тестирования,

X_i - интервалы времени между отказами,

T - продолжительность тестирования.

Функции риска $Z_i(t)$ в момент времени между $i-1$ и $i-m$ отказами определяются следующим образом:

$$Z_i(t) = c(N - n_{i-1}), \text{где } i = 1, \ldots, m;\ T_{i-1} < t < T_i$$

$$T = \sum_{i=1}^{m} X_i$$

Эта функция является линейной внутри каждого интервала времени между отказами, возрастает с меньшим углом наклона. Оценка с и N вычисляется из системы уравнений:

К выходным *показателям надежности* относительно продолжительности T относятся:

число оставшихся ошибок $Mm = N - m$;

среднее время до следующего отказа MTт = (p / (2 (N - m) c))1/2;

среднее время до завершения тестирования и его дисперсия

$$E(T_p) = \sum_{i=1}^{N-m} \sqrt{\frac{\pi}{2ic}},$$

$$Var(T_p) = \sum_{i=1}^{N-m} \frac{2 - \pi/2}{ic}.$$

Функция надежности вычисляется по формуле:

$$R_T(t) = exp(-\frac{(N-m)ct^2}{2}), t \geq 0.$$

Модели пуассоновского типа базируются на выявлении отказов и моделируются неоднородным процессом, который задает $\{M(t), t \geq 0\}$ - неоднородный пуассоновский процесс с функцией интенсивности $\lambda(t)$, что соответствует общему количеству отказов ПС за время его использования t .

Модель Гоело-Окумото. В основе этой модели лежит описание процесса обнаружения ошибок с помощью неоднородного пуассоновского процесса, ее можно рассматривать как модель экспоненциального роста. В этой модели *интенсивность отказов* также зависит от времени. Кроме того, в ней количество выявленных ошибок трактуется как случайная величина, значение которой зависит от теста и других условных факторов.

Исходные данные этой модели:

m - число обнаруженных отказов за время тестирования;

X_i – интервалы времени между отказами;

T – продолжительность тестирования.

Функция среднего числа отказов, обнаруженных к моменту t, имеет вид

$$m(t) = N(1 - e^{-bt}),$$

где b – интенсивность обнаружения отказов и показатель роста надежности $q(t) = b$.

Функция интенсивности $\lambda(t)$ в зависимости от времени работы до отказа равна

$$\lambda(t) = Nb^{-b}, t \geq 0.$$

Оценка b и N получаются из решения уравнений:

$$m/N - 1 + \exp\{(-bT)\} = 0$$
$$m/b - \sum_{i=1}^{m}\{t_i\} - N_m \exp\{(-bT)\} = 0.$$

Выходные *показатели надежности* относительно времени T определяют:

1. среднее число ошибок, которые были обнаружены в интервале $[0, T]$, по формуле $E(N_T) = Nexp(-bT)$,
2. функцию надежности $R_T(t) = exp(N(e^{-bt} - e^{-bt(t+\mathrm{T})})), t \geq 0.$

В этой модели обнаружение ошибки трактуется как случайная величина, значение которой зависит от теста и операционной среды.

В других моделях количество обнаруженных ошибок рассматривается как константа.В моделях роста надежности исходной информацией для расчета надежности являются интервалы времени между отказами тестируемой программы, число отказов и время, для которого определяется надежность программы при отказе. На основании этой информации по моделям определяются *показатели надежности* вида:

вероятность *безотказной* работы;

среднее время до следующего отказа;

число необнаруженных отказов (ошибок);

среднее время дополнительного тестирования программы.

Модель анализа результатов *прогона тестов* использует в своих расчетах общее число экспериментов тестирования и число отказов. Эта модель определяет только вероятность *безотказной* работы программы и выбрана для случаев, когда предыдущие модели нельзя использовать (мало данных, некорректность вычислений). Формула определения вероятности *безотказной* работы по числу проведенных экспериментов имеет вид

$$P = 1 - Nex/N,$$

где Nex – число ошибочных экспериментов, N – число проведенных экспериментов для проверки работы ПС.

Таким образом, можно сделать вывод о том, что модели надежности ПС основаны на времени функционирования и/или количестве отказов (ошибок), полученных в программах в процессе их тестирования или эксплуатации. Модели надежности учитывают случайный марковский и пуассоновский характер соответственно процессов обнаружения ошибок в программах, а также характер и *интенсивность отказов*.

Контрольные вопросы и задания

1. Определите понятие качество ПО.
2. Назовите основные аспекты и уровни модели качества ПО.
3. Определите характеристики качества ПО и их назначение.
4. Какие методы используются при определении показателей качества?

5. Определите метрики программного продукта и их составляющие.
6. Какие стандарты в области качества ПО существуют?
7. Назовите основные цели и задачи системы управления качеством.
8. В чем суть инженерии качества?Назовите критерии классификации моделей надежности.
9. Дайте определение типов моделей надежности и их базис.
10. В чем отличие марковских и пуассоновских моделей надежности?
11. Сформулируйте основные параметры и предположения модели Джелинского.

Внимание! Если Вы увидите ошибку на нашем сайте, выделите её и нажмите Ctrl+Enter.