УДК 004.413

МЕТРИКИ СЛОЖНОСТИ ДЛЯ УПРАВЛЕНИЯ ПРОЦЕССОМ РАЗРАБОТКИ ПРОГРАММНОГО СРЕДСТВА ПО МЕТОДОЛОГИИ SCRUM

А.А. КУЗИКОВ, В.В. БАХТИЗИН

Белорусский государственный университет информатики и радиоэлектроники
П. Бровки, 6, Минск, 220013, Беларусь

Поступила в редакцию 5 июля 2011

Рассматривается влияние сложности программного средства на планирование и выполнение работ по его разработке в командах, работающих по методологии Scrum. Проводится анализ ряда существующих метрик методологии Scrum, применяемых при планировании. Предлагаются новые метрики сложности для оценки организации работ над программным средством при планировании и на всем протяжении итерации.

Ключевые слова: сложность программных средств, разработка программных средств.

Введение

С целью повышения конкурентоспособности программных средств (ПС), когда необходимо не только выпустить качественное программное обеспечение в условиях меняющихся требований, но и сделать это раньше конкурентов, а также ввиду наличия ряда недостатков каскадной модели процесса разработки ПС [1], многие компании рассматривают целесообразность перехода к инкрементной и эволюционной моделям. При этом гибкие методологии разработки ПС, детализирующие и расширяющие инкрементную модель, а также зарекомендовавшие себя в условиях меняющихся требований, адаптированы и достаточно широко практикуются в реальных проектах.

Из многообразия существующих гибких методологий получила достаточно широкое распространение [2] методология Scrum, которая, предоставляя каркас организации и управления процессом разработки ПС, успешно сочетается с методологиями, ориентированными на другие инженерные практики [3–5], например, экстремальное программирование и разработку, управляемую тестами. Фиксированные рамки итераций (от двух до шести недель [3, 4]), на протяжении каждой из которых Scrum-командой выполняются работы процесса разработки ПС по реализации требований заказчика, представленных в журнале требований продукта в виде историй, ориентируют Scrum-команду на реализацию только полезных и важных заказчику требований и на выпуск работоспособных версий ПС на регулярной основе.

Как показывает практика, Scrum-команды, в частности, начинающие работать по гибкой методологии, нередко не способны выполнить все истории, запланированные с учетом производительности работы членов команды. Наиболее распространенными причинами срыва графиков разработки ПС являются:

1) заниженная оценка трудозатрат на выполнение запланированных историй вследствие неубывающей сложности разрабатываемого ПС,

2) неравномерная загруженность членов Scrum-команды в течение итерации в отсутствие учета взаимосвязей между задачами, на которые декомпозирована история.

К концу итерации при невыполнении нормативов времени по реализации Scrum-командой отдельной истории, как правило, имеет место снижение качества разрабатываемого ПС и, следовательно, его ценности для заказчика и конкурентоспособности:

- за счет не реализованной в полном объеме функциональности ПС, ввиду переноса такой истории в журнал продукта для проведения доработки в рамках одной из последующих итераций;

- при сокращении затрат времени на оставшиеся в рамках истории задачи, например, при исключении задач, не выполнимых в рамках текущей итерации, либо уменьшении объемов работ по ним. Нередко уменьшение трудозатрат производится путем сокращения времени на тестирование и документирование ПС, что, в результате, негативно и непредсказуемо отражается на большинстве характеристик качества ПС;

- при сокращении затрат времени на оставшиеся в рамках истории задачи, формирование списка известных проблем, разрешение которых планируется в рамках иной истории. В данном случае, величина негативного эффекта на качество ПС заранее известна.

Метрики Scrum, используемые при планировании

Одной из основных метрик методологии Scrum, учитываемых на этапе планирования каждой итерации, является скорость работы Scrum-команды. Данная метрика эмпирически определяется как средний объем работ, который выполняется Scrum-командой за одну итерацию по отношению к элементам журнала продукта (историям) [4] либо средний суммарный объем историй, работа над которыми выполнена командой за одну итерацию [5]. При этом объем работ, необходимых для выполнения отдельной истории, ассоциируется с ее сложностью. Так, сложность истории принято описывать в единицах трудозатрат, рассчитываемых как отношение объема работ (обычно измеряется в часах) для завершения истории к средней занятости члена команды в течение фиксированного промежутка времени (обычно измеряется в часах в день), то есть

$$\mathrm{Eff}^{*}(t)=\left[\frac{\mathrm{Eff}(t)}{\lambda}\right], \qquad (1)$$

где: $\mathrm{Eff}(t)$ – объем работ, необходимых для завершения истории на момент времени t от начала итерации; λ – средняя занятость члена команды в течение фиксированного промежутка времени; $[\ ]$ – операция получения целой части числа.

Метрика сложности истории (1) обычно рассчитывается для момента времени $t=0$ от начала итерации и не отражает внутреннюю структуру организации работ по выполнению данной истории. Далее в работе предлагаются метрики, которые позволят рассчитать фактическую сложность историй и тем самым предоставить более достоверные данные для этапов планирования выполнения работ в рамках итерации с целью снижения рисков невыполнения командой заданного объема работ.

Термины и определения

Определение 1. Множество членов Scrum-команды можно описать следующим образом:

$$M=\left\{m_l \mid l=\overline{1,L}\right\},$$

где: m_l — l-ый член команды; L — количество членов Scrum-команды.

Определение 2. Любая итерация представима в виде следующего множества историй, каждая из которых, например, отражает требование пользователя к разрабатываемому ПС или является описанием обнаруженного в ПС дефекта:

$$S=\left\{s_k \mid k=\overline{1,K}\right\},$$

где: s_k – k-ая история итерации; K – число историй, выбранных для реализации в рамках итерации.

Определение 3. Каждую историю s_k можно декомпозировать на следующее множество задач – участков работы, которую может самостоятельно выполнить отдельный член команды:

$$W_k = \left\{ w_{k,i} \mid i = \overline{1, N_k} \right\},$$

где: $w_{k,i}$ – i-ая задача k-ой истории; N_k – число задач в k-ой истории.

Поскольку k-ая история декомпозирована на задачи таким образом, что каждая задача может быть выполнена только одним членом команды, то каждой задаче $w_{k,i}$ истории s_k можно поставить в соответствие исполнителя, роль которого играет член Scrum-команды.

Определение 4. Множество исполнителей в рамках k-ой истории можно описать следующим образом:

$$A_k = \left\{ a_{k,i} \mid i = \overline{1, N_k} \right\},$$

где: $a_{k,i}$ – i-ый исполнитель, назначенный на выполнение i-ой задачи k-ой истории итерации; N_k – число исполнителей задач в k-ой истории.

Каждый член команды m_l может быть неоднократно задействован в выполнении различных задач истории s_k, т.е. играть роль исполнителя одной и более задач k-ой истории либо не иметь назначенных задач, что выражается следующим отображением:

$$\text{Assgn} : A_k \to M . \qquad (2)$$

В общем случае, последовательность выполнения задач имеет значение, ввиду наличия зависимостей между задачами. Поскольку множество задач W_k составляет декомпозицию истории s_k, но k-ая история считается выполненной только после прохождения приемки у владельца истории (обычно владельца продукта), то можно полагать, что некоторая конечная задача $w_{k,0}$, представляющая собой работы по приемке истории s_k, будет зависеть от выполнения всего множества задач k-ой истории. Исполнителем задачи $w_{k,0}$, как правило, является лицо, выполняющее приемку истории, например, владелец продукта $a_{k,0}$.

Определение 5. Ориентированный граф $G_k = (V_k, E_k)$ является представлением истории s_k, если множество его вершин $V_k = W_k \cup \{w_{k,0}\}$ есть множество задач истории s_k. Элементами множества E_k являются упорядоченные пары задач вида $(w_{k,i}, w_{k,j}), i \neq j$, причем, задача $w_{k,j}$ зависит от выполнения задачи $w_{k,i}$.

Из *определения* 5 и способа построения графа следует, что

- граф G_k истории s_k не является тривиальным, ввиду того, что $\{w_{k,1},\ w_{k,0}\} \subseteq V_k$;

- граф G_k является связным, так как из любой вершины из множества W_k существует путь в вершину $w_{k,0}$.

Определение 6. Множество W'_k называется множеством начальных задач, если выполнение таких задач не зависит от выполнения каких-либо других задач k-ой истории, т.е.

$$\left(W'_k = \left\{ w'_{k,j} \mid \forall j \in [1, N_k] \right\}\right) \Leftrightarrow \begin{pmatrix} \forall i \in \overline{1, N_k},\ i \neq j \\ w_{k,i} \in W_k,\ \left(w_{k,i}, w'_{k,j}\right) \notin E_k \end{pmatrix}.$$

Применительно к каждой задаче $w_{k,i}$ k-ой истории будем полагать, что задача обладает свойством атомарности (неделимости) в той мере, что

- задача $w_{k,i}$ может быть назначена только одному исполнителю $a_{k,i}$,

- переход от выполнения задачи $w_{k,i}$ к выполнению одной из зависимых от нее задач $w_{k,j}$, например, при $(w_{k,i}, w_{k,j}) \in E_k$ и $\text{indeg}(w_{k,j}) = 1$, возможен только при условии завершения исполнителем $a_{k,i}$ работ над задачей $w_{k,i}$.

Свойство атомарности задачи исключает наличие в орграфе G_k петель и циклов, четко определяя изменение объема работ Scrum-команды над конкретной историей.

Поскольку в случае методологии Scrum объем работ для каждого члена команды выражается в единицах времени (часах, днях), то показатель времени, требующегося исполнителю для завершения задачи, будет отражать риски, связанные с несвоевременным выполнением запланированных работ. Подобным показателем служит временная функция $\tau(w_{k,i}, a_{k,i}, t)$, определяемая отображением

$$\tau : W_k \times A_k \times \left(\mathbf{R}_+ \cup \{0\}\right) \to \mathbf{R}_+ \cup \{0\}, \quad (3)$$

где: $\mathbf{R}_+$ – множество положительных действительных чисел.

Функция (3) отражает время, необходимое исполнителю $a_{k,i}$ для непосредственного выполнения работ над задачей $w_{k,i}$ на момент времени t от начала итерации. Как правило, для хорошо оцененных исполнителем задач функцию τ можно рассматривать как невозрастающую и неотрицательную в течение итерации.

Оценка трудоемкости выполнения работ

Для каждой задачи $w_{k,i}$ следует проанализировать множество W_k на наличие зависимостей. Подобный анализ содействует построению очередности выполнения задач.

Определение 7. $D_{k,i}$ есть такое подмножество задач истории s_k, что начало работ над задачей $w_{k,i}$ непосредственно зависит от результатов их выполнения в порядке логического следования:

$$D_{k,i} = \left\{w_{k,j} \mid \forall j \in [1, N_k],\ j \neq i, \left(w_{k,j}, w_{k,i}\right) \in E_k\right\}. \quad (4)$$

Из *определения* 7 следует, что для любой начальной задачи $w_{k,i} \in W_k'$ $D_{k,i} = \varnothing$.

Негативное влияние на выполнение запланированной работы в фиксированные временные рамки итерации оказывает неэффективное использование рабочего времени членами Scrum-команды в результате незапланированных простоев.

Рассмотрим время, необходимое исполнителю $a_{k,i}$ для завершения всех работ над задачей $w_{k,i}$. Очевидно, что данная величина слагается из времени непосредственного выполнения исполнителем $a_{k,i}$ работ над задачей $w_{k,i}$, а также времени ожидания завершения работ над множеством задач (4), от которых зависит задача $w_{k,i}$, на момент времени t от начала итерации:

$$\mathrm{T_C}\left(w_{k,i}, a_{k,i}, t\right) = \tau\left(w_{k,i}, a_{k,i}, t\right) + \mathrm{T_W}\left(w_{k,i}, a_{k,i}, t\right). \quad (5)$$

Время ожидания $\mathrm{T_W}\left(w_{k,i}, a_{k,i}, t\right)$ исполнителем $a_{k,i}$ начала работ над задачей $w_{k,i}$ можно определить как максимальное время, необходимое для завершения работ над задачами $D_{k,i}$. Следует отметить, что задачи из множества $D_{k,i}$, назначенные разным членам Scrum-команды, предполагают возможность их одновременного выполнения. Таким образом, если в k-ой истории для каждого члена команды m_l имеется не более одной назначенной ему на выполнение задачи $w_{k,i}$, то время ожидания начала выполнения задачи $w_{k,i}$ исполнителем $a_{k,i}$ таким, что $\mathrm{Assgn}\left(a_{k,i}\right) = m_l$, на момент времени t от начала итерации составит:

$$\mathrm{T_W}\left(w_{k,i}, a_{k,i}, t\right) = \max_{\forall w_{k,j} \in D_{k,i}} \mathrm{T_C}\left(w_{k,j}, a_{k,j}, t\right). \quad (6)$$

Если в k-ой истории для некоторого члена команды m_l имеется более одной назначенной ему на выполнение задачи, то возможность одновременного выполнения таких задач членом команды m_l исключается. Для учета данной специфики необходимо преобразовать граф G_k в соответствии с приведенным далее алгоритмом. Время ожидания начала выполнения за-

дачи $w_{k,i}$ исполнителем $a_{k,i}$ на момент времени t от начала итерации также рассчитывается по формуле (6) с учетом влияния, оказанного преобразованным графом G_k на множество $D_{k,i}$.

Функция (6) характеризует потенциальную величину времени простоя члена Scrum-команды, назначенного исполнителем задачи $w_{k,i}$.

Алгоритм преобразования графа истории

Алгоритм преобразования графа k-ой истории из множества S историй, запланированных для реализации в течение итерации, состоит из следующих шагов:

1) выбрать члена команды m_l из множества M членов Scrum-команды;

2) выбрать подмножество задач $Q_{k,l} := \{w_{k,i} \mid i \in [0, N_k]\} \subseteq V_k$, для исполнителей которых выполняется условие $\mathrm{Assgn}(a_{k,i}) = m_l$;

3) если для члена команды m_l в k-ой истории имеется более одной назначенной ему для выполнения задачи, т.е. $|Q_{k,l}| > 1$, то перейти к шагу 4. Иначе перейти к шагу 14;

4) выбрать очередную задачу $w_{k,i}$ из множества $Q_{k,l}$. Для определенности будем полагать, что просмотр множества задач $Q_{k,l}$ для выбора очередной задачи организован слева направо. Если задача $w_{k,i}$ последняя в множестве $Q_{k,l}$, то перейти к шагу 14. Иначе перейти к шагу 5;

5) из задач, следующих за $w_{k,i}$ в множестве $Q_{k,l}$, выбрать очередную задачу $w_{k,j}$. Если подобный выбор задачи $w_{k,j}$ не осуществим, то перейти к шагу 4. Иначе перейти к шагу 6;

6) если в графе G_k существует путь через вершины $w_{k,i}$ и $w_{k,j}$, то перейти к шагу 5. Иначе перейти к шагу 7;

7) если приоритет задачи $w_{k,i}$ выше приоритета задачи $w_{k,j}$, необходимо расширить множество ребер E_k графа G_k за счет нового ребра $(w_{k,i}, w_{k,j})$ и перейти к шагу 5. Иначе перейти к шагу 8;

8) если приоритет задачи $w_{k,i}$ ниже приоритета задачи $w_{k,j}$, необходимо расширить множество ребер E_k графа G_k за счет нового ребра $(w_{k,j}, w_{k,i})$ и перейти к шагу 5. Иначе перейти к шагу 9;

9) расширить множество ребер E_k графа G_k за счет нового ребра $(w_{k,i}, w_{k,j})$ и рассчитать значение $Tw_1 := \mathrm{T_W}(w_{k,0}, a_{k,0}, t)$;

10) заменить ребро $(w_{k,i}, w_{k,j})$ ребром $(w_{k,j}, w_{k,i})$ в множестве ребер E_k графа G_k и рассчитать значение $Tw_2 := \mathrm{T_W}(w_{k,0}, a_{k,0}, t)$;

11) если $Tw_1 < Tw_2$, то перейти к шагу 12. Иначе перейти к шагу 13;

12) заменить ребро $(w_{k,j}, w_{k,i})$ ребром $(w_{k,i}, w_{k,j})$ в множестве ребер E_k графа G_k;

13) перейти к шагу 5;

14) если рассмотрено все множество M членов команды, то завершить алгоритм. Иначе перейти к шагу 1.

Потенциальная реализуемость истории

Рассмотрим функцию $\mathrm{T_C}(w_{k,0}, a_{k,0}, t)$, значение которой соответствует времени, необходимому для завершения работ над k-ой историей на момент времени t от начала итерации. Т.е. значение функции $\mathrm{T_C}(w_{k,0}, a_{k,0}, 0)$ отражает фактическую трудоемкость истории на момент начала итерации.

Определение 8. Потенциальная реализуемость k-ой истории характеризуется следующей величиной

$$\beta_{\mathrm{k}}(t)=\frac{\mathrm{T_C}\left(w_{k,0},a_{k,0},t\right)}{T_I-t}, \qquad (7)$$

где $w_{k,0}$ – конечная задача k-ой истории; $a_{k,0}$ – исполнитель конечной задачи $w_{k,0}$; T_I – время, характеризующее продолжительность итерации, в рамках которой запланированы работы над историей s_k; $t\in[0,T_I)$ – время, прошедшее от начала итерации.

История s_k является потенциально реализуемой в рамках итерации, если $\mathrm{в_k}(t)\le 1$. Иначе, если $\mathrm{в_k}(t)>1$, то существует риск невыполнения k-ой истории за время, ограниченное рамками итерации.

Пример. Пусть $M=\{m_l \mid \forall l=\overline{1,4}\}$ – множество участников Scrum-команды; $A=\{a_{k,i} \mid \forall i=\overline{1,3},\ \mathrm{Assgn}(a_{k,i})=m_i\}\cup\{a_{k,0} \mid \mathrm{Assgn}(a_{k,0})=m_4\}$ – множество исполнителей задач k-ой истории; $V_k=\{w_{k,i} \mid \forall i=\overline{1,3}\}\cup\{w_{k,0}\}$ – множество задач k-ой истории; T_I – продолжительность итерации; $\tau(w_{k,1},a_{k,1},0)=T_I/2$, $\tau(w_{k,2},a_{k,2},0)=T_I/3$, $\tau(w_{k,3},a_{k,3},0)=T_I/4$, $\tau(w_{k,0},a_{k,0},0)=0$ – время выполнения задач первым, вторым, третьим исполнителями и владельцем k-ой истории соответственно (рассматривается момент начала итерации $t=0$). Рассмотрим два предельных случая декомпозиции истории s_k: s_1 и s_2 – отличающиеся только характером взаимосвязей между задачами.

1. Первый вариант декомпозиции (рис. 1). История s_1 декомпозирована на задачи таким образом, что возможно их одновременное выполнение ($W_1=W_1'$).

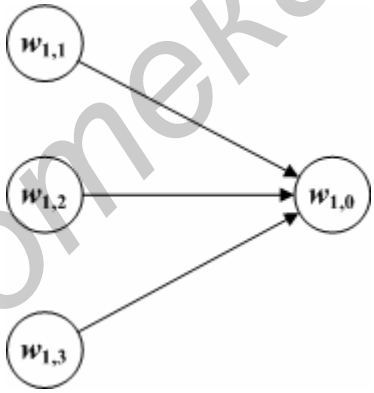

Рис.1. Декомпозиция истории на задачи с возможностью их одновременного выполнения

2. Второй вариант декомпозиции (рис. 2). История s_2 декомпозирована на задачи таким образом, что возможно только последовательное их выполнение ($|D_{2,i}|=1,\ \forall i\in\{0,2,3\}$).

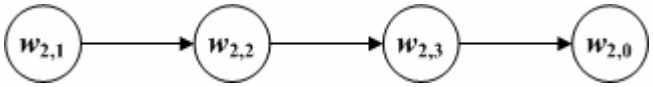

Рис. 2. Декомпозиция истории на задачи с необходимостью их последовательного выполнения

В обоих случаях декомпозиции историй s_1 и s_2 алгоритм преобразования графа истории не влияет на соответствующие множества ребер E_1 и E_2. Следовательно, согласно формуле (5), фактическая трудоемкость историй s_1 и s_2 составит:

$$\mathrm{T_C}(w_{1,0},a_{1,0},0)=0+\max\left(T_I/2,T_I/3,T_I/4\right)=T_I/2,$$
$$\mathrm{T_C}(w_{2,0},a_{2,0},0)=0+\left(T_I/4+\left(T_I/3+T_I/2\right)\right)=13/12\cdot T_I.$$

Согласно формуле (7), потенциальная реализуемость историй s_1 и s_2 на момент начала итерации будет равна

$$\beta_1(0) = \frac{\mathrm{T_C}\left(w_{1,0}, a_{1,0}, 0\right)}{T_I - 0} = 1/2 < 1,$$

$$\beta_2(0) = \frac{\mathrm{T_C}\left(w_{2,0}, a_{2,0}, 0\right)}{T_I - 0} = 13/12 > 1.$$

Метрика (1) для обеих историй s_1 и s_2 составит:

$$\mathrm{Eff}_1^*(0) = \mathrm{Eff}_2^*(0) = \left[\frac{1}{\lambda} \cdot \sum_{i=0}^{3} \tau\left(w_{1,i}, a_{1,i}, 0\right)\right] = \left[\frac{13}{12\lambda} \cdot T_I\right].$$

Несмотря на то, что с точки зрения метрики (1) сложность обеих историй одинакова, только история s_1 потенциально реализуема в рамках одной итерации.

Необходимо отметить, что составление журнала итерации в полном соответствии с текущей скоростью работы Scrum-команды не гарантирует того, что все запланированные истории будут выполнены в рамках данной итерации. Это объясняется тем, что скорость работы Scrum-команды является эмпирически рассчитываемым показателем, чувствительным только к значениям метрики (1), которая, как показано выше, не учитывает структуру задач истории. Например, если при планировании очередной итерации число историй со структурой взаимосвязей между задачами, сходной с s_2, оказалось больше, чем в среднем за предыдущие итерации, то существует риск невыполнения работ в полном объеме.

Заключение

Таким образом, в данной работе рассмотрено влияние сложности ПС на планирование и выполнение работ в команде, в частности, на наличие простоев в работе у членов Scrum-команды, а также на качество разрабатываемого ПС. Предложены метрики сложности, оценивающие фактическую трудоемкость истории (5), величину простоя в работе у членов команды (6) и показатель потенциальной реализуемости запланированных работ в рамках итерации (7). Данные метрики учитывают взаимосвязи между задачами, на которые декомпозируется каждая история в итерации. Поэтому метрики (5)–(7) могут применяться на этапе планирования работ в рамках итерации для первичной оценки распределения нагрузки, выполнимости работ и корректировки приоритетности выполнения задач истории. Наличие зависимости данных метрик от времени, прошедшего от начала итерации, позволяет проводить подобную оценку на регулярной основе.

COMPLEXITY METRICS FOR MANAGING A SCRUM SOFTWARE DEVELOPMENT PROCESS

A.A. KUZIKAU, V.V. BAKHTIZIN

Abstract

An impact of software complexity on planning a software development process and its implementation in teams, practicing Scrum methodology, has been examined. An analysis of a number of Scrum metrics used for planning has been conducted. A set of new complexity measures, aimed at assessing organisational management during planning and implementation phases of Scrum iterations, have been proposed.

Литература

1. *Бахтизин В.В., Глухова Л.А.* Технология разработки программного обеспечения. Минск, 2010.
2. *Benefield G.* // IEEE Computer Society. 2008. P. 461.
3. *Kniberg H.* Scrum and XP from the Trenches. C4Media, 2007.
4. *Schwaber K.* The enterprise and scrum. Microsoft Press Redmond. USA, 2007.
5. *Keith C.* Agile game development with Scrum. Addison-Wesley Professional, 2010.