ソフトウェア工学　80－13
（１９９１．７．１９）

ソフトウェア構成的変化の理論的分類

大森　晃
（株）富士通研究所
国際情報社会科学研究所

青山　幹雄
富士通（株）複合交換機事業部

概　要

ソフトウェアの構成的側面およびその変化を記述し管理するために，ソフトウェア構成管理（ＳＣＭ）の方法論や環境が研究・適用されてきた．本論文では，ＳＣＭの管理対象であるソフトウェア構成的変化そのものについて著者らがすでに展開した理論を概観し，それをソフトウェア構成的変化の分類に適用した．変化のカテゴリーとして1,024の候補があることを論じ，それらを理論的に分析して，理論的に許容できるカテゴリーが80であることを示した．また，それらを実際的な観点から吟味して，理論的にも実際的にも意味のある46のカテゴリーを導いた．これにより，ソフトウェア構成的変化についての見通しをより良いものとした．

A Theoretical Taxonomy of Software Architectural Changes

Akira OHMORI * and Mikio AOYAMA **

* International Institute for Advanced Study of Social Information Science
FUJITSU LABORATORIES LTD., 17-25, Shinkamata 1-Chome, Ota-ku, Tokyo 144, JAPAN.
**Telecommunications Software Division
FUJITSU LIMITED, 629 Shimo-Kodanaka, Nakahara-ku, Kawasaki 211, JAPAN.

Abstract

Methodologies and environments of software configuration management (SCM) have been studied and employed to describe and manage software architectural aspects and their changes. This paper reviews the theory that we have already developed on software architectural changes themselves to be managed by SCM, and applies it to a taxonomical study on software architectural changes. There are 1,024 categorical candidates of changes. Those candidates are analyzed theoretically, and it is shown that there are 80 theoretically acceptable categories. Furthermore, they are reviewed from a practical point of view. In the result, 46 theoretically and practically acceptable categories are made explicit. This leads to a better position where we have an insight into software architectural changes.

1. Introduction

A software system architecturally changes or evolves during development and maintenance. Software configuration management (SCM) addresses such changes. Conventional studies focus on methodologies and environments of SCM [Bers80, Nara87, Yau 87, Baze85, Rend88, Estu89].

The analysis of software architectural changes to be managed has not been treated explicitly so far. We are not placed in a position where we can penetrate into software architectural changes. The theory of SCM is also still in its infancy, and the management methods tend to be ad hoc.

Furthermore, software systems tend to form a family where relationships among software entities are much more complicated, and tend to evolve as a family. A sound and penetrating theoretical framework of SCM becomes increasingly necessary to manage an evolving family of software systems. As a step toward such a framework, it is indispensable to have an insight into software architectural changes themselves.

A theory of software architectural changes was developed by the authors[Ohmo90]. In the theory, a software system configuration is formalized; its architectural changes are analyzed in a set-theoretical manner; and some useful results on software architectural changes are presented.

In this paper, the previous work is briefly reviewed, and the theory is applied to a taxonomical study on software architectural changes. It is shown that categorical complexity of changes can be drastically reduced. A taxonomical table that includes theoretically and practically acceptable categories of changes is presented.

2. Software system configurations

2.1 Entities and relationships

A software system configuration can be described by means of software architectural entities and relationships among them [McCa75, Bers80, Nara87]. A software architectural entity, called an entity in what follows, is a functional unit in a software system.

There are several types of entities, such as modules, programs and subsystems. We can recognize, among others, an entity that uses other entities to implement its function and can be regarded as a delivery unit to a customer. Such an entity is called a system entity.

A system entity usually differs from a software system itself. A software system is a composite of a system entity and other entities. In other words, it contains a system entity as one of the integral parts.

An entity is observed from a macroscopic viewpoint in this paper. Any entity e is expressed as a pair of its name n and its body b, that is, e = (n, b). If an entity e = (n, b) is a program, n and e respectively correspond to its name and the set of its statements.

It is assumed that there is a universal set of entities, which is denoted by E. The name of a system entity might be different from the name of a software system that contains the system entity. In this paper, however, the name of a system entity is regarded as that of a software system, and vice versa.

We can observe it in a software system that an entity uses other entities to implement its function, or is used by other entities to implement their functions. From this observation, a binary relation USE in the set E is introduced as follows: $(e, e') \in USE \subseteq E \times E$[1] if and only if an entity e uses an entity e' in order to implement its function. For a USE relation, we assume that $(e, e) \notin USE$ for any $e \in E$.

2.2 Configurations

Let T be a time set and $t \in T$ be arbitrary; s any software system name. Let $E(s; t) \subseteq E$ be the set of entities that are employed at time t to configure a specific software system named s. Let USE/E(s; t) be the relation induced by USE in E(s; t), that is, $USE/E(s; t) = USE \cap (E(s; t) \times E(s; t))$.

Then, if $SC(s; t) = \langle E(s; t); USE/E(s; t)\rangle$ satisfies the following conditions, it is called a system configuration for s at time t.

(a) E(s; t) contains one, and only one system entity named s.

(b) E(s; t) is a finite set.

(c) For any (n, b) and (n', b') $\in$ E(s; t), if b = b', n = n'.

(d) For any (n, b) and (n', b') $\in$ E(s; t), if n = n', b = b'.

(e) If $Card(E(s; t)) \geq 2$, for any entity $e \in E(s; t)$, there exits an entity $e' \in E(s; t)$ such that $(e, e') \in USE$ or $(e', e) \in USE$, where Card(E(s; t)) is the cardinal number of E(s; t), that is, the number of entities in E(s; t).

A software system configuration is represented by two parts. One is an elemental part E(s; t), and the other is a structural part USE/E(s; t).

3. A theory of software architectural changes

For a system configuration $SC(s; t) = \langle E(s; t); USE/E(s; t)\rangle$, we have some theoretical results on its architectural changes [Ohmo90]. To show them, let us introduce the following notations.

$NAME(SC(s; t)) = \{ n \mid (n, b) \in E(s; t) \}$.

$BODY(SC(s; t)) = \{ b \mid (n, b) \in E(s; t) \}$.

N-struct(SC(s; t))

$= \{(n, n') \mid ((n, b), (n', b')) \in USE/E(s; t) \}$.

[1] The expression 'X $\subseteq$ Y' means that X is a subset of Y; equality is not excluded.

B-struct(SC(s; t))

= $\{(b, b') \mid ((n, b), (n', b')) \in USE/E(s; t)\}$.

Prop. 1

For any system configuration SC(s; t) = <E(s; t); USE/E(s; t)>, Card(E(s; t)) = Card(NAME(SC(s; t)) = Card(BODY(SC(s; t)).†

Prop. 2

Let SC(s; t) = <E(s; t); USE/E(s; t)> be an arbitrary system configuration. If Card(E(s; t)) = 1, USE /E(s; t) = ∅ and therefore N-struct(SC(s; t)) = ∅ and B-struct (SC(s; t)) = ∅.†

Prop. 3

Let SC(s; t) and SC(s'; t') be arbitrary system configurations. If E(s; t) ≠ E(s'; t') and USE/E(s; t) = USE/E(s'; t'), Card(E(s; t)) = 1 and Card(E(s'; t')) = 1.†

Prop. 4

Let SC(s; t) and SC(s'; t') be arbitrary system configurations. If E(s; t) ≠ E(s'; t'), USE/E(s; t) ≠ USE /E(s'; t') and Card(E(s; t)) = Card(E(s'; t')), Card (E(s; t)) ≥ 2 and Card(E(s'; t')) ≥ 2.†

Prop. 5

Let SC(s; t) and SC(s'; t') be arbitrary system configurations; and E(s; t) = E(s'; t'). Then, the following hold.

(a) Card(E(s; t)) = Card(E(s'; t')).

(b) NAME(SC(s; t)) = NAME(SC(s'; t')).

(c) BODY(SC(s; t)) = BODY(SC(s'; t')).

(d) USE/E(s; t) = USE/E(s'; t').

(e) N-struct(SC(s; t)) = N-struct(SC(s'; t')).

(f) B-struct(SC(s; t)) = B-struct(SC(s'; t')).†

Prop. 6

Let SC(s; t) and SC(s'; t') be arbitrary system configurations. If E(s; t) ≠ E(s'; t') and USE/E(s; t) = USE/E(s'; t'), NAME(SC(s; t)) ≠ NAME(SC(s'; t')) or BODY(SC(s; t)) ≠ BODY(SC(s'; t')).†

Prop. 7

Let SC(s; t) and SC(s'; t') be arbitrary system configurations; Card(E(s; t)) ≥ 2; Card(E(s'; t')) ≥ 2; and N-struct(SC(s; t)) = N-struct(SC(s'; t')). Then, NAME(SC(s; t)) = NAME(SC(s'; t')).†

Prop. 8

Let SC(s; t) and SC(s'; t') be arbitrary system configurations; Card(E(s; t)) ≥ 2; Card(E(s'; t')) ≥ 2; and B-struct(SC(s; t)) = B-struct(SC(s'; t')). Then, BODY(SC(s; t)) = BODY(SC(s'; t')).†

Prop. 9

Let SC(s; t) and SC(s'; t') be arbitrary system configurations. If N-struct(SC(s; t)) = N-struct(SC(s'; t')), Card(E(s; t)) = Card(E(s'; t')).†

Prop. 10

Let SC(s; t) and SC(s'; t') be arbitrary system configurations. If B-struct(SC(s; t)) = B-struct(SC(s'; t')), Card(E(s; t)) = Card(E(s'; t')).†

Prop. 11

Let SC(s; t) and SC(s'; t') be arbitrary system configurations. Then, the following hold.

(a) If NAME(SC(s; t)) ∩ NAME(SC(s'; t')) = ∅, N-struct(SC(s; t)) ∩ N-struct(SC(s'; t')) = ∅.

(b) If BODY(SC(s; t)) ∩ BODY(SC(s'; t')) = ∅, B-struct(SC(s; t)) ∩ B-struct(SC(s'; t')) = ∅.†

Prop. 12

Let SC(s; t) and SC(s' t') be arbitrary system configurations; Card(E(s; t)) ≥ 2. Then, the following hold.

(a) If N-struct(SC(s; t)) ⊆ N-struct(SC(s'; t')), NAME (SC(s; t)) ⊆ NAME(SC(s'; t')).

(b) If B-struct(SC(s; t)) ⊆ B-struct(SC(s'; t')), BODY (SC(s; t)) ⊆ BODY(SC(s'; t')).†

4. Taxonomical viewpoints and categorical candidates

This section discusses what viewpoints can be used for a taxonomy of software architectural changes, and how many categorical candidates of changes we can encounter.

Let us consider the case where a system configuration SC(s; t) = <E(s; t); USE/E(s; t)> changes to a system configuration SC(s'; t') = <E(s'; t'); USE /E(s'; t')>. Then, we have four types of changes, including invariant. The first is that E(s; t) = E(s'; t') and USE/E(s; t) = USE/E(s'; t'). The second is that E(s; t) = E(s'; t') and USE/E(s; t) ≠USE/E(s'; t'). The third is that E(s; t) ≠ E(s'; t') and USE/E(s; t) =USE/E(s'; t'). The fourth is that E(s; t) ≠E(s'; t') and USE/E(s; t) ≠USE/E(s'; t').

The upper left part of Fig. 1 shows these types. The elemental parts E(s; t) and E(s'; t') have their nominal aspects and bodily aspects, and so do the structural parts USE/E(s; t) and USE/E(s'; t').

For each type of change, therefore, we can consider sixteen types of changes based on four cases, which are shown on the upper right part of Fig. 1. The first case is whether or not NAME(SC(s; t)) ≠ NAME(SC(s'; t')). The second is whether or not BODY(SC(s; t)) ≠ BODY(SC(s'; t')). The third is whether or not N-struct (SC(s; t)) ≠ N-struct(SC(s'; t')). The fourth is whether or not B-struct(SC(s; t)) ≠ B-struct(SC(s'; t')).

At this point, we have 64 categorical candidates of changes, including 60 candidates for which we can observe some change in at least one aspect. From more detailed observations, we have 16 candidates where only one aspect changes, 24 candidates for two aspects, 16 candidates for three aspects, and 4 candidates for four aspects.

In the above discussions, the difference between SC(s; t) and SC(s'; t') is denoted by a mere inequality. Such a difference can be considered more concretely.

Let us consider, for example, NAME(SC(s; t)) ≠ NAME(SC(s'; t')). We can classify this difference into three cases. The first is that NAME(SC(s; t)) ∩ NAME (SC(s'; t')) = ∅. The second is that NAME (SC(s; t)) ⊂ NAME(SC(s'; t'))[2] or NAME(SC(s'; t')) ⊂ NAME (SC(s; t)). The third is that ¬(NAME(SC(s; t)) ⊂ NAME(SC(s'; t'))), ¬(NAME(SC(s'; t')) ⊂ NAME (SC(s; t))) and NAME(SC(s; t)) ∩ NAME (SC (s'; t')) ≠ ∅. The expression ¬(a statement) means the negation of the statement.

For the other three aspects, their respective differences can be similarly classified into three cases, also, as is shown on the lower part of Fig. 1.

At this point, we have 1,024 categorical candidates of changes, which is derived from the algebraic expression $4\times(1\times3^0+4\times3^1+6\times3^2+4\times3^3+1\times3^4)$. Thus, we have a great number of candidates for software architectural changes even if we roughly observe software architectural changes. However, we can investigate whether or not each of those candidates is theoretically meaningful.

5. Reducing categorical complexity

This section analyzes the categorical candidates theoretically, and describes how the theory can be used for reducing the categorical complexity.

Analysis 1

(See the top-level candidate A and the middle-level candidate 1 in Fig. 1.)

Consider the case where E(s; t) = E(s'; t') and USE /E(s; t) = USE/E(s'; t'). From Prop. 5, we have that NAME(SC(s; t)) = NAME(SC(s'; t')), BODY(SC(s; t)) = BODY(SC(s'; t')), N-struct(SC(s; t)) = N-struct (SC(s'; t')) and B-struct(SC(s; t)) = B-struct(SC(s'; t')). Therefore, we can eliminate 255 out of 256 candidates.†

Analysis 2

(See the top-level candidate B in Fig. 1.)

Consider the case where E(s; t) = E(s'; t') and USE /E(s; t) ≠ USE/E(s'; t'). This type of change shows that the elemental part does not change but the structural part does. However, (d) of Prop. 5 does not allow such a type of change. Thus, all of 256 candidates are not feasible, and so they can be eliminated.†

Analysis 3

(See the top-level candidate C, the middle-level candidates 4, 5 and 11, and the bottom-level viewpoints (a) and (d) in Fig. 1.)

Consider the case where E(s; t) ≠ E(s'; t') and USE /E(s; t) = USE/E(s'; t'). From Props. 3 and 2, we have that N-struct(SC(s; t)) = N-struct(SC(s'; t')) and B-struct (SC(s; t)) = B-struct(SC(s'; t')). From Prop. 6, we have that NAME(SC(s; t)) ≠ NAME(SC(s'; t')) or BODY (SC(s; t)) ≠ BODY(SC(s'; t')). Therefore, we can eliminate 241 out of 256 candidates. For such a type of change, note that we address a system configuration that consists of only a system entity. From this fact, NAME (SC(s; t)) ≠ NAME(SC(s'; t')) implies NAME(SC(s; t)) ∩ NAME(SC(s'; t')) = ∅; and also, BODY(SC(s; t)) ≠ BODY(SC(s'; t')) implies BODY(SC(s; t)) ∩ BODY (SC(s'; t')) = ∅. Thus, we finally have 3 acceptable change categories, that is, we can eliminate 253 out of 256 candidates.†

Analysis 4

(See the top-level candidate D and the middle-level candidates 4, 5, 8, 9, 11, 14 and 15 in Fig. 1.)

Consider the case where E(s; t) ≠ E(s'; t') and USE /E(s; t) ≠ USE/E(s'; t'). From Props. 9, 4 and 7, we cannot allow any type of changes that satisfy both N-struct(SC(s; t)) = N-struct(SC(s'; t')) and NAME (SC(s; t)) ≠ NAME(SC(s'; t')). Similarly, from Props. 10, 4 and 8, we cannot allow any type of changes that satisfy both B-struct(SC(s; t)) = B-struct (SC(s'; t')) and BODY (SC(s; t)) ≠ BODY(SC(s'; t')). These results enable us to eliminate 87 out of 256 candidates.†

Analysis 5

(See the top-level candidate D, the middle-level candidate 7, and the bottom-level viewpoints from (d) to (f) and from (j) to (l) in Fig. 1.)

Consider the case where E(s; t) ≠ E(s'; t') and USE /E(s; t) ≠ USE/E(s'; t'). And let NAME(SC(s; t)) = NAME(SC(s'; t')); BODY(SC(s; t)) ≠ BODY(SC(s'; t')); N-struct(SC(s; t)) = N-struct(SC(s'; t')); and B-struct (SC(s; t)) ≠ B-struct(SC(s'; t')). In this case, we have 9 candidates, but 6 candidates can be eliminated as follows.

(1) From (b) of Prop. 11, we can eliminate 2 candidates.

(2) From Prop. 1, NAME(SC(s; t)) = NAME (SC(s'; t')) implies Card(BODY(SC(s; t)) = Card (BODY(SC(s'; t')). Therefore, neither BODY(SC(s; t)) ⊂ BODY (SC(s'; t')) nor BODY(SC(s'; t')) ⊂ BODY (SC(s; t)) is feasible. This enables us to eliminate 3 candidates.

(3) Let ¬(BODY(SC(s; t)) ⊂ BODY(SC(s'; t'))), ¬(BODY(SC(s'; t')) ⊂ BODY(SC(s; t))) and BODY (SC(s; t)) ∩ BODY(SC(s'; t')) ≠∅; B-struct(SC(s; t)) ⊂ B-struct(SC(s'; t')) or B-struct(SC(s'; t')) ⊂ B-struct (SC(s; t)). From (b) of Prop. 12, this situation is not feasible. Therefore, 1 candidate can be eliminated.†

[2] The expression 'X ⊂ Y' means that X is a proper subset of Y.

Top-level viewpoints and candidates

	E(s; t) ≠E(s'; t')	USE/E(s; t) ≠USE/E(s'; t')
A	No.	No.
B	No.	Yes.
C	Yes.	No.
D	Yes.	Yes.

Middle-level viewpoints and candidates for each top-level candidate

	NAME(SC(s; t)) ≠NAME(SC(s'; t'))	BODY(SC(s; t)) ≠BODY(SC(s'; t'))	N-struct(SC(s; t)) ≠N-struct(SC(s'; t'))	B-struct(SC(s; t)) ≠B-struct(SC(s'; t'))
1	No.	No.	No.	No.
2	No.	No.	No.	Yes.
3	No.	No.	Yes.	No.
4	No.	Yes.	No.	No.
5	Yes.	No.	No.	No.
6	No.	No.	Yes.	Yes.
7	No.	Yes.	No.	Yes.
8	No.	Yes.	Yes.	No.
9	Yes.	No.	No.	Yes.
10	Yes.	No.	Yes.	No.
11	Yes.	Yes.	No.	No.
12	No.	Yes.	Yes.	Yes.
13	Yes.	No.	Yes.	Yes.
14	Yes.	Yes.	No.	Yes.
15	Yes.	Yes.	Yes.	No.
16	Yes.	Yes.	Yes.	Yes.

(Note 1)
'Yes' means the right part, such as E(s'; t'), is different from the left part, such as E(s; t); on the other hand, 'No' means the right part is not different from the left part at all.

(Note 2)
At the bottom level, each viewpoint (each row) in each column is combined with one another as a bottom-level candidate. For example, (a)-(d)-(g)-(j) is a candidate.

Bottom-level viewpoints; and candidates for the middle-level candidate 16

(a)	(d)	(g)	(j)
NAME(SC(s; t))∩NAME(SC(s'; t'))=∅	BODY(SC(s; t))∩BODY(SC(s'; t')) = ∅	N-struct(SC(s; t))∩N-struct(SC(s'; t'))= ∅	B-struct(SC(s; t))∩B-struct(SC(s'; t'))= ∅
(b)	(e)	(h)	(k)
NAME(SC(s; t))⊂NAME(SC(s'; t')) or NAME(SC(s'; t'))⊂NAME(SC(s; t))	BODY (SC(s; t))⊂BODY (SC(s'; t')) or BODY (SC(s'; t'))⊂BODY (SC(s; t))	N-struct(SC(s; t))⊂N-struct(SC(s'; t')) or N-struct(SC(s'; t'))⊂N-struct(SC(s; t))	B-struct(SC(s; t))⊂ B-struct(SC(s'; t')) or B-struct(SC(s'; t'))⊂ B-struct(SC(s; t))
(c)	(f)	(i)	(l)
¬(NAME(SC(s; t))⊂NAME(SC(s'; t'))) and ¬(NAME(SC(s'; t'))⊂NAME(SC(s; t))) and NAME(SC(s; t))∩NAME(SC(s'; t'))≠∅	¬(BODY (SC(s; t))⊂BODY (SC(s'; t'))) and ¬(BODY(SC(s'; t'))⊂BODY (SC(s; t))) and BODY(SC(s; t))∩BODY(SC(s'; t')) ≠ ∅	¬(N-struct(SC(s; t))⊂N-struct(SC(s'; t'))) and ¬(N-struct(SC(s'; t'))⊂N-struct(SC(s; t))) and N-struct(SC(s; t))∩N-struct(SC(s'; t')) ≠ ∅	¬(B-struct(SC(s; t))⊂B-struct(SC(s'; t'))) and ¬(B-struct(SC(s'; t'))⊂B-struct(SC(s; t))) and B-struct(SC(s; t))∩B-struct(SC(s'; t')) ≠ ∅

Fig. 1 Taxonomical viewpoints and categorical candidates

Analysis 6

(See the top-level candidate D, the middle-level candidate 10, and the bottom-level viewpoints from (a) to (c) and from (g) to (i) in Fig. 1.)

Consider the case where E(s; t) ≠ E(s'; t') and USE /E(s; t) ≠ USE/E(s'; t'). And let NAME(SC(s; t)) ≠ NAME(SC(s'; t')); BODY(SC(s; t)) = BODY(SC(s'; t')); N-struct(SC(s; t)) ≠ N-struct(SC(s'; t')); and B-struct (SC(s; t)) = B-struct(SC(s'; t')). In this case, we have 9 candidates. However, from (a) of Prop. 11, Prop. 1 and (a) of Prop. 12, we can eliminate 6 candidates in a similar way to Analysis 5.†

Analysis 7

(See the top-level candidate D, the middle-level candidate 12, and the bottom-level viewpoints from (d) to (l) in Fig. 1.)

Consider the case where E(s; t) ≠ E(s'; t') and USE/E(s; t) ≠ USE/E(s'; t'). And let NAME(SC(s; t)) = NAME(SC(s'; t')); BODY(SC(s; t)) ≠ BODY(SC(s'; t')); N-struct(SC(s; t)) ≠ N-struct(SC(s'; t')); and B-struct(SC(s; t)) ≠ B-struct(SC(s'; t')). In this case, we can eliminate 18 out of 27 candidates as follows.

(1) Let BODY(SC(s; t)) ∩ BODY(SC(s'; t')) = ∅. From (b) of Prop. 11, we can eliminate 6 out of 9 candidates.

(2) Let BODY(SC(s; t)) ⊂ BODY(SC(s'; t')) or BODY(SC(s'; t')) ⊂ BODY(SC(s; t)). From Prop. 1, NAME(SC(s; t)) = NAME(SC(s'; t')) implies Card (BODY(SC(s; t)) = Card(BODY(SC(s'; t')). Therefore, neither BODY(SC(s; t)) ⊂ BODY(SC(s'; t')) nor BODY(SC(s'; t')) ⊂ BODY(SC(s; t)) is feasible. This enables us to eliminate all of 9 candidates.

(3) Let ¬(BODY(SC(s; t)) ⊂ BODY(SC(s'; t'))), ¬(BODY(SC(s'; t')) ⊂ BODY(SC(s; t))) and BODY (SC(s; t)) ∩ BODY(SC(s'; t')) ≠∅. From (b) of Prop. 12, neither B-struct(SC(s; t)) ⊂ B-struct(SC(s'; t')) nor B-struct(SC(s'; t')) ⊂ B-struct(SC(s; t)) is feasible. This implies we can eliminate 3 out of 9 candidates.†

Analysis 8

(See the top-level candidate D, the middle-level candidate 13, and the bottom-level viewpoints from (a) to (c) and from (g) to (l) in Fig. 1.)

Consider the case where E(s; t) ≠ E(s'; t') and USE /E(s; t) ≠ USE/E(s'; t'). And let NAME(SC(s; t)) ≠ NAME(SC(s'; t')); BODY(SC(s; t)) = BODY(SC(s'; t')); N-struct(SC(s; t)) ≠ N-struct(SC(s'; t')); and B-struct (SC(s; t)) ≠ B-struct(SC(s'; t')). In this case, we have 27 candidates. However, from (a) of Prop. 11, Prop. 1 and (a) of Prop. 12, we can eliminate 18 candidates in a similar way to Analysis 7. †

Analysis 9

(See the top-level candidate D, the middle-level candidate 16, and the bottom-level viewpoints from (a) to (l) in Fig. 1.)

Consider the case where E(s; t) ≠ E(s'; t') and USE/E(s; t) ≠ USE/E(s'; t'). And let NAME(SC(s; t)) ≠ NAME(SC(s'; t')); BODY(SC(s; t)) ≠ BODY(SC(s'; t')); N-struct(SC(s; t)) ≠ N-struct(SC(s'; t')); and B-struct(SC(s; t)) ≠ B-struct(SC(s'; t')). In this case, we have 81 candidates. However, we can eliminate 45 candidates with Props. 11 and 12 as follows.

(1) Let NAME(SC(s; t)) ∩ NAME(SC(s'; t')) = ∅; BODY(SC(s; t)) ∩ BODY(SC(s'; t')) = ∅. Then, from Prop. 11, we can only accept the combination of N-struct(SC(s; t)) ∩ N-struct(SC(s'; t')) = ∅ and B-struct (SC(s; t)) ∩ B-struct(SC(s'; t')) = ∅. Therefore, we can eliminate 8 out of 9 candidates.

(2) Let NAME(SC(s; t)) ∩ NAME(SC(s'; t')) = ∅; BODY(SC(s; t)) ⊂ BODY(SC(s'; t')) or BODY(SC(s'; t')) ⊂ BODY(SC(s; t)). Then, from (a) of Prop. 11, we can eliminate 6 out of 9 candidates.

(3) Let NAME(SC(s; t)) ∩ NAME(SC(s'; t')) = ∅; ¬(BODY(SC(s; t)) ⊂ BODY(SC(s'; t'))), ¬(BODY (SC(s'; t')) ⊂ BODY(SC(s; t))) and BODY(SC(s; t)) ∩ BODY(SC(s'; t')) ≠ ∅. Then, from (a) of Prop. 11, we can eliminate 6 out of 9 candidates. From (b) of Prop. 12, we can furthermore eliminate 1 candidate.

(4) Let NAME(SC(s; t)) ⊂ NAME(SC(s'; t')) or NAME(SC(s'; t')) ⊂ NAME(SC(s; t)); BODY(SC(s; t)) ∩ BODY(SC(s'; t')) = ∅. Then, from (b) of Prop. 11, we can eliminate 6 out of 9 candidates.

(5) Let NAME(SC(s; t)) ⊂ NAME(SC(s'; t')) or NAME(SC(s'; t')) ⊂ NAME(SC(s; t)); ¬(BODY(SC(s; t)) ⊂ BODY(SC(s'; t'))), ¬(BODY(SC(s'; t')) ⊂ BODY(SC(s; t))) and BODY(SC(s; t)) ∩ BODY(SC(s'; t')) ≠ ∅. Then, from (b) of Prop. 12, we can eliminate 3 out of 9 candidates.

(6) Let ¬(NAME(SC(s; t)) ⊂ NAME(SC(s'; t'))), ¬(NAME(SC(s'; t')) ⊂ NAME(SC(s; t))) and NAME (SC(s; t)) ∩ NAME(SC(s'; t')) ≠ ∅; BODY(SC(s; t)) ∩ BODY(SC(s'; t')) = ∅. Then, from (b) of Prop. 11, we can eliminate 6 out of 9 candidates. From (a) of Prop. 12, we can furthermore eliminate 1 candidate.

(7) Let ¬(NAME(SC(s; t)) ⊂ NAME(SC(s'; t'))), ¬(NAME(SC(s'; t')) ⊂ NAME(SC(s; t))) and NAME (SC(s; t)) ∩ NAME(SC(s'; t')) ≠ ∅; BODY(SC(s; t)) ⊂ BODY(SC(s'; t')) or BODY(SC(s'; t')) ⊂ BODY (SC(s; t)). Then, from (a) of Prop. 12, we can eliminate 3 out of 9 candidates.

(8) Let ¬(NAME(SC(s; t)) ⊂ NAME(SC(s'; t'))), ¬ (NAME(SC(s'; t')) ⊂ NAME(SC(s; t))) and NAME (SC(s; t)) ∩ NAME(SC(s'; t')) ≠ ∅; ¬(BODY(SC(s; t)) ⊂ BODY(SC(s'; t'))), ¬(BODY(SC(s'; t')) ⊂

BODY(SC(s; t))) and BODY(SC(s; t)) ∩ BODY(SC(s'; t')) ≠ ∅. Then, from (a) of Prop. 12, we can eliminate 3 out of 9 candidates. From (b) of Prop. 12, we can furthermore eliminate 2 candidates.†

From the above analyses, we can conclude that the theoretical analysis makes it possible to eliminate 944 out of 1,024 categorical candidates.

6. Practical review of categories

The theoretical analysis reduces the categorical complexity of software architectural changes to the extent that 80 categories of changes are theoretically acceptable. This section furthermore reviews those acceptable categories from a practical point of view.

The 4 categories that are judged to be acceptable by Analysis 1 and Analysis 3 are practically meaningful, also. Therefore, the review can be restricted to those 76 theoretically acceptable categories in the case where E(s; t) ≠ E(s'; t') and USE/E(s; t) ≠ USE/E(s'; t').

Review 1

(See the middle-level candidate 1 in Fig. 1.)

Consider the case where NAME(SC(s; t)) = NAME (SC(s'; t')); BODY(SC(s; t)) = BODY(SC(s'; t')); N-struct(SC(s; t)) = N-struct(SC(s'; t')); and B-struct (SC(s; t)) = B-struct(SC(s'; t')). In this case, we have 1 category that is theoretically acceptable. However, it implies that the elemental part E(s; t) changes to E(s'; t') without replacement of any entity name or any entity body with a new one. For such a change, we can observe that the name or body of an entity is exchanged for the name or body of another entity within E(s; t). This situation is little meaningful in practice.†

Review 2

(See the middle-level candidate 2 in Fig. 1.)

Consider the case where NAME(SC(s; t)) = NAME (SC(s'; t')); BODY(SC(s; t)) = BODY(SC(s'; t')); N-struct(SC(s; t)) = N-struct(SC(s'; t')); and B-struct (SC(s; t)) ≠ B-struct(SC(s'; t')). In this case, we have 3 categories that are theoretically acceptable. However, each of them implies that the bodily aspect of the structural part B-struct(SC(s; t)) changes to B-struct (SC(s'; t')) without replacement of any entity body with a new one. For such a change, we can observe that the body of an entity is exchanged for the body of another within BODY(SC(s; t)). This is also little meaningful. †

Review 3

(See the middle-level candidate 3 in Fig. 1.)

Consider the case where NAME(SC(s; t)) = NAME (SC(s'; t')); BODY(SC(s; t)) = BODY(SC(s'; t')); N-struct(SC(s; t)) ≠ N-struct(SC(s'; t')); and B-struct(SC(s; t)) = B-struct(SC(s'; t')). In this case, we have 3 categories that are theoretically acceptable. However, each of them is little meaningful in practice, also, due to the observation similar to Review 2.†

Review 4

(See the middle-level candidate 6 in Fig. 1.)

Consider the case where NAME(SC(s; t)) = NAME (SC(s'; t')); BODY(SC(s; t)) = BODY(SC(s'; t')); N-struct(SC(s; t)) ≠ N-struct(SC(s'; t')); and B-struct(SC(s; t)) ≠ B-struct(SC(s'; t')). In this case, we have 9 categories that are theoretically acceptable. In a similar way to Review 2 or Review 3, they can be judged to be little meaningful in practice.†

Review 5

(See the middle-level candidate 12 in Fig. 1.)

Consider the case where NAME(SC(s; t)) = NAME (SC(s'; t')); BODY(SC(s; t)) ≠ BODY(SC(s'; t')); N-struct(SC(s; t)) ≠ N-struct(SC(s'; t')); and B-struct(SC(s; t)) ≠ B-struct(SC(s'; t')). In this case, from Analysis 7, we have 9 categories that are theoretically acceptable. However, they are little meaningful in practice from the same reason that is mentioned in Review 3.†

Review 6

(See the middle-level candidate 13 in Fig. 1.)

Consider the case where NAME(SC(s; t)) ≠ NAME (SC(s'; t')); BODY(SC(s; t)) = BODY(SC(s'; t')); N-struct(SC(s; t)) ≠ N-struct(SC(s'; t')); and B-struct(SC(s; t)) ≠ B-struct(SC(s'; t')). In this case, from Analysis 8, we have 9 categories that are theoretically acceptable. However, they can be judged to be little meaningful in practice based on the discussion in Review 2.†

Thus, from a practical point of view, we can eliminate 34 out of those 76 theoretically acceptable categories. In the result, we have 46 categories of changes that are theoretically and practically meaningful.

7. A taxonomical table

The finally accepted categories are summarized in Table 1. The 'top' column shows the candidates accepted at the top-level in Fig. 1. For each of them, the 'middle' column shows the candidates accepted at the middle-level in Fig. 1. Furthermore, for each candidate in the 'middle' column, the 'bottom' column shows the acceptable combinations of bottom-level viewpoints in Fig. 1. Each combination of a top-candidate, a middle-candidate and a bottom combination corresponds to a finally accepted category, which is numbered.

Let us see the 8th category as an example. The top is D, that is, E(s; t) ≠ E(s'; t') and USE/E(s; t) ≠ USE/E(s'; t'). The middle is 10, that is, NAME(SC(s; t)) ≠ NAME(SC(s'; t')); BODY(SC(s; t)) = BODY(SC(s'; t')); N-struct(SC(s; t)) ≠ N-struct(SC(s'; t')); and B-struct(SC(s; t)) = B-struct(SC(s'; t')). The bottom is (a)-

(g), that is, a combination of NAME(SC(s; t)) ∩ NAME(SC(s'; t')) = ∅ and N-struct(SC(s; t)) ∩ N-struct(SC(s'; t')) = ∅.

8. Conclusions

A theory of software architectural changes has been reviewed, and applied to a taxonomical study on software architectural changes.

There are 1,024 categorical candidates of changes. Those candidates have been analyzed theoretically, and it has been shown that it is possible to eliminate 944 out of 1,024 categorical candidates. We have 80 categories of changes that are theoretically acceptable.

Furthermore, the theoretically acceptable categories have been reviewed from a practical point of view, and it has been shown that it is possible to eliminate 34 out of the 80 theoretically acceptable categories.

In the result, we have 46 categories of changes that are theoretically and practically meaningful. They have been summarized in Table 1, which leads to a better position where we have an insight into software architectural changes themselves. We can study software architectural changes within those limited categories.

References

[Baze85] R. Bazelmans(1985): "Evolution of Configuration Management", ACM SIGSOFT Software Engineering Notes, Vol.10, No.5, Oct., pp.37-46

[Bers80] E. H. Bersoff, V. D. Henderson and S. G. Siegel(1980): Software Configuration Management, Prentice-Hall.

[Estu89] J. Estublier and J. M. Favre(1989): "Structuring Large Versioned Software Products", Proc. IEEE Conf. on Software Maintenance, Oct., Phoenix, pp.404-411.

[McCa75] R. McCarthy(1975): "Applying the Technique of Configuration Management to Software", Quality Progress, Oct.

[Nara87] K. Narayanaswamy and W. Scacchi(1987): "Maintaining Configurations of Evolving Software Systems", IEEE Trans. on Software Engineering, Vol.SE-13, No.3, pp.324-334.

[Ohmo90] A. Ohmori and Mikio Aoyama (1990): "Theoretical Analyses of Software Architectural Changes", IPSJ SIG Reports, 90-SE-73, pp.1-10.

[Rend88] H. S. Render and R. H. Campbell(1988): "CLEMMA: The Design of a Practical Configuration Librarian", Proc. IEEE Conf. on Software Maintenance, Oct., Phoenix, pp.222-228.

[Yau 87] S. S. Yau and J. J. Tsai(1987): "Knowledge Representation of Software Component Interconnection Information for Large-Scale Software Modifications", IEEE Trans. on Software Engineering, Vol.SE-13, No.3, pp.355-361.

No. \ Lev	top	middle	bottom
1	A	1	
2	C	4	(d)
3	C	5	(a)
4	C	11	(a)-(d)
5	D	7	(d)-(j)
6	D	7	(f)-(j)
7	D	7	(f)-(l)
8	D	10	(a)-(g)
9	D	10	(c)-(g)
10	D	10	(c)-(i)
11	D	16	(a)-(d)-(g)-(j)
12	D	16	(a)-(e)-(g)-(j)
13	D	16	(a)-(e)-(g)-(k)
14	D	16	(a)-(e)-(g)-(l)
15	D	16	(a)-(f)-(g)-(j)
16	D	16	(a)-(f)-(g)-(l)
17	D	16	(b)-(d)-(g)-(j)
18	D	16	(b)-(d)-(h)-(j)
19	D	16	(b)-(d)-(i)-(j)
20	D	16	(b)-(e)-(g)-(j)
21	D	16	(b)-(e)-(g)-(k)
22	D	16	(b)-(e)-(g)-(l)
23	D	16	(b)-(e)-(h)-(j)
24	D	16	(b)-(e)-(h)-(k)
25	D	16	(b)-(e)-(h)-(l)
26	D	16	(b)-(e)-(i)-(j)
27	D	16	(b)-(e)-(i)-(k)
28	D	16	(b)-(e)-(i)-(l)
29	D	16	(b)-(f)-(g)-(j)
30	D	16	(b)-(f)-(g)-(l)
31	D	16	(b)-(f)-(h)-(j)
32	D	16	(b)-(f)-(h)-(l)
33	D	16	(b)-(f)-(i)-(j)
34	D	16	(b)-(f)-(i)-(l)
35	D	16	(c)-(d)-(g)-(j)
36	D	16	(c)-(d)-(i)-(j)
37	D	16	(c)-(e)-(g)-(j)
38	D	16	(c)-(e)-(g)-(k)
39	D	16	(c)-(e)-(g)-(l)
40	D	16	(c)-(e)-(i)-(j)
41	D	16	(c)-(e)-(i)-(k)
42	D	16	(c)-(e)-(i)-(l)
43	D	16	(c)-(f)-(g)-(j)
44	D	16	(c)-(f)-(g)-(l)
45	D	16	(c)-(f)-(i)-(j)
46	D	16	(c)-(f)-(i)-(l)

Table 1 A taxonomical table of changes