ISSN 2181-7200

ФАРҒОНА ПОЛИТЕХНИКА ИНСТИТУТИ

ИЛМИЙ-ТЕХНИКА ЖУРНАЛИ

НАУЧНО–ТЕХНИЧЕСКИЙ ЖУРНАЛ ФерПИ

SCIENTIFIC–TECHNICAL JOURNAL of FerPI

2016. спец.вып.

ISSN 2181-7200

ЎЗБЕКИСТОН РЕСПУБЛИКАСИ ОЛИЙ ВА ЎРТА МАХСУС ТАЪЛИМ ВАЗИРЛИГИ

ФАРҒОНА ПОЛИТЕХНИКА ИНСТИТУТИ

ИЛМИЙ–ТЕХНИКА ЖУРНАЛИ

2016 (спец. вып.)

НАУЧНО–ТЕХНИЧЕСКИЙ ЖУРНАЛ ФерПИ

SCIENTIFIC –TECHNICAL JOURNAL of FerPI

ФАРҒОНА – 2016

СОДЕРЖАНИЕ

ФУНДАМЕНТАЛЬНЫЕ НАУКИ

Якубов М.С., Умурзакова Д.М. Методы и алгоритмы распознавания и идентификации разно частотных речевых аудио сигналов 9

МЕХАНИКА

Утаев С.А. Исследование критерия оценки изменения эксплуатационных свойств моторных масел в цилиндро- поршневой группе газовых двигателей 14

Ахмедходжаев Х.Т., Азизов Ш.М. Статистический анализ производственных пильных джинов на выход средней длины волокна 18

Джураев А., Турдалиев В. Динамический анализ пятимассового машинного агрегата 22

СТРОИТЕЛЬСТВО

Мирзажонов М.А., Тулаганов А.А., Отакулов Б.А. Рабочие швы бетонных железобетонных конструкции и факторы влияющих на их прочность 25

ЭНЕРГЕТИКА, ЭЛЕКТРОТЕХНИКА, ЭЛЕКТРОННЫЕ ПРИБОРЫ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

Акбаров Д.Е., Хасанов Х.М., Умаров Ш.А. Приложение логических операций для решения некоторых задач средств обеспечение защиты информации 29

Горовик А.А., Лазарева М.В. Новый метод оценивания сложности и времени разработки программ основанный на генетическом подходе 33

Акбаров Д.Е., Нуритдинов Ж.Т., Дехконов Х.Т., Умаров Ш.А. Модели алгоритмов эффективного метода выполнения операции над действительными числами, представленных в двоичной системе исчислении 40

Мамасодикова Н.Ю., Порубай О.В., У.У. Искандаров, Исмоилов Д.И. Оптоэлектронный метод противодействия средствам несанкционированного съёма информации в волоконно-оптических системах связи 49

Косимов А.А., Пулатов Г.Г. Новый алгоритм оценивания настроек адаптивного регулятора 53

Сиддиков И.Х., Мамасодикова Н.Ю., Йигиталиев З.М. Алгоритмы управления децентрализованных взаимосвязанных динамических объектов в условиях информационных неопределенности 56

Жураев Н.М. Разработка и анализ новых линейно-электрических фильтров применяемых в опто-электрических сетях телекоммуникаций 61

Махсудов А.У., Умаралиев Н.У., Жураев Н.М., Атажанова Х.Р. Исследование аппаратных функций и шумов фотоумножителя для обнаружения сейсмической активности Земли 66

Акбаров Д.Е., Нуритдинов Ж.Т., Умаров Ш.А. Эффективные модели алгоритмов выполнения операции сложения и разности над блоками объедения битов 70

Сиддиков И.Х., Мамасодикова Н.Ю., Комилов А.О., Синтез нейронечеткого пид-регулятора для регулирования давления в трубопроводах 74

Жураев Н.М., Искандаров У. У. Обеспечение безопасности серверов radius с функцией GGSN в мобильной сети связи GSM 81

Кулдашов О.Х., Таджибаева Д.Р., Муминов К.З. Способ защиты информационного сигнала от несанкционированного доступа в волс 85

Позилова Ш.Х., Уринов Э.М. Проектирования автоматизированной диагностирующей компьютерной системы 89

Порубай О.В., Горовик А.А. Система распознавания лица человека на фото и видеоизображениях методом Виолы-Джонса 92

Эргашев С.Ф., Абдурахмонов С.М., Нигматов У.Ж. Автоматическая система управления процессами накапливания и получения горячей воды в солнечном водонагревателе 97

Сотгибоев Н.И., Нематова С.А. Применение вейвлетов Хаара для обработки многомерных сигналов 100

Аббасов Ё.С., Абдулхаев З.Э. Энергосберегающая система солнечного воздушного отопления 103

ХИМИЧЕСКАЯ ТЕХНОЛОГИЯ И ЭКОЛОГИЯ

Собиров М.М., Назирова Р.М., Таджиев С.М., Абдурахмонов С.Ж. Технология получения серосодержащих сложных удобрений 108

Хакимов Ш.З. Отзывчивости сортов озимой пшеницы на уровни минерального питания в условиях орошаемых светлых серозёмах Ферганской долины 113

СОЦИАЛЬНО - ЭКОНОМИЧЕСКИЕ НАУКИ

Хамидов В.С., Полвонов Ф.Ю. МООС: Новые модули повышения качества обученияДля высшего образования 118

тутилишсиз амалга оширилиши ахборот-коммуникация тармоқларида катта ҳажмдаги турли маълумотларни самарали алмашинувининг кафолатини таъминлайди.

Хулоса. Ушбу x ҳамда y ўзгарувчилари ва натижани ифодаловчи z нинг қийматлари “0” ва “1” иборат бўлиб, $x * y = z$ ифодага ҳар хил чинлик жадвалини мос қўювчи амалларнинг сони $2^4 = 16$ та эканлиги аниқланди. Ҳар хил чинлик жадвалларига шартли равишда амаллар ($*_i$, i=1,2,…,16) мос қўйилди. Бу амалларнинг чинлик жадвалидаги “0” ва “1” ларнинг тенг тақсимоти криптобардошли акслантиришлар алгоритмлари моделларини яратиш учун базис бўлиши асосланди. Шунингдек, “0” ва “1” ларнинг тенг тақсимотига эга бўлмаган чинлик жадвалли амаллар ҳам дастурий, аппарат-дастурий, аппарат қурилмаларининг алгоритм ва моделларини яратишда кенг тадбиқларга эгалиги таъкидланди. Олинган натижалар татбиқи ахборот муҳофазаси масалаларидан ташқари ахборот – коммуникацион технологияларининг турли соҳаларидаги масалалар ечимларига ҳам тегишлидир.

Адабиётлар

[1] Шалыто А.А. Логическое управление. Методы аппаратной и программной реализации. –Санкт-Петербург., 1999 г. -779 с.

[2] Предко М. «Руководство по микроконтроллерам». 1-2 Том. Москва: Постмаркет, 2001 г. -543 с.

[3] Бродин В.Б., Калинин А.В. «Системы на микроконтроллерах и БИС программируемой логики». Москва: «ЭКОМ» 2002 г. -655 с.

[4] Баранов В.Н. «Применение микроконтроллеров AVR: схемы, алгоритмы, программы». Москва, 2004 г .-288с.

[5] Фрунзе А.В. «Микроконтроллеры? Это же просто!» 1-3 Том. Москва: ООО «ИВ СКИМЕН», 2002 г.

[6] Молдовян Н. А., Молдовян А. А., Еремеев М. А. Криптография: от примитивов к синтезу алгоритмов. – СПб.: БХВ-Петербург, 2004 г. - 448 с.

[7] Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. –М.: издательство ТРИУМФ, 2003 г. - 816 с.

[8] Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушкин А.В. Основы криптографии: Учебное пособие, 2-е изд. –М.: Гелиос АРВ, 2002 г. -480 с.

[9] Акбаров Д.Е. Ахборот хавфсизлигини таъминлашнинг криптографик усуллари ва уларнинг қўлланилиши. – Тошкент, “Ўзбекистон маркаси”, 2009 й. – 434 б.

[10] Зензин О.С., Иванов М.А. Стандарт криптографической защиты – AES. Конечные поля /Под ред. М. А. Иванова – М.: КУДИЦ-ОБРАЗ, 2002 г. -176 с.

УДК 658.52

НОВЫЙ МЕТОД ОЦЕНИВАНИЯ СЛОЖНОСТИ И ВРЕМЕНИ РАЗРАБОТКИ ПРОГРАММ ОСНОВАННЫЙ НА ГЕНЕТИЧЕСКОМ ПОДХОДЕ

А.А. Горовик, М.В. Лазарева

Ферганский филиал Ташкентского университета информационных технологий

(Получена 22.11.2016 г.)

Бугунги кундаги меҳнат сарфини баҳоловчи усулларнинг асосий муаммоси шундаки уларни ҳар бир муаййан лойиҳага мослаштириш қийинлигидир. Мақолада тубдан янги, ноодатий, бўлган генетик ёндашувга асосланган усули келтирилган. Унинг асосий тамойиллари этиб дастурий таъминотни ишлаб чиқаришдаги меҳнат сарфини сонлив ва сифатли баҳоланишини таъминлаш белгиланган.

Таянч сўзлар: *дастурий таъминот, омехнат сарфини баҳолаш, COCOMO модели, PERT техникаси, хоссалар нуқталари, объуктлар нуқталари, ўхшашликлар бўйича баҳолаш, Паркинсон принципи, баҳолашнинг генетик алгоритми, тестлашнинг регрессион усуллари, LOC, дасту кодини хажмини баҳолаш.*

Основной проблемой современных методов оценки трудозатрат является сложность их адаптации к каждому конкретному проекту. В статье предложен принципиально новый, неклассический генетический подход к этой проблеме, главные принципы которого — обеспечение качественной и количественной оценки трудозатратности разработки программных проектов.

Ключевые слова: *программное обеспечение, оценка трудоемкости, модель COCOMO, техника*

PERT, точки свойств, объектные точки, оценка по аналогии, принцип Паркинсона, генетический алгоритм оценки, регрессионные методы тестирования, LOC, оценка объема программного кода

The main problem of modern methods of evaluation of labor is the difficulty of adapting them to each specific project. This paper proposes is a fundamentally new, non-classical genetic approach to the problem, the main principles of which is providing qualitative and quantitative assessment of labor-intensive software development projects.

***Keywords:** the software, a labour content estimation, model COCOMO, technics PERT, points of properties, objective points, an estimation on analogies, Parkinson's principle, genetic algorithm of an estimation, regressive testing methods, LOC, an estimation of the volume of a programm code.*

Точный расчет ресурсов, необходимых для реализации данного программного продукта с заданными требованиями к качеству, является одной из основных проблем в области управления проектами. Однако при таком расчете возникают сложности учета огромного количества факторов, влияющих на жизненный цикл ПО. Выяснить скрытые закономерности и связи между этими параметрами, а также произвести аналитический расчет в подавляющем большинстве случаев не представляется возможным. В любом программном проекте приходится балансировать между стоимостью, временем, качеством и объемом реализуемой функциональности. На сегодняшний день многие компании сталкиваются с серьезными проблемами в случае неправильных расчетов необходимых сроков:

•при недооценке — непредвиденная трата дополнительных средств, недовольство заказчика невыполнением обязательств в срок, «бессонные ночи» сотрудников, сложность управления «обвальным» проектом, низкое качество конечного продукта, слаборазвитые функции системы и т. д;

•при переоценке — бесполезный расход ресурсов, привлеченных к проекту, отказ заказчика от контракта с данными условиями (в результате возможна потеря рабочих мест) и т.д.

На современном рынке крупных программных систем потери могут исчисляться миллионами долларов.

Точные оценки издержек производства программного обеспечения важны как разработчикам, так и заказчику (клиенту). Они могут использоваться при переговорах о контракте, планировании, контроле и т. п. Таким образом, возникает реальная потребность в разработке методов и средств, позволяющих менеджеру оценить требуемые временные и человеческие ресурсы на основе всех имеющихся характеристик проекта: истории предыдущих подобных проектов, опыта и производительности сотрудников, специфики компании и т. п. Кроме того, требуется возможность перерасчета и уточнения сроков и ресурсов уже на этапе разработки системы с учетом текущих тенденций, наблюдаемых при реализации проекта. Это поможет менеджеру своевременно обнаружить отклонения от установленного графика и принять соответствующие меры в управлении проектом.

Краткий обзор основных понятий и проблем рассматриваемой области

Непосредственные усилия производителей (их трудозатраты) обеспечивают большую часть стоимости разработки ПО, и, как правило, методы оценки необходимых средств (как следствие, и времени) сосредотачиваются именно на этом аспекте и дают оценки в человеко-месяцах, которые впоследствии могут быть преобразованы в продолжительность проекта или стоимость.

На практике часто сталкиваются с тремя проблемами, имеющими принципиальное значение:

1) какую модель оценки выбрать?

2) какую метрику размера ПО использовать (число строк кода (LOC — Lines Of Code), «функциональные точки» (FP — Function Points) или «точки свойств» (feature points))?

3) что можно считать хорошей оценкой?

Выбор модели оценки

Широко применяемый на практике метод оценки трудозатрат — экспертная оценка. Однако такой подход сопряжен со множеством проблем:

−основания для получения оценки не являются явными;

−тяжело находить высококвалифицированных экспертов для каждого нового проекта;

−связь между размером системы и трудозатратами нелинейна. Трудозатраты возрастают экспоненциально с увеличением объема. Поэтому экспертная оценка получается адекватной только в случае, если текущий и предыдущие проекты примерно одинакового масштаба;

−политика руководства, направленная на сокращение затрат, как правило, ставит под сомнение реальный опыт предыдущих проектов и вносит долю «слепого оптимизма».

За три последних десятилетия было разработано множество различных моделей количественной оценки трудозатрат.

Они варьируются от моделей, основанных на эмпирических данных(например, модель «COCOMO» Б. Боема [1]) до чисто аналитических. Эмпирические модели используют данные предыдущих проектов, чтобы оценить текущий (анализируя закономерности, прослеживаемые в предыдущих проектах). С другой стороны, аналитические модели базируются на глобальных предположениях относительно связи различных параметров, таких как скорость устранения разработчиком дефектов и их количество в определенный момент времени. Каждая модель имеет свои достоинства и недостатки, но ключевым фактором при ее рассмотрении, конечно же, является точность.

Выбор метрики размера

Большинство моделей (как эмпирических, так и аналитических) базируются на использовании различных метрик размера разрабатываемой системы (LOC, FP и др.). Точность оценки трудозатрат напрямую зависит от точности оценки размера. Независимо от выбранной метрики размера определить его заранее точно не представляется возможным, поэтому приходится его уточнять (и соответственно производить общую переоценку трудозатрат) уже непосредственно в процессе разработки. Влияние этого фактора заметно уменьшается при использовании достаточно строгой методологии разработки и детальной проработке требований к системе. В силу особой важности данного вопроса для решения исходной задачи более подробно он будет рассмотрен в разделе «Сравнение метрик размера ПО».

Критерии хорошей оценки

По Ройсу [2], хорошая оценка издержек производства программного обеспечения должна быть:

−понятной и поддержанной менеджером проекта и командой разработчиков;

−одобреной всеми заинтересованными лицами как реально осуществимая;

−базирующейся на четкой модели с заслуживающими доверия основаниями, а также на данных подобного проекта (с подобными бизнес-процессами, технологиями, внешней средой, людьми и требованиями);

−определена настолько детально, что ключевые области риска понятны, и вероятность успеха объективно оценена.

Сравнение метрик размера ПО

Размер программного обеспечения — наиболее важный фактор, определяющий трудоемкость реализации ПО. Далее автор опишет пять метрик размера программного обеспечения, используемых на практике. Число строк исходного кода и функциональные точки — самые популярные метрики из пяти рассмотренных.

Число строк кода LOC (Lines Of Code) — число непустых строк исходного текста, исключая комментарии [4]. Несмотря на то, что эта метрика существенно зависит от выбранного языка

программирования, она до сих пор остается самой используемой метрикой размера ПО. Точное число LOC может быть получено только после того, как проект уже закончен. Поэтому оценка размера кода программы до его создания не намного проще оценки реальных трудозатрат, например, в человеко-месяцах.

Типичный метод осуществления такой оценки использует комбинацию экспертных оценок с техникой под названием PERT [6], заключающейся в следующем: пусть имеется n экспертов, каждый i-й эксперт высказывает три предположения относительно конечного размера: L_i — нижняя оценка размера; H_i- — верхняя оценка размера; M_i — наиболее вероятный размер. Тогда размер S можно вычислить как

$$S = \frac{1}{n}\sum_{i=1}^{n} \frac{L_1 + H_1 + 4M_1}{6}.$$

Точность оценки может быть существенно улучшена, если применить PERT не к проекту в целом, а к его отдельным компонентам. В этом случае общую оценку размера можно получить как сумму «локальных» оценок.

Метрики Холстеда

М. Холстедом были предложены такие метрики, как длина кода и объем [7]. Длина кода определяется как

$$N = N_1 + N_2, \quad (2)$$

где N_1 — общее количество появлений операторов в программе; N_2 — общее количество их операндов. Объем кода соответствует объему памяти, требуемой для хранения программы, и вычисляется по формуле:

$$V = N \log(n_1 + n_2), \quad (3)$$

где n_1 — число различных операторов; n_2 — число различных операндов, появляющихся в программе.

Очевидно, что оценить общее число операторов и их операндов до завершения проекта, как правило, еще более затруднительно, чем оценить LOC, поэтому в адрес предложенных Холстедом метрик было высказано множество замечаний [14, 15]. Поддержка такого подхода в последние годы постоянно уменьшается.

Функциональные точки (Function points).

Наиболее удачной заменой количеству строк кода для измерения размера стали функциональные точки *(function points),* впервые предложенные сотрудником IBM А. Альбрехтом в 1979 г. [8]. Применение функциональных точек основано на оценке объема реализуемой функциональности за счет изучения требований, вследствие чего оценка необходимых трудозатрат может быть выполнена на самых ранних стадиях работы над проектом и далее будет уточняться по ходу жизненного цикла, а явная связь между требованиями к создаваемой системе и получаемой оценкой позволяет заказчику понять, за что именно он платит, и во что выльется изменение первоначального задания.

Автор кратко рассмотрит основные принципы данного метода. Общее число функциональных точек программы зависит от количества элементарных процессов пяти типов (рис. 1):

1) входящие транзакции (External inputs (EI)) — получают данные от пользователя;
2) исходящие транзакции (External outputs (EO)) — передают данные пользователю;
3) взаимодействия с пользователем (External inquiries (EQ)) — интерактивные диалоги с пользователем (требующие от него каких-либо действий);
4) файлы внутренней логики (Internal logical files (ILF)) — файлы (логические группы информации), использующиеся во внутренних взаимодействиях системы;
5) файлы внешних взаимодействий (External interface filese (EIF)) — участвуют во

В данной терминологии транзакция — элементарный неделимый замкнутый процесс, имеющий значение для пользователя и переводящий продукт из одного консистентного состояния в другое.

Каждому из пяти типов назначают один из трех уровней сложности (1 = простой, 2 = средний, 3 = сложный), а каждой паре (тип, уровень сложности) ставят в соответствие вес, представляющий собой количество невыровненных функциональных точек (UFP), который изменяется от 3 (для простой входящей транзакции) до 15 (для сложных внутренних файлов). Общая оценка размера в UFP рассчитывается как (4):

$$UFP = \sum_{i=1}^{5} \sum_{j=1}^{3} N_{ij} W_{ij},$$

где $N_{ij} W_{ij}$ — соответственно число и вес элементов системы класса / со сложностью/.

Например, если в системе 2 простых входа (W_{ij} = 3), 2 сложных выхода (W_{ij} = 7) и 1 сложный внутренний файл (W_{ij} = 15), тогда UFP = 2x3 + 2x7 +1x15 = 35.

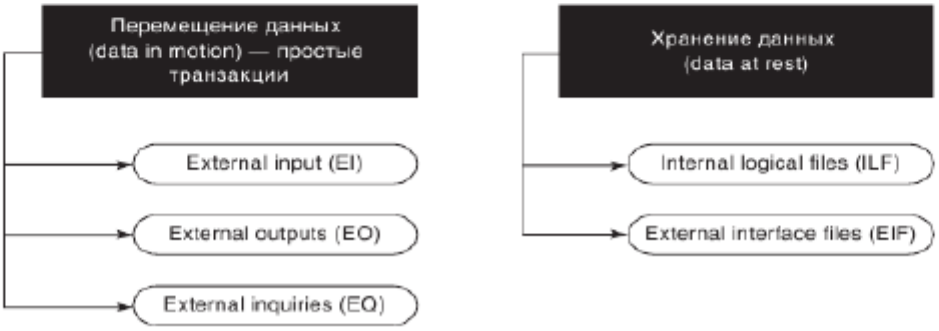

Рис 1. Типы элементарных процессов, используемых в методе FP.

Это число функциональных точек может непосредственно использовать для оценки стоимости/трудоемкости и уточняться с помощью фактора выравнивания (VAF), который вычисляется на основании характеристик общей сложности проекта, таких как: степень распределенности обработки и хранения данных, требования к производительности системы, требования к безопасности и т.д. При этом окончательная оценка размера в выровненных функциональных точках рассчитывается как

$$AFP = (UFP + CFP) \cdot VAF, (5),$$

где CFP — дополнительные функциональные точки, которые потребуются, например, для установки или миграции данных. Общая схема процедуры оценки представлена на рис. 2.

Рис 2. Процедура оценки функционального размера по методу FP.

Число UFP и число строк кода (LOC) связаны линейно:

$$LOC = a \; x \; UFP + b. \; (6)$$

Параметры a и b могут быть получены с помощью линейной регрессии на основании имеющихся данных о завершенных проектах.

1. **Базовая модель COCOMO.** Размер проекта S измеряется в LOC (KLOC), а

трудозатраты — в человеко-месяцах. Создана на основе анализа статистических данных 63 проектов (главным образом, Министерства обороны США) различных типов. Используются три набора параметров {a, to} в зависимости от сложности разрабатываемого программного обеспечения:

a) для простых, легко понимаемых проектов, $a = 2{,}4$, $b = 1{,}05$;
b) для сложных систем, a = 3,0, $b = = 1{,}15$;
c) для встроенных систем, a = 3,6, $b = 1{,}20$.

Модель была проста в применении, но не обеспечивала должной точности.

2. **Детализированная модель COCOMO.** Уточнены наборы параметров {a, to}, кроме того, общая формула приняла вид: $E = M \cdot a \cdot S^b$, где M — уточняющий коэффициент, рассчитывающийся как произведение 15 поправочных факторов из 4-х категорий (факторы конечного продукта, вычислительной среды, персонала, проекта), варьирующихся в диапазоне от 0,7 до 1,66, которые могут быть найдены в специальной таблице. Эти изменения базовой модели позволили существенно улучшить точность оценки, особенно в случае применения метода к отдельным компонентам, а не к системе в целом.

3. **COCOMO II.** (1997 г.). Имеет много общего со своей предшественницей, однако, во многом основана на новых идеях, а также адаптирована к современным методологиям разработки ПО (в частности, если COCOMO подразумевала только каскадную модель жизненного цикла, то COCOMO II пригодна для спиральной и итеративной моделей). При построении COCOMO II для обработки статистических данных использовался байесовский анализ, который дает лучшие результаты для программных проектов, характеризующихся неполнотой и неоднозначностью, в отличие от многофакторного регрессионного, примененного в COCOMO. Также в ней допускается измерять размер проекта не только числом строк кода (LOC), но и более современными функциональными и объектными точками. Помимо прочего, при расчете показателей COCOMO II учитывает уровень зрелости процесса разработки в соответствии с моделями SEI CMM/ CMMI [15].

Основная идея генетического подхода

Главной проблемой современных методов оценки трудозатрат является сложность их адаптации к конкретному проекту (калибровки коэффициентов модели). Каждый проект — это своя команда, свои условия труда, специфика разработки. Обобщенные модели, такие как COCOMO, разработанные с учетом данных о большом количестве проектов, не могут полностью учесть этой специфики.

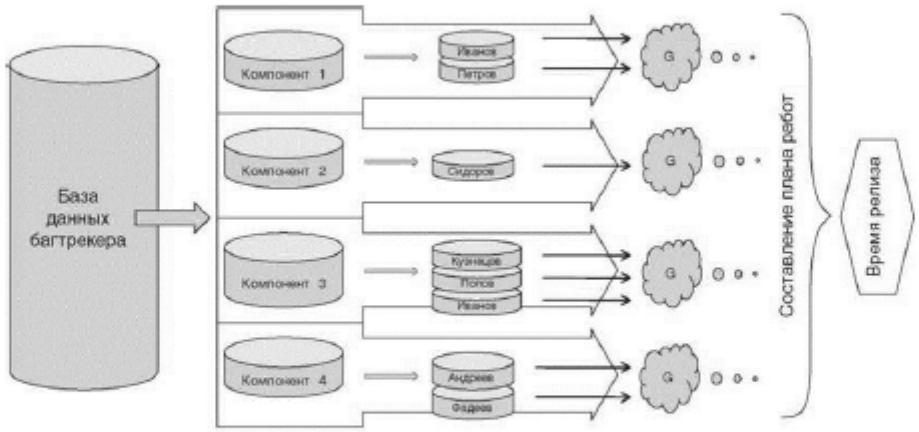

Рис 3. Общая схема методики предсказания метода релиза с применением данного метода.

При решении фиксированной части можно абстрагироваться от многих факторов, определяющих проектную специфику, и использовать какой-либо современный метод оценки трудозатрат так, словно он применяется к какому-нибудь типичному

«среднестатистическому» проекту (например, COCOMO). Это позволяет не учитывать множество дополнительных факторов и существенно упрощает реализацию метода. Главная же инновационная и не имеющая аналогов составляющая предлагаемого подхода заключается в способе решения динамической части задачи. *Динамическая часть* — оценка ресурсов, необходимых для устранения дефектов и других проблем, возникающих в процессе разработки системы, не укладывающихся в имеющиеся модели. Было замечено, что эти дефекты живут и развиваются подобно биологической популяции, подверженной определенным воздействиям окружающей среды — множества сотрудников со своими показателями производительности. Эта среда и порождает такие дефекты, стремится их устранить впоследствии.

Итак, выявлены три принципиально новые идеи, предлагаемые в подходе:

1) разделение исходной задачи на фиксированную и динамическую составляющие;
2) интерпретация множества дефектов как хромосомной популяции;
3) использование имеющейся в багтре- кере (системе контроля дефектов) информации для уточнения оценки, полученной с помощью какого-либо метода на ранних стадиях жизненного цикла.

Кроме того, этот подход позволяет провести все необходимые расчеты любому менеджеру, даже не являющемуся экспертом в области предсказания сроков, так как база данных может быть проанализирована автоматически.

Условия применимости

Несмотря на достаточную универсальность метода, он все же требует определенных условий, при которых его применение оправдывается в полной мере.

1. Наличие багтрекера, включающего обязательные поля:
   a. Время сообщения об ошибке.
   b. Время установления статуса «принято к исполнению».
   c. Время установления статуса «исправлено».
   d. Промежуточная версия компонента.
   e. Сотрудник, ответственный за исправление.
2. Наличие в распоряжении какого-либо средства для вычисления фиксированной составляющей (например, пакета на основе COCOMO II).
3. Базы данных ошибок от предыдущих проектов (чем больше, тем лучше), в которых участвовали те же разработчики, что и в текущем проекте — для расчета сроков на самых начальных его этапах. В случае отсутствия таких баз данных для этого коллектива можно воспользоваться базами данных другого коллектива, работавшего над проектами, подобными текущему, но эффективность метода заметно ухудшается.

Полученные результаты

Описанные идеи позволили разработать прототип системы уточнения оценок трудозатрат, основанных на модели COCOMO II. По базам данных багтрекеров трех opensource-проектов (Firefox, Fedora, KDE) произведены оценки, которые в среднем оказались на 4%

Проект	Имитируемая дата осуществления предсказания	Фактическая дата релиза	COCOMO II	COCOMO II + Genetic	Увеличение точности, %
Firefox 49.0.2	15.08.2016	20.10.2016	24.10.2016	26.10.2016	2,7
Fedora 25	10.08.2016	11.10.2016	3.10.2016	14.10.2016	8,1
KDE 5.7.2	4.06.2016	19.07.2016	28.07.2016	29.07.2016	1,3

точнее, чем оценки, полученные с помощью COCOMO II, без уточнения посредством рассмотренного метода (табл. 1). В дальнейшем планируется дополнить разрабатываемую систему упрощенными версиями существующих методов, что сделает возможным ее

использование как полноценного самостоятельного программного пакета. Увеличение точности оценки для некоторых opensource-проектов

Перспективы развития предложенных идей

На сегодняшний день существует достаточное количество развитых средств анализа репозиториев проекта (например, SolidSTA), однако, они не осуществляют никаких предсказаний касательно реальных сроков, как правило, приспособлены только для использования менеджерами или аналитиками и, по сути, обеспечивают лишь сбор статистики и визуализацию, не производя подробного анализа, так как при этом возникают сложности учета огромного количества факторов, колеблющихся от проекта к проекту.

Целью дальнейших исследований и разработок является создание расширения для IDE, которое должно удовлетворять следующим требованиям:

−предоставить возможность контроля производительности и каждого отдельного разработчика, и всей команды в целом;

−на основании анализа проектных репозиториев сигнализировать об отклонениях от запланированного графика и в режиме one-click производить перерасчет сроков сдачи проекта (и/или отдельного разрабатываемого модуля/сборки) в эксплуатацию на текущий момент;

−обладать достаточной степенью универсальности (не должно ориентироваться на какую-либо конкретную модель разработки, язык программирования и т. п.);

−должно быть рассчитано не только на менеджеров проекта, но и на рядовых разработчиков, а также всех, кто заинтересован в получении представления о текущих темпах разработки и укпадки в запланированный график (для самоконтроля, проведения различных форм аналитики и др.);

−предоставить возможность наглядной визуализации результатов анализа.

Список литературы

[1] Boehm B. Software Cost Estimation with Cocomo II. NewJersey, Prentice-Hall, 1981.
[2] Royce W. Software project management: a unified framework. Reading, Addison-Wesley Professional, 1998.
[3] Boehm B. Software engineering economics. New Jersey, Prentice-Hall, 1981.
[4] Fenton N. Software Metrics: A Rigorous and Practical Approach, London, Chapman and Hall, 1991.
[5] Parkinson G. Parkinson's Law and Other Studies in Administration NY, Houghton-Miffin, 1962.
[6] Aron J. Estimating Resource for Large Programming Systems Rome, NATO Science Committee, 1969.
[7] Halstead M. Elements of software science, NY, Elsevier, 1977.
[8] Albrecht J., Gaffney J. E. Software function, source lines of codes, and development effort prediction: a software science validation, IEEE Trans Software Eng. SE-9, 1983. P. 639-648.
[9] Jones C. Applied Software Measurement, Assuring Productivityand Quality, NY, McGraw-Hill, 1977.
[10] St-PierreD., Maya M., Abran A., Desharnais J. and Bourque P. Full Function Points: Counting Practice Manual, Technical Report 1997-04, University of Quebec at Montreal, 1997.

УДК 681.3

ИККИЛИК САНОҚ СИСТЕМАСИДА ИФОДАЛАНГАН ҲАҚИҚИЙ СОНЛАР УСТИДА АМАЛЛАР БАЖАРИШНИНГ САМАРАЛИ УСУЛЛАРИ АЛГОРИТМЛАРИ МОДЕЛЛАРИ

Д.Е. Акбаров, Ж.Т. Нуритдинов, Х.Т. Деҳқонов, Ш.А. Умаров[1],

Қўқон давлат педагогика институти, sht00357@gmail.com , bardosh9295@mail.ru , usha003@inbox.uz
[1]Тошкент ахборот технологиялари университети Фарғона филиали, sht003@umail.uz

Мақолада иккилик саноқ системасида ифодаланган ҳақиқий сонлар устида амаллар бажаришнинг самарали усуллари алгоритмлари моделлари келтирилди. Амалларнинг битлар устида бажарилиши таклиф этилган алгоритм ва моделлар етарли катта сонлар устида тез ҳамда хотирадан кам фойдаланган ҳолда амаллар бажариш имкониятини беради. Бундан ташқари