УДК004.422

С.Н.Баранов, А.М.Тележкин

МЕТРИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ПРОГРАММНЫХ РАЗРАБОТОК

Баранов С.Н., Тележкин А.М. **Метрическое обеспечение программных разработок.**
Аннотация. В статье систематизируются и сводятся воедино свыше 110 различных процессных, проектных и продуктовых метрик, установившихся в практике промышленной разработки программных продуктов, из которых в каждом отдельном программном проекте используется только их небольшая часть. Выбор, сбор, накопление и регулярный анализ этих метрических данных позволяет создавать качественное программное обеспечение в заданных ограничениях и определять направления для дальнейшего совершенствования процесса разработки в организации и проектной группе, принимая решения на основе фактов, а не предположений.
Ключевые слова: Технология программирования, проектные метрики, процесс разработки ПО, модель CMMI.

Baranov S.N., Telezhkin A.M. **Metrics for Software Development.**
Abstract. Over 110 different process, project, and product metrics used in the current practice of software development are assembled and classified in this paper, of which only a small number is used in each particular project. Selecting, collecting, accumulating and regular analysis of these metric data allows for creation of high-quality software under the given constraints and for identifying ways to further improve the software development process in the given organization or project team through decision making based on facts rather than assumptions or opinions.
Keywords: software engineering, project metrics, software development process, CMMI.

1. Введение. Создание высококачественного программного обеспечения (ПО) в современную эпоху информационных технологий давно стало одним из важных и существенных высокотехнологичных производств в развитых и развивающихся странах. Огромная сложность ПО как технического объекта, по сравнению с механизмами и техническими средствами предшествующих технологических эпох, является следствием огромного количества состояний, в которых может находиться программная система (под состоянием такой системы обычно понимается совокупность единовременных значений всех ее переменных, изменяющихся с течением времени).

Любое конкретное поведение программной системы рассматривается как некоторый путь в дискретном пространстве состояний; перебрать все такие пути при тестировании невозможно практически. Например, если в программе имеется всего N независимых 16-разрядных переменных, то нижняя оценка для общего числа S ее состояний выражается равенством $S=2^{16\times N}$, а число путей длины L в этом пространстве из какого-либо начального состояния задается «экспоненциальной башней» высоты L из значений S, что обозначается ме-

тафорой «экспоненциальный взрыв» (exponential explosion).

Вместе с тем, неправильное поведение программы на каком-либо конкретном пути из множества всех допустимых путей (которое является подмножеством всех возможных путей в данном пространстве состояний), как правило, приводит к неправильному срабатыванию того устройства, в котором она установлена и может привести даже к катастрофическим последствиям.

Дисциплина программирования, сложившаяся в индустрии программного обеспечения за последние 40 лет, создала ряд формализованных моделей для процесса разработки высококачественных программных продуктов (см. рисунок 1), из которых в настоящее время наиболее полной является модель [1], разработанная Институтом технологии программирования при университете Карнеги-Меллон.

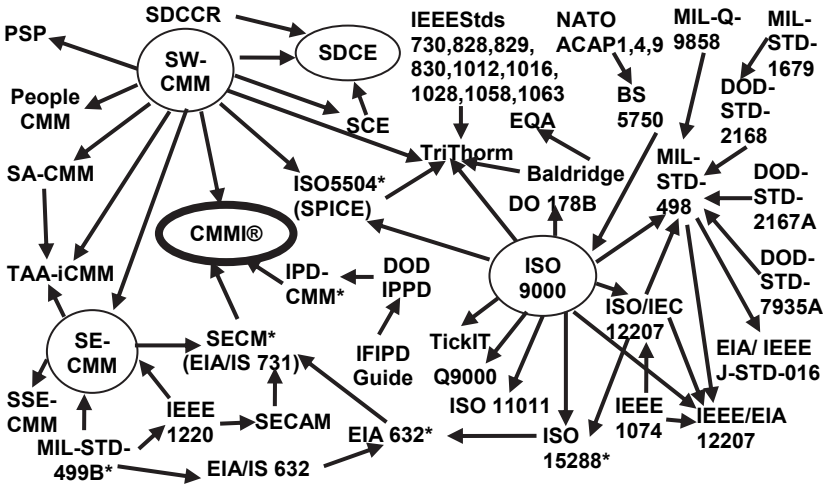

Рис. 1. Взаимосвязанные модели процесса разработки

Метрическая программа этой модели предполагает постоянный сбор и анализ различных метрик, характеризующих текущее состояние разрабатываемого продукта, ход процесса разработки и достигнутый уровень зрелости организации-разработчика. Выбор, сбор, накопление и регулярный анализ этих метрических данных позволяет создавать качественное программное обеспечение в заданных ограничениях и определять направления для дальнейшего совершенствования процесса разработки в организации и проектной группе, принимая решения на основе фактов, а не предположений. Вместе с тем, модель CMMI,

как и многие другие, не определяет ни конкретный состав этих метрик, ни способы их сбора, хранения и анализа. В данной работе систематизируется накопленный опыт по метрическому обеспечению программных проектов для его дальнейшего применения в практических разработках. Наряду с общемировым опытом, использован опыт авторов в научных исследованиях и в промышленных разработках в компаниях Моторола [2-3], Эксиджен [4] и других.

2. Классификация метрик. На сегодняшний день известно свыше 500 различных измеряемых показателей, так или иначе относящихся к разработке программных продуктов. Разумеется, в каждом отдельном проекте используется только небольшая часть этого множества. В данной работе собраны и приведены в систему 112 различных метрик, некоторые из которых, в свою очередь, обеспечивают целых куст отдельных метрических показателей (всего свыше 230). Выбор состава метрик, способов их сбора и анализа определяется целями и задачами проекта и рядом других сопутствующих условий, которые к тому же могут меняться в процессе работы. Огромную роль в метрической программе играет наличие достоверной исторической базы данных (ИБД) проектов, как уже выполненных, так и находящихся в процессе разработки в данной организации, что позволяет проводить объективный анализ этих данных и делать оценки для текущих и будущих разработок применительно к данным условиям. Создание и поддержание такой ИБД является важной задачей, а сама база – существенным информационным активом организации-разработчика.

По своему назначению метрики можно разделить на три основных класса: продуктовые, проектные и процессные; причем, различаются метрики базовые – основные, ни от чего не зависящие, общие для многих разработок, производные – выводимые из базовых и представляющие дополнительную информацию, и специфические – индивидуальные для каждого отдельного продукта, проекта или процесса. Общими для всех метрик характеристиками являются единица измерения, способ измерения и фаза (фазы) жизненного цикла, на которой эта метрика может быть измерена. Значения многих метрик изменяются в процессе разработки, так что отслеживание их динамики является важной задачей для оперативного управления проектом.

По ИБД для некоторых часто используемых метрик определяются их референсные значения: среднее, лучшее и, возможно, наихудшее по организации-разработчику, а также аналогичные данные по промышленности на определенный срок. Далее эти три типа метрик рассматриваются более подробно. Для каждой метрики указывается ее название с англоязычным эквивалентом, определение ее сущности,

если она не следует из ее названия, и единица измерения, если это не просто число или процент. Наряду с метриками, продукты, проекты и процессы характеризуются еще и атрибутами (свойствами), как правило, в описательном виде.

2.1. Продуктовые метрики характеризуют непосредственно измеряемые свойства самого программного продукта (изделия).

1. Исходные требования – Source requirements. Число исходных требований, общее и по типам (функциональные, интерфейсные, по производительности, системные и т.д.).
2. Размер онтологии – Ontology size. Число разных онтологических сущностей, задействованных в описании исходных требований.
3. Изменчивость требований – Volatility of requirements. Число изменений, добавлений и удалений в формулировках исходных требований, совершенных по инициативе заказчика, абсолютное и в отношении к общему числу требований.
4. Полнота требований – Completeness of requirements. Число требований, добавленных по инициативе разработчиков, абсолютное и в отношении к общему числу требований.
5. Противоречивость требований – Inconsistency of requirements. Число изменений и удалений требований, совершенных по инициативе разработчиков, абсолютное и в отношении к общему числу.
6. Завершенность требований – Finalization of requirements. Число незакрытых вопросов по уточнению исходных требований, общее и по типам требований.
7. Системные компоненты – System components. Число и наименования различных системных компонентов в продукте. Определяется по высокоуровневому проекту и структуре разбиения работ.
8. Программные модули – Software modules. Число и наименования различных программных модулей в продукте. Определяется по компонентам высокоуровневого проекта и структуре разбиения работ.
9. Используемые интерфейсы – Interfaces used. Число и наименования различных интерфейсов, используемых в продукте. Определяется по высокоуровневому проекту и структуре разбиения работ.
10. Входные и выходные каналы – I/O channels. Число и наименования различных входных и выходных каналов в продукте. Соотносится со сложностью по Холстеду [5] высокоуровневого проекта.
11. Используемые языки программирования – Programming languages used. Число и названия языков программирования.
12. Используемые базы данных – Databases used.
13. Используемые серверы приложений – Application servers used. Число и названия таких серверов.

14. Используемые операционные системы – Operating systems used. Число и названия таких операционных систем.

15. Используемые технологии – Technologies used.

16. Размер кода – Code size. Число строк кода на входном языке программирования, не считая пустые строки и строки, состоящие только из комментариев; единица измерения – KLOC или KAELOC.

17. Размер кода для модульного тестирования – Code size for unit testing. Число строк кода на входном языке программирования для модульного тестирования; единица измерения – KLOC или KAELOC.

18. Ветвления в коде – Code branches. Число всех операторов ветвления и цикла в исходном коде.

19. Параллелизм кода – Code parallelism. Число параллельно исполняемых ветвей в исходном коде.

20. Сложность кода по Холстеду – Halstead complexity. Четыре основных показателя: число различных/всех (η_1/N_1) операторов и число различных/всех (η_2/N_2) операндов в программе и пять производных, определяемых через них: N – длина программы, V – объем программы, L* – оценка длины реализации программы, λ – уровень языка программирования, I – интеллектуальное содержание программы [5].

21. Цикломатическая сложность кода (сложность по Мак-Кейбу) – Cyclomatic complexity (McCabe complexity). Цикломатическое число $\lambda(G) = m - n + 2 \times p$, где m – число вершин, n – число дуг и p – число компонентов связности в графе G передач управления в программе [6].

22. Качество кода – Code quality. Оценка числа остаточных дефектов в коде (как известных, так и еще не выявленных), абсолютное и в отношении к общему размеру кода; единица измерения – число/KLOC или число/KAELOC, либо в виде $N\sigma$.

23. Повторное использование кода – Code reuse. Процент повторно использованного кода в общем коде поставки.

24. Размер кода для повторного использования – Code for reuse. Число строк вновь разрабатываемого кода с дополнительной целью его дальнейшего повторного использования; единица измерения – KLOC (KAELOC).

25. Покрытие тестами требований – Requirement coverage with tests. Процент числа требований, покрытых тестами, от общего числа требований, суммарный и по типам требований.

26. Пост-релизные дефекты – Post-release faults. Число дефектов, обнаруженных в продукте после поставки заказчику в течение определенного срока (обычно 6 или 12 месяцев), абсолютное и в отношении к размеру всего поставляемого кода; единица измерения – число/KLOC

или число/KAELOC, либо в виде Nσ.

27. Покрытие тестами кода – Code coverage with tests. Процент кода, покрытого тестами, от всего кода.

28. Покрытие тестами ветвлений в коде – Branch coverage with tests. Процент числа ветвлений, покрытых тестами, от числа ветвлений во всем поставляемом коде.

29. Размер документации – Documentation size. Число страниц в разработанной документации, поставляемой заказчику.

30. Повторное использование документации – Documentation reuse. Процент повторно использованной документации в общем объеме документации, поставляемой заказчику (по числу страниц).

31. Использованные патенты – Patents used. Число и наименования запатентованных решений, использованных в данном продукте.

32. Инновационность продукта – Product innovativeness. Число признанных инновационных решений, использованных в данном продукте, сделанных участниками проекта и реализованных в нем.

33. Пакет поставки – Delivery package. Метрические данные по видам составляющих компонентов пакета поставки, включая исходных код, выполняемый код, документацию, отчеты о дефектах, материалы тестирования, спецификации, планы, процедуры, технологии оценивания, предложения по инновациям и т.д.

34. Обратная связь по продукту – Product feedback. Число предложений, отзывов, рекламаций и т.д., полученных в течение определенного периода (обычно 6 или 12 месяцев) после поставки продукта.

2.2. Проектные метрики измеряемым образом характеризуют сам проект по созданию и сопровождению программного продукта.

1. Трудоемкость – Effort. Суммарные трудозатраты, общие и по фазам проекта, по видам работ и категориям исполнителей, по плану и по факту на данный момент; единица измерения – число человеко-месяцев, человеко-дней или человеко-часов.

2. Производительность труда – Performance. Отношение размера разработанного в данном проекте кода в законченном продукте к общей трудоемкости на его создание, плановое и по факту; единица измерения – KLOC (KAELOC) на человеко-день (человеко-месяц).

3. Длительность проекта – Project duration. Число календарных дней или месяцев, отведенных на разработку, плановое и по факту.

4. Экономия от автоматизации разработки кода – Cost saving due to code development automation. Экономия затрат на разработку за счет автоматизации разработки кода, плановая и по факту; единица измерения – руб.

5. Автоматизация разработки кода – Code development automa-

tion. Процент автоматически созданного кода в общем размере кода, плановый и по факту.

6. Автоматизация разработки тестового кода – Test development automation. Процент автоматически созданного тестового кода в общем размере тестового кода (код тестов плюс код тестового окружения), плановый и по факту.

7. Автоматизация прогона тестов – Test run automation. Процент тестов, исполняемых без участия тестировщика, в общем числе тестов в эталонном тестовом наборе, плановый и по факту.

8. Экономия от автоматизации тестирования – Cost saving due to test automation. Экономия затрат на тестирование за счет автоматизации разработки и прогона тестов, плановая и по факту; единица измерения – руб.

9. Методы и инструменты – Methods and tools. Число и наименования различных методов и инструментов, используемых в разработке, включая лицензии на ПО.

10. Стоимость проекта – Project cost. Суммарная текущая стоимость проекта в денежном выражении, плановая и по факту, общая и по видам затрат, которые включают затраты на заработную плату, оборудование, ПО, специальные компоненты, обучение и переподготовку, книги и журналы, командировки, аренду, услуги сторонних организаций, расходы на юридическое сопровождение, накладные и представительские расходы, расходы на обеспечение секретности, стоимость исправления дефектов и т.д.; единица измерения – руб.

11. Стоимость разработчика – Cost of software developer. Стоимость программной разработки в суммарных затратах на одного участника в месяц, плановая и по факту, включая расходы на оборудование и накладные расходы; единица измерения – руб./месяц.

12. Предел стоимости – NTE (Not to Exceed). Суммарная величина всех затрат на проект, которую нельзя превысить ни при каких обстоятельствах; превышение означает немедленное прекращение проекта; единица измерения – руб.

13. Стоимость строки кода – Cost of a line of code. Отношение суммарных затрат на проект к размеру кода в LOC или AELOC в законченном программном продукте; единица измерения – руб.

14. Частота совершения ошибок – Error rate. Число совершаемых ошибок на единицу размера кода и документации; единица измерения – число/KLOC (KAELOC) для кода и число/стр. для документации.

15. Стоимость обеспечения качества – Cost of quality. Процент суммарных трудозатрат на деятельности по обеспечению качества

продукта, включая обзоры, тестирование, повышение квалификации и т.п., в суммарных трудозатратах на проект, плановый и по факту.

16. Стоимость переделок – Cost of poor quality. Процент суммарных трудозатрат на нахождение, исправление и повторную проверку исправления дефектов с учетом трудозатрат на регрессионное тестирование, вызванное исправлениями, в суммарных трудозатратах на проект, плановый и по факту.

17. Разработчики – Developers. Число лиц, создающих код продукта и сопроводительную документацию, плановое и по факту.

18. Распределенность разработки – Distributed (multi-site) development. Число отдельных групп (часто разнесенных по разным площадкам), работающих над проектом, если оно больше единицы.

19. Тестировщики – Testers. Число инженеров, тестирующих продукт, плановое и по факту.

20. Штат проекта – Staffing. Число ставок, включая руководство, разработчиков, тестировщиков и поддержку, плановое и по факту.

21. Женщины – Women. Число женщин, участвующих в проекте, абсолютное и как % от общего числа участников проекта.

22. Запланированные поставки – Planned deliverables. Число и наименования поставок заказчику, включенных в план разработки.

23. Точность поставок – On-time delivery. Процент поставок заказчику, совершенных вовремя, в общем числе всех совершенных поставок, плановый и по факту.

24. Точность планирования – Schedule accuracy. Процент отклонений фактических показателей хода проекта от запланированных в графике проекта в общем числе запланированных показателей.

25. Точность оценивания – Estimation accuracy. Процент отклонений фактических значений параметров проекта от их первоначальных оценок, по каждому параметру в отдельности и усредненный по всем учтенным параметрам.

26. Завершенность разработки – Development completion. Процент объема выполненных работ (по числу завершенных работ в структуре разбиения работ, по трудоемкости, по затратам и т.д.) от запланированного на данный момент, плановый и по факту.

27. Завершенность тестирования – Testing completion. Процент выполненных работ по тестированию (по числу созданных тестов, совершенных тестовых прогонов, успешно прошедших тестов и т.д.) от запланированного на данный момент, плановый и по факту.

28. Перепланирования проекта – Project replannings. Число утвержденных изменений плана в процессе работы, плановое и по факту.

29. Отчеты об ошибках – Defect reports. Число полученных отче-

тов об ошибках за данный период времени.

30. Плотность выявленных дефектов – Defect density. Отношение числа выявленных дефектов в коде и документе к его размеру; единица измерения – число/KLOC или число/KAELOC для кода и число/стр. для документов, либо в виде Nσ.

31. Эффективность сдерживания дефектов – Defect containment effectiveness. Процент числа дефектов, выявленных и устраненных до сдачи продукта, от сделанной оценки общего числа дефектов.

32. Эффективность сдерживания серьезных (уровень 3 и выше по 5-бальной шкале) дефектов – MFC (Major Fault Containment). Процент серьезных ошибок, не перешедших в дефекты, от общего числа всех выявленных серьезных дефектов, плановое и по факту.

33. Эффективность сдерживания дефектов по фазам проекта – Defect containment effectiveness on project phases. Процент дефектов, выявленных и устраненных на данной фазе, и относящихся к ней, от оценки общего числа дефектов, относящихся к данной фазе.

34. Удовлетворенность заказчика – Customer satisfaction. Число баллов (обычно по 10-бальной шкале, где 10 – высшая оценка), поставленных заказчиком при анкетировании, плановое и по факту.

35. Размер тестового набора – Test suite size. Число тестов в эталонном тестовом наборе для проверки готовности продукта к выпуску.

36. Процент прохождения тестов – Tests passed. Процент прошедших тестов в общем числе тестов в эталонном тестовом наборе, плановый и по факту.

37. Циклы тестирования – Test cycles. Число полных циклов системного тестирования, совершенных до выпуска продукта, плановое и по факту.

38. Опыт совместной работы данного коллектива – Project team continuity. Среднее время работы одного человека в данном коллективе; единица измерения – число месяцев или лет.

39. Результативность (эффективность) взаимодействия внутри коллектива – Effectiveness of team interaction. Процент числа проблем, разрешенных внутри коллектива, в общем числе проблем, поднятых в ходе исполнения проекта.

40. Рациональность (экономичность) взаимодействия внутри коллектива – Efficiency of team interaction. Средние затраты на разрешение одной проблемы внутри коллектива разработчиков (число взаимодействий, длительность процесса разрешения, понесенные трудозатраты и т.д.); единица измерения – в зависимости от способа.

41. Опыт работы с данной платформой – Platform experience. Срок работы участников проекта с данной платформой, суммарный по

всем разработчикам и тестировщикам, и в пересчете на 1 человека; единица измерения – число месяцев или лет.

42. Опыт работы в данной предметной области – Subject domain experience. Срок работы участников проекта в данной предметной области, суммарный по всем разработчикам и тестировщикам, и в пересчете на 1 человека; единица измерения – число месяцев или лет.

43. Опыт работы в данной методологии и с данными инструментальными средствами – Methodandtoolsexperience. Срок работы участников проекта в данной методологии и с данными инструментальными средствами, суммарный по всем разработчикам и тестировщикам, и в пересчете на 1 человека; единица измерения – число месяцев или лет.

44. Время жизни дефекта – Defect longevity. Среднее и максимальное время от момента обнаружения дефекта до сообщения о его закрытии по всем выявленным и закрытым дефектам, плановое и по факту; единица измерения – число дней.

45. Процент незакрытых дефектов – Still open defects. Процент числа еще не закрытых дефектов в общем числе обнаруженных дефектов на данный момент времени.

46. Результативность (эффективность) обзоров – Review effectiveness. Процент числа дефектов, выявленных на обзорах, в общем числе выявленных дефектов на данный момент времени.

47. Рациональность (экономичность) обзоров – Review efficiency. Отношение суммарных трудозатрат на подготовку и проведение обзоров к общему числу найденных на обзорах дефектов – т.е., средняя стоимость нахождения одного дефекта путем обзора; единица измерения – человеко-час.

48. Инновационность проекта – Project innovativeness. Число рацпредложений и заявок на изобретение, поступивших от участников проекта по его тематике в ходе выполнения проекта и в течение некоторого срока (обычно до 6 месяцев) после его завершения.

49. Публикационность проекта – Project publicity. Число публикаций, включая защиты диссертаций, сделанных участниками проекта по его тематике в ходе выполнения проекта и в течение некоторого срока (обычно до 2 лет) после его завершения. Учитываются как открытые публикации, так и публикации для служебного пользования.

50. Загруженность оборудования – Equipment usage (load). Процент времени использования оборудования в суммарном времени, в течение которого оно находится в распоряжении участников проекта (по видам оборудования и лицензиям на ПО), плановый и по факту.

51. Использование уникального оборудования – Unique equipment usage. Суммарное время, потраченное на настройку и подготовку

уникального оборудования организации или заказчика и прогон на нем создаваемого продукта, плановое и по факту; единица измерения – число человеко-месяцев, человеко-дней или человеко-часов.

52. Проектные риски – Project risks. Общее число различных проектных рисков, включенных в план управления рисками.

53. Сработавшие риски – Triggered risks. Число актуализировавшихся проектных рисков, абсолютное и в отношении к общему числу проектных рисков, включенных в план управления рисками.

54. Переоценка рисков – Risk reassessment. Число пересмотров плана управления рисками (сколько раз производился пересмотр рисков по их перечню и характеристикам), плановое и по факту.

55. Смягчение последствий рисков – Risk mitigation. Трудозатраты на предотвращение и смягчение последствий рисков, плановые и по факту.

56. Воздействие рисков – Risk impact. Убытки в денежном выражении, которые проект может понести при срабатывании всех рисков и каждого в отдельности, плановые и по факту.

57. Декомпозиция проекта на продукты – Project decomposition to products. Число самостоятельных продуктов, появляющихся в результате данного проекта, если оно больше единицы.

58. Декомпозиция проекта на функциональные подгруппы – Project decomposition to functional subgroups. Число функциональных групп, на которое в итоге был разбит проект.

59. Компьютеры – Computers. Число компьютеров (настольных и др., включая тестовые и сервера), выделенных для данной группы.

60. Достаточность ресурсов – Sufficiency of resources. Процент необходимых по плану ресурсов в фактическом их объеме по видам ресурсов и усредненный по всем видам ресурсов.

61. Постоянство разработчиков – Staff continuity. Процент лиц, работавших от начала до конца проекта, в общем числе разработчиков.

62. Интенсивность взаимодействия внутри организации – Intensity of interaction within the organization. Число человеко-часов потраченных на общение внутри организации. Подсчитывается взаимодействие между разработчиками, между разработчиками и тестировщиками, между руководством проекта и командой.

63. Интенсивность взаимодействия с заказчиком – Intensity of interaction with customer. Число человеко-часов потраченных на общение с заказчиком.

64. Результативность (эффективность) тестирования – Effectiveness of testing. Число дефектов, выявленных путем тестирования, абсолютное и в отношении к общему числу выявленных дефектов.

65. Рациональность (экономичность) тестирования – Efficiency of testing. Трудозатраты на нахождение дефектов путем тестирования, абсолютные и в отношении к общему числу выявленных дефектов.

66. Метрики – Metrics. Число и наименования используемых в проекте метрик.

2.3. Процессные метрики характеризуют применяемый в создании продукта технологический процесс разработки и, соответственно, уровень зрелости организации, в которой этот процесс установлен.

1. Уровень зрелости разработчиков – Maturity level. Определяется путем официального оценивания или самооценивания; единица измерения – номер уровня по модели CMMI.

2. Повышение квалификации – Training. Среднее число часов для одного сотрудника на обучение и переподготовку в течение года, плановое и фактическое.

3. Выполненные разработки (проекты) – Completed projects. Число и наименования разработок (проектов), выполненных в организации за последние несколько лет. Определяется по ИБД организации.

4. База заказчиков – Customer base. Число и наименования разных заказчиков, для которых выполнялись разработки за последние несколько лет. Определяется по ИБД организации.

5. Портфель патентов – Patent portfolio. Число и наименования патентов, полученных данной организацией за пять несколько лет.

6. Текучесть кадров – Attrition rate. Процент сотрудников, покинувших организацию, в общем числе сотрудников в расчете на год.

7. Нарушения процесса – Process violations. Число зафиксированных нарушений процесса разработки за период по их категориям.

8. Улучшения процесса – Process improvements. Число поданных предложений по улучшению процесса за период.

9. Выработка ресурса – Burn-down chart. Число человеко-часов, остающихся на выполнение данной задачи; в методике Scrum наглядно отображает "сгорание" выделенного ресурса на выполнение задач по всему проекту в целом и по каждому спринту в отдельности.

10. Процессные активы – Project assets. Число шаблонов документов, моделей жизненного цикла, программных и тестовых наработок и других процессных активов, накопленных в организации.

11. Используемость процессных активов – Project assets usage. Процент повторно использованных процессных активов из базы процессных активов организации в общем числе процессных активов по отдельным активам и их группам.

12. Опытность команды – Staff experience. Процент ведущих опытных разработчиков в общем числе разработчиков.

2.4. Референсные значения для некоторых метрик по промышленности США за 2000 г. (см. [7]) приведены в таблице 1. Поскольку данные по отечественной промышленности программного обеспечения малодоступны, то организации-разработчики постепенно накапливают собственные данные по уже выполненным проектам и в дальнейшем используют их для сравнения и расчетов.

Таблица 1.Эталонные данные по промышленности США

Метрика	Единица измерения	Среднее	Лучшее
Производительность труда	KAELOC на человеко-месяц	3.23	7.14
Стоимость строки кода	Доллары США на KAELOC	4,334	1,962
Плотность дефектов	Число дефектов на KAELOC	15.6	8.1
Эффективность сдерживания дефектов	%	95.0	99.5
Пост-релизные дефекты	Число дефектов на KAELOC	0.780	0.041

В таблице 2 приведены усредненные данные отечественного предприятия Эксиджен [4], накапливавшиеся в течение 2008-2010 гг., по 69 выполненным проектам общим объемом 3 309 378 строк кода (LOC) на языке Си, что составляет 8 273 445 эквивалентных строк на языке ассемблера (AELOC) — см. коэффициенты пересчета в таблице 4.

Таблица 2. Пример метрических данных из базы данных Эксиджен

Метрика	мин.	макс.	среднее
Размер проекта (KAELOC)	3 578	1 535 498	135 630,2
Число найденных дефектов	9	4 484	362,0
Число разработанных тестов	4	10 890	987,7
% покрытия кода при модульном тестировании	11	93	62,4
Цикломатическая сложность кода	2	50	17,4

В модели SEER-SEM [8] интересны данные по типичному распределению трудоемкости в жизненном цикле разработки, которые можно использовать как начальное приближение при планировании проекта, а при планировании инспекций (обзоров) кода и документов можно опираться на следующие данные (таблица 3), пока в организации не накоплены собственные результаты по этим метрикам.

Таблица 3. Оптимальный темп инспекций

Шаг инспекции	Программный код	Текст
Запуск	15 мин	30 мин
Обзор	500 строк кода в час	560 строк текста в час
Подготовка	100 строк кода в час	140 строк текста в час
Инспекционное совещание	125 строк кода в час	140 строк текста в час
Выводы	30 мин	45 мин

Если разные компоненты программного продукта создаются на разных языках программирования, то размер этого продукта определяется в KAELOC – K Assembler Equivalent Lines Of Code, а для пересчета KLOC в KAELOC используются переходные коэффициенты, эмпирически установленные для разных языков программирования (таблица 4). С помощью этих же коэффициентов можно сравнивать размеры программных продуктов, написанных на разных языках.

Таблица 4. Пересчет KLOC в KAELOC для некоторых языков

Язык программирования	Коэффициент пересчета	Язык программирования	Коэффициент пересчета
Ada	4,5	LISP	1,5
Assembler	1,0	Macro-Assembler	1,0
C	2,5	Pascal	3,5
C++	11,0	Query languages	25,0
Forth	5,0	Unix shell scripts	1,5
FORTRAN	3,0	4-th Generation Languages	16,0

При определении качества программы (оценки числа остаточных дефектов) часто используется подход Nσ (уровни сигма), где σ – среднеквадратическое отклонение при нормальном распределении вероятности наличия причины дефекта (ошибки) в исходном коде, а N – уровень сигма. Принятый в промышленности его пересчет в число дефектов (ошибок) в строках исходного кода приведен в таблице 5.

Таблица 5. Уровни сигма и оценка числа остаточных дефектов

Уровень сигма	Число дефектов (ошибок) на 1 миллион строк исходного кода	Уровень сигма	Число дефектов (ошибок) на 1 миллион строк исходного кода
1	691 462	4	6 210
2	308 538	5	233
3	66 807	6	3,4

В настоящее время в индустрии программного обеспечения уровень шесть сигма принят за эталон качества для надежного ПО.

3. Сбор и хранение метрических данных. Сбор и хранение метрических данных в организации обычно осуществляется группой обеспечения качества (SQA – Software Quality Assurance), которая ведет (курирует) каждый проект в организации и таким образом участвует в разработке. Собираемые для анализа метрические данные обычно хранятся в специальной базе данных, которая является частью процессных активов организации.

Член группы обеспечения качества, приписанный к данному проекту, ведет регулярный сбор метрических данных и проводит их анализ с помощью специальных инструментов. Результаты анализа сообщаются руководителю проекта и руководству организации, а также всем другим заинтересованным лицам. Задача группы обеспечения качества – отслеживать соответствие всех технологических процессов в разработке ПО принятым моделям и стандартам и заданным плановым показателям. При выявлении отклонений хода проекта от плана немедленно возбуждается анализ причин такого отклонения, и вырабатываются соответствующие поправочные действия.

4. Представление и анализ метрик. Собираемые метрики о ходе проекта регулярно анализируются его участниками для принятия решений о необходимости тех или иных поправочных действий для успешного завершения проекта. Регулярность такого анализа определяется выбранной моделью жизненного цикла и технологии разработки. Например, в технологии Scrum такой анализ выполняется ежедневно в начале каждого рабочего дня, тогда как в модели CMMI обычно приняты еженедельные отчеты всех участников разработки и ежемесячные операционные обзоры с участием руководства организации.

В конце разработки обычно проводится ретроспективный анализ всего хода работ, и делаются выводы на будущее по дальнейшим изменениям и усовершенствованиям типового процесса разработки в данной организации в соответствии с ее целями. В [10] приведены примеры ежемесячных одностраничных отчетов о ходе проекта, которые генерируются автоматически средствами MS Excel по данным из ИБД проекта на запрошенную дату. На таких ежемесячных операционных обзорах также сообщаются и анализируются и другие метрические показатели хода проекта: S-кривые по размеру кода и объему созданной документации, количеству найденных и закрытых дефектов, трудоемкости по разным типам работ и другие, определенные в проектном плане. Эти диаграммы, показывающие динамику изменения соответствующего показателя с течением времени, также строятся по табличным данным, которые допускают автоматическую обработку и более развитыми инструментами, например, такими как MathLab и др.

5. Планирование проекта на основе метрик. План разработки – это проигрывание будущих работ и инструмент для определения роли каждого участника. Он увязывает отдельные части разработки вместе, служит точкой отсчета для изменений и помогает понять, когда цель проекта достигнута, и проект может быть закончен. Этот план строится на основании структуры разбиения работ и оценок трудозатрат на исполнения выявленных блоков работы, которые устанавливаются по аналогичным проектам, выполненным ранее.

Общая схема предварительного планирования программного проекта представлена на рисунке 2. Она детализирует переход от исходных начальных требований к будущему программному продукту, полученных от заказчика, к начальной версии плана управления проектом в виде дорожной карты из ряда последовательных шагов.

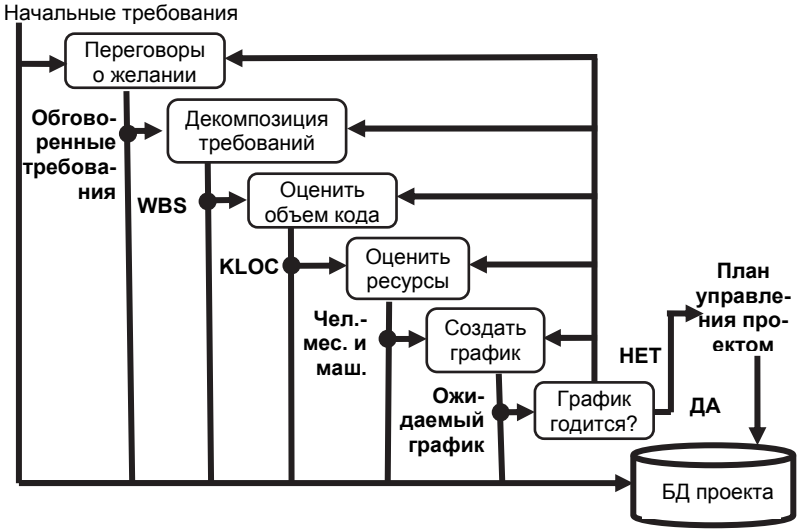

Рис. 2 Дорожная карта для планирования проекта

Исходя из начальных требований строится структура разбиения работ (Work Breakdown Structure – WBS). На следующем шаге эти требования уточняются и разносятся по подсистемам будущего продукта. Таким образом, на выходе этого шага появляется WBS, которая служит основой для дальнейшего планирования. По структуре разбиения работ и количеству требований, подлежащих реализации в каждом компоненте будущего продукта оценивается объем кода, подлежащий разработке в тысячах строк программного текста KLOC или в их ассемблерном эквиваленте KAELOC. Здесь существенным является при-

влечение уже известных данных по выполненным проектом для обоснования этих оценок. Затем, исходя из полученных оценок и трудоемкости разработки тысячи строк кода (с поправками на сложность, опыт исполнителей и другие факторы), оцениваются необходимые трудозатраты в человеко-месяцах и для некоторых проектов – необходимые дополнительные ресурсы по машинному времени и уникальному оборудованию. На основании этих данных строится ожидаемый график исполнения работ, учитывающих полученные оценки трудозатрат и взаимосвязи между компонентами в структуре разбиения работ. Заключительным шагом является принятие решения о том, устраивает ли полученный график работ как заказчика, так и исполнителя, и если это не так, то происходит возврат к одному из предыдущих шагов для внесения необходимых поправок.

Все получаемые исходные данные и промежуточные оценки вместе с их обоснованием накапливаются в базе данных проекта для последующего хранения и анализа при принятии решений.

Из целого ряда инструментов для составления предварительного плана работ в промышленности чаще всего используется модель издержки затрат COCOMO – COnstructive COst MOdel, предлагающая три формулы для расчета важнейших плановых показателей:

$$Трудоемкость = a_b \times (KLOC)^{b_b} [человеко - месяцев]$$

$$Срок\_разработки = c_b \times (Трудоемкость)^{d_b} [месяцев]$$

$$Число\_разработчиков = \frac{Трудоемкомкость}{Срок\_разработки} [человек]$$

где коэффициенты a_b, b_b, c_b и d_b, разные для разных типов проектов и определенные усреднением по огромной и представительной выборке конкретных программных проектов, выполненных в разное время разными разработчиками, представлены в таблице 6:

Таблица 6. Коэффициенты базовой модели COCOMO

Тип проекта	a_b	b_b	c_b	d_b
Органический	2,4	1,05	2,5	0,38
Полуразделенный	3,0	1,12	2,5	0,35
Встроенный	4,6	1,20	2,5	0,32

Для органического (organic) типа характерна небольшая команда разработчиков с достаточным опытом разработки, и нежесткими требованиями к создаваемому продукту. Полуразделенный (semi-detached) тип предполагает команду среднего размера со смешанным

опытом работы и различными по степени жесткости требованиями к продукту. Отличительной особенностью встроенного (embedded) типа является большое число жестких требований.

Таким образом, входные параметры для получения оценок трудоемкости, срока разработки и числа разработчиков в этой модели – это тип проекта и объем кода, который надо создать. Накапливая данные по фактическим трудозатратам в данной организации, можно совершенствовать эту модель, уточняя значения ее коэффициентов.

В начале 2000-х готов была предложена более развитая модель COCOMO II [9], со значительно большим числом характеристик программного проекта и в силу этого дающая более точные оценки. Новой группой входных данных для оценки трудоемкости, по сравнению с базовой моделью COCOMO, являются так называемые масштабируемые показатели программного обеспечения, подлежащего разработке. В так называемой промежуточной модели (intermediate model) COCOMO II их значения выбираются по 6-ти бальной шкале: очень низкий, низкий, номинальный, высокий, очень высокий и сверхвысокий. В значении «номинальный» все эти коэффициенты равны 1 и отклоняются в ту или иную сторону для понижающих или повышающих оценок. В виде числовых множителей эти значения входят в формулу для трудоемкости как сомножители поправочного коэффициента EAF (effort adjustment factor), который изменяется в диапазоне от 0,9 до 1,4:

$$Трудоемкость = a_i \times (KLOC)^{b_i} \times EAF[человеко-месяцев]$$

где коэффициенты a_i и b_i те же, что и a_b и b_b, в базовой модели, за исключением органического: a_i = 3,2 и встроенного: a_i = 2,8 проектов. Формулы для оценки срока разработки и числа разработчиков и соответствующие коэффициенты c_i и d_i в промежуточной модели те же, что и c_b и d_b в базовой модели. В продвинутой модели (advanced model) COCOMO II, дополнительно учитываются следующие показатели:

– *Атрибуты продукта – Product attributes*
  – Код для повторного использования – Developed for reusability;
  – Соответствие документации по продукту потребностям жизненного цикла – Documentation match to lifecycle needs;
– *Атрибуты команды разработчиков – Personnel attributes*
  – Преемственность коллектива – Personnel continuity;
  – Опыт работы с платформой – Platform experience;
– *Атрибуты проекта – Project attributes*

– Работа на нескольких площадках – Multi-site development.

В этой модели, наряду с показателями стоимости, учитываются еще и показатели масштаба ПО (Software Scale Drivers), значения которых также задаются по 6-ти бальной шкале, и к которым относятся:

– Наличие опыта аналогичных разработок – Precedentness;

– Гибкость самой разработки – Development flexibility;

– Решения по архитектуре/рискам – Architectural/risk resolution;

– Сплоченность команды разработчиков – Team cohesion;

– Уровень зрелости процесса разработки – Process maturity.

Разумеется, все оценки являются приблизительными и должны анализироваться и корректироваться в ходе проекта по мере того, как в ИБД проекта появляются и накапливаются фактические данные по трудозатратам на отдельных этапах и подэтапах разработки.

Наряду с представленными методами, планирование проекта можно осуществлять на основе проекта-аналога, который, по мнению эксперта, может служить дополнительным источником более точных данных, чем усредненные значения по ИБД организации, для оценивания соответствующих характеристик нового проекта. Система САМПО+ [11] поддержки создания ИБД позволяет на основе данных ИБД проводить объективную оценку выполнимости новых проектов при помощи автоматизированного поиска проекта-аналога.

Поиск аналога базируется на методологии рекомендуемого множества характеристик, которое формируется исходя из критерия функциональной пригодности информации, то есть ее актуальности, полноты и корректности. Только наличие всех трех критериев функциональной пригодности позволяет говорить о возможности классификации проектов в ИБД. Эта методология в общем случае состоит в последовательной проверке функциональной пригодности информации относительно: 1) *внесенных* в базу выполненных проектов; 2) *добавляемых* в базу выполненных проектов; и 3) *инициируемых* проектов.

Информация, хранимая в ИБД компании, функционально пригодна относительно *выполненных* проектов, если классификация проектов, полученная после кластеризации на всех предметно значимых поисковых наборах, согласуется с мнением экспертов.

Информация, хранимая в ИБД, функционально пригодна относительно *добавляемого* проекта, если данный проект может быть вписан в существующие классификаторы, либо путем включения в существующий класс, либо путем добавления нового класса в один из существующих классификаторов.

Информация, хранимая в ИБД, функционально пригодна отно-

сительно *инициируемого* проекта, если ему может быть сопоставлен какой-либо выполнявшийся ранее проект-аналог из ИБД.

В случае неудачи на каком-либо из перечисленных шагов состав характеристик ИБД расширяется и проводится ее повторное исследование на полноту, которое состоит из трех вышеуказанных этапов.

В рамках подхода, ориентированного на привлечение предметных знаний эксперта [12], с помощью классического алгоритма, основанного на вычислении оценок [13], модифицированного для решения поставленной задачи, формируется множество проектов (рекомендуемое множество), которые представляются эксперту в качестве возможных аналогов исследуемому проекту.

6. Заключение. Рассмотренный перечень измеряемых показателей является основой для объективного количественного управления ходом разработки ПО при заданных ограничениях. Не претендуя на исчерпывающую полноту, он, тем не менее, задает направление для постоянного совершенствования процесса разработки ПО в соответствии с моделью зрелости способностей CMMI для создания высококачественных и экономичных программных продуктов.

Специально рассмотрены вопросы планирования проекта с использованием методологии моделирования и принятия решений на основе алгоритмических сетей [14, 15] и исторической базы метрических данных по уже выполненным проектам, реализованной в системе САМПО+.

Задачей дальнейших исследований является расширение и уточнение этого списка, а также сбор и анализ фактических данных по выполненным проектам с целью установления текущей "базовой линии" как точки отчета для дальнейших улучшений процесса.

Литература

1. CMMI® for Development. Version 1.3 // Carnegie-Mellon University. 2010. 450 p.
2. *Babkin A.V.* Moving to CMMI: Approach, Results and Lessons Learned // Proceedings of St. Petersburg IEEE Chapters. 2005. vol. 2. pp. 183–188.
3. *Baranov S.* An Industrial Technology of Test Automation Based on Verified Behavioral Models of Requirement Specifications for Telecommunication Applications // IEEE EUROCON. St. Petersburg 2009. pp. 122–129.
4. *Тележкин А.М.* Создание исторических баз данных при помощи системы САМПО+ // Региональная информатика (РИ-2012). Труды конференции. Санкт-Петербург, 24-26 октября 2012 г. СПб. 2013. pp 84–90.
5. *Холстед М.Х.* Начала науки о программах // М.: Фин. и статистика. 1981. 128 с.
6. *Watson A.H., McCabe Th.J., Dolores R.* Structured Testing: a Testing Methodology Using the Cyclomatic Complexity Metric // National Institute of Standards and Technology Special Publication 500-235. 1996. 123 p.
7. *Capers, J.* Software Assessments, Benchmarks, and Best Practice // Addison-Wesley.

2000. 339 p.
8. Сайт продукта SEER® by Galorath. URL: http://www.galorath.com. (Дата обращения: 18.10.2014).
9. COCOMO® II - Constructive Cost Model. URL: http://csse.usc.edu/tools/COCOMOII.php. (Дата обращения: 18.10.2014).
10. *Баранов С.Н. и др.* Процесс разработки программных изделий // М.: Наука. 2000. 176 с.
11. *Тележкин А.М.* Система САМПО+ для создания и анализа исторической базы данных проектов // Известия ВУЗов. Приборостроение. 2014. Т.57. №11. С. 58–62.
12. *Морозов В.П.* Поддержка принятия решений, ориентированная на знания эксперта // Региональная информатика (РИ-2010). Труды конференции. Санкт-Петербург, 20-22 октября 2010 г. СПОИСУ. СПб. 2011. С. 69–73.
13. *Журавлев Ю.И., Никифоров В.В.* Алгоритмы распознавания, основанные на вычислении оценок // Кибернетика. 1971. №3. С. 1–11.
14. *Иванищев В.В., Марлей В.Е.* Введение в теорию алгоритмических сетей // СПб. Изд-во СПбГТУ. 2000. 179 с.
15. *Соколов Б.В., Михайлов В.В., Морозов В.П.* Методология и технология автоматизации аналитико-имитационного моделирования на основе алгоритмических сетей // Шестая всероссийская научно-практическая конференция «Имитационное моделирование. Теория и практика» ИММОД-2013. Казань. 2013. С. 256–262.

References

1. CMMI® for Development. Version 1.3. Carnegie-Mellon University. 2010. 450 p.
2. Babkin A.V. Moving to CMMI: Approach, Results and Lessons Learned. Proceedings of St. Petersburg IEEE Chapters. 2005. vol.2. 2005. pp. 183–188.
3. Baranov S. An Industrial Technology of Test Automation Based on Verified Behavioral Models of Requirement Specifications for Telecommunication Applications. IEEE EUROCON.St. Petersburg. 2009. pp. 122–129.
4. Telezhkin A.M. [Creating Historical Databases with the SAMPO+ System]. *Regional'naja informatika (RI-2012). Trudy konferencii* [Regional Informatics (RI-2012). St.Petersburg, 24-26 October 2012. Conference Proceedings]. St.Petersburg, 2013. pp. 84–90. (In Russ.)
5. Halstead M.H. *Nachala nauki o programmah* [Elements of Software Science] M.: Finansy i statistika. 1981. 128 p. (In Russ.)
6. Watson A.H., McCabe Th.J., Dolores R. Structured Testing: a Testing Methodology Using the Cyclomatic Complexity Metric. National Institute of Standards and Technology Special Publication 500-235. 1996. 123 p.
7. Capers J. Software Assessments, Benchmarks, and Best Practice. Addison-Wesley. 2000. 339 p.
8. Site SEER® by Galorath. Available at: http://www.galorath.com. (Acsessed: 18.10/2014).
9. COCOMO® II - Constructive Cost Model. Available at: http://csse.usc.edu/tools/COCOMOII.php. (Acsessed: 18.10/2014).
10. Baranov S.N. et al. *Process razrabotki programmnyh izdelij* [Software Development Process]. M: Nauka 2000. 176 p. (In Russ.)
11. Telezhkin A.M. [The SAMPO+ System for Creation and Analysis of Project Historical Database]. *Izvestiya vuzov. Priborostroyenie – Transactions of Universities. In-*

strumentation. 2014. vol. 57. no. 11. pp. 58–62. (In Russ.)
12. Morozov V.P. [Decision-Making Support Oriented to Experts' Knowledge]. *Regional'naja informatika (RI-2010). Trudy konferencii* [Regional Informatics (RI-2010). Conference Proceedings]. SPOISU. SPb. 2011. pp. 69–73. (In Russ.)
13. Zhuravlev Yu.I., Nikiforov V.V. [Recognition Algorithms basec of Estimate Calculations]. *Kibernetika – Cybernetics*. 1971. vol. 3. pp. 1–11. (In Russ.)
14. Ivanishchev V.V., Marley V.Ye. *Vvedenie v teoriju algoritmicheskih setej* [Introduction to the Theory of Algorithmic Networks]. SPb. Izd-vo SPbGTU. 2000. 179 p. (In Russ.)
15. Sokolov B.V., Mikhailov V.V., Morozov V.P. [Methodology and Technology for Automation of Analytical and Imitational Modeling Based on Algorithmic Networks]. *Shestaja vserossijskaja nauchno-prakticheskaja konferencija «Imitacionnoe modelirovanie. Teorija i praktika» IMMOD-2013* [Sixth All-Russia Scientific and Practical Conference "Imitational Modeling. Theory and Practice" IMMOD-2013]. Kazan. 2013. pp. 256–262. (In Russ.)

Баранов Сергей Николаевич — д-р физ.-мат. наук, главный научный сотрудник, Санкт-Петербургский институт информатики и автоматизации Российской академии наук (СПИИРАН), сотрудник международной научной лаборатории университета ИТМО. Область научных интересов: технология программирования, формальные методы. Число научных публикаций — 100. snbaranov@googlemail.com; 199178, Санкт-Петербург, 14 линия, д. 39; р.т. +7-812-328-0887.

Baranov Sergey Nikolaevich — Ph.D., Dr. Sci., chief researcher, St. Petersburg Institute for Informatics and Automation of Russian Academy of Sciences (SPIIRAS), researcher of international scientific laboratory of the university ITMO. Scientific interests: software engineering, formal methods in software development. The number of publications — 100. snbaranov@googlemail.com; 39, 14-th Line, St. Petersburg, 199178, Russia; office phone +7-812-328-0887.

Тележкин Александр Михайлович — аспирант, Санкт-Петербургский институт информатики и автоматизации Российской академии наук (СПИИРАН). Область научных интересов: технология программирования, формальные методы в разработке программного обеспечения. Число научных публикаций — 6. telezhkin@gmail.com; 192102, Санкт-Петербург, ул. Фучика 4к; р.т. +7-812-325-05-64.

Telezhkin Alexander Mikhailovich - Ph.D. student, St. Petersburg Institute for Informatics and Automation of Russian Academy of Sciences (SPIIRAS). Scientific interests: software engineering; formal methods in software development. The number of publications — 6. telezhkin@gmail.com; 4K, Fuchika st., St. Petersburg, 192102, Russia; office phone +7-812-325-05-64.

Поддержка исследований. Работа выполнена при государственной финансовой поддержке ведущих университетов Российской Федерации (субсидия 074-U01), университет ИТМО.

Acknowledgements. This research is partially supported by the leading universities of the Russian Federation (grant 074-U01), university ITMO.

РЕФЕРАТ

Баранов С.Н., Тележкин А.М. **Метрическое обеспечение программных разработок.**

Рассмотренный в статье перечень из 112 измеряемых показателей является основой для объективного количественного управления ходом разработки ПО при заданных ограничениях. Не претендуя на исчерпывающую полноту, он, тем не менее, задает направление для постоянного совершенствования процесса разработки ПО в соответствии с моделью зрелости способностей CMMI для создания высококачественных и экономичных программных продуктов.

Специально рассмотрены вопросы планирования проекта с использованием методологии моделирования и принятия решений на основе алгоритмических сетей и исторической базы метрических данных по уже выполненным проектам, реализованной в системе САМПО+.

Сформулирована задача дальнейших исследований – расширение и уточнение этого списка, а также сбор и анализ фактических данных по выполненным проектам с целью установления текущей "базовой линии" как точки отчета для дальнейших улучшений процесса.

SUMMARY

Baranov S.N., Telezhkin A.M. **Metrics for Software Development.**

The considered list of 112 measurable indications is a basis for objective quantitative managing a software development project under given constraints. Being incomplete, it nevertheless determines the direction for continuous improvement of the software process in accordance with the CMMI model for creating creating high-quality and efficient software products.

Planning issues under the methodology of modeling and decision-making based on algorithmic networks and historical database of metrics of completed projects within the system SAMPO+ are considered as well.

Directions for further study are defined: extending and refining the list of metrics as well as accumulating and analyzing actual data on completed projects in order to establish project and process baselines for further process improvements.