Software Engineering and TRIZ (2)

Step-wise Refinement and the Jackson Method Reviewed with TRIZ

Toru Nakagawa
(Osaka Gakuin Univ., Japan)
http://www.osaka-gu.ac.jp/php/nakagawa/TRIZ/eTRIZ/

ETRIA Conference
"TRIZ Future 2005"
Graz, Austria
November 16-18, 2005

This series of research is going to review
basic concepts in Software Engineering & Computer Science
one by one with the eyes of TRIZ.

Three aims:

(1) To clarify the concepts in SE/CS from the TRIZ views
(2) To feed the knowledge in SE/CS back into TRIZ
(3) To clarify how to apply TRIZ to software development and software-based problems.

Text in SE: "Program Engineering --
Implementation, Design, Analysis, and Testing"
by Osamu Shigo (in Japanese, 2002)

In our previous paper (TRIZCON2005):

Structured Programming Reviewed with TRIZ:

Conventional teaching in SE/CS:

"**Structured Programming is the theory**
as originally proposed with the three constructs
and is compromised with practice
by adding four or so extra constructs."

Our view with TRIZ (over the conflict resolution):

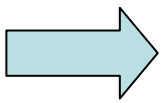
"**Structured Programming** in its final form is
the programming style with tree-type procedural structure
(or non-skewed nesting of procedures) inside modules
by using 3 + 4 or so basic control constructs. "

Feedbacks from Structured Programming onto TRIZ:

- Step-wise Refinement ==>
  TRIZ Inventive Principle 1 (Segmentation) should contain
  "**Segmentation of Problems**" Sub-Principle.

- Basic constructs ==>
  TRIZ Inventive Principle 6 (Universality) should contain
  "**Universal Standards**" Sub-Principle.

- Hierachical structure of procedures ==>
  TRIZ Inventive Principle 7 (Nesting) should contain
  "**Hierarchy of Systems**" Sub-Principle.



In the present paper, we go ahead to review
Step-wise Refinement and
the Jackson Method
(or 'Jackson Structured Programming').

Step-wise Refinement
at the step of detailed design of program modules

Top-down style of designing (in SE)

⟺ **TRIZ** Segmentation (Pr. #1) and
Local Quality (Pr. #3).

⇨ **Sub-Principle: "Segmentation of Problems" (in Pr. #1)**

Usage of pseudo-code (in SE)

to describe the procedural controls clearly
while describing procedural contents in a natural lanuage

⟺ **TRIZ** Intermediary (Pr. #24)
Prior Action (Pr. #10)
Partial or Excessive Action (Pr. #16)
Homogeneity (Pr. #33)

Criteria of good Step-wise Refinement *(by Shigo)* (1)

(in SE) "At any stage of refinement,
the whole procedure should be made understandable
with the description down to the level,
without referring to any further detail and
without being forced to read unnecessary details."

⇒ (in TRIZ) "At any level of system description/design,
the whole technical system should be made understandable
with the description of the parts down to the level,
without referring to any further detail and
without being forced to read unnecessary details of parts."

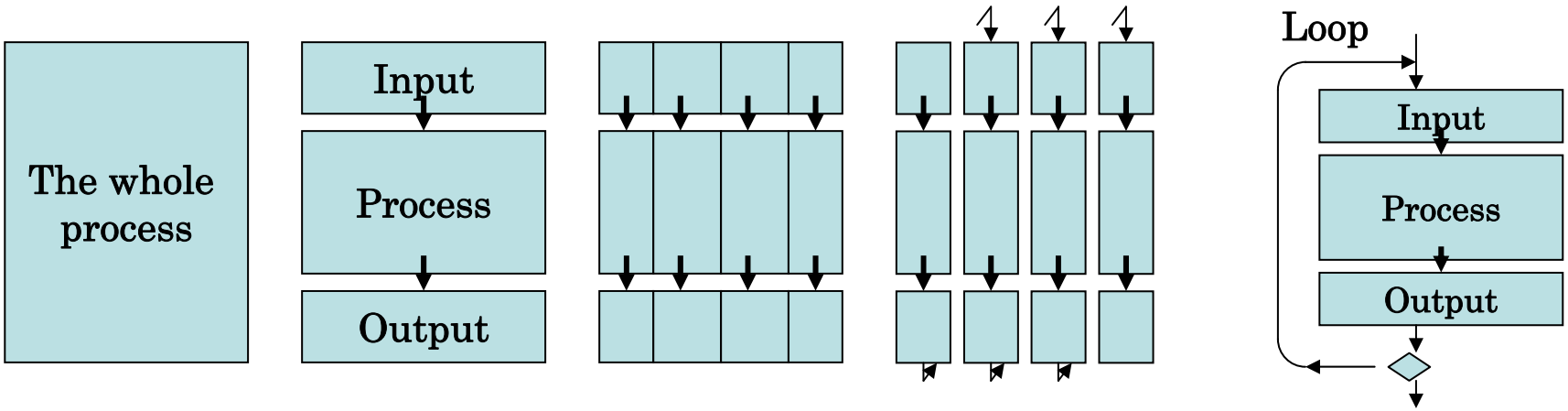
This partly corresponds to TRIZ Law of System Completeness
and the concept of hierarchy of systems;
but it states more clearly.

(TRIZ should learn more in this context.
E.g., TRIZ Principles do not have a hierachical structure.)

Another Criterion of good Stepwise Refinement *(by Shigo)*

"To construct a (software) system
so as to be able to refine each part of it individually."

A 'poor' case of practice (by Shigo):



TRIZ View: Reasonable case of refinement in 'Another Dimension'

'Refining instructions individually' is just a first approximation.

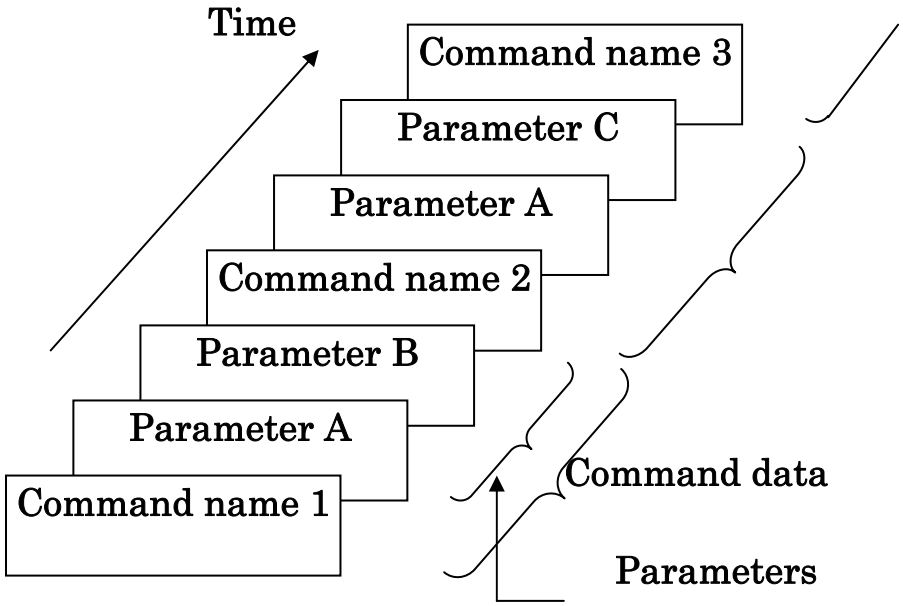
Software engineers should better be prepared to
introduce 'Another Dimension' (or a different aspect of view)
in the Step-wise Refinement.

The Jackson Method (or Jackson Structured Programming):

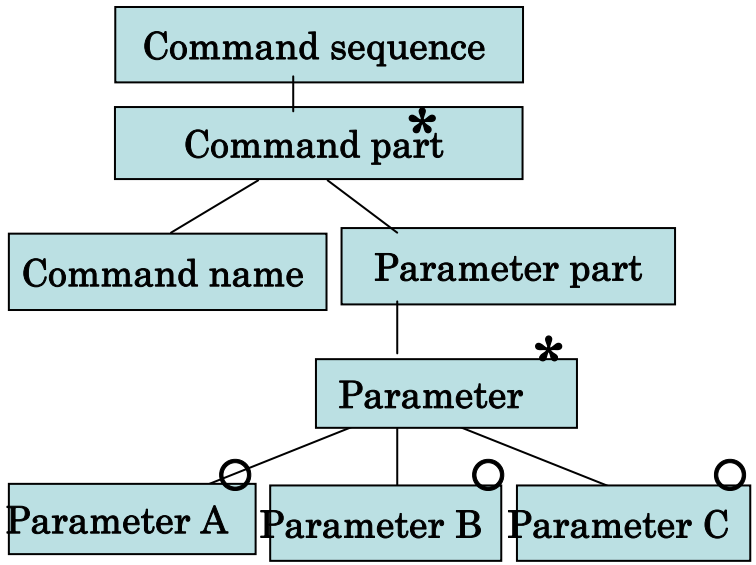
for constructing software procedures for business applications.

(1) Clarify the data structures of the input and of the output.
(2) Construct a program structure
so as to reflect the data structures of input and output data.

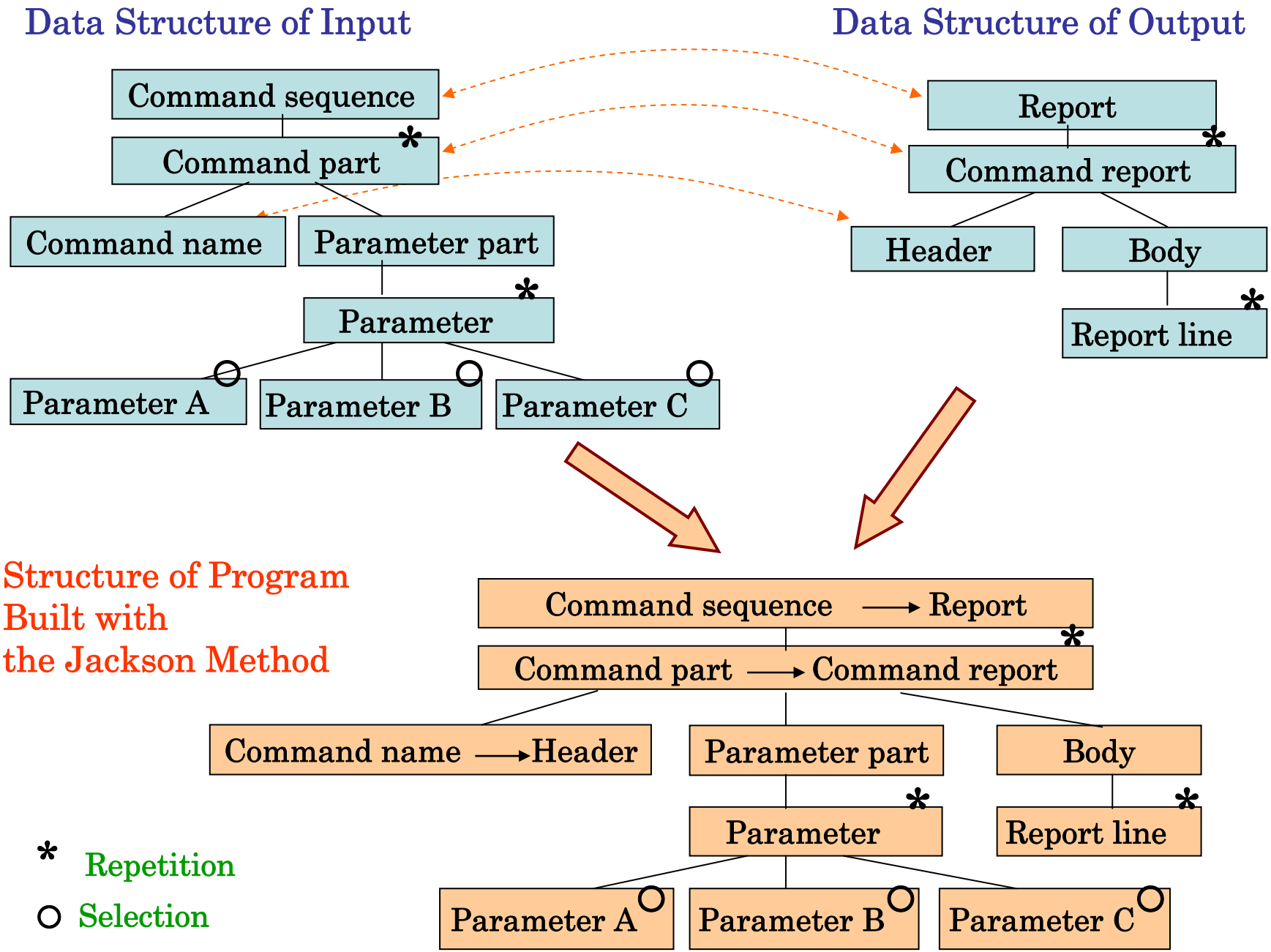
Shigo's Example: Reporting the Command Inputs



Data Structure of the Input

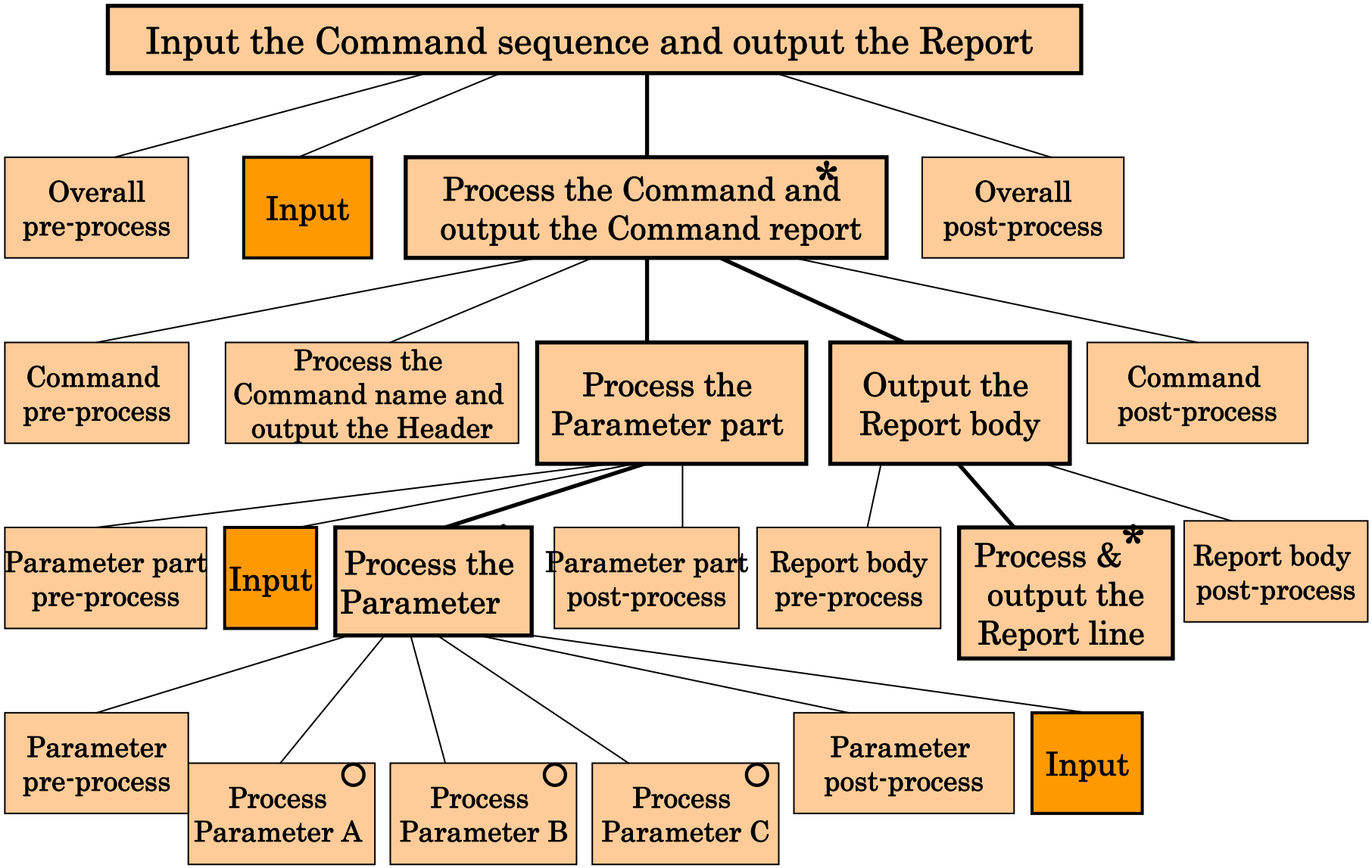


* Repetition ○ Selection


Data Structure of Input
Data Structure of Output
Command sequence
Command part
*
Command name
Parameter part
Parameter
*
Parameter A
Parameter B
Parameter C
Report
Command report
*
Header
Body
Report line
*
Structure of Program
Built with
the Jackson Method
Command sequence → Report
*
Command part → Command report
Command name → Header
Parameter part
Body
Parameter
*
Report line
*
Parameter A
Parameter B
Parameter C
* Repetition
Selection

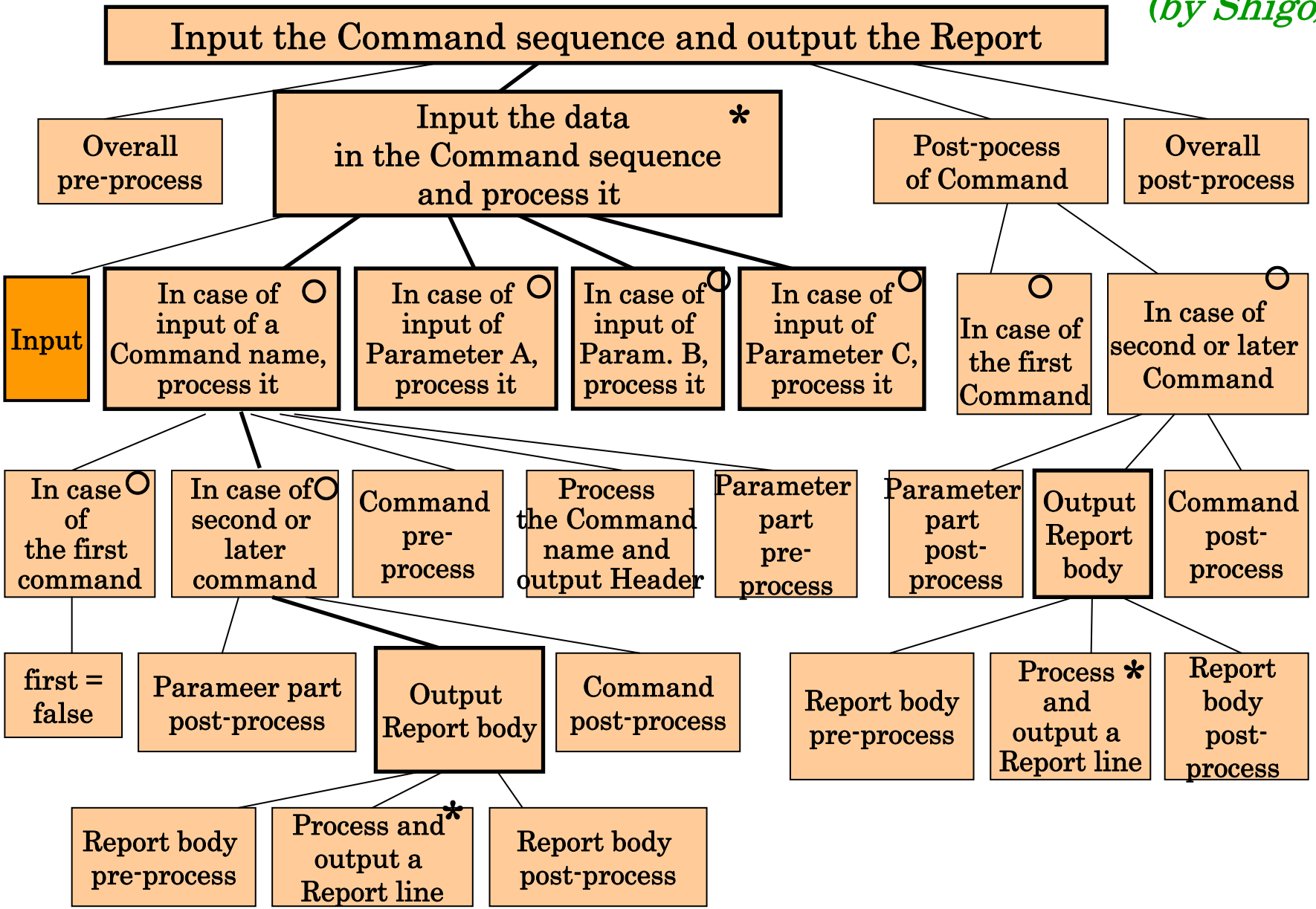
Structure of Program Built with the Jackson Method

(by Shigo)

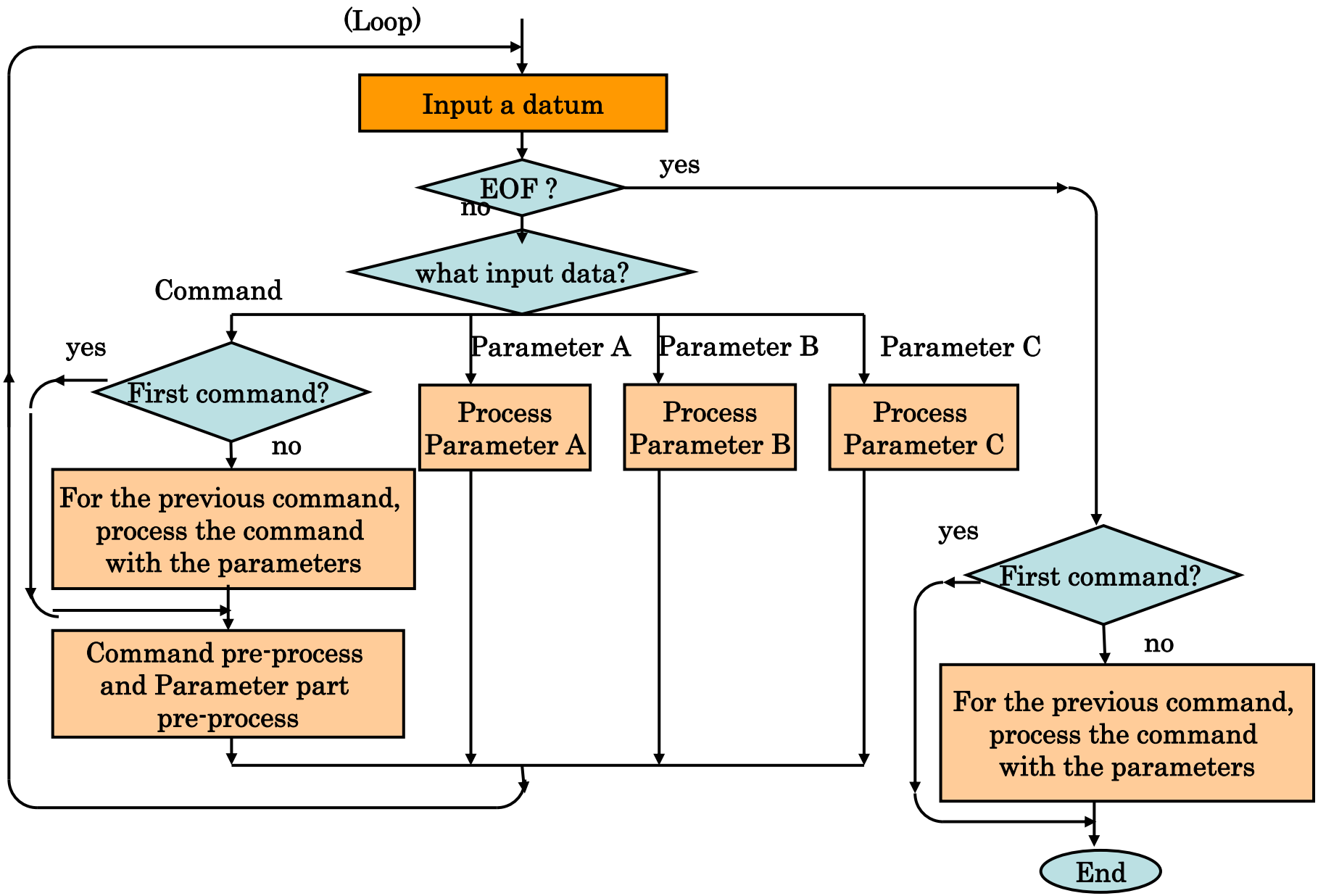


Structure of Program Built with Conventional Non-Jackson Method

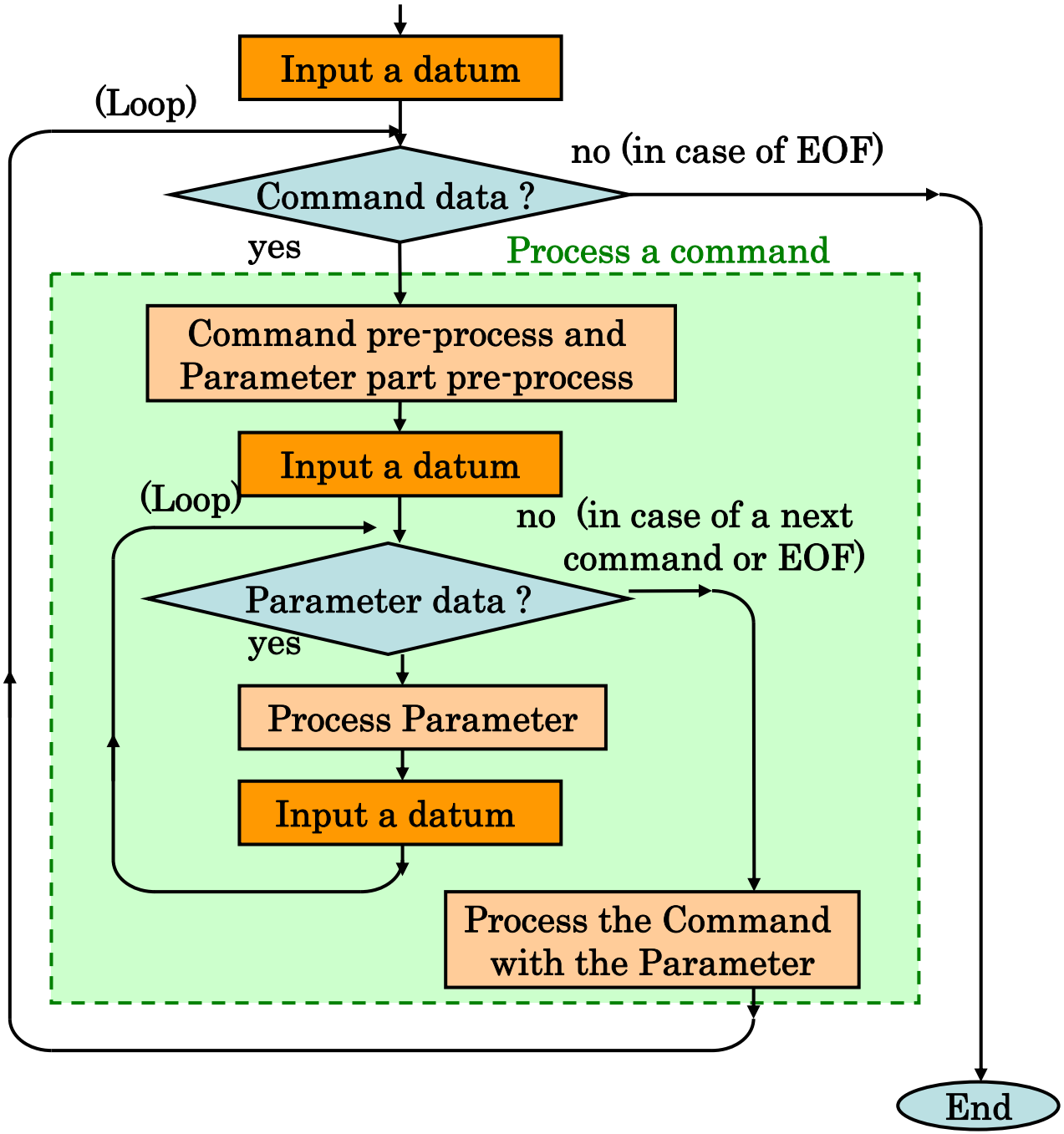
(by Shigo)



Flowchart of Program Built with Conventional Non-Jackson Method

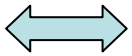


Flowchart of Program Built with the Jackson Method



Philosophy in the Jackson Method

(in SE) "to build a software module
so as to reflect the structure of its input and output"



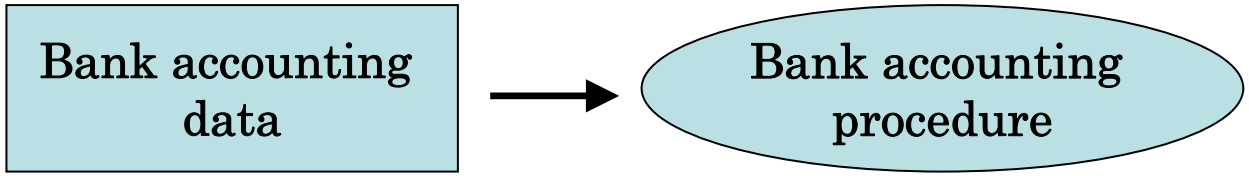
(in TRIZ) "to build the structure of a technical system
so as to reflect the structures of the materials and information
which are to be handled with and produced by it"

Relevant to TRIZ principles, e.g.,

- 9-Windows method
- Asymmetry (Pr. #4)
- Flexible Shell and Membrane (Pr. #30)
- Parameter Change (Pr.#35)
- Composite Materials (Pr. #40)
- Selective Introduction of substances (Inventive Standards)
- Flexible Systems (Inventive Standards Da5)

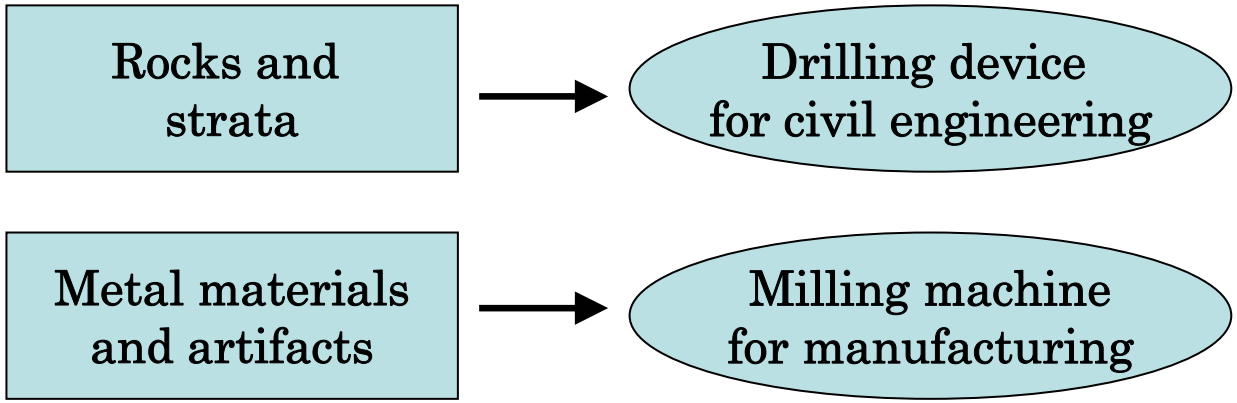
However, the matching in software technologies is better and deeper:

Software: "to build a software module
so as to reflect the structure of its input and output"



Hardware:

"to build the structure of a technical system
so as to reflect the structures of the materials and information
which are to be handled with and produced by it"



Desciplines concerning the inputs/outputs of a system
are often very different from that of the system itself. ➡Difficulty

Feedbacks to TRIZ from the Jackson Method

We should pay more attention to
'the Objects we are going to handle with the System'
in contrast to 'the component Objects of (i.e. within) the System'.

(in TRIZ)

Sub-principle "Introducing the Structures of Objects":
"The Objects to be processed by the technical system
should be examined closely in their structures,
especially from the aspects of spatial, temporal, causal,
and logical structures; and
the structues of the Objects should be introduced
in designing the structures of technical systems."

(This should be a sub-pinciple of
the Laws of Completeness of Technical Systems.)

Concluding Remarks

The concept of Step-wise Refinement is important
in analyzing problems and in designing (software) systems.

The concept of data structures of inputs/outputs is useful
in designing the structure of processing systems.

These concepts in SE/CS are in harmony with TRIZ principles
and have been developed in a more detailed and logical way
than in TRIZ and hard-technologies.

TRIZ can learn a lot from basic concepts in SE/CS,
probably more than TRIZ can provide to SE/CS.

TRIZ has contributed in some extent to SE/CS, e.g.,
in the cases of 'Refinement in Another Dimension'
to make the Step-wise Refinement more flexible.

The present approach is slow but steady
to explore the ways of applying TRIZ to software development.