Principle of Lyee Software

Fumio Negoro
The Institute of Computer Based Software Methodology and Technology, Tokyo, Japan
E-mail: f-negoro@lyee.co.jp

Abstract— **A methodology that we have developed enables us to produce source programs directly from an intention. The methodology is based on philosophical hypotheses and transforms them into a technical formula. Source programs generated with this methodology and those generated with conventional methodologies are different in their structures and the approach for producing the programs. The advantageous characteristics of this methodology help solve the current software problems.**

Keywords— **Philosophical hypotheses, technical formula, non-physical world, consciousness model, equivalent atom, units, critical state, scenario function, pallets, word, signification vectors, pallet control function, pallet function, and harmonization formula.**

I. Introduction

THE source program should express a user's inherently ambiguous intention as it is in a programming language. A new methodology to fulfill this purpose has been created and is presented in this paper. The methodology is based on philosophical hypotheses and transforms them into technical formula. The formula can be written in any general programming language. Lyee's[1] source programs are determined by the formula.

At this juncture, let us briefly explain why we incorporate philosophical thinking into software methodology. Scientific and engineering perspectives based on the premise that we could share the same cognition about various phenomena have been developed to elucidate the physical world. Concerning the non-physical world, however, we have yet to fully develop such perspectives.

Since software is significantly influenced by an ambiguous intention, an approach for development of software is the same as that for comprehension of the non-physical world. In order to comprehend the non-physical world, we have to take a philosophical thinking and construct hypotheses. There should not be contradictions among the hypotheses.

This methodology presented here resolves serious problems surrounding software. It should be also noted that this methodology is being used in actual business fields and has achieved high performance[2].

[1]Lyee stands for the governmentaL methodologY for softwarE providencE. This methodology was patented in the U.S., Singapore, and New Zealand as of September, 2000, and its applications for a patent are under examination in twenty-three other nations.

[2]In Japan, our methodology for software development is actually being implemented for two large-scale systems and two middle-sized such as Catena corporation's sales management system and National Mutual Insurance Federation of Agriculture Cooperative' operational management system for 401K.

II. Basic Idea of the Model

The mental process of establishing cognition is considered to compare two entities and then to clarify the difference between them. In our intrinsic process of cognition, we grasp an entity using another entity and a functional relationship between the two entities. However, when we try to comprehend non-physical existence, our intrinsic process using a functional relationship is not helpful.

While there is an intrinsic disadvantage of our mental process, a model expressing an intention in a programming language is constructed to clarify two non-physical entities without using a functional relationship between them. This model is called the consciousness model and constructed based on hypotheses. A basic idea of the model is shown in Figure 1.

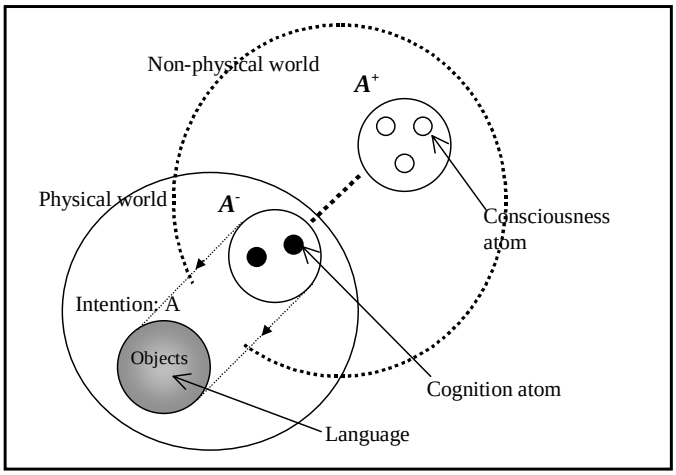

Fig. 1. Basic Idea of the Consciousness Model

The hypotheses are formed on a basis of the idea that there might be some cause that determines the intention. The cause is a process of establishing cognition. This process is constructed in the model using a pair of sets. This pair is defined to appear only one at a time. Consequently, there is no cause-and-effect relation between one pair and another.

One set of the pair is denoted as A^+ and the other is A^-. An element of these sets is undividable and ultimate entity called a logical atom. The logical atom represents *being* with its conceptual space. An element of A^+ is called an consciousness atom, while an element of A^- is called a cognition atom.

When A^- appears in the physical world, A^- is transformed into an object with already existing memories. An

object herein means something which can be expressed in natural or programming languages, and the languages themselves. Only an objectified A^- remains in our memories. Once A^- is objectified, the pair of A^+ and A^- disappears.

In our habitual way of cognition, source programs are naturally made out of specifications. A cause-and-effect relationship between specifications and source programs is intentionally made up.

On the contrary, the hypotheses of our model do not form such a cause-and-effect relationship. This is because an object of specifications is a natural language and that of a source program is a programming language, and the natural and programming languages are not the same entities. In addition, a pair of A^+ and A^- cannot produce different objects simultaneously. Accordingly, there is no a cause-and-effect relationship between the specifications and the source programs at all. This is a key idea to understand this methodology.

The hypotheses using logical atoms need to be further developed so as to construct a realistic model that can be applied to non-physical existence. The advanced hypotheses are presented as follows:

A. Consciousness unit

Figure 2 shows the structure of a consciousness unit. The left-hand circle is a subset of the set of logical atoms. This subset is called a dense structure. The right-hand circle is one logical atom which is equivalent to the left-hand set and called an equivalent atom. The equivalent atom belongs to the dense structure. This relationship is called a consciousness unit. All the logical atoms to establish the consciousness unit are called consciousness atoms.

Being equivalent means that the size of a conceptual space of one equivalent logical atom is approximately the same as the size of its dense structure.

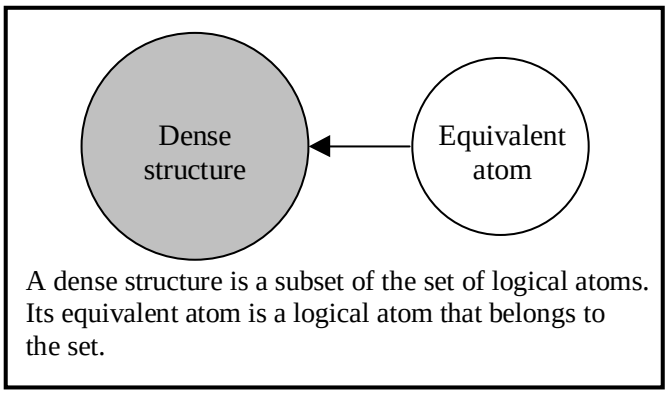

Fig. 2. Consciousness Unit

B. Cognition unit

Figure 3 shows the structure of a cognition unit. The left-hand circle is a subset of the set of logical atoms. This subset is also called a dense structure. The right-hand circle is one logical atom which is equivalent to the left-hand set and called an equivalent atom. The equivalent atom does not belong to the dense structure. This relationship is called a cognition unit. All the logical atoms to establish the cognition unit are called cognition atoms.

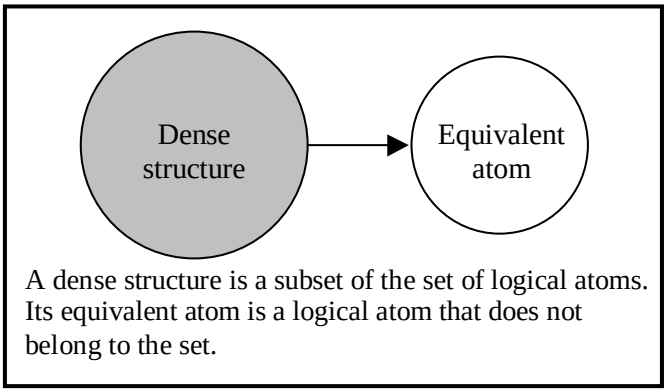

Fig. 3. Cognition Unit

C. Critical state

A certain type of cognition unit makes an operation called coupling. Coupling is a process that two cognition units are likely to generate another cognition unit. Coupling is supposed to continue. As a result, the number of cognition atoms which belong to the dense structure of the cognition unit does increase. Finally, that dense structure contains all the cognition atoms so that it has to take a boundary atom as its equivalent atom. The process from the first coupling to the coupling that takes a boundary atom is called a coupling history.

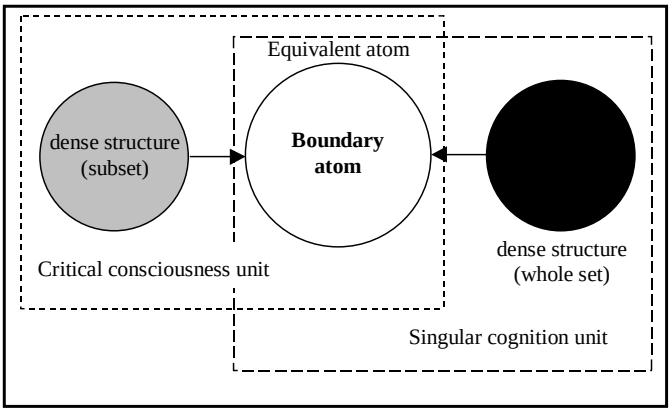

Fig. 4. Critical State

The boundary atom is a neutral logical atom between the consciousness atoms and the cognition atoms. The cognition unit taking the boundary atom as its equivalent atom is called a singular cognition unit, and the consciousness unit taking the boundary atom as its equivalent atom is called a critical consciousness unit. The critical consciousness unit and the singular cognition unit are bridged via the boundary atom as their equivalent atom. This state is

called a critical state. A^+ and A^- are determined by the critical state. This is shown as Figure 4.

D. Whole set and subset

Cognition in the physical world is established when two subsets are established. When one of the subsets becomes a whole set, cognition cannot be established. Therefore, a hypothetical world is constructed where a whole set is never formed. In our model, the physical world corresponds to A^- whereas the hypothetical world corresponds to A^+.

By forming the critical state between two different worlds, a mechanism that establishes cognition is constructed.

III. Architecture of the Model

In order to construct a model that enables us to express an intention as it is in a programming language, the notions described in Figures 1, 2, 3 and 4 are further developed and formulated into rules as follows: There are four types of cognition units. Based on their process of formation, these types of units are named as initial units, actual units, power units and natural units. Formation of the units generates relationships among different types of units as shown in Figure 5.

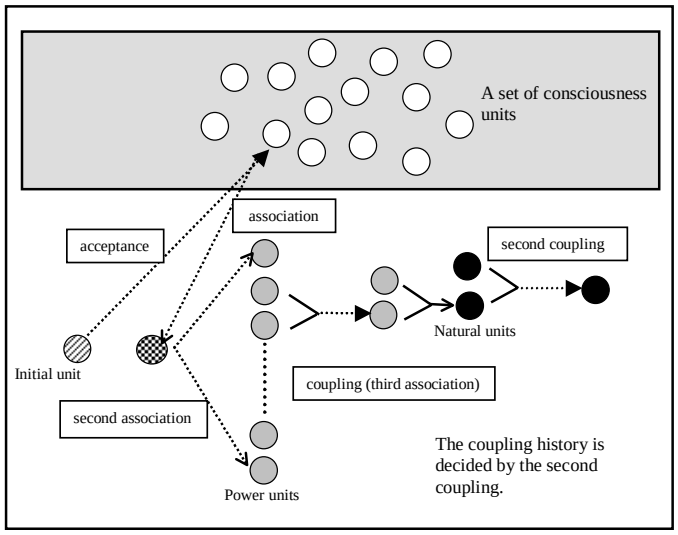

Fig. 5. Relationships among Units

A. Relationships among units

An initial unit is one that is created first among the four types of units. An initial unit forms a relationship with a consciousness unit, which is called acceptance. The consciousness unit having a relationship of acceptance with an initial unit generates a new cognition unit. This newly generated unit is called an actual unit, and generation of the actual unit is called association. The actual unit generates a plural number of cognition units called power units. Generation of power units by the actual unit is called the second association. The power units further generates power units by an operation of coupling.

Coupling interminably continues. When a generated power unit meets a certain condition, that particular power unit becomes a natural unit. Then, natural units continue coupling, which is specifically called the second coupling. The units are generated in this manner and the number of units increases.

A set of equivalent atoms of the consciousness units is called the consciousness space. A set of equivalent atoms of the initial, actual, power and natural units are respectively called, the initial space, the actual space, the power space and the natural space.

The consciousness space and the natural space are related to each other through the initial, actual and power spaces. Thus, the critical state between the consciousness space and the natural space is formed using the aforementioned relationships. A^+ is represented as the consciousness space, and A^- is represented as the natural space.

B. Scheme of establishing cognition

Figure 6 is a scheme of establishing cognition. This scheme shows a state at one moment. One moment later a new state is created.

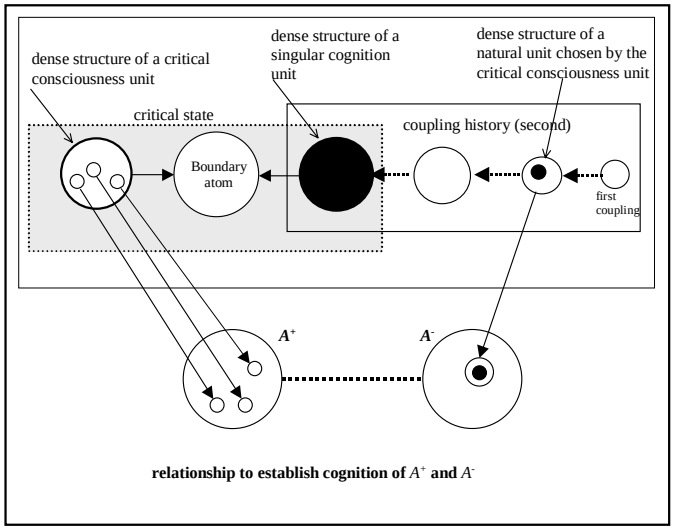

Fig. 6. Scheme of Establishing Cognition

A^+ and A^- reflect the state at one moment of the critical state, the history of coupling and spaces. Accordingly, A^+ and A^- momentarily exist.

Elements of A^- are cognition atoms that are chosen from the history of coupling that has reached the critical state, provided the number of cognition atoms belonging to the dense structure of the cognition unit in the history of coupling agrees with the number of consciousness atoms belonging to the dense structure of the critical consciousness unit.

C. Summary of the consciousness model

Based on the rules constructed above, the concept described in Figure 1 is re-described as in Figure 7. The rules discussed here are used to construct another model that

transform the consciousness model into realization with a programming language.

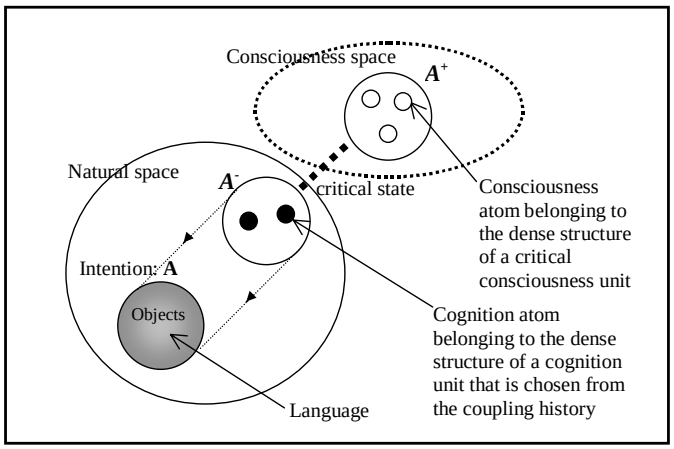

Fig. 7. Concept of the Consciousness Model

IV. Realization of the Consciousness Model

An intention is comprehended as a relationship among five hypothetical spaces. Expressing the relationship in a programming language is realization of the consciousness model.

Three spaces of the initial space, the actual space and the power space are replaced by axes in the realization model. In addition, the consciousness space and the natural space are functionally determined in a relation to the coordinates on the three axes. This model is shown in the Figure 8 and called the three-dimension-like space model.

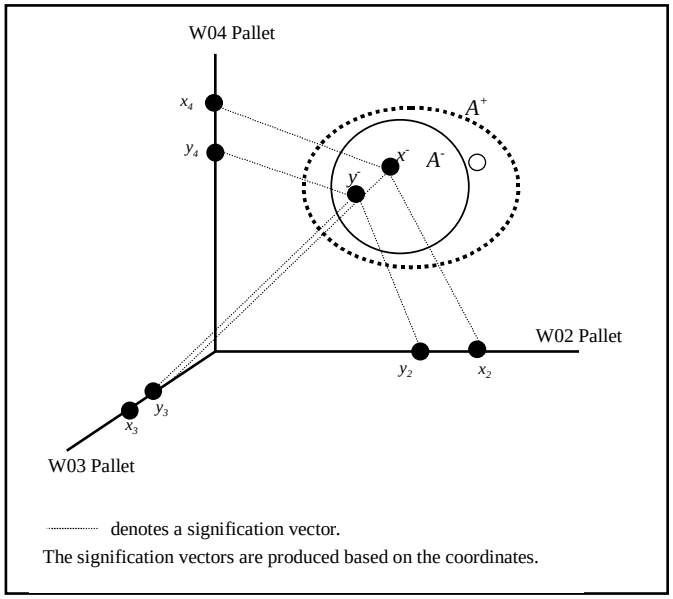

Fig. 8. Three-dimension-like Space Model

A. Scenario function

The three-dimension-like space model is formulated in order to realize it. The formula is called the scenario function and denoted as $T_U{}^1$, whose components are shown in Figure 9. The components are written in a programming language.

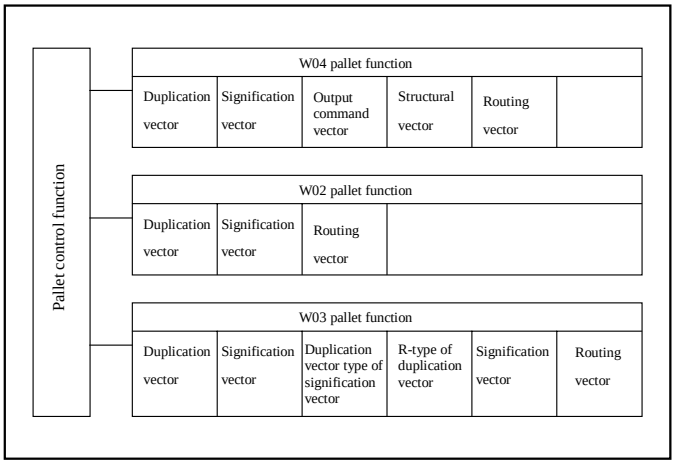

Fig. 9. Scenario Function

Notwithstanding that an intention is always an ambiguous being, software should express the intention as it is in a programming language without using specifications. If such software is realized, the environment related to software is improved. The scenario function actually realizes it as a formula.

B. Pallets

The axes in the three-dimension-like space model are called pallets for realization. More specifically, the initial, actual and power spaces are defined as $W02$, $W03$, and $W04$ pallets. In other words, a pallet is a set of coordinates that are cognition atoms existing as equivalent atoms. The equivalent atoms are elements that form the spaces.

The pallets consist of signification vectors, duplicate vectors, other vectors and data areas to determine A^-.

C. Words

An objectified intention is considered to be a set of words. Based on our cognition, the reason why the set of words is formed is explained with empirical knowledge.

On the contrary, based on the hypotheses presented here, the set is formed with A^+ and A^-, which is not necessary to explain.

With the hypotheses, a word is treated as follows: A word is consisted of a cause of meaning and its identifier. The identifier corresponds to an equivalent atom, and the cause of meaning corresponds to a logical atom belonging to a dense structure having the equivalent atom. Thus, a word has the identical structure with that of a unit.

D. Signification vectors

A signification vector is an element for establishing cognition and made for each word that reflects an intention. A coordinate on an axis in the three-dimension-like space model is given to each word by the rules derived from the hypotheses.

The coordinate on the axis of a word corresponds to a cognition atom being an equivalent atom. The cognition atom should be one that belongs to a dense structure of the cognition unit in the coupling history. Furthermore, the cognition unit must correspond to the critical consciousness unit. The correspondence between a word and a cognition atom is described as the above in the consciousness model.

Whereas a cognition atom belonging to A^- is also considered to be a word, its coordinate is defined as a functional relationship among three coordinates on each axis.

An identifier of a word is given to the coordinates on an axis, and a cause of meaning is expressed as a coordinate defined by functional relationship. This functional relationship is expressed as the signification vector and shown in Figure 10.

The signification vector is a key concept of this methodology. Detailed explanation on it is out of scope of this paper.

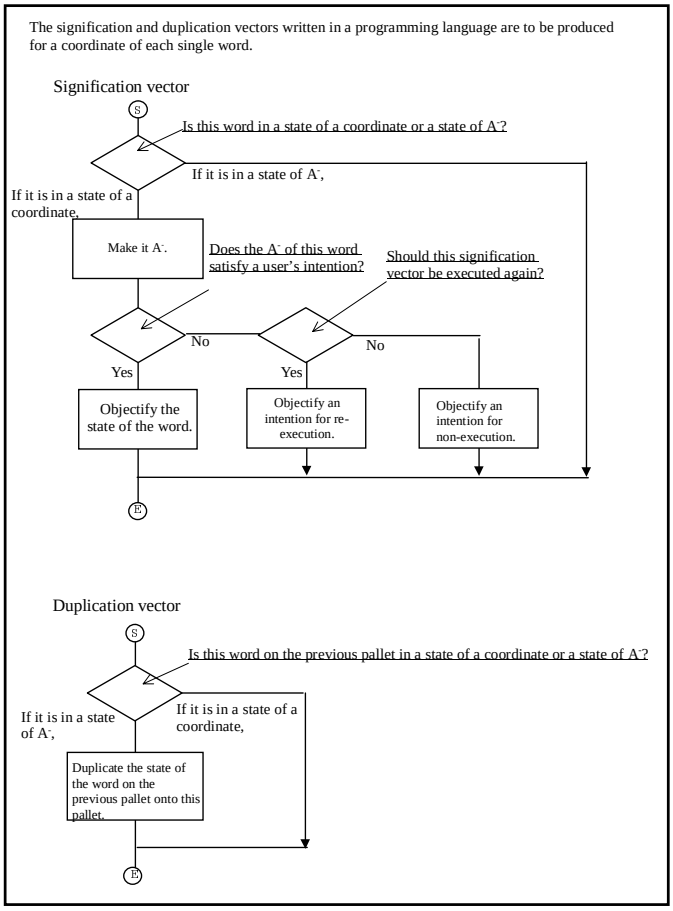

Fig. 10. Structures of Vectors

E. Duplication vectors

The five spaces are being synchronized, and this state is to be realized with the three kinds of pallets. This vector generates the synchronous state by handing over the state of $W02$ to $W03$ and the state of $W03$ to $W04$. This vector is shown at the bottom of Figure 10. The state of $W04$ is not carried over to $W02$ due to the non-synchronous relationship between the initial and power spaces.

F. Pallet control function and pallet function

A triple of $W04$, $W02$, and $W03$ pallets is called a base structure. They are executed on a computer in a linear order of $W04$, $W02$, and $W03$. Termination of execution is judged in $W02$, and when execution of another base structure is required, it is done via $W03$. A program to control execution among pallets is called the pallet control function, which is denoted as Φ.

A program that controls pallet itself is called a pallet function. As the pallet function holds controlling rules, each pallet function is denoted as ${\Phi_P}^4$, ${\Phi_P}^2$ and ${\Phi_P}^3$. Based on the above, the scenario function is expressed as follows:

$$ {T_U}^1{}_{,1} = \Phi\left({\Phi_P}^2(W02) + {\Phi_P}^3(W03) + {\Phi_P}^4(W04)\right) $$

When a plural number of base structures are connected, the scenario function is expressed as follows:

$$ {T_U}^1{}_{,i} = \Phi\Sigma_i\left({\Phi_P}^2(W02) + {\Phi_P}^3(W03) + {\Phi_P}^4(W04)\right) $$

where i is the number of the base structures.

G. Other vectors

The route vector is provided to each pallet and determines which pallet should be executed next. The structural vector is provided to $W04$ pallet and initializes data area. The command vector is provided to $W02$ for receiving data, and that provided to $W04$ for outputting it. See Figure 10.

V. Harmonization Formula

The relationship that establishes ${T_U}^1$ in an intention A is formulated as follows:

$$ \left({T_U}^1 \rightarrow A\right) = {T_A}^1 $$

This formula is called the harmonization formula. ${T_A}^1$ is the redefined ${T_U}^1$ using a set of words reflecting the intention A.

In this methodology, developing software means to complete the harmonization formula. This resolves serious problems that occur in the conventional software development methodologies.

The state of program of ${T_A}^1$ is different from that of conventional programs. When ${T_A}^1$ is executed on a computer, A^+ and A^- are constructed in the computer. Conventional programs to satisfy the intention A is denoted as P_A. P_A and ${T_A}^1$ are completely different in their structures and the way of thinking to develop the programs. The difference

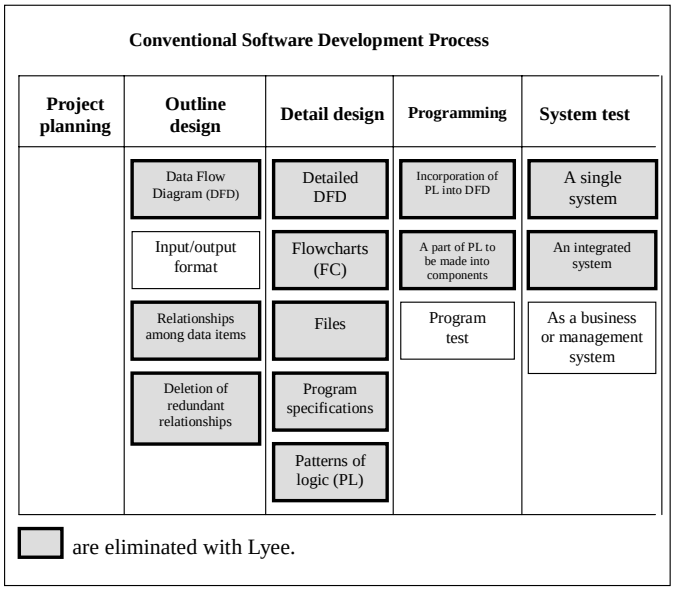

Fig. 11. Software Development Process.

of development process between the conventional and this methodologies is shown in Figure 11.

${T_A}^1$ expresses a static state of an intention A whereas P_A does not. The execution of ${T_A}^1$ means that ${T_A}^1$ is rewritten by a computer. This rewritten state corresponds to iteration of ${T_A}^1$. Therefore, this state is denoted by ${T_A}^n$. As a result of execution, both P_A and ${T_A}^n$ become equivalent.

VI. Conclusion

This methodology produces source programs which could express a user's intention as it is in a programming language with the scenario function and the harmonization formula. Our approach is a theory-oriented methodology.

This methodology resolves the following problems we often encounter in the conventional methodologies:

- Spaghetti programs
- Tests for verification
- Excessive documentation
- Various programming languages
- Unreasonable standardization

The methodology simplifies the designing process. At the same time, it determines source programs at the highest level of automation. It also guarantees a short period of time for software development and a high maintainability.

Proceedings of the 2000 International Conference on Information Society in the 21st Century (IS2000). (Aizu-Wakamatsu, Fukushima, Japan, Nov., 2000), pp.441-446.