УДК 004.413.4

Сороков А.С.,
студент
3 курс, институт информационных технологий
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«МИРЭА — Российский технологический университет»
Россия, г. Москва

РАЗРАБОТКА МАССОВОГО ОТКРЫТОГО СОФТА И МОДЕЛЬ НАДЕЖНОСТИ ТИПА ДЖЕЛИНСКИ-МОРАНДА

Аннотация: *Статья посвящена проблеме системного анализа массового открытого софта, структуры программного обеспечения, закономерностей и моделированию, прогнозированию его надежности. Рассмотрены различные типы программных комплексов, системное толкование категории «надежность программы». Предложена модифицированная модель типа Джелински-Моранда, которая исследована. Результаты практически применимы при идентификации надежности программного обеспечения.*

Ключевые слова: *программное обеспечение, моделирование, надежность, Джелински-Моранда, системный.*

Abstract: *The article is devoted to the problem of system analysis of mass open software, software structure, laws and modeling, predicting its reliability. Various types of software systems, system interpretation of the category "program reliability" are considered. A modified model of the Jelinski-Moranda type, which has been investigated, is proposed. The results are practically applicable when identifying software reliability.*

Keywords: *program providing, modeling, reliability, Dzhelinski-Moranda, system.*

Введение

Планирование проекта – актуальная стадия реализации программного обеспечения (ПО), установления степени согласованности спецификаций с целями и ресурсами. Методы разработки ПО многообразны, но все должны соответствовать уровню автоматизации (интеллектуализации) и методике проектирования.

Методика, не опирающаяся на автоматизацию, в большей степени применяется при исследовании небольших и несложных по структуре программ. Трудоемкость такого ПО незначительна. Но сложные комплексы – другое дело. Нужны коллективы программистов, требования по данным, спецификациям, логике, отладке (верификации) и др. При несоблюдении ограничений уменьшается эффективность и качество продукта, возрастают трудовые затраты, повышается цена.

Автоматизированное планирование и проектирование появилось с потребностью сокращения расходов, уменьшения сроков, внедрения стандартных «заготовок» алгоритмов (проектов), неоднократно используемых с разными целями, а также координации работ и ресурсов в коллективе создателей. Автоматизация и интеллектуализация потребует технологического, программного и инфраструктурного «перевооружения». Это по силам лишь большим компаниям, которым доступны открытые коды к программам, их использование в организации удаленного доступа к информационным ресурсам.

Анализ надежности программ, комплексов также по силам коллективам разработчиков для больших задач, например, связанных с космосом [1, С.69]. Единства в понимании и использовании категории «надежность ПО» пока нет (например, [2, С.105]).

Можно определить структуру, закономерность деятельности составных частей ПО. Главный структурный акцент – моделирование обработки сведений, инфологическое и архитектурное структурирование, модификация состава функций и процедур.

В работе проделан системный анализ современного состояния различных типов программных комплексов, соотносительно с категорией «надежность программного обеспечения», эволюцией этой категории. Также предложена модифицированная модель типа Джелински-Моранда.

Модули и принципы структуризации ПО

Структура – связь модулей ПО, независимых его функциональных частей, обеспечивающих установленные требования к определенной внутренней структуре ПО. Структуризация проектов производится для анализа логики, программирования, верификации (тестирования) и модификации. Сложные (алгоритмически, инфологически) программные комплексы разрабатываются командой. Проектируется большой командой, реализуются небольшими программистскими группами (5-7 человек). Регулировать выполнение проекта можно, лишь соблюдая принципы структуризации:

1) разделять деятельность по исполнителям, гарантируя применимую для них загрузку и срок исполнения;
2) создавать календарные графики реализации проекта и вести координацию по ходу реализации продукта;
3) осуществлять стандартизацию, унификацию и контроль трудовых затрат;
4) структурное и инфологическое разбиение проекта на модули сопровождается подбором релевантного инструментария;
5) элементы ПО соединены информационно, интерфейсно, справочно, функционально и требованиями по качеству, стоимости и платформе (платформам).

Ведется независимая, сложная настройка модулей и всего комплекса, проверка работоспособности (устойчивости) модулей, программных средств, прототипирование (изготовление проверочного образца), на котором делается заключение об использовании ПО.

Проблемы массового открытого софта (Open-Sourse Soft)

В индустрии проектирования, разработки массового ПО выделим критические проблемы:

1) защита ПО (или интеллектуальной собственности);
2) развитие массово эксплуатируемого ПО (его сопровождение);
3) обеспечение переносимости (мультиплатформенности) ПО.

Решение проблем выполняется на основе «клиент-серверной» архитектуры (технологии), позволяющей разбить весь программный комплекс на части (модули):

1) «клиент» (клиентскую, отвечающую за отображение данных на рабочем месте пользователя);
2) «сервер» (серверную, отвечающую за все остальное, часто скрытое от пользователя и наиболее важное, например, веб-браузерные приложения).

Важные технологии (программы) переносят на сервер, у клиентских приложений прямого доступа к ним нет для безопасности, эффективности взаимодействий, но есть доступ к необходимым копиям ресурсов в любой момент работы у клиента.

Правом разработки модификации, дополнений к серверному ядру обладают административная или сторонняя группа разработчиков. Разработчик ПО концентрирует усилия лишь на самой платформе, сервисно-интерфейсные не волнуют его. Непрофильную деятельность часто отдают аутсорсингу, за счет экономии ресурсов (особенно, временных) создавая качественный профильный продукт, улучшая сервис и решения для него.

Сторонним разработчикам предоставляется необходимый инструментарий создания сервиса платформы. Работа осуществляется по

модели типа SaaS. Защищённый сервер всей платформы реализует валидацию клиентских расширений (плагинов), исключая риски несанкционированного доступа и уязвимости «от клиента».

Открытое ПО (ОПО, OSS) – ПО с доступным всем исходным кодом. Обычно размещается на официальном сайте или форуме разработчика и копируется многими другими впоследствии, согласно лицензии. ОПО доступно для изучения, анализа, модификации, что позволяет многим успешно (и не очень) изменять программы, модули ОПО, используя доступные коды, дорабатывая их или уже «исправленные», исправляя ошибки и уязвимости.

С помощью использования существующих исходных кодов (согласно лицензиям) или анализа примененных алгоритмов, структур и технологий. Анализ «исходников» – лучший способ разработки (модификации) подобных программ.

ОПО предполагает:

1) использование программы законным способом;
2) изучение ее работы, модификация «под себя»;
3) распространение копий программы, согласно лицензионным правам (не ущемляя прав третьих лиц);
4) улучшение программы с необходимыми ссылками на оригинал;
5) публикация внесенных улучшений в СМИ.

Есть проекты, закрытые по процессу разработки (непрозрачные), но с открытым кодом, например, Oracle VirtualBox. Закрытая функциональность может стать конкурентным преимуществом бизнес-компании, разрабатывающей (использующей) открытый проект.

Закрытые подсистемы (компоненты) могут оказаться системообразующими (эмерджентными), формирующими программный продукт. Даже при отсутствии технической поддержки. Например, библиотечная ассоциация США (American Library Association) считает, что не

стоит больше полагаться лишь на дорогое проприетарное ПО [3, с.2]. Об этом говорит и аналитика по библиотечным информационным системам [4, с.71].

В РФ правовое регулирование ОПО осуществляется ФЗ, подзаконными актами. Например, Распоряжением Правительства РФ 2007 от 18 октября, №1447-р и др. В ч.4 ГК РФ речь идет о правовом статусе «лицензия» (авторский, лицензионный договор). Такой договор – письменный (авторский договор бывает и в устной форме). В РФ использование ОПО регулируется офертой (публичным договором присоединения). Для нерезидентов применимо право местожительства/работы.

Парадигма «Открытый код» – способ мышления, взаимодействий отличающийся интеллектуальной свободой и реализуемый на основе принципов прозрачности, сотрудничества, вовлеченности, свободного обмена участников.

ОПО создается совместной и контролируемой массовой деятельностью, инновационными решениями. Закрытое ПО (CSS) не распространяется массово, не общедоступно, зашифровано. Исключительные права принадлежат разработчикам. Поэтому есть ограничения, например, на копирование, модификацию и распространение кода. Есть еще и свободное ОПО (FOSS), Фонд СПО (FSF) для защиты прав, инициатив (OSI). Существуют также разнообразные бесплатные лицензии ПО, допускающие использование, модификацию и продажу (GPL, LGPL и др.). Например, лицензия MIT, BSD (разрешена модификация), GNU, Apache (модификация, распространение), SUSE (распространение).

Свободное ПО не тождественно бесплатному ПО, которое относится к несвободному. Условно-бесплатное – это ПО, которое уже скомпилированное, предоставляемо бесплатно, но без «исходников», с возможными ограничениями режимов, функционала. Они снимаются после соответствующих выплат пользователем разработчику. Часто условно-

бесплатное ПО предоставляется лишь возмездным образом и без прав на модификацию [5, С.1].

Модель типа Джелински-Моранда

Многие существующие модели надежности – вероятностные, базируются на различных законах распределений, но они не всегда отражают реальности [6, С.98].

Модель Джелински–Моранды [7, С.157] предполагает экспоненциальное распределение отказов ПО и их нтенсивность, пропорционально количеству оставшихся ошибок.

Хотя данная модель – «древняя» и простейшая, она стала основой многих развитых моделей, используемых при управлении программными проектами [8, С.608].

Строится модель на гипотезах:

1) интенсивность *R(t)* выявления ошибок (уязвимостей) в момент времени $t \in [0; T]$ считается (и справедливо) пропорциональной текущему (не обнаруженному к моменту) *t* количеству ошибок;
2) между двумя последовательными моментами обнаружения ошибок *R(t)* не изменяется;
3) ошибки в программе распределены равновероятно и независимо, с равной степенью важности;
4) время между отказами (до следующего отказа) распределено по экспоненциальному закону;
5) исправление ошибок не привносит новых ошибок в программу.

Формализуя указанные предложения, можно записать функцию риска ошибок в виде:

$$R(t) = k(R_* - i + 1),$$

где k – идентифицируемый или экспертно задаваемый коэффициент масштабирования, R_* – идентифицируемое или задаваемое количество ошибок, оставшихся в начале отладки (тестирования, верификации), т.е. если

за t времени обнаружено $i-1$ ошибок, то в программе еще есть $R_* - (i-1)$ необнаруженных ошибок.

Согласно гипотезам, все

$$\Delta t_i = t_i - t_{i-1},\ i = 1,2,\dots,n-1,\ t_n = T,\ t_0 = 0,$$

распределены экспоненциально:

$$P(\Delta t_i) = e^{-k(R_* - (i-1))\Delta t_i},$$

а плотность вероятности отказа –

$$q(\Delta t_i) = k(R_* - i + 1)e^{-k(R_* - i + 1)\Delta t_i}.$$

Функция L правдоподобия (по гипотезам) представима в следующем виде [9, С.282]:

$$L(\Delta t_1, \Delta t_2, \dots, \Delta t_n) = \prod_{i=1}^{n} q(\Delta t_i).$$

Логарифмируя, получим:

$$\ln L = \sum_{i=1}^{n}(\ln(k(R_* - i + 1) - k(R_* - i + 1)\Delta t_i).$$

Условия экстремума L дают:

$$\begin{cases} \dfrac{\partial \ln L}{\partial R_*} = \displaystyle\sum_{i=1}^{n}\left(\frac{1}{R_* - i + 1} - k\Delta t_i\right) = 0\,, \\ \dfrac{\partial \ln L}{\partial k} = \displaystyle\sum_{i=1}^{n}(\frac{1}{k} - (R_* - i + 1)\Delta t_i) = 0 \end{cases}$$

Отсюда находим оценку максимального правдоподобия:

$$k = n/((R^* - 1)\sum_{i=1}^{n}\Delta t_i - \sum_{i=1}^{n} i\Delta t_i),$$

где R^* – оценка максимального правдоподобна для R_*, которая находится из соотношения:

$$\sum_{i=1}^{n}\frac{1}{R^* - i + 1} = (n - \sum_{i=1}^{n}\Delta t_i)/((R^* + 1)\sum_{i=1}^{n}\Delta t_i - \sum_{i=1}^{n} i\Delta t_i).$$

Решение уравнения – целочисленное (т.е. R^* – целое). Это и усложняет, и облегчает идентификацию.

Конечное решение R^* при $R^* \geq n$ существует лишь при условии выполнения неравенства:

$$\sum_{i=1}^{n}(i-1)\Delta t_i \,/ \sum_{i=1}^{n}(i-1) > (1/n)\sum_{i=1}^{n}\Delta t_i$$

Если $R^* < n$, то $R^* \to \infty$.

Последнее условие можно переписать:

$$\sum_{i=1}^{n} i\Delta t_i / \sum_{i=1}^{n}\Delta t_i > \frac{n+1}{2}.$$

Предложим модификации модели Джелински-Миранды, которые несложно идентифицируемы. Например, с использованием алгоритма, аналогичного методу наименьших квадратов.

1. Если допускать, что функция риска меняется между моментами $t_{i-1},\ t_i, i = 2, \dots, n$, а $N(t)$ – количество ошибок, обнаруженных к моменту t,

$$R(t) = k(R_* - N(t)),$$

то получим дифференциальное уравнение

$$\frac{\partial R(t)}{\partial t} + kR(t) = 0$$

с начальным условием

$$N(0) = 0\,, R(0) = kR_*$$

и решением

$$R(t) = kR_* e^{-kt}\ .$$

2. Если в промежутке $[t_{i-1}; t_i], i = 2, \dots, n$ меняется не только функция риска, но и параметр k, то уравнение примет вид:

$$\frac{\partial R(t)}{\partial t} = k'(t)\big(R_* - N(y)\big) - k(t)(R_* - N(t)).$$

3. Интересен случай, когда меняется и R_*. Тогда получаем:

$$\frac{\partial R}{\partial t} = k'(t)\big(R_*(t) - N(t)\big) - k(t)\left(\frac{\partial R_*(t)}{\partial t} - \frac{\partial N(t)}{\partial t}\right).$$

В России планируются сильные изменения в области ОПО. В частности, реализуется Стратегия развития ОПО до 2024 г. [10, С.1]. Все разработчики

смогут использовать ОПО, не нарушая лицензионное соглашения, права разработчика. Планируются льготы ИТ-компаниям, участвующим в финансировании такого ПО.

Open Source «погружается» в эволюционную стратегию ИТ-отрасли, создавая новую инфраструктуру и культуру разработки ПО.

Использованные источники:

1. Самарин Н.Н. Модель безопасного функционирования программного обеспечения, формализующая контроль использования памяти и обращений к ней процессора /Н.Н. Самарин // Наукоемкие технологии в космических исследованиях Земли. –2021. –Т.13. –№1. –С.68–79. doi: 10.36724/2409-5419-2021-13-1-68-79.
2. Смагин В.А., Дорохов А.Н. Основы теории надежности программного обеспечения. –СПб.: 2009. –303 с.
3. Brooke T. Open Source Integrated Library System Public Libraries / Tony Brooke // SJSY: School of Information. –2013. –Vol.3. –P. 1–21.
4. Изместьева О.В., Матусевич Д.С. Зарубежное свободное программное обеспечение автоматизированных библиотечно-информационных систем / О.В. Изместьева, Д.С. Матусевич. Научные и технические библиотеки. –2020. –№3. –С.69-78. https://doi.org/10.33186/1027-3689-2020-3-69-78.
5. Официальный сайт «Frequently Answered Questions. Open Source Initiative», URL: http://opensource.org/faq#free-software (дата обращения: 26.11.2021).
6. Лаврищева Е.М., Зеленов С.В., Пакулин Н.В. Методы оценки надежности программных и технических систем / Е.М. Лаврищева, С.В. Зеленов, Н.В. Пакулин // Труды ИСП РАН. –2019. –Т.31. –№5. –С.95-108. DOI: 10.15514/ISPRAS-2019-31(5)-7
7. Майерс Г. Надежность программного обеспечения / пер. с англ. Ю.Ю. Галимова. –М.: Мир, 1980. –356 с.

8. Фатрелл Р., Шафер Д., Шафер Л. Управление программными проектами. Достижение оптимального качества при минимуме затрат / Р.Т. Фатрелл, Д.Ф. Шафер, Л.И. Шафер. –М., Вильямс. –2003. –1125 с.
9. Магнус Я.Р. Эконометрика. Начальный курс: учебник / Я.Р. Магнус, П.К. Катышев, А.А. Пересецкий. –М.: Дело. –2004. –576 с.
10. Стратегия развития программного обеспечения с открытым кодом в России до 2024 года [Электронный ресурс]. URL: https://d-russia.ru/wp-content/uploads/2021/10/proekt_strategii_opo_30_09_21.pdf (дата обращения: 05.11.2021).