Рекурсивная модель жизненного цикла разработки программ и ее применение

М.М.Мучник, к.т.н., с.н.с., доцент

Компания “Сканпрог”, 04070,Украина, Киев, вул.Фроловська, 6/8, корп.1, Факс: (380-44) 463-76-14, Тел.: (380-44) 463-76-15, E-mail: muchnik@padco.kiev.ua

Аннотация. В докладе рассмотрены различные модели жизненного цикла разработки программ и предложена новая – рекурсивная модель жизненного цикла. Проведен сравнительный анализ рекурсивной и базовых моделей жизненного цикла, рассмотрено применение рекурсивной модели в других информационных технологиях.

Вступление

На протяжении истории развития программирования как науки были разработаны различные модели жизненного цикла (ЖЦ) разработки программ, методологии программирования и соответствующие им технологические средства разработки программ, которые позволили существенно упростить и упорядочить процесс разработки программных продуктов.

В настоящее время разработан ряд международных стандартов, регламентирующих терминологию в этой сфере и подходы к рассмотрению моделей жизненного цикла разработки программ, выделены базовые типы моделей жизненного цикла, такие как каскадная, инкрементная и эволюционная [1,2].

1 Базовые модели жизненного цикла

Каскадная модель.В рамках *каскадной модели* предполагается, что жизненный цикл разработки включает следующие, выполняемые последовательно этапы:

- *установление потребностей пользователя;*
- *определение требований;*
- *проектирование (конструирование);*
- *изготовление;*
- *испытание; корректировка;*
- *поставка или использование*.

Если последовательные работы перекрываются, то некоторые этапы могут быть частично выполнены параллельно.

Инкрементная модель. *Инкрементная модель* предполагает, что процесс создания программы начинается после получения от заказчика набора требований и состоит в разработке последовательности конструкций, все более приближающихся к итоговой системе. Вначале реализуется наиболее очевидная и необходимая часть требований, в последующую конструкцию добавляют дополнительные требования и так далее до тех пор, пока не будет закончено создание системы. Для каждой конструкции выполняют необходимые процессы, работы и задачи, например, анализ требований и создание архитектуры могут быть выполнены одновременно, в то время как разработка технического проекта, программирование и тестирование, компоновка и отладка реализуются при создании каждой последующей конструкции.

При таком подходе, называемом также запланированным усовершенствованием продукта, для разработки каждой конструкции работы и задачи процесса разработки выполняют последовательно или частично параллельно с перекрытием. Работы и задачи процесса разработки обычно выполняют неоднократно в той же последовательности для всех конструкций. Процессы сопровождения и эксплуатации могут быть реализованы параллельно с процессом разработки.

Эволюционная модель. В соответствии с *эволюционной моделью* также предполагается, что программный комплекс разрабатывается в виде последовательности конструкций, но, в отличие от инкрементной модели, подразумевается, что требования не могут быть полностью осознаны и сформулированы предварительно, поэтому они устанавливаются частично и уточняются в каждой последующей конструкции.

При таком подходе для каждой конструкции работы и задачи процесса разработки выполняют последовательно или параллельно с частичным перекрытием.

Работы и задачи процесса разработки обычно выполняют не однократно, возможно в той же (или в разной) последовательности для всех конструкций. Процессы сопровождения и эксплуатации могут быть реализованы параллельно с процессом разработки. Процессы заказа и поставки, а также вспомогательные и организационные процессы обычно выполняют параллельно с процессом разработки.

Надо сказать, что этими типами моделей ЖЦ их разнообразие не исчерпывается. Так, при более детальном уровне рассмотрения ЖЦ отдельно выделяются итерационные циклические формы, основанные на технике макетирования системы.

Таким образом, можно говорить о том, что используемые в настоящее время базовые модели ЖЦ предполагают как последовательное (в каскадной модели), так и итеративное (в инкрементной и эволюционной моделях) выполнение этапов ЖЦ.

В то же время, как отмечается в [2], интерпретация данных моделей различными участвующими в процессе разработки программы людьми может существенно различаться, ряд реальных процессов разработки программ достаточно сложно объяснить в рамках указанных моделей, что позволяет говорить о том, что вопрос разработки адекватной модели жизненного цикла разработки программ и до настоящего времени продолжает оставаться актуальным.

2 Типовой жизненный цикл

В [1] модель жизненного цикла определена как структура, состоящая из процессов, работ и задач, включающих в себя разработку, эксплуатацию и сопровождение программного продукта, охватывающая жизнь системы от установления требований к ней до прекращения ее использования, т.е. можно указать следующие этапы:

1. Формулирование требований (что нужно разработать);
2. Проектирование (как это сделать);
3. Разработка (реализация) программного продукта в соответствии с проектом;
4. Тестирование (проверка соответствия программы исходным требованиям). Если не соответствует, переход к п.2;
5. Эксплуатация и сопровождение

   5.1. Анализ необходимости модификации в соответстствии с потребностями заказчика (если требуется, переход к п.1);

   5.2. Анализ необходимости снятия с эксплуатации (если эксплуатация должна продолжаться, переход к п.5);

   5.3. Снятие программы с эксплуатации.

Блок-схема этого процесса представлена на Рис.1, а граф-схема - на Рис.2.

На Рис. 2 не показаны подэтапы 5.1 –5.3, поскольку они не являются самостоятельными и выполняются в рамках этапа 5.

Описанная модель ЖЦ соответствует как последовательным, так и итеративным моделям жизненного цикла, однако ничего принципиально нового, по сравнению с уже известными, такая модель не дает.

3 Рекурсивная модель жизненного цикла

Как показывает весь опыт разработки различных программ, начинать непосредственную разработку программы сразу после анализа поставленной задачи и разработки алгоритма ее решения можно только для относительно небольших задач. Для разработки больших программных комплексов вначале необходимо четко выделить различные элементы комплекса (входящие в комплекс программы и данные) и определить логические связи между ними, т.е. спроектировать структуру комплекса. Для этого были предложены различные подходы, например, методологии процедурно- и объектно-ориентированного программирования.

Однако, с помощью какой-бы методологии ни проводилась разработка, очевидны различия в процессах реализации небольших и простых программ и больших и сложных программных комплексов.

Для небольших и простых программ, которые могут быть реализованы в виде одного модуля, процесс проектирования (п. 2) сводится к проектированию внутренней структуры этого модуля, а жизненный цикл действительно является линейным или итеративным, поскольку сразу после этапа проектирования можно переходить к этапам разработки и тестирования.

Для сложных программных комплексов предварительно необходимо провести логическое и физическое проектирования структуры комплекса, т.е. разбить комплекс на уровни и выделить отдельные модули, вычленить классы и объекты и т.д. Как отмечено в [3], «модули выполняют роль физических контейнеров, в которые помещаются определения классов и объектов при логическом проектированиии системы».

В процессе проектирования строится иерархически-модульная структура (“дерево”) [4] программного комплекса, узлы которого соответствуют модулям определенного уровня. Модули самого низкого уровня, уже не содержащие обращений к другим модулям, можно рассматривать как модули-листья такого дерева (см.Рис. 3). Однако независимо от того, представляется ли программа одним модулем, или группой модулей, каждый модуль в иерархически-модульной структуре программного комплекса представляет собой программу, которая должна пройти полный жизненный цикл разработки.

Для входящих в иерархически-модульную структуру модулей верхних уровней проведение полного жизненного цикла в процессе нисходящего проектирования невозможно, поскольку еще отсутствует текст вызываемых ими модулей. Только модули-листы представляют собой самостоятельные программы, поскольку не содержат обращений к другим модулям.

Следовательно, в отличие от других инженерных изделий, переход от этапа проектирования к этапам разработки и тестирования текста модулей-листов сразу же после завершения проектирования этих модулей (т.е. до завершения проектирования всего программного комплекса), вполне возможен и оправдан, поскольку модули-листы независимы от других модулей программного комплекса [5].

Как было отмечено в [6], нисходящее проектирование представляет собой рекурсивный процесс, при котором для проектирования структуры модулей на очередном уровне используется одна и та же рекурсивная процедура, но процесс проектирования рассматривался отдельно от процесса разработки, поскольку предполагалось, что процесс разработки текста программы, как и процесс разработки любого инженерного изделия, происходит после окончания проектирования всего программного комплекса.

Если же после окончания проектирования модулей-листов сразу переходить к их разработке и тестированию, то не только проектирование, но и весь процесс разработки программного комплекса можно рассматривать как рекурсивную процедуру (назовем ее процедурой “жизненный цикл”), в процессе последовательных вызовов которой формируется иерархически-модульная структура программного комплекса. При проектировании структуры каждого модуля на каждом уровне формулируются требования к модулям нижних уровней, которые передаются процедуре формирования этих модулей как параметры. Рекурсивные вызовы продолжаются до тех пор, пока структура модуля очередного уровня уже не требует дальнейшей детализации (достигнут модуль-лист). Получение отлаженного программного кода модуля–листа является “точкой возврата” рекурсивной процедуры “жизненный цикл”.

После завершения тестирования текста модуля-листа этот текст “возвращается” в процедуру формирования модуля предыдущего уровня для сборки с другими такими модулями и т.д. вверх до получения текста всего программного комплекса.

Таким образом, полный текст программного комплекса формируется в процессе возврата из последовательности рекурсивных вызовов процедуры “жизненный цикл” (т.е. в процессе “обратного прохода” по дереву программного комплекса). На Рис. 3 стрелки, направленные сверху-вниз, показывают передачу требований модулям нижних уровней, а стрелки, направленные снизу-вверх – “возврат” отлаженного текста модулей. На Рис. 4 этот процесс представлен в виде граф-диаграммы для одной вертикали этого дерева.

Если модуль какого-либо уровня не прошел тестирования и оказалось, что это произошло вследствие некорректного проектирования или некорректности модулей нижних уровней, то в процедуре “жизненный цикл” происходит итеративный возврат к этапу проектирования и снова начинаются рекурсивные вызовы этой процедуры для получения корректного текста. При изменении требований к какому-либо модулю, для него и всех “подчиненных” ему модулей повторяется процесс проектирования (модули других ветвей, независимые от данного, при этом не изменяются). “Подчиненные” модули, требования к которым не изменились, также могут использоваться без изменений. Из этого следует, что, если разработка программы производится в соответствии с итеративно-рекурсивной моделью, то изменения будут быстрыми и локализованными.

Следует отметить, что рекурсивная модель вполне адекватна как модульному программированию (при процедурно-ориентированном подходе), так и сборочному и компонентному программированиям (при объектно-ориентированном подходе), поскольку, в последнем случае, процесс нисходящего проектирования останавливается при наличии в библиотеке необходимого класса или объекта. В настоящее время в Международном Соломоновом университете соответствии с рекурсивной моделью реализуется система сборочного программирования «Легенда».

Таким образом, если расссматривать процесс разработки любого программного комплекса как итеративно-рекурсивную процедуру, можно получить вполне естественные ответы на многие из возникавших ранее вопросов:

1. Различие в разработки простых и сложных программ можно интерпретировать как вопрос количества уровней в иерархической структуре программного комплекса. Для небольших программ, не требующих детализации в виде нескольких модулей, проектирование сводится к проектированию структуры одного модуля, а жизненный цикл для таких программ последователен или итеративен. Реализация же больших и сложных программных комплексов, приобретающих в процессе проектирования многоуровневую структуру, проводится в соответствии с итеративно-рекурсивной моделью жизненного цикла. Очевидно, что проектирование и разработка таких комплексов будет требовать более жесткой дисциплины всего процесса проектирования и разработки.
2. Нисходящее и восходящее проектирование в соответствии с предложенной моделью можно рассматривать не как альтернативные стратегии, а как две составляющие одного процесса.
3. Последовательные и итеративные модели жизненного цикла включаются в предложенную модель как составляющие каждого уровня.

4 Применение рекурсивной модели в других информационных технологиях

Рассмотренная выше рекурсивная модель может быть использована не только в качестве модели ЖЦ, но и в качестве модели других сложных информационных процессов, которых имеют (или формируют в результате некоторого процесса, например, анализа или проектирования) иерархическую структуру.

Например, в [7] было предложено использовать эту модель для представления процесса сбора отчтных данных в корпоративных объединениях, имеющих иерархическую структуру.

Поскольку любое предприятие или организация построены по иерархическому принципу, то можно утверждать, что на таких объектах документооборот также производится в соответствии с их иерархической структурой, а это означает, что системы документооборота также должны быть построены в соответствии с рекурсивной моделью.

Рассмотрение многих информационных технологий показало, что принципы их построения соответствуют рекурсивной модели, однако, это можно рассматривать скорее как соответствие здравому смыслу разработчика, чем явное следование рекурсивной модели. Вероятно, рекурсивная модель отражает принципы мышления человека при решении сложных задач и поэтому может быть использована в различных приложениях.

ВЫВОДЫ.

1. Предложена итеративно-рекурсивная модель жизненного цикла разработки программ, обеспечивающая:
   - Общий подход к разработке малых и больших программ;
   - Обобщение методов нисходящего и восходящего проектирования;
   - Обобщение последовательных и итеративных моделей жизненного цикла;
   - Управляемое повторное использование программ.
2. Рекурсивная модель может быть использована не только в качестве модели жизненного цикла, но и для любых информационных технологий, создаваемых для сложных объектов, изначально имеющих иерархическую структуру, либо таких, в которых иерархическая структура создается в результате некоторого процесса.

Литература

1. ГОСТ Р ИСО/МЭК 12207-99. Информационная технология. Процессы жизненного цикла программных средств
2. Васютович В., Самотохин С., Никифоров Г. Регламентация жизненного цикла программных средств. – Открытые ситемы, Директор ИС, #07-08/2000.
3. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++, 2-е изд.- М.: «Издательство Бином», СПб.: «Невский диалект», 2001.- 560 с.
4. Мучник М.М., Левицкий А.А. Инструментальные средства подддержки технологии многоуровневого проектирования программных комплексов с иерархически-мо-дульной структурой. - В кн.: 7-я Всесоюзная школа-семинар “Параллельное программирование и высокопроизводительные системы”. - Киев, ИК АН УССР, 1986г.
5. Мучник М.М. Жизненный цикл разработки программ: рекурсивная модель.- Корпоративные системы, № 3, 2003, с.74-76
6. Зиглер К. Методы проектирования программных систем:- М.:Мир,1985.-328 с.
7. Мучник М.М. Модель систем сбора отчетных данных в корпорациях.- Корпоративные системы, №1, 2003, с.69-71
8.

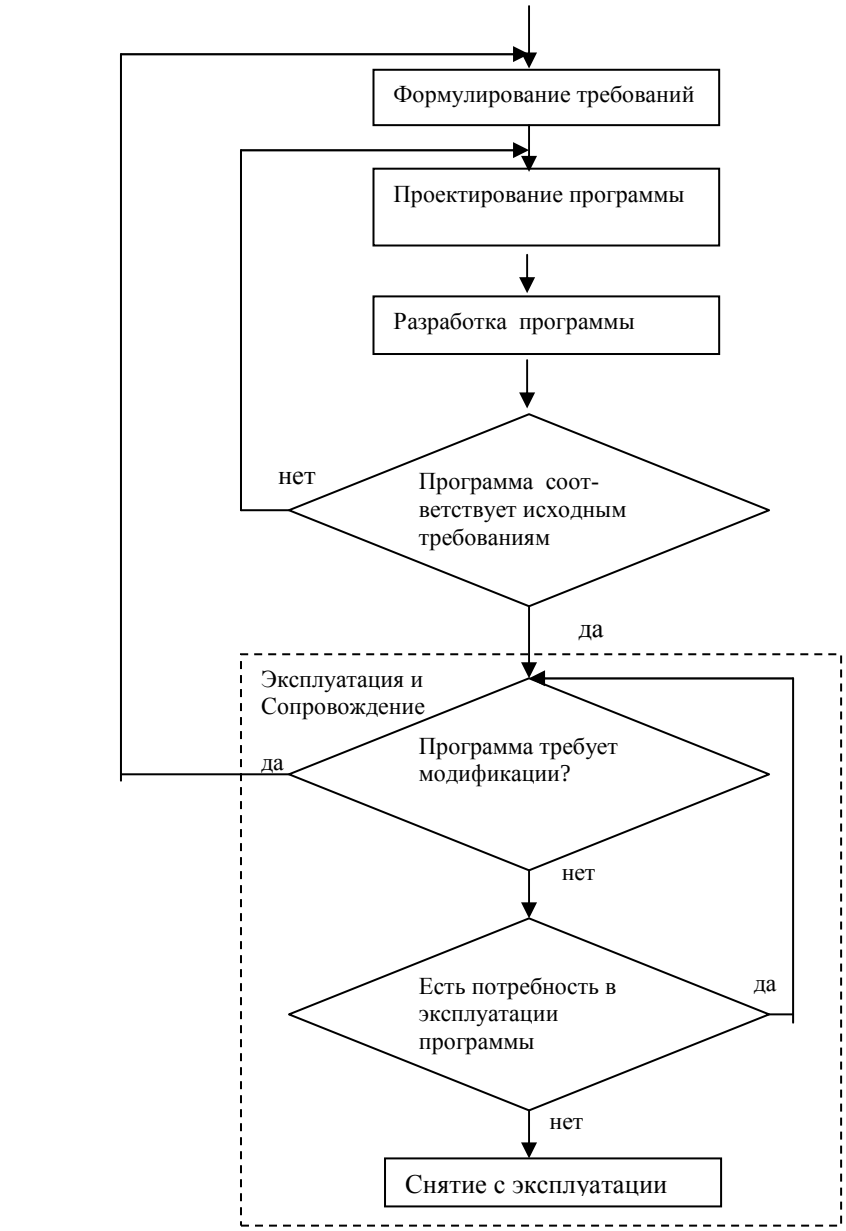

Рисунок 1. Блок-схема типового жизненного цикла

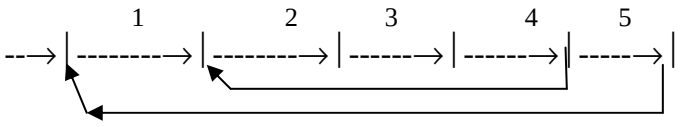

Рис.2. Граф-схема типового жизненного цикла

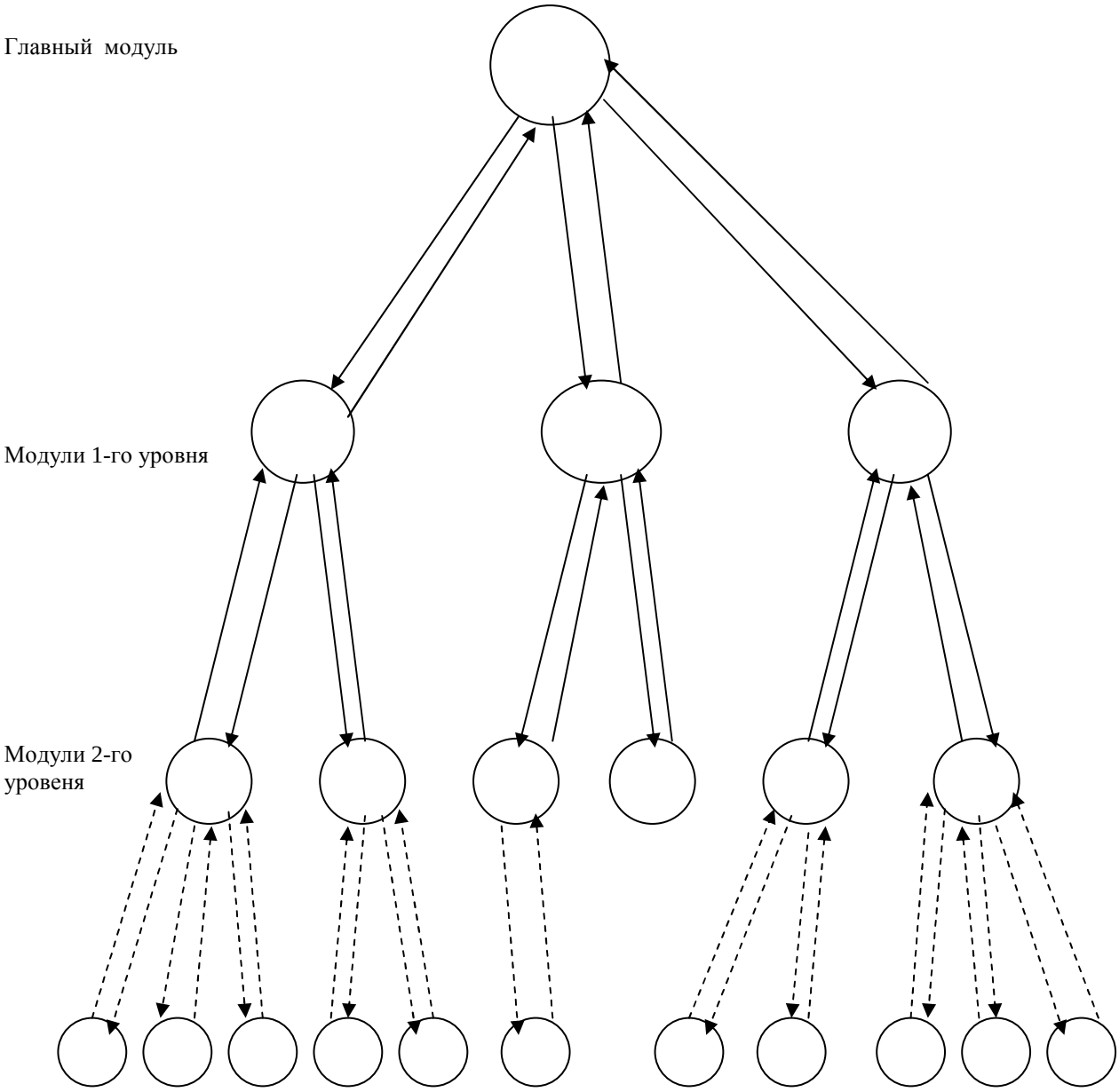

Рисунок 3. Иерархически-модульная структура программного комплекса

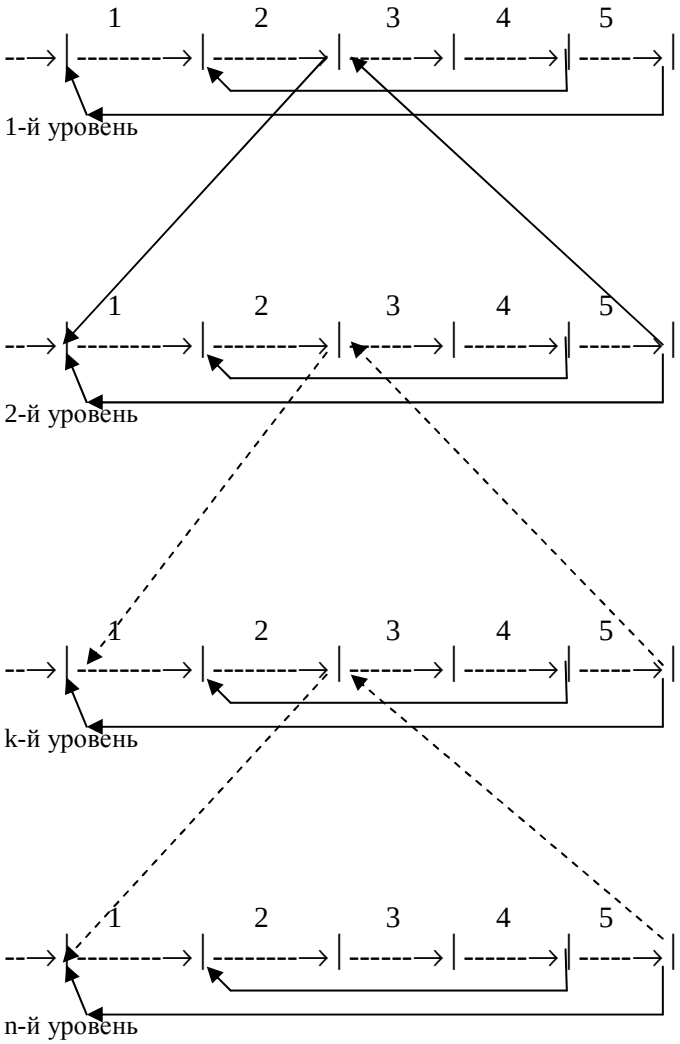

Рисунок 4. Граф-диаграмма рекурсивного жизненного цикла разработки программного продукта