Тема: ПОДХОДЫ К ПРОЕКТИРОВАНИЮ СЛОЖНЫХ СИСТЕМ. Методика Джексона.

Содержание:

1. Введение

В настоящий момент во всем мире наиболее широко используются два основных подхода к проектированию сложных систем:

- **структурный и**
- **объектно–ориентированный подходы.**

Использование того или иного метода зависит от следующих факторов:

1. **квалификация разработчиков,**
2. **масштаб проекта,**
3. **затраты на разработку проекта,**

результат, который должен быть получен.

2. Структурное программирование

Методика Джексона

Метод проектирования Джексона (Jackson System Development, JSD) - это метод спецификации и дизайна систем. В основе этого метода лежат базовые идеи структурного программированя Джексона (Jackson Structure programming, JSP).

В JSD используются для моделирования два типа диаграмм - *Entity Structure Diagrams* (ESD -диаграммы сущностей) и *Network Diagrams.*

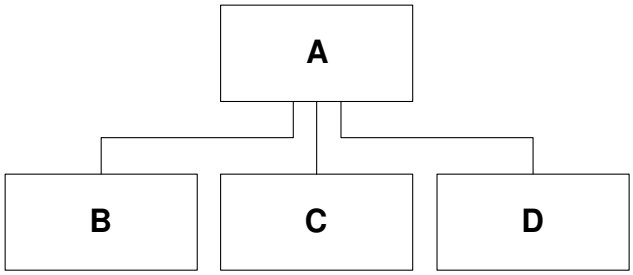
В JSD имеется три основных этапа:

- этап моделирования (результатом этого этапа являются ESD)
- *network*-этап
- этап внедрения

Основная идея методики Джексона состоит в том, что «форма», или структура подлежащих обработке данных будет определять «форму» или структуру обрабатывающей программы. В этой методике основные конструкции структурного программирования применяются для построения структуры входных и выходных данных, и те же структуры используются для построения программы. При этом одна и та же нотация служит для обозначения и данных, и программ.

Используемая в методике Джексона нотация содержит следующие конструкции.

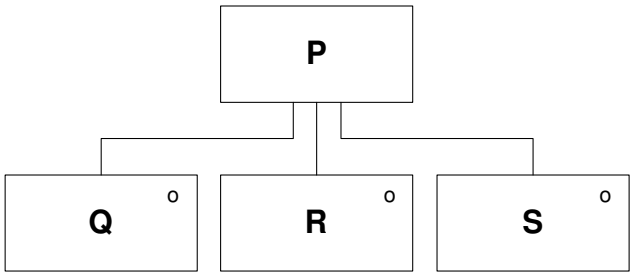
Конструкция последовательности. Эта конструкция показывает, что стоящий выше по иерархии сущность (объект) состоит из объектов, лежащих ниже, в указанном порядке слева направо.



Конструкция последовательности

На этом примере отражено, что блок «A» состоит из блоков «B», «C» и «D».

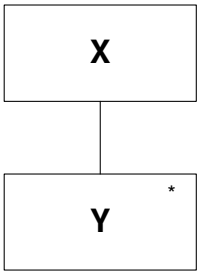
Конструкция выбора. Эта конструкция показывает, что лежащий выше по иерархии объект состоит из ровно из одного из объектов, находящихся уровнем ниже по иерархии. Она отличается от последовательности наличием значка «o» в правом верхнем углу каждого прямоугольника второго уровня. Конструкция выбора должны содержать две или более конструкции второго уровня иерархии.



Конструкция выбора

На этом примере показано, что блок «P» состоит или из блока «Q», или из блока «R», или из блока «S».

Конструкция повторения. Эта конструкция показывает, что объект X состоит из нуля или более объектов, лежащих ниже по иерархии. Конструкция повторения включает одну и только одну конструкцию второго уровня и обозначается символом «*» в прямоугольнике второго уровня.



Конструкция повторения

На этом примере сказано, что объект «X» состоит из нуля или более повторяющихся объектов «Y».

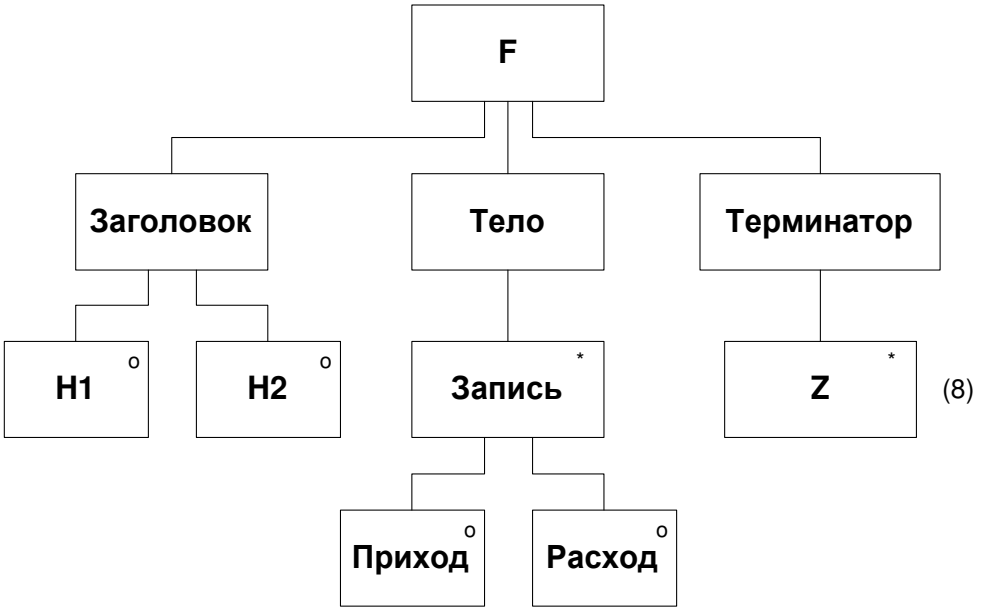
Все эти три конструкции структурного программирования необходимы и достаточны для построения любой программы.

Каждый из приведенных выше элементов A, B, C, D, P, Q, R, S, X, Y стал бы оператором или даже процессом программы.

Далее по методике Джексона выполняются следующие этапы:

1. Изобразить структуры входных и выходных данных.
2. Идентифицировать связи обработки (соответствия) между структурами данных.
3. Сформировать структуру программы на основании структур данных и соответствий.
4. Перечислить все выделения исполняемых операций для структуры программы.
5. Написать программу в «структурированном изложении».

Пример элементарной конструкции согласно методике Джексона, описывающей формат файла данных.



(8)

Описание формата файла F

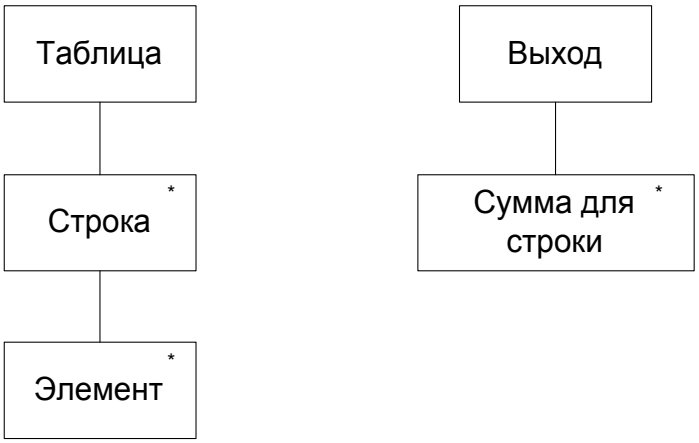
Файл F - это последовательность, которая представляет собой «Заголовок», за которым следует «Тело файла», а затем «Терминатор»: признак конца файла – строго в

указанном порядке. Заголовок представляет собой выбор из компонент либо «H1», либо «H2». Повторение записи (ноль или более записей) образует «Тело Файла», и аналогично «Терминатор» является повторением компонента «Z». Число «8» в круглых скобках указывает, что «Терминатор» фактически состоит из восьми элементов типа «Z».

Все компоненты низкого уровня, которые не разлагаются на подкомпоненты, считаются **элементарными**. Конечно, каждая элементарная конструкция может иметь собственную структуру, но ее на данном этапе может быть не целесообразно разлагать далее.

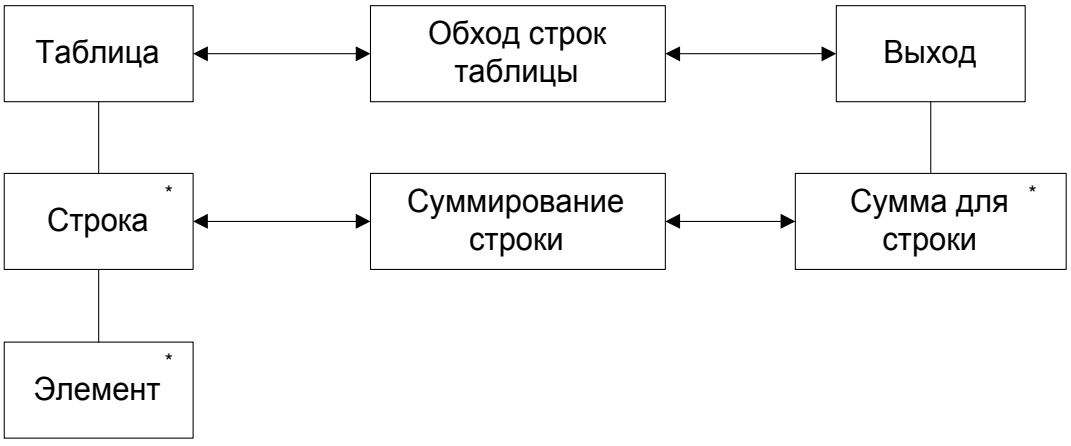
Простой пример взаимосвязи.

Рассмотрим программу, которая суммирует построчно элементы в таблице и выдает их сумму в виде чисел.



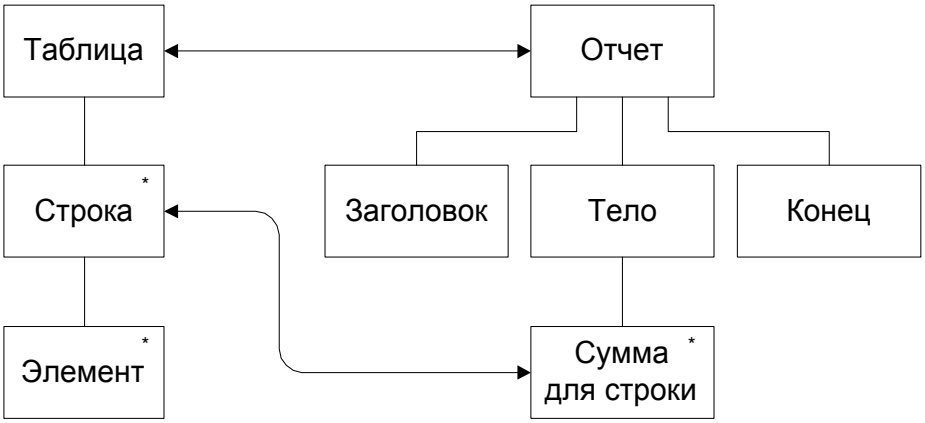
Структуры входных и выходных данных

Рассмотрим вариант программы, которая может обрабатывать входные данные.



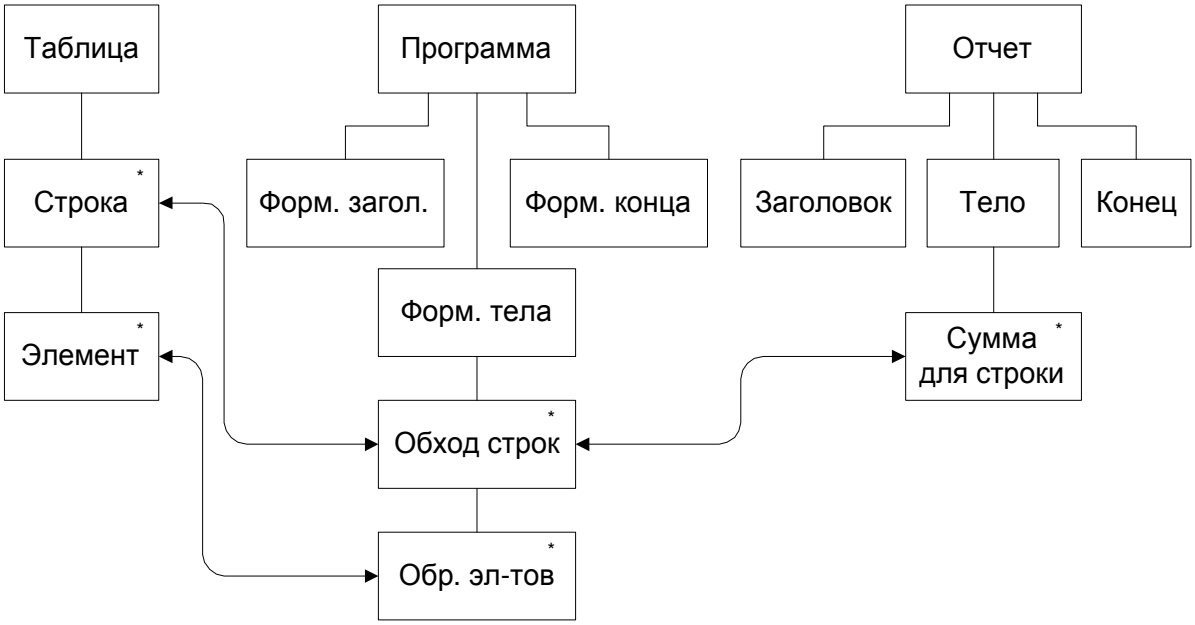
Соответствие данных и программы

Усложним структуру выходных данных. Пусть нам потребуется выдавать данные в виде файла, имеющего некоторую структуру.



Усложненные структуры входных и выходных данных

Используя данную структуру, мы получим следующую структуру программы.



Усложненая программа суммирования

После окончательной стадии, когда получена диаграмма Джексона для разрабатываемого алгоритма, все исполняемые операции переписываются, и записывается программа на псевдокоде.

К достоинствам методики Джексона следует отнести высокую эффективность, простоту, интуитивную понятность, возможность свести к минимуму число ошибок, наличие инструментальных средств поддержки данной методики для ряда языков программирования.

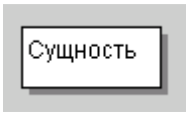
3. ИТАК: “10 правил”

Entity Structure Diagram (ESD).

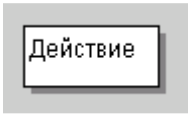
ESD – очень важная часть JSD, которая является результатом моделирования. при помощи ESD записываются действия системы в порядке выполнения.

1.Диаграммы Джексона (*Entity Structure Diagrams* - ESD) – это иерархическое дерево: между различными действиями на одном и том же уровне и между действиями различных ветвей соединений быть не должно.

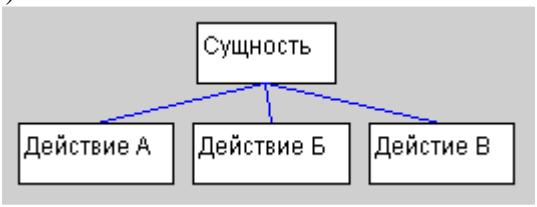
2. Сущность (entity) - это объект, физический или абстрактный, который сам что-то делает в системе, или с чем производит действия сама система, или и то, и другое. Таким образом, сущностью может быть, например, человек или какой-то проект (или программа).



3. Действие (activity) – это происходящее с сущностью событие или производимое сущностью событие. Каждое событие, возможно, поделить в свою очередь на более мелкие события.



4. В диаграмме может быть только одна сущность и эта сущность «корень» дерева.
5. Последовательность (sequence) есть декомпозиция действий или сущности на одно или больше действий, где составляющие события происходят в точном порядке (слева направо).



Ситуация, представленная на рисунке, описывается при помощи псевдокода следующим образом:

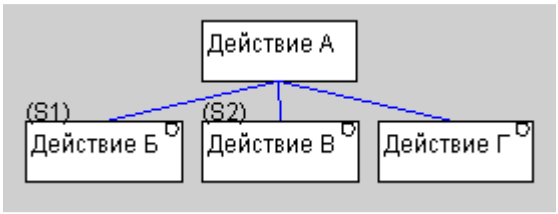
```
begin /*  Сущность*/

  Действие А;
  Действие Б;
  Действие В;
end
```

или

```
{/*  Сущность*/
```

```
  Действие А;
  Действие Б;
  Действие В;
}
```

6. Выбор – декомпозиция компонента схемы на два или больше действия, из которых в зависимости от условия выполняется только одно. Связанное с выбором условие записывается рядом с конструкцией выбора, и ни в коем случае, внутри конструкции выбора.

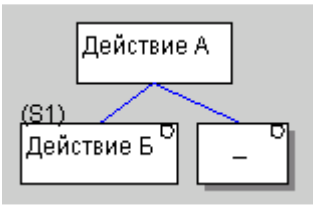


Ситуация, представленная на рисунке, описывается при помощи псевдокода следующим образом (S1, S2 - условия):

```
if <S1> then
  Действие Б
else if <S2> then
  Действие В
else
  Действие Г
```

Для большей наглядности рекомендуется действия заключать в фигурные скобки или блоки *begin* и *end*.

7. «Нулевая» конструкция (пустой компонент) используется тогда, когда выбор возможен, но не надо ничего делать. В случае может быть только один пустой компонент.



Ситуация, представленная на рисунке, описывается при помощи псевдокода следующим образом S1 - условие):

```
if <S1> then
  Действие Б
```

8. Повторение – декомпозиция компонента схемы, которая содержит только одно действие, которое в зависимости от условия произойдет или нет. Связанное с повторением условие записывается рядом с конструкцией повторения, и ни в коем случае, внутри конструкции выбора.

Ситуация, представленная на рисунке, описывается при помощи псевдокода следующим образом (l1 - условие):

while <l1> **do**
Действие Б

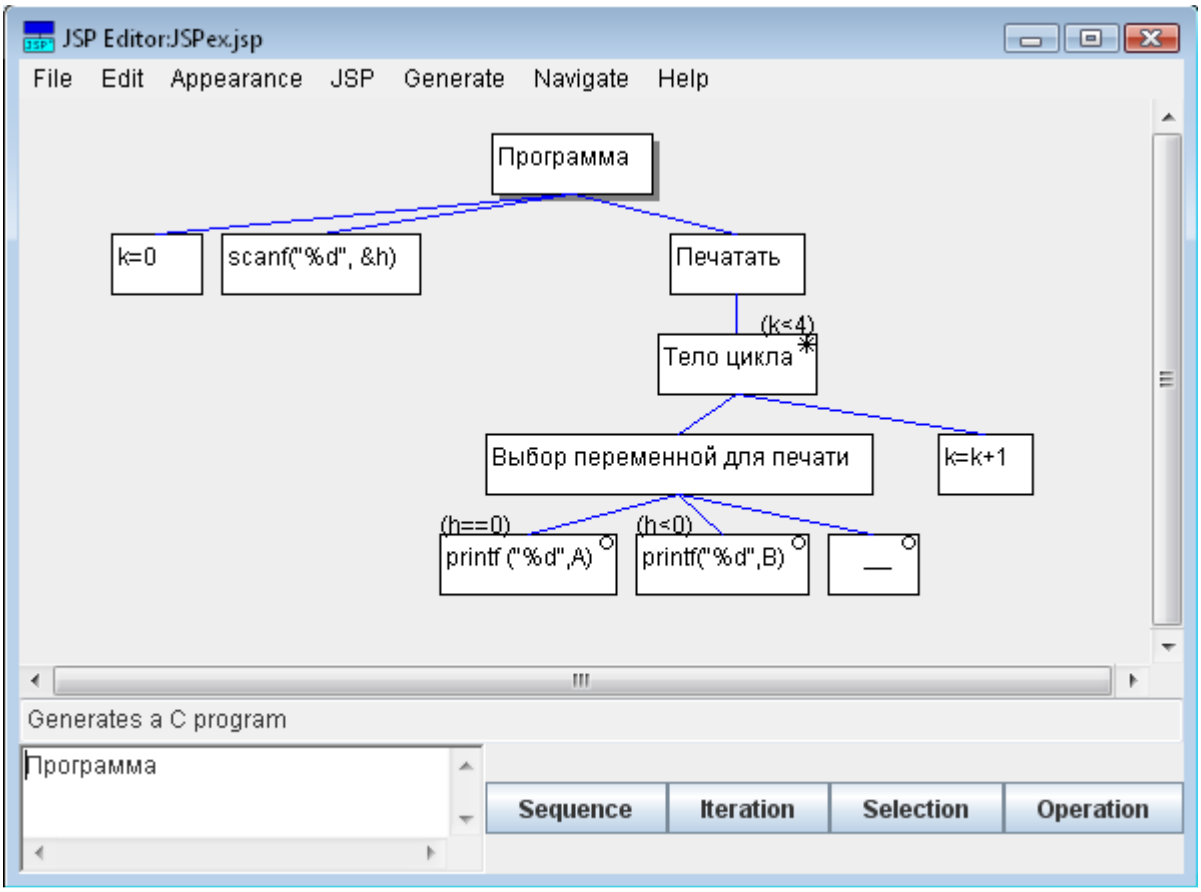
9. Конструкция повторения – единственный потомок своего родителя.
10. Потомки одного родителя должны быть одного типа, смешанные конструкции запрещены.

В таблицу действий и атрибутов должны быть помещены все соответствующие действия, их описания и связанные с ними атрибуты.

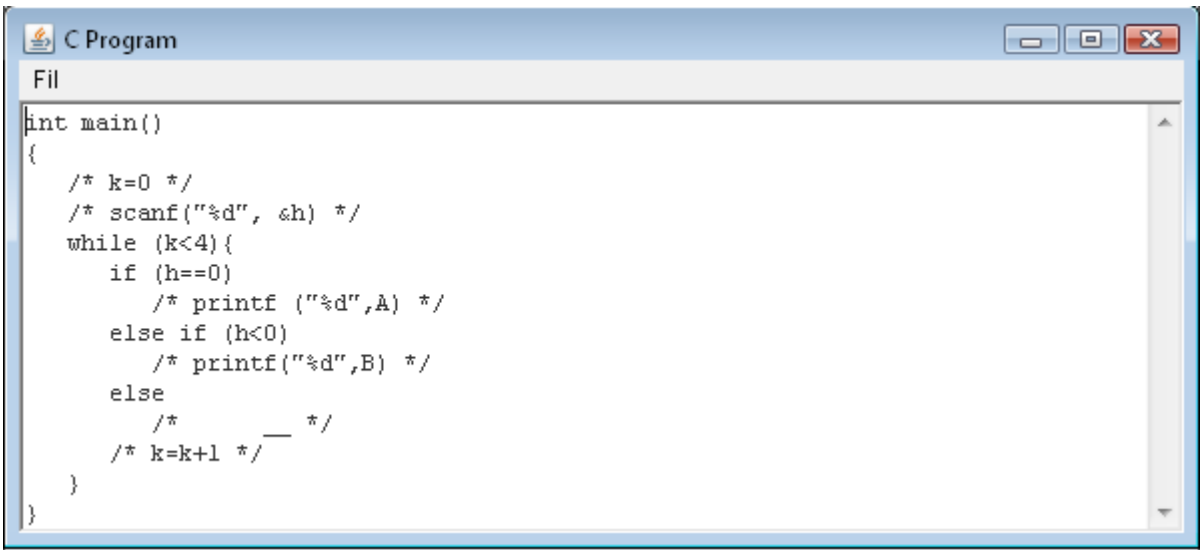
действие	**описание действий**	**атрибуты действий**
действие1	...	...
действие2	...	...
...	...	...

Пример:

Следующая программа должна в зависимости от значения переменной h печатать значения переменной A или B.



JSP генерирует скелет С-программы.



действие	**описание действий**	**атрибуты действий**
Печатать	Печатать значения переменных A или B	k<4
…	...	...
...	...	...

Составлено: **Марина Брик**
9.10.2007
*Обновлено:***26.10.2010**

При составлении использованы материалы:

How to Draw Jackson System Development (JSD) Diagrams
http://www.smartdraw.com/resources/centers/software/jsd.htm

Tutorial on JSP & JSD
http://cisx2.uma.maine.edu/NickTemp/JSP&JSDLec/jsd.html

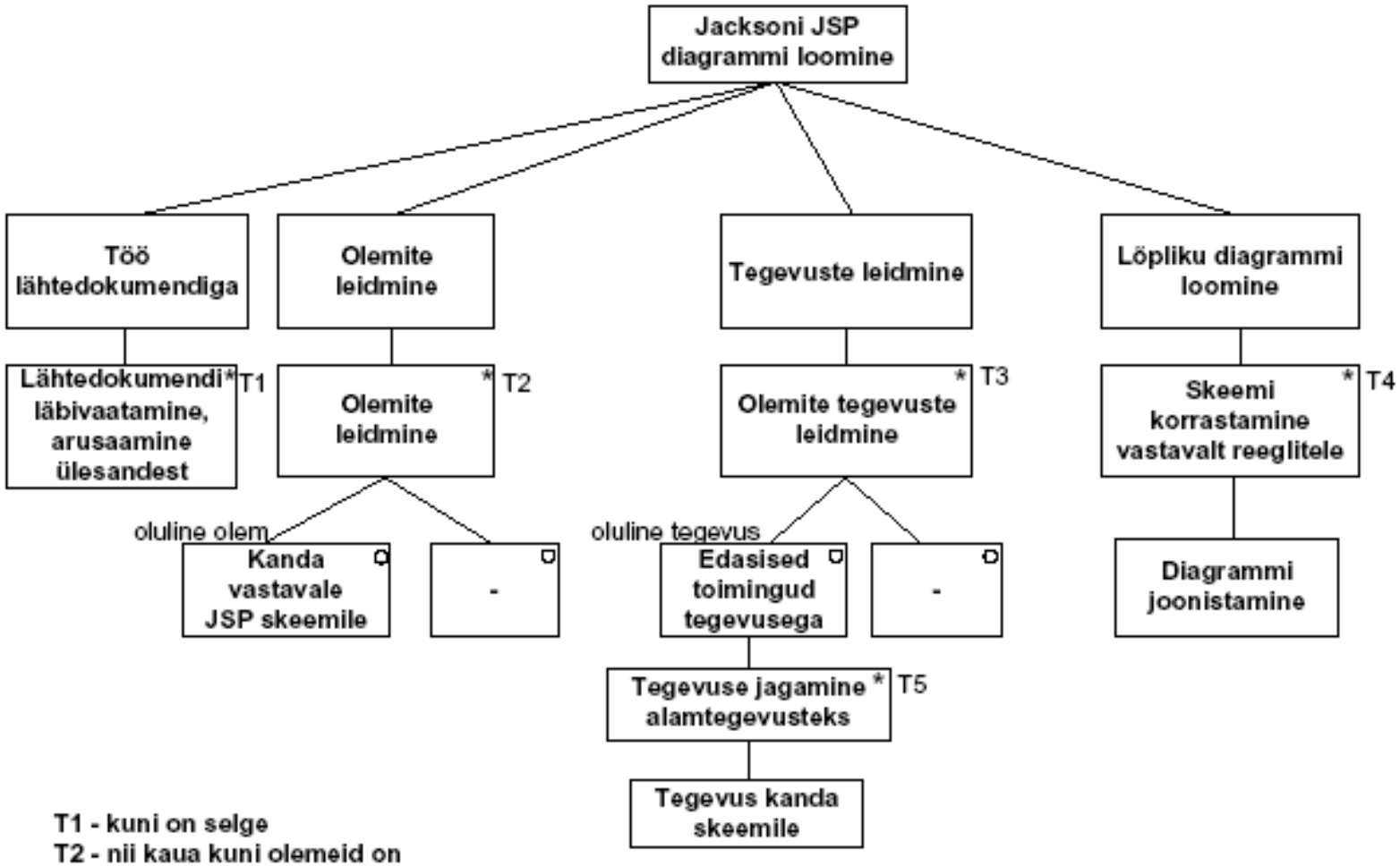
M. Jackson. JSP In Perspective. sd&m Pioneers` Conference, Bonn, June 2001, 12p.

Материалы H. Kruus, T. Robal, К. Григорьевой и др.

При составлении диаграмм использован редактор:
http://www.his.se/english/university/contact/staff/henrik-engstrom/jsp-editor/

Приложение

Построение диаграммы Джексона можно описать при помощи самой диаграммы Джексона:



T1 - kuni on selge
T2 - nii kaua kuni olemeid on
T3 - nii kaua kui tegevusi on
T4 - seni,kuni skeem vastab kõikidele reeglitele
T5 - atomaarse tegevuse leidmiseni, st tegevus pole enam alamtegevusteks jaotatav