FASSI 2018

The Fourth International Conference on Fundamentals and Advances in Software Systems Integration

ISBN: 978-1-61208-666-8

September 16 - 20, 2018

Venice, Italy

FASSI 2018 Editors

Chris Ireland, Open University, UK

FASSI 2018

Forward

The Fourth International Conference on Fundamentals and Advances in Software Systems Integration (FASSI 2018), held between September 16, 2018 and September 20, 2018 in Venice, Italy, continued a series of events started in 2015 and covering research in the field of software system integration.

On the surface, the question of how to integrate two software systems appears to be a technical concern, one that involves addressing issues, such as how to exchange data (Hohpe 2012), and which software systems are responsible for which part of a business process. Furthermore, because we can build interfaces between software systems we might therefore believe that the problems of software integration have been solved. But those responsible for the design of a software system face a number of trade-offs. For example the decoupling of software components is one way to reduce assumptions, such as those about where code is executed and when it is executed (Hohpe 2012). However, decoupling introduces other problems because it leads to an increase in the number of connections and introduces issues of availability, responsiveness and synchronicity of changes (Hohpe 2012).

The objective of this conference is to work toward on understanding of these issues, the trade-offs and the problems of software integration and to explore strategies for dealing with them. We are interested to receive paper from researchers working in the field of software system integration.

We take here the opportunity to warmly thank all the members of the FASSI 2018 technical program committee, as well as all the reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated their time and effort to contribute to FASSI 2018. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

We also gratefully thank the members of the FASSI 2018 organizing committee for their help in handling the logistics and for their work that made this professional meeting a success.

We hope that FASSI 2018 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in the field software systems integration. We also hope that Venice, Italy provided a pleasant environment during the conference and everyone saved some time to enjoy the unique charm of the city.

FASSI 2018 Chairs

FASSI Steering Committee

Chris Ireland, The Open University, UK

Hironori Washizaki, Waseda University / National Institute of Informatics / System Information, Japan

Keijiro Araki, Kyushu University, Japan

FASSI 2018
Committee

FASSI Steering Committee
Chris Ireland, The Open University, UK
Hironori Washizaki, Waseda University / National Institute of Informatics / System Information, Japan
Keijiro Araki, Kyushu University, Japan

FASSI 2018 Technical Program Committee
Frank J. Affonso, Universidade Estadual Paulista – UNESP, Brazil
Harvey Alférez, Montemorelos University, Mexico
Keijiro Araki, Kyushu University, Japan
Doo-Hwan Bae, School of Computing - KAIST, South Korea
Imen Ben Mansour, University of Manouba, Tunisia
Silvia Bonfanti, University of Bergamo, Italy
Michael Franklin Bosu, Waikato Institute of Technology, New Zealand
Graeme Burnett, University of Glasgow/Enhyper Ltd., UK
Yudith Cardinale, Universidad Simón Bolívar, Caracas, Venezuela
Stephen Clyde, Utah State University, USA
Marian Daun, University of Duisburg-Essen | paluno - The Ruhr Institute for Software Technology, Essen, Germany
Nitish Devadiga, Datarista Inc. / Carnegie Mellon University, USA
Jorge Edison Lascano, Universidad de las Fuerzas Armadas ESPE, Quito, Ecuador
Leire Etxeberria Elorza, Mondragon Unibertsitatea, Spain
Ip-Shing Fan, Cranfield University, UK
Marie Farrell, University of Liverpool, UK
Peter Forbrig, University of Rostock, Germany
Atef Gharbi, LISI | INSAT, Tunisia
Hamza Gharsellaoui, ENI Carthage | Carthage University, Tunisia / Al Jouf College of Technology | TVTC, KSA
Fotios Gioulekas, Aristotle University of Thessaloniki, Greece
Afef Gueidi, University Tunis El Manar / Carthage University, Tunisia
Mohammad Mahdi Hassan, Al Qassim University, Buraidah, Saudi Arabia
Shinpei Hayashi, Tokyo Institute of Technology, Japan
Alan Hayes, University of Bath, UK
Samedi Heng, Université Catholique de Louvain, Belgium
Nikolas Herbst, University of Würzburg, Germany
Uwe Hohenstein, Siemens AG, Munich, Germany
LiGuo Huang, Southern Methodist University, USA
Anca Daniela Ionita, University Politehnica of Bucharest, Romania
Chris Ireland, The Open University, UK
Slinger Jansen, Utrecht University, Netherlands

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

Prototyping for a Parallel Programming Tool

Kyoko Iwasawa
Department of Computer Science
Takushoku University
Hachioji Tokyo, Japan
e-mail: kiwasawa@cs.takushoku-u.ac.jp

***Abstract*—We propose a tool to enable even beginners in parallel processing to develop a parallelization program using Open Multi-Processing (OpenMP) directives. Our proposed tool is characterized by its analysis of source programs for C and OpenMP directives written by users and its display of parallel structure diagrams. Further, the discovery of source program bugs is facilitated by the static analysis of interactive data access regions and decisions on the feasibility of parallelization using these parallel structure diagrams. While our proposed tool currently handles only basic OpenMP directives, our aim is to improve the analysis of parallel structure diagrams by including more complex simultaneous processing and more precise data access.**

Keywords-parallel programming; OpenMP directive; data flow analysis.

I. INTRODUCTION

While the recent years have seen a proliferation in systems capable of parallel execution, including multicore and General Purpose computing on Graphics Processing Units (GPGPU), in general, the development of programs for parallel execution is difficult. While this is also the case with algorithm development, writing parallel processing code in an editing environment for the coding of sequential processing easily produces errors. Further, it is difficult to identify the errors, because in parallel programs the execution results are not reproducible.

Therefore, we propose a programming environment particularly for beginners in parallel processing, using a parallel structure that is easily understood visually and also statically analyses the feasibility of parallel execution from the execution statement data access regions during program editing. We are developing the prototype of this tool.

Open Multi-Processing (OpenMP) is an application programming interface that supports multi-platform shared memory multiprocessing programming by the OpenMP Architecture Review Boards [1]. The details of OpenMP spec are written in [2] and [3]. There are several tools for OpenMP programming. [4] and [5] are integrated tools for OpenMP programming, which include compiler and parallel execution environments. They have various functions and can be somewhat difficult for beginners of parallel programming. We simplify the analyzing method in [6] and [7] because our proposed tool does not generate parallel object code, but suggests user appropriate directives for parallelization.

The rest of paper is structured as follows. Section II presents the overview of the proposed tool; Section III describes the parallel structure diagrammatic display; Section IV explains the access region analytical method; Section V explains the parallelization feasibility decision method. Finally, we conclude and present the future issues in Section VI.

II. PROPOSED TOOL OVERVIEW

Our proposed tool is an environment for the C programming language used in creating and editing programs that give parallelization directions using OpenMP directives. It has the following three main functions.

(1) OpenMP directive analysis
(2) Parallel structure graph display
(3) Interactive, static data flow analysis, and parallelization feasibility decisions.

In addition, it display the structure written in OpenMP in an easy understood manner for users not accustomed to parallel processing, as well as for beginners to perform debugging by displaying static analytical results interactively.

Figure 1 shows the overall proposed tool structure.

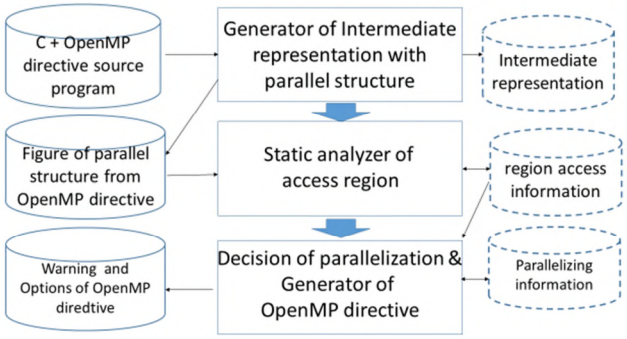

Figure 1. Overall tool structure

A source program with an OpenMP directive parallel execution direction is entered into a C program to analyze the C programming language execution directions and OpenMP directives. Subsequently, they are joined in an intermediate representation. This is formed and displayed as the parallel structure diagram in Figure 3. In this diagram, the user selects the quadrangle in the execution direction and the elliptical shape in the parallel execution direction to decide the data access region and parallel execution feasibility.

 ISBN: 978-1-61208-666-8

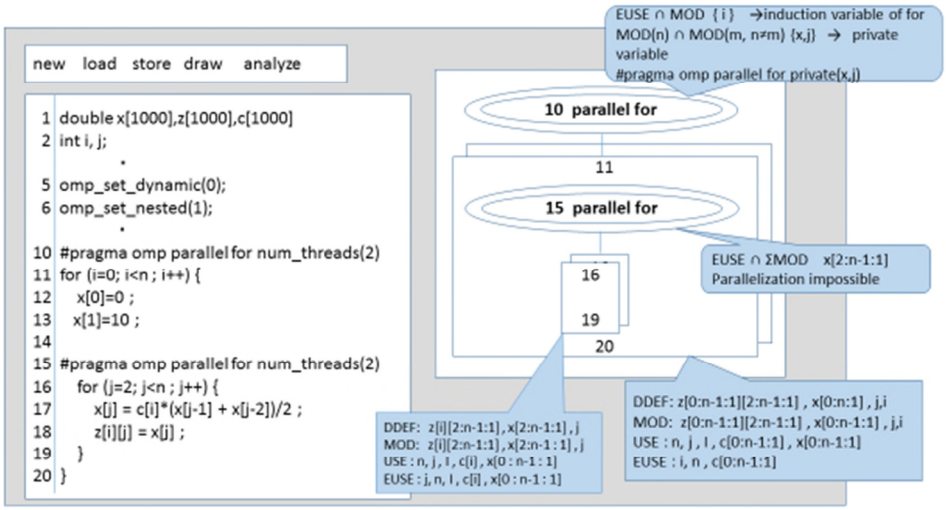

Figure 2. Display screen

Figure 2 shows the display screen and an analytical example. In the editing window on the left, the user performs the parallelization program coding using the C programming language script and OpenMP directives. The command "draw" is selected for the tool to display a parallel structure diagram on the left. The details of this parallel structure diagram are presented in Section III. The command 'analyze' is selected to enable the selection of the diagram quadrangle and elliptical shape (line number and OpenMP directive). Selecting one of these displays the parallel block access regions for that OpenMP directive and the parallelization decision.

III. GRAPH DRAWING OF OPENMP DIRECTIVE ANALYTICAL RESULTS

The tool analyzes syntax and context of OpenMP directives in the C source program, and these directives are reflected in the intermediate representation. This displays the diagram expressing the parallel structure in a graph from this intermediate representation. This is a graph structure with quadrangles expressing the parallel execution unit and ellipses expressing parallel execution direction as nodes.

Quadrangles do not display the execution directions merely by inserting the first and last direction numbers. Ellipses have line numbers and OpenMP directives as labels.

Although there are many OpenMP directives, the current parallel structure diagrams are expressing only for the following three basic types thought necessary for beginners as subjects of analysis.

(1)#pragma omp parallel

(2)#pragma omp parallel for

(3)#pragma omp parallel sections and #pragma omp section

The "parallel for" for the do-all-type parallel processing is expressed in double ellipses, and their loops are expressed by overlapping quadrangles. The "parallel sections" that express parallel-case type parallel processing are single ellipses. The nested parallel execution is expressed by drawing ellipses and quadrangles in other quadrangles.

The requirements of parallel structure graph to express directives are the following:

- To distinguish between the execution of same statements in parallel for the number of threads (#pragma omp parallel) and the execution of different statements in parallel (#pragma omp parallel for, #pragma omp parallel sections).
- To arrange statements to be executed simultaneously, side by side.
- To arrange sequential statements vertically and clarify the order of execution by using connected line.
- To show the synchronization point.
- To disclose parallel nesting structure.

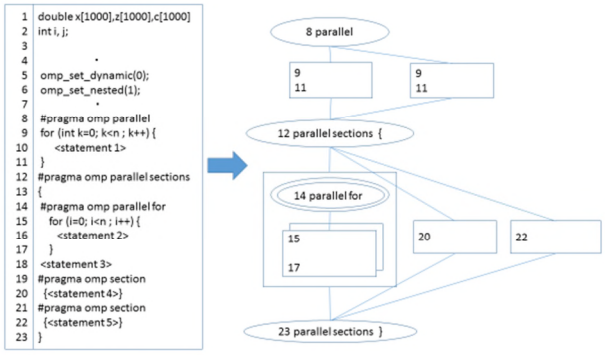

Figure 3. Source program and parallel structure diagrams

Figure 3 shows an example of source program and parallel structure graph. Since the eighth line is a "parallel" directive, it directs to execute the entire following for-loop parallel by the thread number. On the other hand, the 14th line is a "parallel for" directive that divides the loop

 ISBN: 978-1-61208-666-8

repetition and directs to execute by dividing them in a parallel manner. This is expressed with a double ellipse directive and overlapping quadrangles such that the difference can be intuitively understood. The 12th line is a "parallel section" directive, and the quadrangles are able to execute parallel and the synchronizing point for the "}" in the 23rd line to clarify its scope. While internal statements of each quadrangle and the overlapping quadrangle execute sequentially, when there is any parallel directive graph shows nested parallel execution.

IV. STATIC ANALYSIS OF DATA ACCESS REGION

When the user selects the quadrangle in a generated parallel structure graph from an OpenMP directive analysis, the tool finds and displays the data access region by its execution. Additionally, when user selects the ellipse that expresses parallel directive, the tools decides the parallel execution feasibility. An example is shown in Figure 2.

Access region analysis to decide parallelization feasibility analyses what regions are accessed in what order according to a control flow.

A. Access Types

Four data access types are available:

- Possible use (USE)
  Data that might be used within a certain scope (flow graph pass)
- Possible exposed use (EUSE: Exposed USE)
  Data that might be used within a certain scope before definition (flow graph pass)
- Possible definitions (MOD: MODified)
  Data that might be updated within a certain scope before definition (flow graph pass)
- Definitely defined (DDEF: Definitely Defined)
  Data that is definitely updated within a certain scope (flow graph pass)

The 'flow graph pass' above widens the scope of analysis to the parallelization block through the process of one statement → basic block → loop i-th iteration → all repetitions loop → outer loop.

While the 'possible use' and 'possible definitions' are control flow insensitive, 'possible exposed use' and 'definitely defined' are control flow sensitive. These regions are related as follows:

Possible use ⊆ Possible exposed use

Possible definitions ⊆ Definitely defined

As understood from the analytical methods in Section V, the 'possible use' and 'possible definitions' are required to guarantee safety.

B. Method 1: [fusing]

In the if-then-else structure, when node 1 is 'then' and node 2 is 'else', the tool integrates the access regions as in Figure 4 (+ is union and * is intersection).

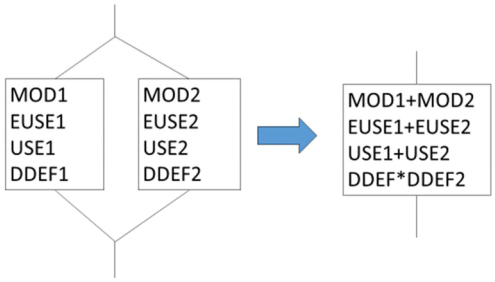

Figure 4. Access region integration (conditional branches)

C. Method 2: [join]

After fusing the if-then-else structure, the nodes sometimes line up in a row. Node 1 is the priority node and node 2 is the next node. The tool integrates the access region as in Figure 5 (- is the difference set excluding the intersection set from the first operand).

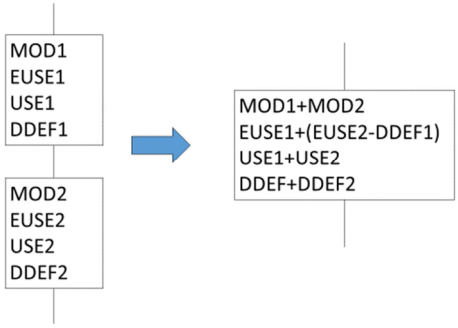

Figure 5 . Access region integration (connection)

D. Method 3: [expansion of loops]

Concerning loops, the access region of the i-th iteration is analysed by *Method 1* and *Method 2* and the access region of the entire loop is analysed, as shown in Figure 6. The information of data access in a loop is expanded.

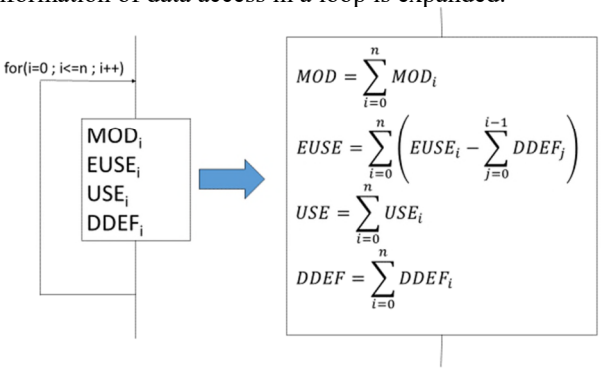

Figure 6. Access region expansion

The balloons indicating the quadrangles in Figure 2 contains an example of the analysis results.

V. PARALLEL EXECUTION FEASIBILITY DECISION

By using the parallel structure graph, the user knows feasibility of parallel execution. The tool decides whether each iteration of a loop can be executed independently for the do-all type, and for the parallel-case type the tool decides whether the parallel blocks surrounded by the section directives can be executed independently. When dependency that impedes parallel execution occurs, the tool displays it as the reason of impossible of parallelization. In some cases, the usage of the reduction instructions and the privatization of variables are confirmed.

Decisions of parallel execution are conducted using access regions as follows.

A. do-all Type

The entire loop-accessed region is checked for reliance upon the following three types of loop-carried data dependence.

$$\forall_i (0 \le i \le n, n: loop\ iteration\ number)\ EUSE_i \cap \sum_{j=0}^{i-1} MOD_j \neq \emptyset \quad (1)$$

→ loop carried flow dependence

$$\forall_i (0 \le i \le n, n: loop\ iteration\ number)\ MOD_i \cap \sum_{j=0}^{i-1} (USE_j - EUSE_j) \neq \emptyset \quad (2)$$

→ loop carried anti dependence

$$\forall_i (0 \le i \le n, n: loop\ iteration\ number) MOD_i \cap \sum_{j=0}^{i-1} MOD_j \neq \emptyset \quad (3)$$

→ loop carried output dependence

When condition (1) is satisfied, confirming data that is detected causes loop carried data dependence. It becomes parallelization impeding factor. When conditions (2) and (3) are satisfied, confirming data that is detected causes loop carried anti and output data dependence. In this case, parallelization might be possible by privatisation of these confirming data. The tool recommends users to add private clause to parallel for directive.

The balloon indicating the ellipse with the parallelization direction in Figure 2 contains an example of the analysis results.

B. parallel-case Type

In a parallel-case-type parallel processing, whether all quadrangles that are connecting the parallel section ellipses can be independently executed is decided as follows. They are similar to do-all case. If defined area of a given section is not overlapping with defined and used regions of any other sections, these sections can be executed independently. The overlapping of regions causes memory hazard.

$$\forall_i (0 \le i \le n, n: section\ number)\ EUSE_i \cap \sum_{j=0}^{n} MOD_{j \neq i} \neq \emptyset \quad (4)$$

→ flow dependence

$$\forall_i (0 \le i \le n, n: section\ number)\ MOD_i \cap \sum_{j=0}^{n} (USE_{j \neq i} - EUSE_{j \neq i}) \neq \emptyset \quad (5)$$

→ anti dependence

$$\forall_i (0 \le i \le n, n: section\ number)\ \ MOD_i \cap \sum_{j=0}^{n} MOD_{j \neq i} \neq \emptyset \quad (6)$$

→ output dependence

The case of condition (4) is satisfied and there is flow dependence, which inhibit parallel execution without any synchronization. When condition (5) or condition (6) is satisfied, the tool recommend user to privatize confirming data that is detected.

VI. CONCLUSION

In a structure as presented herein, providing a parallel program development environment allows the meaning of the written OpenMP directives to be easily understood and mistakes in directives to be easily recognized by beginners not accustomed to parallel processing. Further, this enables the detection and correction of errors peculiar to parallel processing at an early development stage for an accurate static analysis. Inserting OpenMP directives into C programs, such as parallel structures graph enables the easy understanding of parallel structure mistakes and missing synchronous processes because when necessary OpenMP directive is missed out the graph does not have parallel structure. Then the tool makes comments reason why the tool cannot make parallel structure.

Currently, the prototype of the proposed tool is under developing. The GUI specifications have developed as they are considered. We are going to connect the result of static analysis to parallel structure graph. In the future, we would like to increase the types of OpenMP directives for analysis, display complex synchronous processes in an easily understood manner, and provide appropriate advice from the analytical results. Once the parallel structure specifications are established, we would like for users to draw parallel structure graph, input execution statements in them, and for the tool to generate C and OpenMP source programs.

REFERENCES

[1] http://www.openmp.org/, “HOME OpenMP”, 2018.03.19

[2] B. Chapman, G. Jost, and R. Van Def Pas, “Using OpenMP: Portable Shared Memory Parallel Programming”, MIT Press, 2008

[3] R. Chandra, R. Menon, L. Dagum, D. Kohr, D. Maydan, J. McDonald, “Parallel Programming in OpenMP”, Morgan Kaufmann, 2000

[4] O. Harnandez, C. Liao, and B.Chapman, “Dragon: A Static and Dynamic Tool for OpenMP”, Internationa Workshop on OpenMP Applocation and Tools, pp.54-66, 2004

 ISBN: 978-1-61208-666-8

[5] https://pm.bsc.es/ompss, “Programming Models@BSC”, 2018.03.20

[6] http://coins-compiler.osdn.jp/international/index.html, “COINS project”, 2018.03.19

[7] K. Iwasawa, Automatic Parallelizing “Method of Loops by Conditional Region Analysis”, Proceedings of the 16th IASTED International Conference Applied Informatics, pp.310-313, 1998

[8] T., Watanabe, T. Fujise, K. Mori, K. Iwasawa, and I. Nakata, “Design assists for embedded systems in the COINS Compiler Infrastructure”, Proceedings of the 10th International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems, pp.60-09, 2007

 ISBN: 978-1-61208-666-8

Empirical Exploration of the Software Integration Success Factors in Global Software Development

Analyses based on Company Size and Practitioners' Experiences

Muhammad Ilyas
Software Engineering Research Group (SERG_UOM),
Department of Computer Science & IT,
University of Malakand, KPK, Pakistan
e-mail: milyasmkd@uom.edu.pk

Siffat Ullah Khan
Software Engineering Research Group (SERG_UOM),
Department of Computer Science & IT,
University of Malakand, KPK, Pakistan
e-mail: siffatullah@uom.edu.pk

***Abstract* — Software Integration is the most important and complicated phase of software development process. The integration phase becomes even more challenging in Global Software Development (GSD) environment. In our previous study, we identified nine Critical Success Factors (CSFs) using Systematic Literature Review (SLR). Further, for validation of the identified CSFs and for identification of additional success factors, we conducted an industrial survey in GSD environment. In this paper, we present some important analyses of the identified software integration CSFs in GSD environment, based on practitioners' experiences and company size, through industrial survey.**

Keywords-Software Integration; Success Factors; Empirical Study; Global Software Development.

I. INTRODUCTION

The advances in Information and Communication Technologies (ICTs) have resulted in an increase in software use and its size. The software development process has also changed from local to global software development [1]. Global software development paradigm has been adopted by many software vendors, from the last two decades, because of the perceived benefits that can be gained from GSD [1] e.g., cost savings, reduced time to market, proximity to market and customers' access to large skilled labor force, etc. However, in spite of the benefits gained from GSD, vendors also face communication, coordination, knowledge sharing and control problems due to temporal, cultural and linguistic differences [2]-[5]. These problems have also made software integration process more complicated [6]-[8]. Many of the uncovered problems of the previous phases start appearing in the integration phase [9]. These problems not only increase the workload of the global teams but also decrease the quality of the final working product. Researchers reported that more than 50% of the software development projects suffer from cost overrun and/or time overrun problem(s) due to the complexities and incompatibilities found at the software integration stage [10]. Keeping in mind the importance of the integration stage, we proposed the following research questions.

RQ-1: What are the critical success factors (CSFs), as identified in the literature and real-world practice, to be adopted by GSD vendors at various stages of the product integration in GSD environment?

RQ-2: Do the identified critical success factors, as identified in the survey, vary with the level of experience?

RQ-3: Do the identified critical success factors, as identified in the survey, vary with the organization size?

In order to answer RQ1, we identified a list of nine Critical Success Factors (CSFs), as shown in Table 1, in our previous study using Systematic Literature Review (SLR) method. To answer RQ1, Table 1 shows a list of nine Critical Success Factors (CSFs) identified in our previous study, via Systematic Literature Review (SLR) method [11]-[13]. These findings were further validated through a questionnaire survey in the industry. In this paper, we present analyses of the empirical data regarding the identified CSFs based on different variables such as expert's level of experience and organization's size. Thus, we have tried to answer RQ2 and RQ3 in this paper, whereas RQ1 has already been published [11][13].

 ISBN: 978-1-61208-666-8

TABLE 1. LIST OF SOFTWARE INTEGRATION CSFS

S.No	Software integration Critical Success Factor (CSFs)
1	Consistency in requirements and architecture design
2	Intra and inter team communication and coordination
3	Component/unit testing prior to integration
4	Advance and uniform development environment and training
5	Efficient incremental/continues integration
6	Efficient specification for interface compatibility
7	Proper documentation & configuration management
8	Early integration planning and centralized P3 management
9	Careful evaluation of the Commercial Off-The-Shelf/Open Source Software (COTS/OSS) components

The remaining of the paper is organized as follows. In Section 2, background and motivation is presented, while the research methodology is presented in Section 3. Results are presented in Section 4. Finally, Section 5 discusses the limitations of the study and Section 6 presents the conclusion and future work.

II. BACKGROUND

Lorson [9] defines the integration process as a set of procedures for combining components into one larger component, product, subsystem or system. It is the integration stage that enables the organization to assess the overall system functionality and performance that a system may have. In software systems, the integration is the first stage where the overall results of the software development efforts can be observed. Thus integration is a critical phase in the overall software development process.

Paloheimo [14] reported that "as the integration phase is usually the last to follow in a software development process, the unnoticed problems in the preceding phases tend to accumulate in this final phase". The author recommended joint/shared milestones, and incremental integration for successful integration of the software components in the GSD environment.

Van Moll et al. [10] report that the majority of the GSD projects suffer because of the integration complexities. The authors of the study recommended good planning, better monitoring and control, and assigning responsibilities to each and every team member in a well defined manner.

Vasilescu et al. [15] have quantitatively analyzed the continuous integration practice of software engineering. They have concluded that the success or failure of a build process is dependent on the way the code is modified. The code can be modified in two ways:

- Direct change in the code: In this case, a small group of developers, who have the write access to the main project repository, modifies the code.
- Indirect/pull request: In this case, developers who fork the main repository, change their copies locally and tender pull request for review and merge.

Their analysis showed that pull request method of code change is more likely to cause integration testing failures as compared to the direct method. The main limitation of their results is their applicability to open source projects only.

Adams et al. [16] reported in an empirical study that, although the reuse of COTS/OSS components is the best practice, the integration process of these components may also introduce unexpected maintenance costs. They pointed out a need of increased empirical research in software engineering for successful reuse and integration of COTS/OSS software components [11][13].

III. RESEARCH METHODOLOGY

The empirical methods such as case studies, controlled experiments, surveys and post-mortem analysis are essential to the researchers for evaluation and validation of research results in the field of software engineering because the software development process is human intensive work [17]-[19]. In survey design, a survey or questionnaire is administered to a small group of people, also called the *sample*, for identification of trends in characteristics, opinions, attitudes or behavior of a large group of people, also called *population* [20]. Interview and questionnaire are the two main methods of gathering the quantitative or qualitative data. In both methods, a sample representing a population is studied. The results obtained from the survey are analyzed for derivation of explanatory and descriptive conclusions. These conclusions are then generalized to the population from which the sample was taken and studied [17]. In view of the available resources and diverse range of respondents, we have used the questionnaire method as the data collection tool.

The purpose of conducting the survey was to validate the findings of the SLR through industry practitioners and to identify new practices, if any. A similar approach has been used by other researchers [5][6][18].

 ISBN: 978-1-61208-666-8

We have designed the questionnaire survey based on the inputs from our previously published SLR study [21]. The questionnaire survey was properly conducted as done by other researchers [5][22]. We have used both open and close ended questions in this survey. The close ended questions were used as an instrument for collecting self-reported data. In our case, we have used the close ended questions for collecting data about the software integration success factors identified through SLR. We have also used open ended questions to gain the tacit knowledge on the success factors from the industry experts.

The questionnaire used in the survey was designed for eliciting the significance that each respondent has placed on each software integration success factors as identified through SLR. In order to expose the importance of each factor, we have used a seven point Likert scale i.e., Extremely Agree, Moderately Agree, Slightly Agree, Not Sure, Slightly Disagree, Moderately Disagree and Extremely Disagree. The respondents were requested to mention each practice relative value. We used a 7 point Likert scale in the survey, however, for the analysis purposes mentioned in this paper, we have considered Extremely Agree (EA) view point of the survey participants. The number of responses got for the other 6 view points were very low and are therefore not analyzed in this paper.

IV. RESULTS

This section discusses the results and examines the identified software integration critical success factors for each of the Research Questions stated in Section I.

RQ-2 Do the identified critical success factors, as identified in the survey, vary with the level of experience?

We received consent from 232 experts for participation in the survey. A total of 99 experts participated in the survey from 22 different countries. We received a total of 96 valid responses from participants of the questionnaire survey and have used seven point Likert scale (EA: Extremely Agree, MA: Moderately Agree, SA: Slightly Agree, NS: Not Sure, SD: Slightly Disagree, MD: Moderately Disagree and ED: Extremely Disagree). In order to answer RQ-1, we classified the survey participants into three groups, as shown in Table 2, based on their experience level, as follows:

- Junior level experts (JLE): 1 to 5 years experience
- Intermediate level experts (ILE): 5+ to 10 years experience
- Senior level experts (SLE): 10+ experience

It should be noted that these three classes of experts were defined after discussion with the industry experts and external reviewers. Other researchers may however define their own criteria for deciding different levels for experts.

TABLE 2. SUCCESS FACTORS, EXTREMELY AGREE VIEW POINT OF EXPERTS HAVING DIFFERENT EXPERIENCE LEVELS

Critical Success Factors (CSFs)	**Expert's experience level**			**Chi Square Test (Linear-by-linear Association ∝=0.05, Df =1)**	
	Junior (1 to 5y) (n=39)	***Intermediate (5+ to 10 y) (n=26)***	***Senior (10+ y) (n=31)***		
	% of EA	***% of EA***	***% of EA***	X^2	***P***
CSF1-Consistency in requirements and architecture design	72	81	87	2.462	0.117
CSF2-Intra and inter team communication and coordination	74	58	77	3.897	0.048
CSF3-Component/Unit testing prior to integration	54	65	55	0.20	0.888
CSF4-Advance & uniform development environment and training	38	54	45	0.385	0.535
CSF5-Efficient incremental/continuous integration	38	46	48	0.710	0.399
CSF6-Efficient specification for interface compatibility	31	46	39	0.548	0.459
CSF7-Proper documentation & configuration management	44	42	45	0.014	0.904
CSF8-Early integration planning and centralized P3 management	23	27	35	1.275	0.259
CSF9-Careful evaluation of the COTS/OTS components	51	54	55	0.090	0.765

 ISBN: 978-1-61208-666-8

The data in Table 2 shows that all CSFs excluding CSF8 "Early integration planning and centralized P3 management" have been cited by >=30% in the sample of extremely agree responses from the three levels of experts. The most common success factors which have >=50% of extremely agree responses in the sample, across all three level of experts are CSF1-"Consistency in requirements and architecture design", CSF2-"Intra and inter team communication and coordination", CSF3-"Component/Unit testing prior to integration" and CSF9-"Careful evaluation of the COTS/OTS components". It is worth mentioning that the factor "Consistency in requirements and architecture design" is the top ranked factor for all three experience levels of experts. Therefore, proper care should be taken at the design time of software architecture and gathering and specification of requirements because consistent software architecture is positively correlated with the ease of the integration process [23]. Similarly, Kommeren et al. [24] suggested that, for achieving a unified interpretation of requirements, they should be discussed repeatedly with all the development teams. This will result in an optimal design of software components that can be easily integrated. On the other hand, any deficiency in the common understanding of requirements may yield poor design decisions leading to delay in the integration process and the project as a whole.

RQ-3: Do the identified critical success factors, as identified in the survey, vary with the organization size?

According to the Australian Bureau of Statistics [25] definition of organization size, we divided the questionnaires on the basis of organization size into three groups as follows:

- Small (<20 employees)
- Medium (20 – 199 employees)
- Large (>=200 employees)

In order to answer RQ-3, the distribution of the success factors reported by various groups of experts, in the survey, from the three size of organization, is presented in Table 3.

The data in Table 3 shows that all success factors have cited as extremely agree across various groups of experts in all three types of organizations. It should also be noted that all CSFs have been reported with >=30% by experts of all three size organizations except CSF8-"Early integration planning and centralized P3 management", which has 20% occurrence in the large size organization. The reason of low frequency for CSF8 may be that large organizations may have already implemented better planning and management for the activities related to software integration. Again, there are some factors which have got >=50% in the extremely agree response sample in two or more than two types of organizations. These factors are CSF1-"Consistency in requirements and architecture design", CSF2-"Intra and inter team communication and coordination", CSF3-"Component/Unit testing prior to integration", CSF4-"Advance & uniform development environment and training", CSF7-"Proper documentation & configuration management" and CSF9-"Careful evaluation of the COTS/OTS components".

It should be noted that the CSF1-"Consistency in requirements and architecture design" and CSF2-"Intra and inter team communication and coordination" are the two top most ranked factors which have got >=50% across the experts of all the three size of organizations i.e., small, medium and large. Further, Chi Square Test shows that there is no significant difference because no column has $p<0.05$. Hence, it is obvious that these factors should be implemented on priority basis in organizations of all sizes.

TABLE 3. SUCCESS FACTORS, EXTREMELY AGREE VIEW POINT OF EXPERTS ACROSS VARIOUS SIZE OF ORGANIZATIONS

Critical Success Factors (CSFs)	**Company size**			**Chi Square Test (Linear-by-linear Association ∝=0.05, Df =1)**	
	Small (n=16)	***Medium (n=36)***	***Large (n=44)***		
	% of EA	***% of EA***	***% of EA***	X^2	***P***
CSF1-Consistency in requirements and architecture design	63	89	77	0.389	0.533
CSF2-Intra and inter team communication and coordination	75	83	59	3.145	0.076
CSF3-Component/Unit testing prior to integration	63	67	48	1.983	0.159
CSF4-Advance & uniform development environment and training	50	53	36	1.592	0.207
CSF5-Efficient incremental/continuous integration	56	44	39	1.401	0.237
CSF6-Efficient specification for interface compatibility	38	44	32	0.009	0.926
CSF7-Proper documentation & configuration management	63	50	32	2.245	0.234
CSF8-Early integration planning and centralized P3 management	44	31	20	3.260	0.071
CSF9-Careful evaluation of the COTS/OTS components	56	67	41	2.646	0.104

 ISBN: 978-1-61208-666-8

V. LIMITATIONS

The data presented for analysis in this paper was obtained by conducting a questionnaire survey in the GSD industry. A general problem with the survey is that it has a very low response rate and has the possibility of subjective biasness. The results of the survey exhibit opinions of the respondents about a phenomenon under investigation. Literature reveals that the opinions obtained through a survey may be biased as well as different from the real population distribution [26]. In our study, we have tried to explore the perceptions and experiences of GSD experts, but it was not possible to verify these perceptions and experiences directly. Moreover, practitioner's opinions and perceptions may not be accurate. Additionally, the respondents of the questionnaire survey were self-selecting. However, the results of piloting studies give a satisfactory level of internal validity since the variables incorporated in this research study were obtained from comprehensive literature review and piloting of the questions survey. The external validity is addressed by receiving survey responses from a total of 96 experts, among which 56 experts belong to 22 different countries, providing a good representative sample.

VI. CONCLUSION AND FUTURE WORK

The analyses presented in this paper show that our identified software integration practices are important from various experts point of view. This means that implementation of these success factors may help GSD vendors to easily and effectively integrate their software components. The frequency percentages of each CSF in the questionnaire survey (EA: extremely agree) show the relative importance of each factor within the group of software integration success factors. The implementation of software integration CSFs, especially those reported with greater percentage, may boost the performance of GSD vendors by effectively integrating their software components.

Further analysis of the CSFs based on different variables, such as expert's position, time etc. is reserved for future work.

The ultimate aim of this research work is to develop our proposed Software Integration Model (SIM) for GSD vendors [27].

REFERENCES

[1] P. J. Agerfalk, B. Fitzgerald, H. H. Olsson and E. O. Conchuir, "Benefits of global software development: the known and unknown," in *Making Globally Distributed Software Development a Success Story*, ed: Springer, 2008, pp. 1-9.

[2] R. A. Khan, S. U. Khan and M. Niazi, "Communication and Coordination Challenges Mitigation in Offshore Software Development Outsourcing Relationships: Findings from Systematic Literature Review," in *The Tenth International Conference on Software Engineering Advances (ICSEA 2015)*, Barcelona, Spain, 2015, pp. 45-51.

[3] M. Jimenez, M. Piattini and A. Vizcaino, "Challenges and improvements in distributed software development: A systematic review," *Advances in Software Engineering*, vol. 2009, pp. 1-14, 2009.

[4] P. J. Agerfalk *et al.*, "A framework for considering opportunities and threats in distributed software development," in *Proceedings of the International Workshop on Distributed Software Development*, Paris, 29, 2005, pp. 47-61.

[5] A. W. Khan and S. U. Khan, "Solutions for Critical Challenges in Offshore Software Outsourcing Contract," *Pakistan Academy of Sciences*, vol. 52, pp. 331-344, 2015.

[6] M. Niazi, S. Mahmood, M. Alshayeb and A. Hroub, "Empirical investigation of the challenges of the existing tools used in global software development projects," *IET Software*, vol. 9, pp. 135-143, 2015.

[7] M. Ilyas and S. U. Khan, "Software integration in global software development: Challenges for GSD vendors," *Journal of Software: Evolution and Process*, vol. 29, pp. 1-17, 2017.

[8] M. Ilyas and S. U. Khan, "Software integration challenges for GSD Vendors: An exploratory study using a systematic literature review," *Journal of Computers*, vol. 12, pp. 416-422, 2017.

[9] S. Larsson, "Key Elements of the Product Integration Process," Ph.D. Thesis Proposal, Department of Computer science and Electronics, Malardalen University Sweden, Malardalen, 2007.

[10] J. Van Moll and R. Ammerlaan, "Identifying Pitfalls of System Integration-An Exploratory Study," in *IEEE International Conference on Software Testing Verification and Validation Workshop (ICSTW)*, Lillehammer, 2008, pp. 331-338.

[11] M. Ilyas and S. U. Khan, "Software Integration in Global Software Development: Success Factors for GSD vendors," in *16th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2015)*, Takamatsu, Japan, 2015, pp. 119-124.

[12] M. Ilyas and S. U. Khan, "Software Integration Challenges in Global Software Development Environment: A Systematic Literature Review Protocol," *IOSR Journal of Computer Engineering (IOSRJCE)*, vol. 1, pp. 29-38, 2012.

[13] M. Ilyas and S. U. Khan, "An Exploratory Study of Success Factors in Software Integration for GSD Vendors " *Proceedings of the Pakistan Academy of Sciences*, vol. 53, pp. 239–253 (2016), 2016.

[14] H. Paloheimo, "Feasibility of SW Architecture Integration in a Distributed R&D Environment," HUT / SoberIT 2003, Fall T-76.6512003.

[15] B. Vasilescu, S. Van Schuylenburg, J. Wulms, A. Serebrenik and M. G. van den Brand, "Continuous integration in a social-coding world: Empirical evidence from GitHub," in *International Conference on Software Maintenance and Evolution (ICSME)*, BC, Canada, 2014, pp. 401-405.

[16] B. Adams, R. Kavanagh, A. E. Hassan and D. M. German, "An empirical study of integration activities in distributions of open source software," *Empirical Software Engineering*, pp. 1-42, 2015.

 ISBN: 978-1-61208-666-8

[17] C. Wohlin, M. Höst and K. Henningsson, "Empirical research methods in software engineering," in *Empirical methods and studies in software engineering*, ed: Springer, 2003, pp. 7-23.
[18] T. Ambreen, N. Ikram, M. Usman and M. Niazi, "Empirical research in requirements engineering: trends and opportunities," *Requirements Engineering*, vol. 21, pp. 1-33, 2016.
[19] M. Alshayeb, "Empirical investigation of refactoring effect on software quality," *Information and software technology*, vol. 51, pp. 1319-1326, 2009.
[20] J. W. Creswell, *Educational research: Planning, conducting, and evaluating quantitative and qualitative research*, 4th ed.: Prentice Hall, 2008.
[21] M. Ilyas and S. U. Khan, "Practices for Software Integration Success Factors in GSD Environment," in *15th International Conference on Computer and Information Science (ICIS), 2016 IEEE/ACIS*, Okayama, Japan, 2016, pp. 1-6.
[22] M. Niazi, D. Wilson and D. Zowghi, "A Framework for Assisting the Design of Effective Software Process Improvement Implementation Strategies," *Journal of Systems and Software*, vol. 78, pp. 204-222, 2004.
[23] R. Land and I. Crnkovic, "Software systems in-house integration: Architecture, process practices, and strategy selection," *Information and Software Technology (IST)*, vol. 49, pp. 419-444, 2006.
[24] R. Kommeren and P. I. Parviainen, "Philips experiences in global distributed software development," *Empirical Software Engineering*, vol. 12, pp. 647-660, 2007.
[25] D. Trewin, "Small Business in Australia," Australian Bureau of Statistics, Canberra 2001.
[26] B. A. Kitchenham *et al.*, "Preliminary guidelines for empirical research in software engineering," *IEEE Transactions on software engineering*, vol. 28, pp. 721-734, 2002.
[27] M. Ilyas and S. U. Khan, "Software Integration Model for Global Software Development," in *15th International Multitopic Conference (INMIC)*, Islamabad, Pakistan, 2012, pp. 452-457.

 ISBN: 978-1-61208-666-8

Applying Quality Requirements Framework to an IoT System

Tsuyoshi Nakajima
Department of Information Science and Engineering
Shibaura Institute of Technology
Tokyo, Japan
e-mail: tsnaka@shibaura-it.ac.jp

***Abstract*—Modern information and communication technology systems more focus on their quality requirements since they have been increasing their complexity. This paper shows how the quality requirements framework of the ISO/IEC 25030 can be applied to an Internet of things application, Elderly monitoring system. The results of this application indicate the usefulness of the framework.**

Keywords. Quality requirements; SQuaRE; IoT.

I. INTRODUCTION

Information and Communication Technology (ICT) systems are increasingly used to perform a wide variety of organizational functions and personal activities. The quality of these products enables and impacts various business, regulatory and information technology stakeholders. High-quality ICT systems are hence essential to provide value, and avoid potential negative consequences, for the stakeholders.

To develop such high-quality ICT systems, it is important to define quality requirements, because finding the right balance of quality requirements, in addition to well-specified functional requirements, is a critical success factor to meet the stakeholders' objectives.

Furthermore, the complexity of ICT systems has grown exponentially with the advent of modern digital technologies like Internet of Things (IoT). This has also led to focus on more and more quality requirements that are critical to modern ICT systems.

ISO/IEC 25030 Quality requirements has been published in 2007, and its revision process has been going on to expand its scope from software to ICT systems [1]. The standard belongs to ISO/IEC 25000 series: Systems and software Quality Requirements and Evaluation (SQuaRE) has been developed as the successor of the other standards on product-related quality, including ISO/IEC 9126.

This paper shows how the quality requirements framework of the ISO/IEC 25030 revision works [1], in case that it is applied to an IoT system. Section II explains the quality requirements framework and section III describe the target IoT system, and then the framework is applied to the system in section IV.

II. QUALITY REQUIREMENTS FRAMEWORK

A. Architecture of the SQuaRE series

The SQuaRE series consists of five main divisions and on extension division. The divisions within the SQuaRE series are:

- **ISO/IEC 2500n - Quality Management Division**. The standards that form this division define all common models, terms and definitions used by all other standards in the SQuaRE series. The division also provides requirements and guidance for the planning and management of a project.
- **ISO/IEC 2501n - Quality Model Division**. The standards that form this division provide quality models for system/software products, quality in use, data, and IT services. Practical guidance on the use of the quality model is also provided.
- **ISO/IEC 2502n - Quality Measurement Division**. The standards that form this division include a system/software product quality measurement reference model, definitions of quality measures, and practical guidance for their application. This division presents internal measures of software quality, external measures of software quality, quality in use measures and data quality measures. Quality measure elements forming foundations for the quality measures are defined and presented.
- **ISO/IEC 2503n - Quality Requirements Division**. The standard that forms this division helps specifying quality requirements. These quality requirements can be used in the process of quality requirements elicitation for a system/software product to be developed, designing a process for achieving necessary quality, or as inputs for an evaluation process.
- **ISO/IEC 2504n - Quality Evaluation Division**. The standards that form this division provide requirements, recommendations and guidelines for system/software product evaluation, whether performed by independent evaluators, acquirers or developers. The support for documenting a measure as an Evaluation Module is also presented.

B. Quality requirements and quality models/measures

Quality In Use Requirements (QIURs) specify the required levels of quality from the stakeholders' point of view. These requirements are derived from the needs of various stakeholders. QIURs relate to the outcome when the product is used in a particular context of use, and QIURs can be used as the target for validation of the product.

QIURs can be specified using quality in use model (ISO/IEC 25010 [2]) and measures (ISO/IEC 25022 [4]). **Figure 1** describes characteristics and subcharacteristics of quality in use model.

 ISBN: 978-1-61208-666-8

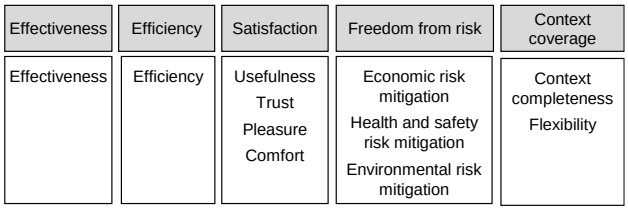

Figure 1. Quality in use model [2]

Product Quality Requirements (PQRs) specify levels of quality required from the viewpoint of the ICT product. Most of them are derived from stakeholder quality requirements including QIURs, which can be used as targets for verification and validation of the target ICT product.

PQRs can be specified using product quality model (ISO/IEC 25010[2]) and measures (ISO/IEC 25023[5]). Figure 2 describes characteristics and subcharacteristics of product quality model.

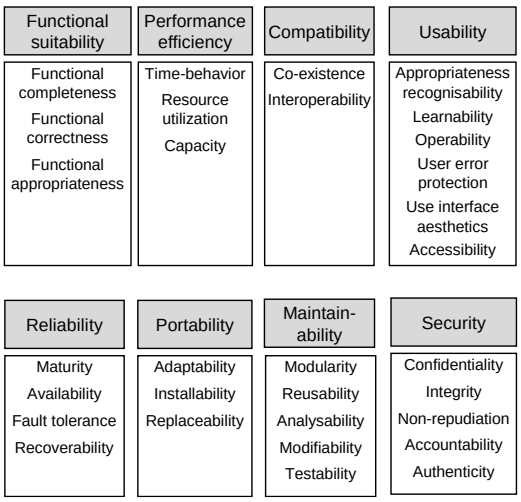

Figure 2. Product quality model [2]

The Data Quality Requirements (DQRs) specify levels of quality required for the data associated with the product. These include requirements derived from QIURs and PQRs of input and output products. DQRs can be used for verification and validation from the data side.

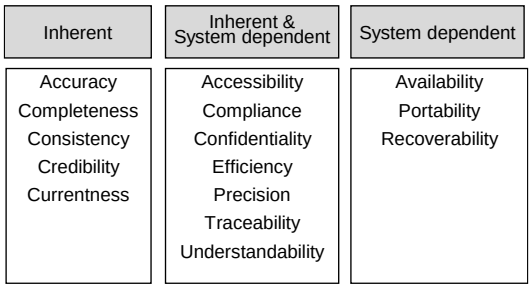

Figure 3. Data quality model [3]

DQRs can be specified using data quality model (ISO/IEC 25012[3]) and measures (ISO/IEC 25024[6]). Figure 3 describes 15 characteristics of data quality model, which are categorized by inherent and/or system dependent.

C. *Quality requiremnets framework*

The revision of ISO/IEC 25030[1] will provide a framework for quality requirements, which consists of concept of the quality requirements, and processes and methods to elicit, define, use and govern them.

There are three important points:

- To elicit quality requirements, not only direct users of the ICT product but also indirect users (using results of the product) and other stakeholders, such as developers, regulatory body, and society at large should be taken into account.
- QIURs should be considered first because most of PQRs are derived from QIURs, and they should be deployed into PQRSs and DQRs of its sub-products (smaller ICT products, software, data, hardware and communication facilities) to meet them.
- Quality requirements should be defined quantitatively, in order not to be vague and unverifiable requirements that depend on subjective judgement for their interpretation.

III. IoT SYSTEM AND TARGET SYSTEM

A. *Characteristics of IoT systems*

The IoT envisages a future in which digital and physical things or objects can be connected by means of suitable information and communication technologies, to enable a range of applications and services. The IoT's characteristics include [7]:

- many relevant stakeholders involvement
- device and network heterogeneity and openness
- resource constrained
- spontaneous interaction
- increased security attack-surface

These characteristics will make development of the diverse applications and services a very challenging task.

B. *Target system*

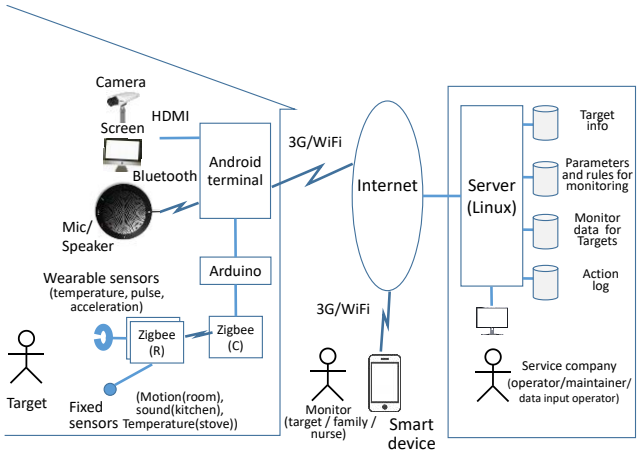

Figure 4. Elderly monitoring system [8]

The target IoT system, to which SQuaRE's quality requirements framework is applied, is Elderly monitoring system. Figure 4 shows its system architecture.

Figure 5 describes use cases of elderly monitoring system.

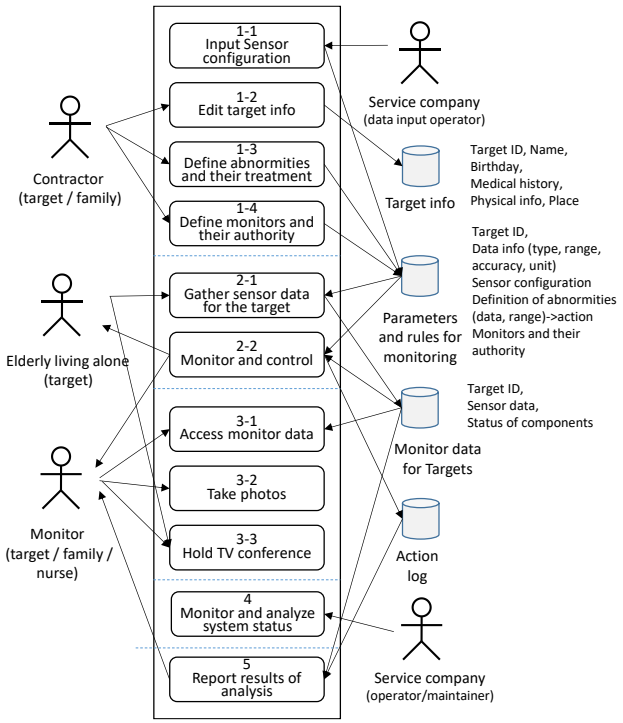

Figure 5. Use cases of elderly monitoring system (written by the author)

IV. APPLICATION OF THE FRAMEWORK

A. *Stakeholder identification and select important QIURs*

The quality requirements framework needs identification of stakeholders of the target system, including not only direct users but also indirect user and other stakeholders.

Table 1 lists identified stakeholders for the elderly monitoring system, including:

- Direct user: contractor, elderly living alone, family, nurse, and service company's operators
- Indirect user: service company's managers
- Other stakeholder: Developer, Ambulance

For all stakeholders, their goals to achieve through using the target system are extracted. The direct users must have use case of the system **(**Figure 5) in which they involved for achieving their goal. The indirect users and other stakeholders do not use the system directly, and the former uses the results of the system and the latter may get influenced from the system in an indirect way. Therefore, they do not have relevant use cases.

Based on stakeholders' goals and use cases, important quality in use characteristics/subcharacteristics (Table 1) should be selected with target outcomes and consequences.

Table 1. QIURs selection based on stakeholders' goal

Stakeholder	Goal	Use case	QIUR (with target outcomes and consequences)
Service company's manager (**indirect user**)	Customer satisfaction	NA	**Usefulness** **Trust**
	Prevention from incidents	NA	**Freedom from risks:** prevention from * incidents by system faults or malfunctions * incidents by normal operation * privacy leakage * malfunction by malicious attach
Service company's operator (**direct user**)	Monitor all equipment, and take actions if something wrong with them.	4	**Efficiency**: system monitor and control **Effectiveness**: preventive actions before disfunction or malfunction
	Maintain and update system and equipment.	1−1	**Efficiency:** maintenance activities
Contractor (**direct user**)	Inform the service company of what he/she wants them to do.	1−2 1−3	**Efficiency**: operation for input **Freedom from risks**: prevention from wrong input
Elderly living alone (**direct user**)	Detect designated abnormalities for the target, and take actions.	2−2	**Effectiveness**: early treatment **Trust**: correct results on good timing
	Obtain his/her own current body condition and behavioral pattern.	5	**Effectiveness**: obtain info on current body condition and behavioral pattern to provide objective insights.
Family (**direct user**)	Confirm target's normality.	3−1 3−2	**Effectiveness**: see target's condition anytime and anywhere
	Be informed of target's serious abnormalities.	2−2	**Trust**: correct results on good timing
			Freedom from risks: prevention from * overlook of serious abnormalities * unnecessary notice on trivial abnormalities
Nurse (**direct user**)	Confirm target's normality.	3−1 3−2	**Effectiveness**: remote nursing **Efficiency**: early notice of patient's abnormalities
	Be informed of target's all abnormalities.	2−2	**Effectiveness**: early treatment **Trust**: correct results on good timing
			Freedom from risks: prevention from overlook of serious abnormalities
	Create reports for asking doctors to diagnose abnormalities.	5	**Efficiency**: automatic reporting
Developer (**Other stakeholder**)	Achieve QCD goal	NA	**Efficiency**: development activities
	Update the system to implement new functions periodically	NA	**Efficiency**: maintenance activities
Ambulance (**Other stakeholder**)	Dispatch ambulance cars on demand (by nurse's call)	NA	**Freedom from risks**: prevention from unnecessary dispatches of ambulance cars

* direct user: person who interacts with the product
* indirect user: person who receives output from a system, but does not interact with the system, for example executive manager, service acquirer

B. *Drivation of PQRs and DQRs*

Figure 6 describes how quality requirements derive others in the system hierarchy.

The primary source of quality requirements is the users, from whom first QIURs for the information system including the target entities are elicited and documented. Then, they evolve into PQRs and DQRs for the target entities. Other stakeholders, such as developers and regulatory bodies, also give some quality requirements on the target entities. Finally, other entities give some requirements as constraints to the target entities, including non-target ICT products, software and data which are connected to or used in the targets, and hardware and communication which are used in them.

Table 2 shows how to derive PQRs and DQRs from QIURs which are partially selected from Table 1. For PQRs and DQRs, important product quality characteristics/ subcharacteristics (Figure 2) and data quality characteristics/ subcharacteristics (Figure 3) are selected to meet the corresponding QIURs.

 ISBN: 978-1-61208-666-8

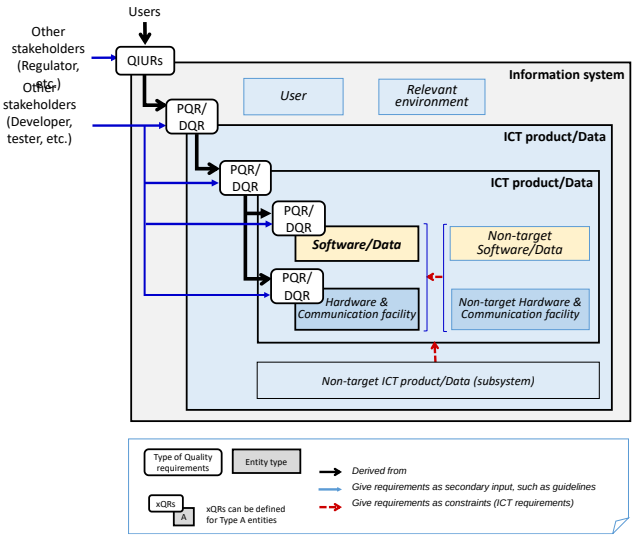

Figure 6. Derivation of quality requirements [1]

Some PQRs for the target product may be deployed into subcomponents to meet them (denoted with ->). DQRs are identified for the data files or data base used in the system (Figure 6).

Table 2. Derivation PQRs and DQRs from QIURs

Stakeholder	Use case	QIUR (with target outcomes and consequences)	PQR	DQR
Service company's manager (**indirect user**)	NA	**Freedom from risks:** prevention from		
		* incidents by system faults or malfunctions	**Maturity** ->**Availability** for server ->**Maturity** for IoT devices	**Recoverability** of all data
			Time-behavior ->Throughput of server	**Efficiency** and **Accessibility** on Monitor data for target
		* incidents by normal operation	**Maturity**: exhaustive testing	**Consistency** and **Currentness** on Monitor data for target
		* privacy leakage	**Confidentiality** on server	**Confidentiality** on Target info
		* malfunction by malicious attach	**Integrity**: IoT devices, network	**Traceability** on Parameters and rules for monitoring
Contractor (**direct user**)	1-2 1-3	**Efficiency**: operation for input	**Operability** and **Accessability** on Web user interface	**Understandability** on Parameters and rules for monitoring
		Freedom from risks: prevention from wrong input	**Learnability** and **User error protection** on : Web user interface	**Accuracy**, **Completeness** and **Consistency** on Parameters and rules for monitoring
Elderly living alone (**direct user**)	2-2	**Effectiveness**: early treatment **Trust**: correct results on good timing	**Functional suitability** and **Functional completeness** for detecting abnormalities	
	5	**Effectiveness**: obtain info on current body condition and behavioral pattern to provide objective insights.	**Functional suitability**: inclusion of useful information	**Understandability** of reports

V. SUMMARY AND FUTURE WORK

Modern ICT systems like IoT systems should put more focus on their quality requirements. This paper provides the brief introduction of ISO/IEC 25000 (SQuaRE) series, which define quality models and measures, and how to define quality requirements and evaluate quality of the ICT products.

And then, the IoT systems' unique characteristics compared to the other information systems are mentioned, including many relevant stakeholders' involvement, device and network level heterogeneity and openness, resource constrained, spontaneous interaction, and increased security attack-surface, which may make development of the diverse applications and services a very challenging task.

To solve this problem, we apply the quality requirements framework of the ISO/IEC 25030 revision to an IoT system, Elderly monitoring system. The results of this application make us believe the usefulness of the framework.

More application of the framework to a variety of IoT systems and much larger scale ones should be needed to clarify its limitations and problems.

REFERENCES

[1] ISO/IEC 25030 DIS, Systems and Software engineering — Quality requirements framework.

[2] ISO/IEC 25010:2011, Systems and Software engineering — System and software quality models.

[3] ISO/IEC 25012:2008, Systems and Software engineering — Data quality model.

[4] ISO/IEC 25022:2016, Systems and Software engineering — Measurement of quality in use.

[5] ISO/IEC 25023:2016, Systems and Software engineering — Measurement of system and software product quality.

[6] ISO/IEC 25024:2015, Systems and Software engineering — Measurement of data quality.

[7] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, "Middleware for internet of things, a survey," IEEE Internet of Things Journal, Vol. 3, No. 1, pp. 70-95, 2016.

[8] S. Okazaki et al, "An Intelligent Space System and its Communication Method to Achieve the Low Energy Consumption," IEEJ-C Vol. 136, No. 12, pp. 1804-1814, 2016 (in Japanese).

 ISBN: 978-1-61208-666-8

Integration of Data Providing and Analyzing System and its Application to Higher Education Institutional Data

Masaaki Ida

National Institution for Academic Degrees and
Quality Enhancement of Higher Education
Tokyo, Japan 187–8587
Email: ida@niad.ac.jp

***Abstract*—There exist various kinds of data providing and analyzing service on Web sites. Japanese College and University Portraits is an information system consisting of databases with Web services for providing information concerning various activities undertaken by universities and junior colleges, covering national, prefectural, municipal, and private institutions. This paper describes the outline of this integrated system and related analysis systems. Especially, we focus on data providing service in several ways including research results conducted by the research department of National Institution for Academic Degrees and Quality Enhancement of Higher Education. A further advanced and integrated data analysis and data visualization system can be developed by using Web APIs with various multivariate analysis methods. Canonical correlation analysis is one of the basic and requisite data analysis and visualization skills for data analysts in this Big Data era. Therefore, because data is received through Web APIs, the development of an integrated data analysis system equipped with canonical correlation analysis is desirable. This article also presents a work-in-progress result of the canonical correlation analysis for higher education institutional data.**

Keywords–Higher education institutional data; data providing; Web API; visualization.

I. INTRODUCTION

Education-related databases are important for college selections or various quality assurance activities, such as reporting and data analysis in higher education institution. Therefore, data service of higher education institutional data is desired to be developed. However, Institutional data of universities, e.g., the number of various kinds of academic staffs, are difficult to analyze because they were not necessarily standardized and integrated in each university itself or even in national level education-related agencies. Some advanced higher education integrated data systems are progressively developing. The most famous and useful system is the Integrated Postsecondary Education Data System [1], which has been developed by National Center for Education Statistics (NCES) in the United States. The system collects and analyzes basic institution information about universities and colleges in the U.S. The system standardizes and accumulates this information nationwide. This system comprehensively holds general and basic institution data. Moreover, this system is equipped with data analysis tools to conduct university comparative analysis. There exist other web-based university database systems in the U.S. and other countries. These databases are well-organized and comprehensive systems with easy Web-based operation on their Web sites. However, in order to cooperate or integrate with other information systems, e.g., in-house database developed in individual institutions, or external database services, more improved systems are expected to be equipped with various Web service functions and standardized data sets. In this paper, in Section II, the integration of data providing and analyzing system in Japan is described. In Section III, Web API and data analysis is described. As an example of data analysis, canonical correlation analysis is introduced with a numerical example.

II. INTEGRATION OF DATA PROVIDING AND ANALYZING SYSTEM

A. Japanese college and university portraits

In Japan, Ministry of Education, Culture, Sports, Science and Technology collects basic information about higher education institutions in Japan. This law-based basic statistical data includes yearly information of higher education institutions, such as the number of faculties or staffs, the number of enrolled students by grade (undergraduate, graduate, foreign student), the number of graduates by subsequent course, the number of those who are employed after graduation by each industry and by occupation, faculties, facilities, and financial data. However, these are published as statistical data, so that detailed information of individual universities are not published.

Japanese College and University Portraits is an information system consisting of database with Web services for providing information concerning various activities undertaken by universities and junior colleges, covering national, prefectural, municipal, and private institutions [2]. System operation started in March 2015. The system is managed by National Institution for Academic Degrees and Quality Enhancement of Higher Education, Japan (NIAD-QE) associated with Promotion and Mutual Aid Corporation for Private Schools of Japan.

The purposes of the system are as follows

- Information Dissemination: The Portrait Website will be used not only by those who intend to participate in higher education as students, but also by stakeholders in various areas of society, such as government and industry. The database is also expected to be an information source contributing to improve international society's understanding of higher education institutions in Japan.
- Monitor and Analysis of Institution Activities: The system is expected to be used by higher education institutions to monitor and analyze the status of their own educational activities for internal quality assurance and enhancement.
- Workload Reduction: Collection and publication of fundamental and standardized data in the database system will assist higher education institutions when they

 ISBN: 978-1-61208-666-8

respond to various surveys and external evaluation. Workload reduction of institutions based on accurate data are expected to be accomplished by the system.

Data items stored in the system are in multiple levels, Institution level and faculty level, for example, general information of higher education institution, objectives of education and research, characteristics, education system organization, campus, university evaluation, student support, policies of education, academic program, admission, faculty, enrollment, scholarship, completion, post graduate pathways, employment, research activities, international activity, student life, financial information and so on.

The database system consists of three databases, three circles in Figure 1, with basic organization data located in the common part of circles, which is regarded as university data warehouse. These database are classified in the following three categories:

- University common publication data: published common education data over national, prefectural, private institution,
- University basic data: corresponding to school basic survey,
- National university evaluation data: used for national university evaluation.

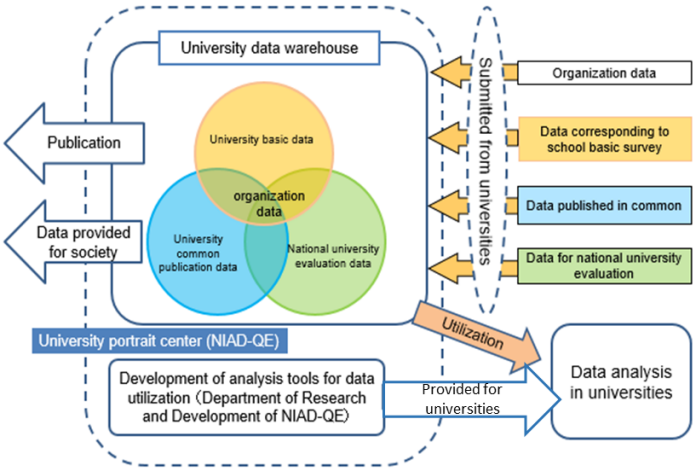

Figure 1. Japanese College and University Portraits.

Each data is registered by data-sheet submission from higher education institutions. The stored data are used for publication and provided for data utilization for the sake of society and universities.

B. Data providing service

Data providing and utilization services are explained in the following ways. These functions are partially equipped in the working system. Data analysis tools for data utilization are now being developed by research department of NIAD-QE.

1) Data providing via Website: The system has search (retrieval) functions, (1) simple search by university name, faculty name and location, (2) detailed search, e.g., by entrance examination, student financial aid and so on, and (3) keyword search. Adding to searching and utilizing the data which are ordinary in table format in Web pages, we can download data tables, and utilize them in spreadsheet software in user side. Then it is possible to extract necessary data for analysis, and to conduct data analysis by using personal analysis tools or personal Business Intelligence (BI) tools on user's local environment, which are popular tools in these days.

2) Data providing via BI tool: Highly-detailed and flexible data analysis can be attained by Structured Query Language (SQL). However, expert ability is required for such advanced treatment of database. In case that we intend to try advanced data analysis without expert ability, full-scale BI tools are candidates of effective analysis with great potential, which is equipped in the Portrait system. BI tool makes it possible for system registered users to utilize the database more conveniently with some useful BI functions, such as filter, formula, chart and drill-down functions. We can generate various kinds of easily understandable data tables and charts, and also generate data analysis report file in PDF format or spread sheet format. This BI tool of the system was used to generate data analysis reports in National University Corporation Evaluation in Japan.

3) Data providing via Web-based analysis system: Data analysis and data visualization tool are being developed by research department of NIAD-QE. Figure 2 shows an example of comparative analysis of universities. We refer European university comparison and visualization systems, U-Map [3] and U-Multirank [4]. They are new higher education transparency tools for multi-dimensional mapping and ranking [5]. In this figure, data table includes selected eight indicators in columns for selected 13 universities in rows for corresponding fiscal year. Values of indicators are transformed into relative classes or groups (e.g., four level: quantile point), which are expressed by the number of star marks. Chart in lower side shows feature of three universities selected from universities in this data table. Fan-shaped parts of the charts, surrounding center circle, correspond to the amount of indicators. Relative analysis by class or group is helpful for understanding whole aspect of higher education institutions with multiple features.

4) Data system integration: Data providing via Web API: Web API is a Web Application Programming Interface for performing computer processing via Internet. This mechanism makes it possible for registered users to access external database through the internet. The advantages to use Web API are to obtain data when necessary, to obtain only necessary part of data by query (search), to obtain standardized and latest data that might be updated recently, and to have possibility to provide more useful and valuable information combining with other multiple external data sources provided by other Web APIs such as official government statistics or location information Web service. Moreover, this type of Web services has an effect for developing application modules with independency, which leads to improvement of maintenance and redesign of database application system.

Research department of NIAD-QE is developing various kinds of Web APIs and their applications which are suitable for data analysis and data dissemination. Web APIs for university basic survey (for national and prefectural) and university financial data have being developed. Output form can be selected in JSON or XML formats. University basic survey sheets consist of detailed university information cards in university level or department level, e.g., institutional structure, faculty member and staff (sheet 7: number of students, number of

 ISBN: 978-1-61208-666-8

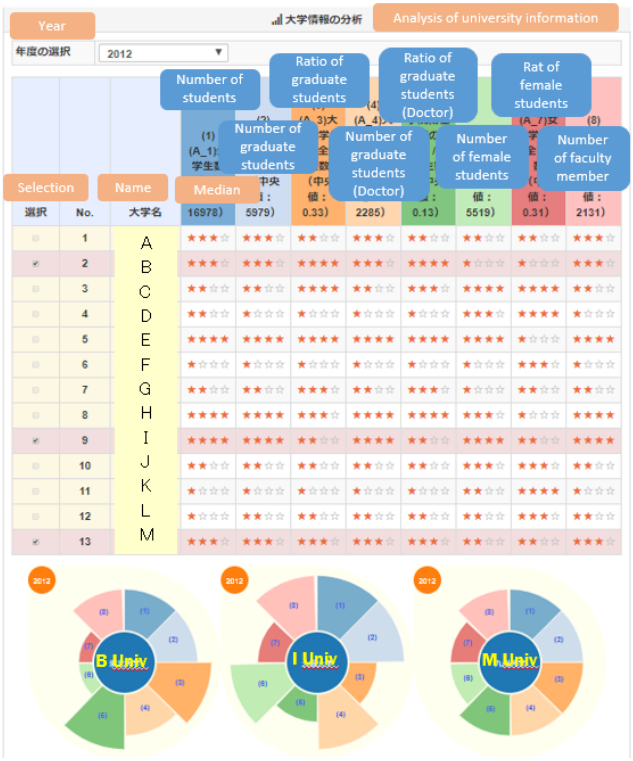

Figure 2. Data Analysis and ata Visualization Tool.

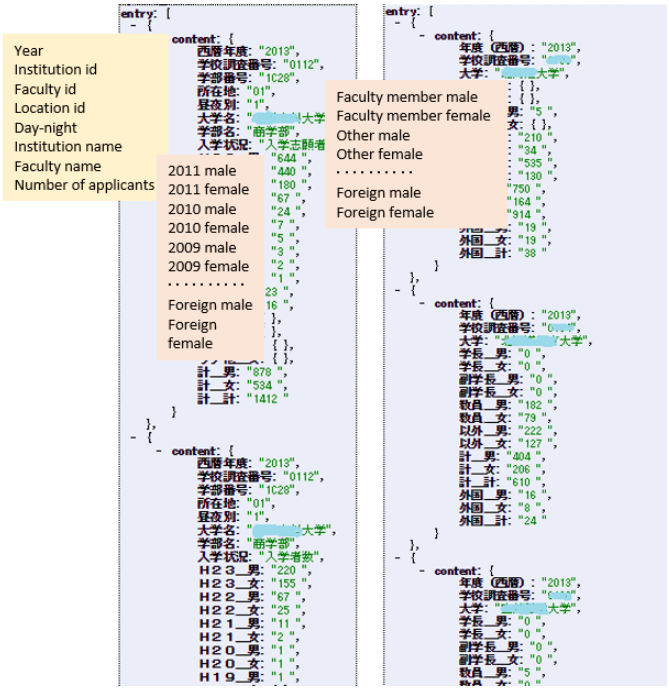

Figure 3. Web API of university basic information (JSON format).

academic staffs), student (sheet 8: number of students of each department), graduate student (sheet 9), foreign student (sheet 11), facility (sheet 20), graduation and employment (sheet 30) and so on. This type of detailed university Web API development is early attempt in higher education field.

The followings are examples retrieved by survey year and institution code of Japanese universities. Figure 3 (left) shows an example of Web API output concerning applicant and enrollment figures by undergraduate school in faculty level. The elements in Japanese language mean university name, faculty name, and the number of undergraduate students in every fiscal year. Figure 3 (right) shows an example of output of the Web API concerning faculty members from survey in university level.

API key is issued for registered user to access and use API functions for security. Retrieved data is provided through cryptographic protocol that provides communication security over computer network.

III. Web API and Data Analysis

A. Analysis and visualization process

Generally, there exist various kinds of data analysis services on Web sites. Further complex and advanced data analysis tool and data visualization applications can be developed by data integration mechanism using Web API functions. By utilizing Web APIs, we can develop flexible integrated Web applications with data tables and charts generation, and data analysis system. With flexibilities of API mechanism more useful and user-friendly data visualization system can be developed.

The analysis and visualization process is as follows:

1) Database query by university name or department name with various indicators is submitted to university information Web API site with API key for registered user, which is also developed by research department of NIAD-QE.
2) Data in JSON or XML format are received by Web programming on server side or client side.
3) Analysis and visualization of various indicators with effective graphic libraries are conducted, and comparison of multiple indicators with sorting functions on data tables or charts is made.
4) Moreover, analysis system can be programmed to combine with other databases using API functions, e.g., various official statistical data API or map API on outer Web service sites. These Web service combination, or mash up programing, can be easily applied using Web API functions.

B. Canonical correlation analysis

Canonical correlation analysis (CCA) is a core analysis method in multivariate analysis field. CCA is a generalized method of corresponding analysis that is useful for questionnaire analysis [6]–[8]. Two multiple variable data matrices, X and Y are expressed with $n \times p$ and $n \times q$ real data matrices, X_R and Y_R. We define the following matrices for data average and deviation:

$$Q_n = I_n - (1/n)\mathbf{1}_n\mathbf{1}_n^{\mathrm{T}}$$

$$X = Q_n X_R, \quad Y = Q_n Y_R$$

where $\mathbf{1}_n$ means $(1, 1, \cdots, 1)^{\mathrm{T}}$, and Q_n means averaging.

Calculating correlation matrices, R_{XX}, R_{YY}, R_{XY}, for X, Y, then, the result of canonical correlation analysis is the singular value and corresponding singular vectors, μ with $\boldsymbol{a}$ and $\boldsymbol{b}$ satisfy the following matrix equation:

$$\begin{pmatrix} R_{XX} & R_{XY} \\ R_{YX} & R_{YY} \end{pmatrix} \begin{pmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{pmatrix} = (1+\mu) \begin{pmatrix} R_{XX} & O \\ O & R_{YY} \end{pmatrix} \begin{pmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{pmatrix}$$

We call the singular value μ as the first canonical correlation coefficient μ^1 for maximum value, and μ^2 for second one, and μ^3 for third one and so on. And its corresponding vectors, $\boldsymbol{a}^i$ and $\boldsymbol{b}^i$ for μ^i, are called score vectors. Similarly, we calculate the vectors $\boldsymbol{f}^i$ and $\boldsymbol{g}^i$ as follows;

$$\boldsymbol{f}^i = X\boldsymbol{a}^i \quad (1)$$

$$\boldsymbol{g}^i = Y\boldsymbol{b}^i \quad (2)$$

In this paper, we consider $\mu^i, \boldsymbol{a}^i, \boldsymbol{b}^i, \boldsymbol{f}^i, \boldsymbol{g}^i$ for understanding the arrangement of each element and tendency of whole data set.

C. Numerical example

As an example of canonical correlation analysis with Web API, we show the result of analysis for university financial data (work in progress); In this case, the number of items for X expressing incomes is four (management expenses grant, tuition, research grant, donation), and the number of items for Y expressing expenses is three (general management expenses, research expenses, education expenses). Figure 4 shows the result of canonical correlation analysis in two dimensions; $(\boldsymbol{a}^1, \boldsymbol{a}^2)$ and $(\boldsymbol{b}^1, \boldsymbol{b}^2)$ for μ^1, μ^2. For $\boldsymbol{f}^i, \boldsymbol{g}^i$, Figure 5 show the arrangement of each university incomes and expenses in two dimensions.

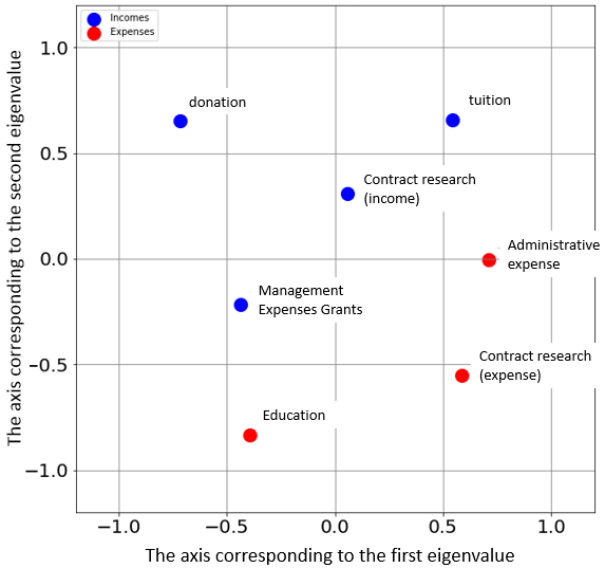

Figure 4. Example of canonical correlation analysis (1).

Figure 4 shows visually summarized information in two dimensions, which are high accumulation contribution of eigenvalues. We can grasp the global feature of financial situations of universities by this arrangement. Figure 5 shows the proximity between universities (university ID) and the tendency of whole data set.

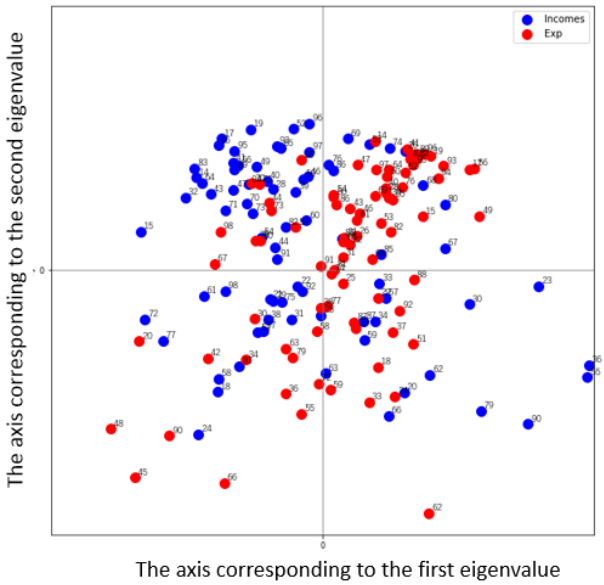

Figure 5. Example of canonical correlation analysis (2).

In this way, we can take a global view of the clustering. various comprehensive considerations on overall of accumulated data can be taken by executing the analysis. Various comprehensive considerations on overall accumulated data can be taken by executing the analysis. Those abilities will deepen the global understanding on the relations of accumulated multiple information, and which have promising possibility leads to new knowledge discovery.

IV. Conclusion

This paper describes the outline of the integrated system, Japanese College and University Portraits and related data analysis systems. Especially, we focus on several data providing services and utilization of Web API function. This type of university Web API development is early attempt in higher education field. In order to handle more general university data, coordination of differences between the data definition is needed for useful comparison. We hope that our development and attempt will play an important role as an infrastructure for data utilization and data analysis in higher education quality assurance.

References

[1] "Integrated Postsecondary Education Data System, IPEDS", URL; nces.ed.gov/ipeds [accessed: 2018-07-20].

[2] "Japanese College and University Portraits", URL; top.univ-info.niad.ac.jp [accessed: 2018-07-20]

[3] "U-Map", URL; http://www.u-map.eu [accessed: 2018-07-20].

[4] "U-Multirank", URL; http://www.umultirank.org [accessed: 2018-07-20].

[5] D. F. Westerheijden, "Multi-dimensional Mapping and Ranking New Higher Education Transparency Tools", NIAD-UE University Quality Assurance Forum 2014 Keynote speech, URL; www.niad.ac.jp/n_kenkyukai/no13_2014forum_keynote.pdf [accessed: 2018-07-20].

[6] B. Thompson, *Canonical Correlation Analysis: Uses and Interpretation*, Sage Publications, 1985.

[7] J. P. Benzecri, *Correspondence Analysis Handbook*, Marcel Dekker, 1992.

[8] M. Greenacre, *Correspondence Analysis in Practice, Second Edition*, Chapman and Hall/CRC, 2007.

 ISBN: 978-1-61208-666-8