Trajectory Mining on Capability Space: Its Concept and Potential Application

Hiroshi TSUJI Ryosuke SAGA
Osaka Prefecture University
{tsuji, saga}@cs.osakafu-u.ac.jp

Amrit TIWANA
University of Georgia
tiwana@uga.edu

Dietrich ALBERT
Graz Univ. of Tech. and Univ. of Graz
dietrich.albert@uni-graz.at

Abstract

To find the growth path for companies, this paper proposes the concept of trajectory mining. The central idea is to find the constraints and normal growth paths from the volumes of the trajectories on the capability space where capability space is spanned by multi-dimensional axes. The identified trajectory shows the reasonable direction for future growth. An application for software outsourcing clients demonstrates the proposed concept.

1. Introduction

To assign credit to software development organizations, the Capability Maturity Model (CMM) was proposed as an evaluation method that considers both their capability and maturity [1]. Because the framework of CMM can be extended to a wide range of evaluation not only for IT companies but also for individuals (hereafter both be referred to as simply "units"), the framework has been extended to Capability Maturity Model Integration (CMMI) [2-3] and been applied to various application domains such as information security management and project management. Such works have inspired us to develop support systems for personal learning [4], science teacher development [5] and organizational security management [6].

In CMMI, all units start at the lowest level and climb up the levels step by step. The final destination is the highest level for all capabilities. In between these two states there are potential intermediary states and paths that we refer to as the capability space. However, some states may not be reachable if there are prerequisite constraints preceding them. Furthermore, there may be normal paths for most units moving from one state to another state. To find the effective paths is an important task, and to avoid the ineffective paths can be a promising strategy for the unit.

Once we have a sufficient number of trajectories that express the growth trajectory on the capability space from the initial state to the current state, we might have the prospect of finding such constraints. If we can find constraints among states, we may recommend a unit to follow a direction in accordance with antecedent and prerequisite conditions.

Originally such a growth path was a kind of tacit knowledge. No unit knew either the explicit way or others' paths. The audits themselves are just data and do not guide growth strategy. The audits have seldom been recorded explicitly. Once we regard the recorded audit as externalized knowledge, we have the chance to combine them. Analyzing the recorded audit, we should find a strategy for further growth. Once a unit learns strategy, it becomes smarter. Such a stream is SECI model proposed in [7] in a wide sense. Recorded audit analysis requires computing power. To implement idea as a digitized information system [8], the formal approach is required.

The purpose of this research is to propose the formal concept of trajectory mining on the capability space for implementing an information system. As the term (which is similar to data mining [9]) suggests, trajectory mining extracts knowledge for deciding growth strategy. First, the terms will be defined. Next, we will describe the visualization method illustrating the operations and algorithms for finding constraints as trajectory mining. Introducing the method for collecting trajectories in a smart manner, we will describe an application that explores the software outsourcing capability space. Future research will be also implied.

2. Term Definition

Let us first define basic terms for trajectory mining for our proposal. "Unit $\boldsymbol{U_i}$" is a growing object that includes both organizations (including sports teams) and individuals (like students). "Capability $\boldsymbol{C_j}$" is a growth axis. We suppose there are more than three capabilities for one application domain. For a simple example, there are four capabilities in the language study domain (reading, writing, speaking and hearing) where a unit is a person.

1530-1605/12 $26.00 © 2012 IEEE
DOI 10.1109/HICSS.2013.572

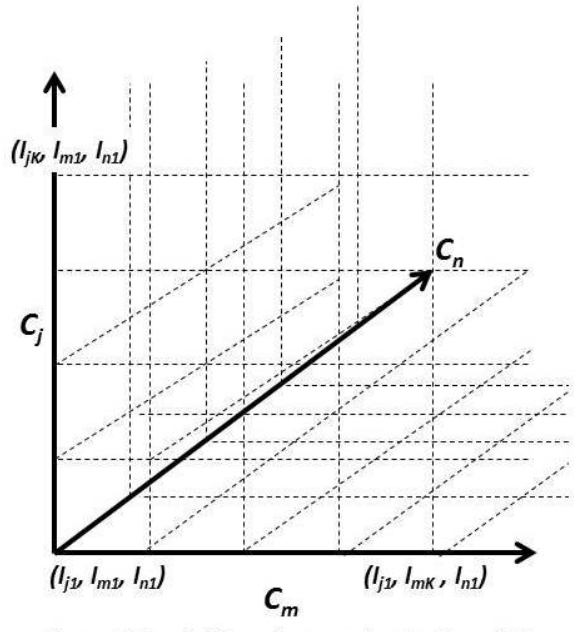

Figure 1 Capability sub-space for C_i, C_m and C_n

Each capability has some "levels l_{jk}". For example, *low, fair, good, very good and excellent* at speaking a language. For notation simplification, hereafter let us assume each capability has *K* levels l_{jk} : *k=1, 2, ..., K*. "Capability state" is a vector expressed as the capability levels at their axes. For example, *(reading, writing, speaking, hearing) = (very good, fair, good, low)*.

At first (*t=1*), each unit stands at the lowest level l_{j1} for all capability C_j . The final goal is to achieve the highest level l_{jK} for all capability C_j . Their examples are initial state *(low, low, low, low)* and final state *(excellent, excellent, excellent, excellent)*. "The time-series variable $X_i(t)$" is a capability state for unit U_i at time t.

Then

$$X_i(1) = (l_{11}, l_{21}, l_{31}, \ldots\ldots) \text{ for all } U_i.$$

assuming that all units start growing at the same time (a person's capability state at a specific starting day is *(low, low, low, low)*. This assumption is not essential and can be relaxed in the future.

We also suppose no unit can degrade its capability along the time axis (once one achieved "*very good*", s/he shall not later be "*good*", "*fair*" or "*low*"). This assumption can be also relaxed in the future work. Here, we suppose that there are semi-ordinal relations on growth:

$$X_i(t) < X_i(s) \text{ iff } t < s.$$

This assumption implies that as time passes, at least one capability level should be updated. For our first proposal, we initially assume that the difference between $X_i(t+1)$ and $X_i(t)$ is just one element in their vectors.

We present another concept called "growth path $P_i(t)$" for Unit U_i. Then $P_i(t)$ is the sequence $\{X_i(1), X_i(2), X_i(3), \ldots, X_i(t)\}$.

Let us define the capability space and the capability structure next. The basic idea derives from the analogy of knowledge space theory proposed in [10] for knowledge acquisition while there are some similar concepts called rough set for reasoning [11] and version space for machine learning [12, 13].

Capability space is a set of potential capability states. Then the cardinality of the set of capability states is the K^{th} power of the number of capability axes. The capability subspace is a set of state vectors that consist of some capability axes. The example of three dimensional capability subspace spanned by C_j, C_m and C_n is illustrated in Figure 1.

To define the capability structure, we should remember that there may be constraints when a unit achieves a capability level. Sometimes levels l_{jk} can be achieved without any condition if a unit has already achieved levels l_{jh} where $k=h+1$. However, achieving levels l_{jp} may sometimes require another condition even if a unit has already achieved levels l_{jk} where $p=k+1$: for example, l_{m3} should be achieved before a unit achieves l_{j4} (see Figure 2); in other words, l_{j3} **and** l_{m3} are prerequisites for achieving l_{j4},

For example, hearing level "*good*" may be achieved without any constraint if one has already achieved hearing level "*fair*". However, speaking level "*good*" is possible only if one has already achieved not only speaking level "*fair*" but also hearing level "*good*".

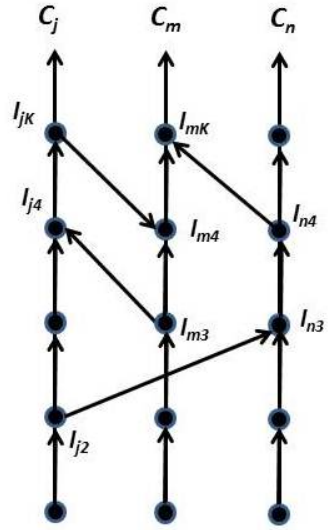

Figure 2 Example of capability structure for C_i, C_m and C_n

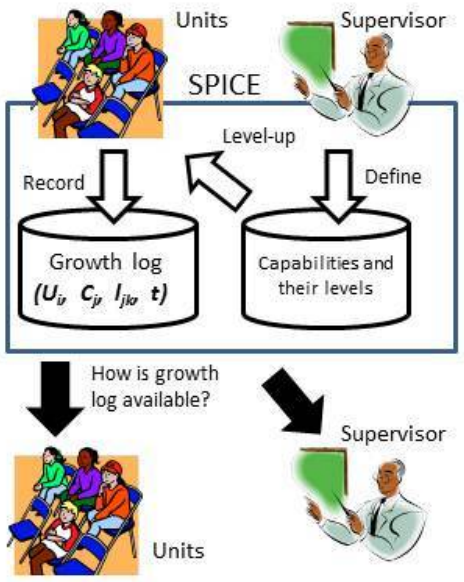

Figure 3 SPICE and research questions

Let us illustrate this simple example of capability structure in Figure 2 which shows three capability axes and four extra conditions for growth.

Note that there is an algorithm called ISM (Interpretive Structural Modeling) [14] that identifies such ordinal or hierarchical structures from a set of pair-wise ordinal relations ("*speaking fair*" => "*speaking good*" and "*hearing good*" => "*speaking good*" are examples of pair-wise ordinal relations). The algorithm is based on graph theory [15] and finds out reach-ability matrix from adjacent matrix.

Finally, let us define "supervisor ***S***" for units. The supervisor is responsible for designing capabilities and their levels in an application domain. It also should give units advice how to improve their capability, and may update its design as time passes. While the task for such design is not simple, it is well-known that CMMI is a wide framework and there have been varieties of capability definition.

3. Research Questions

We have already developed capability enhancement systems [4] that records $X_i(t)$ for all units ***i***. The basic architecture is called the *SPIral Capability Enhancement support system* (SPICE). Some functions of SPICE are provided for units while others are prepared for a supervisor. Note that the term "spiral" is important as the name expresses.

At first the supervisor defines capabilities and their levels. We suppose that it takes a lot of time by multiple experts as CMM was defined. The initial levels for capabilities are equal to ***1*** for all units as defined before. Then as each unit achieves an upper level along the defined capability, it updates its current state. At that time, SPICE records the growth logs for units: (U_i, C_j, I_{jk}, t).

Our major issue is what SPICE can do for the units and the supervisor after it accumulates volumes of growth logs. There are two viewpoints as strategy: one from a unit and the other from a supervisor.

To support the growing units, there are the following research sub-questions:

(Q-u1) To review each growth history, how does SPICE handle growth paths?

(Q-u2) To confirm each current position, how does SPICE show the difference among units?

(Q-u3) To suggest the growth direction, how does SPICE give advice?

To support the supervisor, there are also the following research sub-questions:

(Q-s1) To review the growth histories, how does SPICE find out constraints for growing units?

(Q-s2) To confirm consistency among growing units (whether there is a normal path or not), how does SPICE evaluate?

(Q-s3) To redesign capability axes and their levels, how does SPICE suggest to the supervisor?

If most units follow the same path from the initial state to the goal, we say that "*they are consistent among them* (homogeneous)". On the other hand, when each unit takes a different path from the others, "*they are inconsistent among them* (heterogeneous)". The stakeholders of SPICE and the research questions are illustrated in Figure 3. Note that the growth logs are stored and not modified like the case of a data-warehouse [16].

4. Trajectory Mining

Let us answer the research sub-questions in this section. We must remember that the stored log data, (U_i, C_j, I_{jk}, t), is peculiar multi-dimensional. Therefore, it is difficult to be aware of growth without reducing dimensions and transformation. As growth logs are recorded in a data-warehouse for analysis, we have the chance to introduce On-Line Analytical Program (OLAP) operations [16] such as slicing and dicing to analyze stored multi-dimensional data.

On the other hand, simple enumeration of growth logs has given us little information even if the dimensions are reduced. Thus, the visualization that gives us a bird's-eye view is an attractive approach.

Slicing operation refers to the ability to reduce dimensions to simplify the stored data. Two-dimensional visualization refers to the ability to clarify the position in the capability space [17].

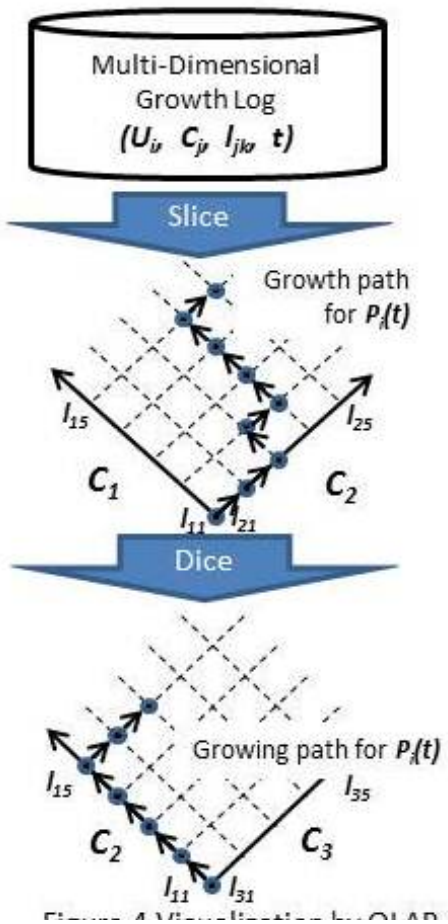

Figure 4 Visualization by OLAP operations for a growth path

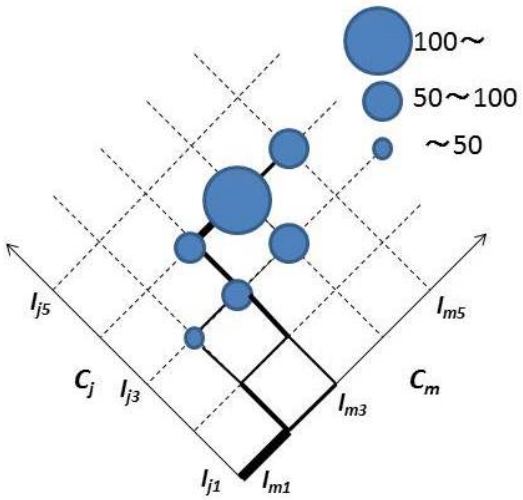

Figure 5 Unit distribution with trajectory at time=***T***

For **(Q-u1)**, let us slice the multi-dimensional cube by two capability axes C_1 and C_2 and filter the unit dimension by U_i. Then we have a chance to connect capability state time by time as a growth path on the capability subspace.

An example growth path is shown in Figure 4. From this figure, the growth order between C_1 and C_2 becomes clear for U_i.

By changing one capability axis C_1 into another axis C_3, we can operate the log data warehouse as if we were dicing the cube. Figure 4 also includes an example of dicing operation where we can find that U_i did not improve his C_3 until he achieved l_{25} for C_2.

For **(Q-u2)**, let us slice the cube using two capability axes C_j and C_m. We filter the time dimension by time = T. Then we count the number of units state by state and also the number of paths between capability states. Changing the size of the node and the thickness of the link in accordance with the counted numbers, we illustrate the distribution of units at time = T and of path until time = T. Dicing operation also allows re-combining the dimension to see different slices of the distribution. The example is shown in Figure 5 where we can see the following:

(1) More than 100 units are now l_{j4} and l_{m4}.
(2) Fewer than 50 units are now l_{j3} and l_{m2}.
(3) All units grow along capability C_m first.
(4) There are two ways at (l_{j1}, l_{m2}) but growth along C_j is more common.
(5) There is one way at (l_{j1}, l_{m3}) and no chance to stay at (l_{j1}, l_{m4}).

Note that not all units always start growing at the same time. Therefore, there are options with regards to time stamp: relative time stamp and absolute time stamp. Although the origin for the relative time stamp is the same, that for the absolute time stamp is different unit by unit.

For **(Q-u3)**, there are two cases for U_i. To consider these options, let us introduce the Pareto (no inferior) capability state. If unit U_i exists at the Pareto capability state, it has achieved the highest levels for at least one combination of capabilities. If not, there is at least one unit superior to unit U_i.

(1) If unit U_i does not exist at the Pareto capability state, then unit superior to it are collected. Then some of them are filtered out if they did not previously exist at the current U_i state. Then the number of growth path should be calculated. Finally, we can depict the potential growth direction.
(2) If unit U_i exists at the Pareto capability state, we notify it with this fact and encourage it to be a pioneer.

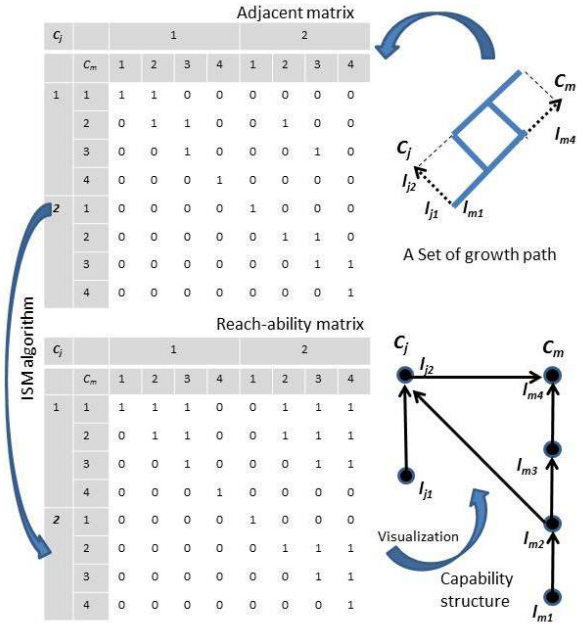

Figure 6 Example for identifying capability structure

For **(Q-s1)**, let us construct adjacent matrix ("number of capability levels" x "number of capability levels") where the element is equal to

1 if at least one unit grew from l_{jk} to its either of neighbors ($l_{j+1\,k}$, $l_{j\,k+1}$),
1 if it is diagonal element l_{jj} , and
0 otherwise.

Let us show an example in Figure 6. Suppose that there is a set of growth paths as shown in the upper right. There are two levels for capability ***j*** and four levels for capability ***m***. Then the matrix size is 8 x 8 as shown in upper left. Because there are six paths between capability states, there are fourteen "1"s (6+8) in the matrix. For example, there is a path, (l_{j1}, l_{m3}) to (l_{j2}, l_{m3}), and then the corresponding entry is 1.

By multiplying the adjacent matrix with itself repeatedly, the ISM (Interpretive Structural Modeling) algorithm [14] determines the reach-ability matrix which is well-known in graph theory [15]. The matrix expresses some restrictive capability levels that must be achieved before other levels can be achieved. An example is shown in the bottom left of Figure 6. From the reach-ability matrix, we can produce the capability structure. Once we create the capability structure, the constraints between two capabilities become easy to identify. An example capability structure is shown in the bottom right of Figure 6: To reach level 2 of capability ***j*** one must achieve level 2 of capability ***m*** and to reach level 4 of capability ***m*** one must achieve level 2 of capability ***j***. The algorithm will also be enhanced if we consider the thicknesses of paths in the future.

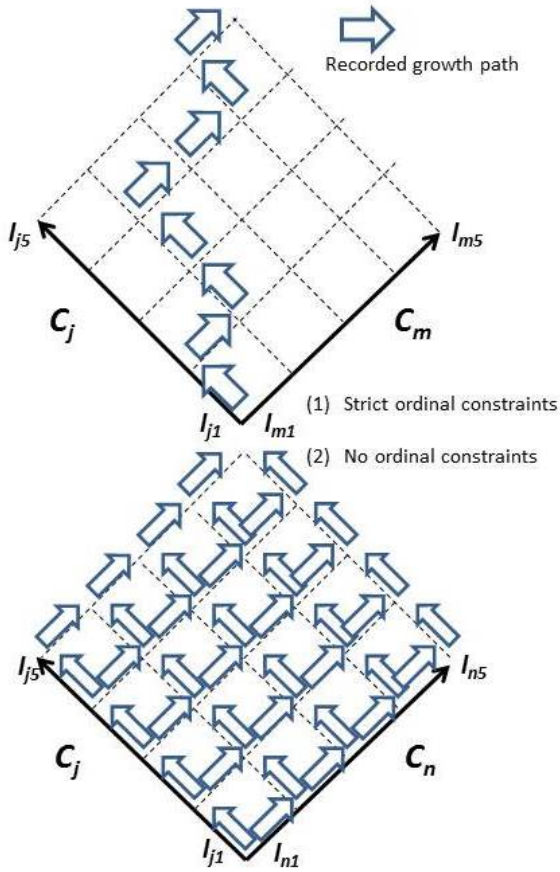

Figure 7 Examples of consistency between two capability axes

On the basis of the capability structure, the supervisor may identify the ordinal constraints for unit growth among capabilities and to give growing units appropriate advice for direction or priority. This might be regarded as a normal path.

For **(Q-s2)**, first we should consider two kinds of index on paths. One is the number of potential paths varieties and the other is the number of recorded path varieties. Their ratio can be regarded as the density. Again let us slice the capability space by selecting two capability axes for simplification: C_1 and C_2. Then the initial state is (l_{11}, l_{21}) and the final state is (l_{1K}, l_{2K}).

Then the former combinatorial number is ${}_8C_4 = 70$ if $K=5$. If all units followed the same path (in other words, they are consistent on two capabilities and there are strict ordinal constraints between them), the density is *1/70*. On the other hand, if there are more than seventy units and there are seventy sorts of recorded paths (in other words, they are inconsistent on two capabilities and there is no ordinal constraints among them), the density is *70/70 = 1*.

Thus, by calculating the density between two capabilities, SPICE finds consistent and inconsistent

pairs. Figure 7 illustrates a consistent pair (C_j and C_m) and an inconsistent pair (C_j and C_n). If the density is close to *1*, the supervisor does not need to advise units about the growth direction because there is no ordinal constraint. Consistent and inconsistent subspaces can also be found by identifying the two capabilities.

For **(Q-s3)**, let us consider whether, at first, supervisor ***S*** could define appropriate capability axes and their levels which is probably a difficult task for him. It may take a lot of time even if some experts join the task. Sometimes, it may miss some axes or include redundant axes. Once he is aware of the inappropriateness in his design, there should be a chance to redesign them.

Until now, we have supposed that the difference between $X_i(t+1)$ and $X_i(t)$ is just one element in their vectors. However, this cannot always be true.

In fact there are two cases: (1) the difference is more than two elements and (2) there is no difference. For the former case, a capability axis may be redundant or the number of levels may be too large. On the other hand, for the latter, another capability axis may be introduced. Alternatively the number of levels may be too small.

We have proposed answers for the research sub-questions described in Section 3. Because the concept on trajectory mining is still new, there are still a lot of discussion issues for strategic knowledge management. For example, there are some kinds of units based on their demographics. Each cluster may have different characteristics based on their capability structure and consistency. This shows that the clustering on capability space is an interesting research topic.

Another idea to support the supervisor for such redesign is to introduce a time span. The requisite time for growing one level along a capability is naturally different from each other. If SPICE calculates the average time and deviation for growth between capability states, it may suggest a shortest path for both growing units and the supervisor.

5. Application for Evaluating Software Outsourcing Clients

As a potential application of the proposed trajectory mining, we design capability axes and their levels for evaluating IT companies who are outsourcing software development to Asian countries such as India and China. The basic idea comes from JEITA's (Japan Electronics and Information Technology Industries Association) discussion in which more than ten experts participated for three years [18]. Although in the past Japanese IT companies had developed software using their domestic affiliate companies, they are now concerned with methods to reduce costs and how to procure the volumes of engineers [19-21].

As a first rough analysis, it can be said that there are four maturity stages as shown in Table 1. Because the first stage is a trial, both client and vendor have low capability levels. On the other hand, they have high capability levels at the fourth stage. To promote offshore outsourcing, they should upgrade their capability to improve their stages.

Table 1 Overview of offshore software development maturity stages

	Client View	Vendor View	Method
1	Try for Cost Reduction	Learn Technical Skill and project Management	Bridge Engineer
2	Receive Returns	Improve Quality based on credit	Traditional CMMI
3	Enlarge Scales	Rock-in Clients for long term	Knowledge Management
4	Balance Portfolio	Acquire Domain Knowledge	Collaborative Creation

Our concerns based on interviews with Japanese IT companies are

(1) How has each company improved its capability?
(2) How effective are they in managing vendors?
(3) In which order should they improve their capability?
(4) Are there any ordinal constraints for improving the capability?
(5) Are there differences among companies?
(6) Is our questionnaire appropriate?

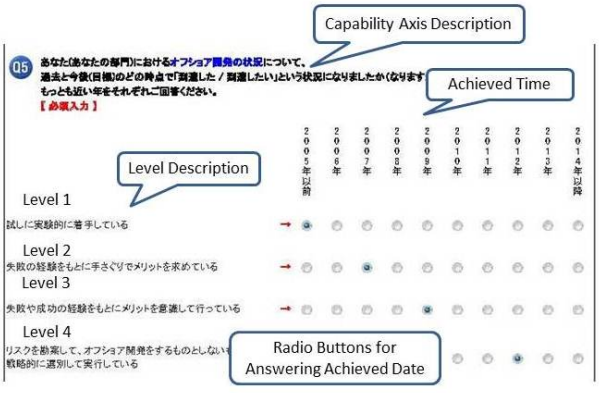

Figure 8 An example growth logs collection web interface for client managers

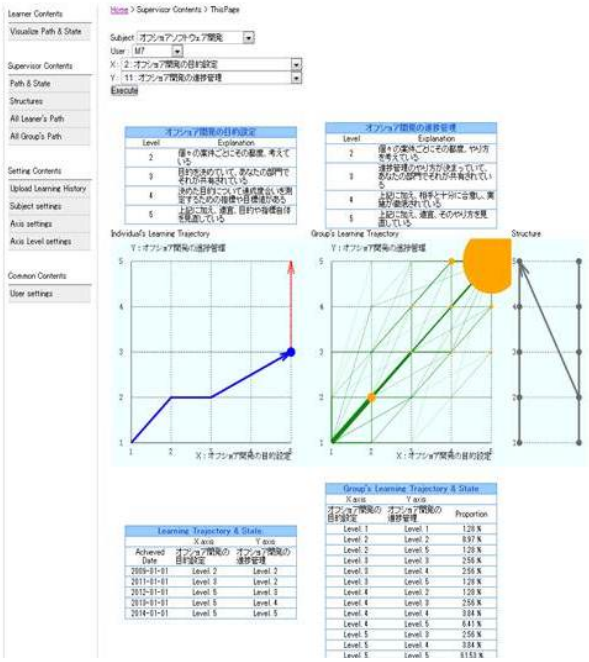

Figure 9 A screen example of SPICE: Personal and group growth paths

To collect the companies' growth audit, we designed a questionnaire that includes thirteen capabilities C_j and their levels l_{jk}. Referring to JEITA report [18], we asked some experts to review the capabilities and their levels. Then number of level for each capability is less than five. An example list of capabilities and their levels defined by those experts are shown in Table 2.

The question treats (U_i, C_j, l_{jk}, t) where t='05-'14 year by year. In other words, it asks clients (units) about their states for ten years: their past state (t='05-'11), current state (t='12) and future planned state (t='13-'14). An example questionnaire is shown in Figure 8. Such a questionnaire structure (just select radio buttons in the web browser) allows us to collect growth paths $P_i(t)$ in a short period from multiple clients.

The questionnaire also asks about control parameters such as vendor countries (India, China, Vietnam and others) and software categories (embedded software, middleware and customer applications). Our survey includes 213 responses (units) U_i from project managers in Japanese IT companies. Because some responses have extraordinary audits or missing answers, we use 73 responses.

Let us show example screen shots. Figure 9 visualizes individual growth path in the left side and current distribution in the center. The identified constraint in capability structure is also shown in right side. Note that there are diagonal lines meaning that the unit improves multiple capabilities at the same period. We can change a unit as a filtering option or capability axis as a slicing/dicing option.

Figure 10 shows bird-eye-view for capability structure where we can identify the existence of constraints for growth. The detailed description for analysis procedure on business strategy is an important topic but beyond the scope of this paper.

Table 2 Example list of capabilities and their levels

#	Capability	Levels
1	General Status	No Experience, Trial, Repeat based on Failure Experience, Repeat based on Success Experience, Repeat based Good Practice, Repeat based on Clear Strategy
2	Goal Setting	Not Clear, Define by Project, Qualitative Goal, Quantitative Goal, Optimization
3	Outsource Process Selection	Not clear, Subjective decision by Project, Check-list for Decision, Check-list for Decision with Quantitative Measure, Optimization
4	Vendor Selection	Not Defined, Reputation-based, By Original Criteria, Strategic Criteria, Optimization
5	Cost Estimate	Not Clear, Project by Project, Optional Estimate Method, Obligatory Process for Estimate, Optimization
6	Gap Analysis Frequency	No Case, Seldom Case(-20%), Some Cases(20-80%), Most Cases(80%-), All Cases

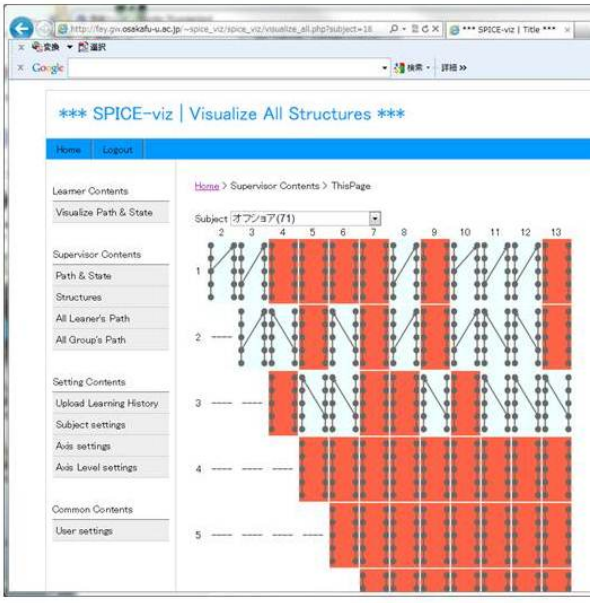

Figure 10 An example of screen of SPICE: bird-eye-view for capability structures

6. Conclusion

This paper has proposed the basic idea of trajectory mining on capability space and shown the formal description. The method consists of two operations: (1) visualization and (2) capability structure identification. The visualization includes a bird's eye view of a specific time stamp and trajectory as the growth history. The capability structure identification shows the constraints among capability states. Because the basic idea comes from combination of CMMI [2-3], knowledge space theory [10] and ISM [14], the paper addressed the relationship between our proposal and them.

Connecting the new approach with an already existing system (SPICE), this paper has also demonstrated a potential application for rating offshore software outsourcing companies. The user screens for collecting trajectories and for mining growth trajectory have been described. Such trajectory disappears time by time unless we propose that trajectory record works for knowledge management. In this sense, our proposal will contribute to strategic knowledge management for such companies.

The current proposal is rather theoretical but is applicable to varieties of potential application such as personal learning and organization compliance because the base theory CMMI has been applied widely. Of course, there are still a lot of discussion issues both on technology and on application domains.

7. Acknowledgement

The authors would like to express sincere thanks to Mr. Yu Nakamura and Mr. Takeo Ichinotsubo for their prototyping. This work is partially supported by Japan Society for the Promotion of Science KAKENHI-C grant (23500049).

8. References

[1] W. Humphrey, Managing the Software Process, Addison Wesley (1989)

[2] CMMI Product Team: \CMMI for Development, Version1.3",http://www.sei.cmu.edu/library/abstracts/ reports/10tr033.cfm (2011-03-31 accessed)

[3] M.B. Chrissis, M. Konrad, S. Shrum: Guidelines for Process Integration and Product Improvement, Pearson Education, Inc. (2003)

[4] Y. Nakamura and H. Tsuji, Spiral Capability Enhancement Support System, Proc. of the 2nd International Symposium on Aware Computing, pp.233-238 (2010)

[5] M. Sakoda, Y. Wada, H. Tsuji and K. Seta, Social Network Service with Maturity Level for Science Teachers. Proc. of IEEE International Conference on Systems, Man and Cybernetics, pp.1718-1723 (2009)

[6] Y. Kuo, Y. Nakamura, M. Sakoda, H. Tsuji, C. Lee, Giving Awareness of Maturity by Capability Assessment, Proc. of FUZZ-IEEE 2011, pp.1055-1060 (2011)

[7] I. Nonaka and H. Takeuchi, The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation, Oxford Univ. Pr. (1995).

[8] A. Tiwana, The Knowledge Management Toolkit, Orchestrating IT, Strategy and knowledge Platforms, Prentice Hall PTR, Upper Saddle River, NJ (2002)

[9] J. Han, M. Kamber, J. Pei, Data Mining: Concepts and Techniques, Elsevier Inc., (2006)

[10] J. Doignon and J. Falmagne, Spaces for the assessment of knowledge, International Journal of Man-Machine Studies, Vol.23, pp.175-196 (1985)

[11] M. Inuiguchi, Analysis of Information Table by Rough Set Theory, System, control and information: The Institute of Systems, Control and Information Engineers, Vol.49, No.5, pp.165-172 (2005)

[12] T. M. Mitchell, Machine Learning, Boston, McGraw-Hill (1997)

[13] M.R. Genesereth and N.J. Nilsson, Logical Foundations of Artificial Intelligence, Morgan Kaufmann (1986)

[14] J. N. Warfield, Interpretive structural modeling. In: Olsen, S. A. (ed.), Group planning and problem solving methods in engineering management. John Wiley and Sons, Inc. (1982)

[15] N. Hartsfield and G. Ringel, Pearls in Graph Theory: A Comprehensive Introduction, Academic Press, Inc. (1990)

[16] W.H. Inmon, Building the Data Warehouse, Fourth Edition, Wiley Publishing Inc. (2005)

[17] Y. Nakamura, H. Tsuji, K. Seta, K. Hashimoto, D. Albert, Visualization of Learner's State and Learning Paths with Knowledge Structures: A. Konig et al. (Eds.): KES 2011, Part IV, LNAI 6884, pp.261-270 (2011)

[18] JEITA, Survey report for optimizing software development resources, http://home.jeita.or.jp/is/publica/2008/is-08-jyousi-3.html, in Japanese (2007)

[19] H. Tsuji, A. Sakurai, K. Yoshida, A. Tiwana and A. Bush, Questionnaire-based Risk Assessment Scheme for Japanese Offshore Software Outsourcing, Lecture Notes on Computer Science 4716, B. Meyer and M. Joseph (Eds.), pp.114-127, Sprinter-Verlag Berlin Heidelberg (2007)

[20] A. Tiwana, A. Bush, H. Tsuji, A. Sakurai, and K. Yoshida, Myths and Paradoxes in Japanese IT Outsourcing, Communications of the ACM, Vol. 51, No.1, pp.141-145 (2008)

[21] A. Bush, A. Tiwana, H. Tsuji: An empirical investigation of the drivers of software outsourcing decisions in Japanese Organizations, Information and Software Technology, Journal of Information and Software Technology archive, Volume 50 Issue 6, May, pp.499-510 (2008)