Software Interoperability Tools: Standardized Capability-Profiling Methodology ISO16100

Michiko Matsuda, Qian Wang

▶ **To cite this version:**

Michiko Matsuda, Qian Wang. Software Interoperability Tools: Standardized Capability-Profiling Methodology ISO16100. IFIP TC 5 International Conference on Enterprise Architecture, Integration and Interoperability (EAI2N) / Held as Part of World Computer Congress (WCC), Sep 2010, Brisbane, Australia. pp.140-151, 10.1007/978-3-642-15509-3_13 . hal-01054826

HAL Id: hal-01054826

https://inria.hal.science/hal-01054826

Submitted on 8 Aug 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Software Interoperability Tools: Standardized Capability-Profiling Methodology ISO16100

Michiko Matsuda[1] and Qian Wang[2]

[1] Kanagawa Institute of Technology, 1030 Shimo-ogino, Atsugi-shi, Kanagawa 243-0292, Japan, matsuda@ic.kanagawa-it.ac.jp
[2] South-East university, No2, Si Pai Lou, Nanjing, 210096, China, qwang@seu.ac.jp

Abstract. The ISO 16100 series has been developed for Manufacturing software interoperability through capability profiling. Theses international standards are also applicable and usable for developing general software applications including enterprise applications. In this paper, ISO 16100 methodology and its usage in the trial implemented environment are introduced and discussed as software interoperability tools. The capability profile is created by filling an adequate template. The templates are prepared corresponding to activity classes which construct the application. In the development stage of new application software, the adequate software units for reuse can be found to match the requirement described in the required capability profile. The matching algorithm for capability profiles using an application domain dictionary is also provided when the profiles come from different activity class trees.

Keywords: Software interoperability, Capability profiling, Capability template, Application domain data, Profile matcher, International standard

1 Introduction

In the development of application software systems in any area such as business, game and manufacturing, there is globalization from building an application system by developing all software units by themselves to building a application system by combining software units which are provided by various vendors. To follow this change, key technologies concentrate on how to skillfully find and use software units which are provided by various vendors and how to reuse existing software units. However, there is no standardized mechanism to search for good and proper software units in the world. Even if proper units could be found, big efforts are required to know the precise capabilities and assurances of the software units because of a lack of standardized description methods of their capabilities and assurances. On the other hand, for the vendor of software units, there exists no distribution chain. Even if they have a distribution chain, there is no standardized mechanism to show the precise capability and assurance of their software units.

This paper provides the methodology to resolve the above mentioned problems. When this mechanism is provided as an international standard, it will provide the interoperability and assurance of software units using capability profiling. This paper is associated with the ISO/TC 184/SC 5/WG 4[1] activities: ISO16100 series [1][2][3][4][5][6] which is titled 'Manufacturing Software Capability Profiling for Interoperability.' This methodology proposed in ISO 16100 is applicable not only in the manufacturing enterprise but also in other area's enterprise such as in the chemical and amusement industry. The ISO 16100 series enables manufacturing software integration by providing the following : 1) standard interface specifications that allow information exchange among software in industrial automation systems developed by different vendors, 2) software capability profiling using a standardized method to enable users to select software that meet their functional requirements, and 3) a conformance test method that ensures the integrity of the software integration.

2 Software Interoperability using Capability Profiling Methodology

In the ISO16100 series, the interoperability framework for manufacturing software is based upon a more general interoperability framework for applications. An integrated manufacturing application is modeled as a combination of a set of manufacturing processes, a set of manufacturing resources and a set of information units whose data structure, semantics, and behavior can be shared and exchanged among the manufacturing resources. Manufacturing resources are to support the processes and information exchanges required by the application.

[1] Authors are experts in ISO/TC 184/SC 5/WG 4 whose convenor is Prof. M. Matsuda.

The set of integrated manufacturing resources forms a manufacturing system architecture that fulfils a set of manufacturing application requirements. These manufacturing resources, including the manufacturing software units, provide the services, activities and functions associated with the manufacturing processes.
In an appropriate operating environment, the combined capabilities of the various software units provide the required functionality to control and monitor the manufacturing processes according to the production plan and the allocated equipment. A manufacturing process is composed of a set of manufacturing activities. A manufacturing software unit consists of one or more manufacturing software sub-units, performing a definite function or role within a manufacturing activity while supporting a common information exchange mechanism with other software units. The manufacturing software interoperability of a set of manufacturing activities are described in terms of the interoperability of the set of manufacturing software units associated with each manufacturing activity. These relationships are used for capability profiling.
Interoperability on manufacturing software is the capability of a software unit to support a particular usage of an interface specification in exchanging a set of application information with another software unit. The interoperability of software units is described in terms of their capabilities that are associated with the aspects of services, activities, function, interface and structure. Fig. 1 depicts the use of a capability profile concept to integrate interoperable software. In Fig. 1, the left side shows the capability profile registration flow for software vendors for wide distribution of their software units. The software unit's capability profile definition is registered in an appropriate database after passing the conformance test which assures the software unit profile and the software unit itself. On the other hand, the right side shows the flow for finding capability profiles for development of new applications through reuse of adequate software units. The required profiles are compared to existing profiles in the database. When a match occurs, the software unit being profiled is considered to be ready for reuse and integration. When no match occurs, a new software unit with the required capabilities will be developed, profiled, and registered in the capability profile database.
The capability profiling methodology is defined in terms of the rules and elements provided in following sections. The methodology makes use of the domain-specific attributes and methods associated with each specific software unit to describe capability profiles in terms of software unit name, functions, and other needed class properties.

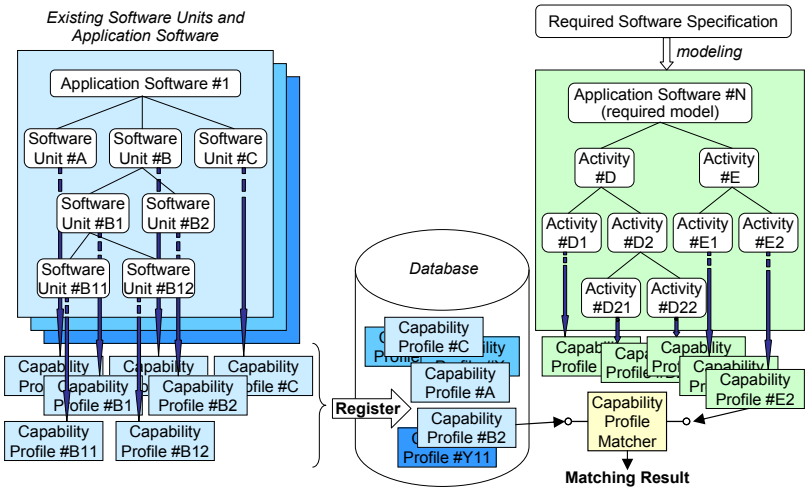

Fig. 1. Software interoperability through capability profiles.

3 Rules and Elements in ISO 16100 for Software Interoperability

3.1 MDM (Manufacturing Domain Model) and MDDs (Manufacturing Domain Data)

The manufacturing domain that includes discrete, batch, and continuous control encompasses many types of industries. For manufacturing software, the interface between plant management systems and floor control systems is described by the same method regardless of whether control systems are discrete, batch, or continuous. Similarly, the control flow inside a control system is also described by the same method regardless of whether the system is discrete, batch, or continuous. However, a key aspect of terminology is delicately different in each domain. A key aspect of a terminology for capability profiles is its ability to identify the contents that constitute a capability definition. A terminology is constructed that provides a means for the interchange of the capability

information. The terminology describes a partial set of activities undertaken within the lifecycle of a manufacturing enterprise and domain. For capability profiling of software units, the applicable manufacturing domain must be focused, and the set of terminology must be common in the domain.
Two new elements are introduced for capability profiling. One is the ***MDM*** (Manufacturing Domain Model). MDM is a model of an applicable manufacturing domain for manufacturing software. Another one is the ***MDD*** (Manufacturing Domain Data). The MDM is a particular view of a manufacturing domain, consisting of MDDs and relationships among them, corresponding to the domain's applications. The MDD represents information about manufacturing resources, manufacturing activities, or items exchanged among manufacturing resources within a particular manufacturing domain. A set of MDDs works like a terminology set in the applicable domain. MDDs represent different types of manufacturing information, including those that are exchanged between the resources within an application and between applications. Fig. 2[5] shows an example of a structure of a MDM with multiple MDDs. Within a specific manufacturing domain, a manufacturing application can be represented as a set of MDDs. An MDD provides information about various aspects of a manufacturing application such as manufacturing resources, manufacturing processes, manufacturing information exchanged, and relationships among the resources, processes and information exchanged. Each MDD within a specific manufacturing domain consists of attributes, operation types and a mapping between them. The mapping identifies the operation types associated with an attribute. In a typical mapping, not all operation types will be associated with an MDD's particular attribute. Typical operation types include initialization of an attribute's value and rewriting an attribute's value. The MDD exchanged among manufacturing functions or among manufacturing activities is descriptively named such that each MDD is unique in the target manufacturing domain.

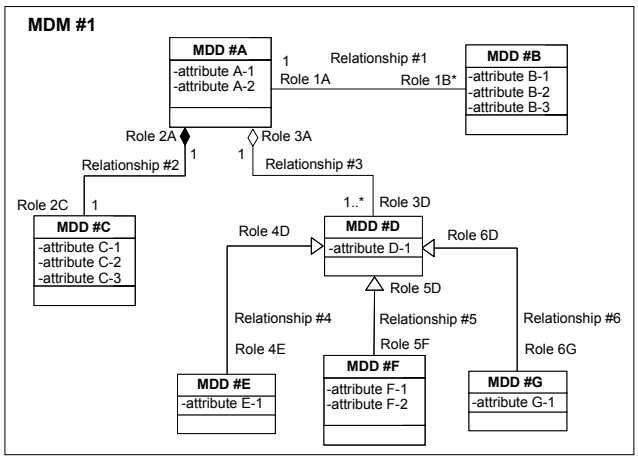

Fig. 2. Target Manufacturing Domain Model and Manufacturing Domain Data in the manufacturing area.

The MDM creator represents a manufacturing application as a set of MDDs within a specific manufacturing domain. An MDD provides information about various aspects of a manufacturing application such as:

- Manufacturing resources (ex. manufacturing software unit, equipment, automation devices, personnel, material, work-in-process inventory),
- Manufacturing processes (ex. operations, activities),
- Manufacturing information exchanged (ex. product data, recipe, manufacturing data, quality data),
- Relationships among the resources, processes and information exchanged.(ex. data flow, network configuration, work flow).

3.2 CCS (Capability Class Structure) and Capability Template

A manufacturing application is modeled as an activity tree structure that is both nested and hierarchical. The activity tree is structured based on the MDM from the requirements of the manufacturing application. To distinguish a particular activity in an activity tree, an activity has an unambiguous and unique name, along with semantic information expressed in terms of a sequence of MDDs. The ***CCS*** (Capability Class Structure) is formed from the activities in the activity tree. As shown in Fig. 3, a CCS corresponds to the activity tree and a capability class is unique when the activity can be pointed to in the activity tree. The capability of a software unit is expressed in terms of capability classes. At each level, the software unit is modeled as a set of capability classes organized in a similar structure. These classes also denote the manufacturing function, resource, and information handled by the software unit according to the requirements of the manufacturing process. As a result, an activity tree and a CCS have a one to one mapping.

If a manufacturing software vendor wants to widely distribute his developed software unit, the vendor makes several profiles for one component corresponding to each CCS. When a system integrator or manufacturing application developer wants to search proper software units, the developer creates the profile by filling the ***Capability Template*** with required capabilities. A software vendor registers a capability profile of a software unit so that it is widely available to many potential users of the software unit.
Usually the vendor who is the supplier of software components, and the application developer who is the user of software components, are in the same MDM but not in the same CCS. To allow matching existing capability profiles of software component and required capability profiles derived from multiple CCSs, MDDs are used.

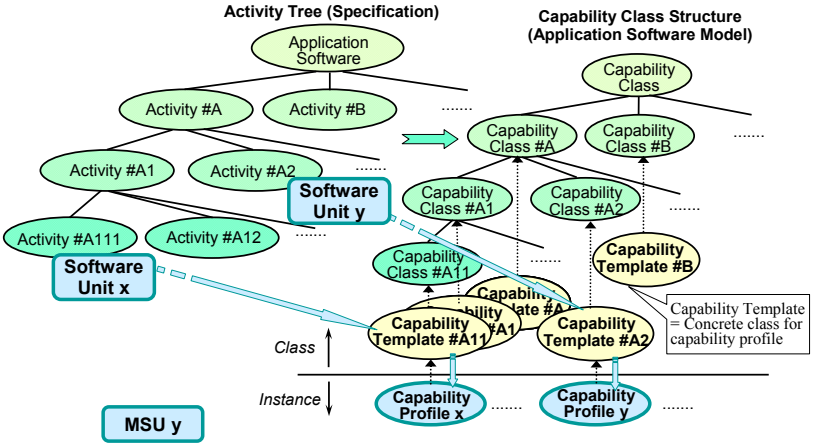

Fig. 3. Capability Classes Structure and Capability Template.

Fig. 4 shows the conceptual structure of the Capability Template. The sets of elements in the Capability Template are filled by the concrete values corresponding to the target profiled software unit's capabilities. The profile is described using XML. A capability profile template contains a Common Part and a Specific Part. The Specific Part contains the elements: Reference MDM Name, list of MDD objects, and Capability Definition (e.g. time ordered access to MDD objects).

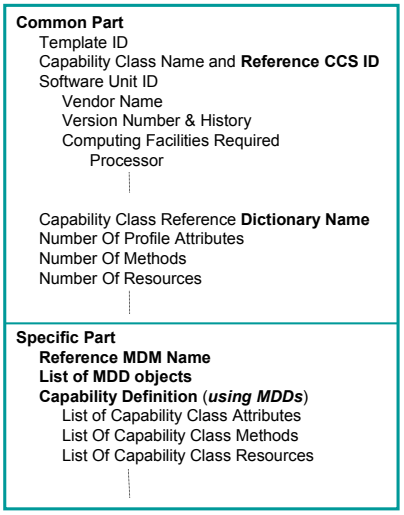

Fig. 4. Structure of Capability Template.

4 Interoperability Tool: Matcher of Capability Profiles

4.1 Interoperability Framework for Application Development

To increase the efficiency of application software development, it is desirable to reuse software units previously deployed in a similar application. For this purpose, the required software unit for the new application needs to be

compared to the software unit capability that will be re-used. In this comparison, the reference CCS of the software unit to be re-used may not be the same as the CCS of the target application. In this case, the application developer is not limited by the existing CCSs within the same application domain.

In Fig. 5, the flow shows the procedure that a software vendor performs to prepare and register a capability profile of a software unit. After choosing a suitable MDM, the first step is to analyze the set of activities that the software unit enables. The software unit can enable one or more activities. The second step is to identify the capability class corresponding to each activity and search for the associated CCS to which the capability class belongs. If a software unit provides capabilities for two or more activities, those activities can belong to the same CCS or to a different CCS. The third step is to select the capability template for each capability class identified. If there is no suitable CCS, the fourth step is to construct the appropriate CCS and register it, and then to generate the corresponding template and register it. The last step is to create the software unit capability profile by filling in the capability template and register it[7].

When a new manufacturing application is developed, the following procedures are performed as shown in Fig. 5. The first step is to analyze the functional capability requirements of the manufacturing application and create an activity tree in the appropriate MDM. The second step is to create a CCS using existing or new capability classes to match the activity tree, or select an existing CCS. The third step is to fill in the corresponding capability template for each capability class in the created or selected CCS to create the set of required capability profiles. The sets of elements in the template are satisfied by the concrete values for the requirement upon the software unit capabilities. The profile is described using XML. The required profile contains a set of mandatory and optional capabilities. The fourth step is to compare the set of required capability profiles to the available set of software unit capability profiles using a capability profile matcher, to find a set of existing software units that matches the set of required capability profiles. The fifth step is to select the set of existing software units that meets the requirements of the new manufacturing application. If the set of software units that meets the requirements is not found, a set of the missing software units has to be developed. The last step is to combine the set of reused software units and any set of developed software units to meet the requirements of the new manufacturing application[7].

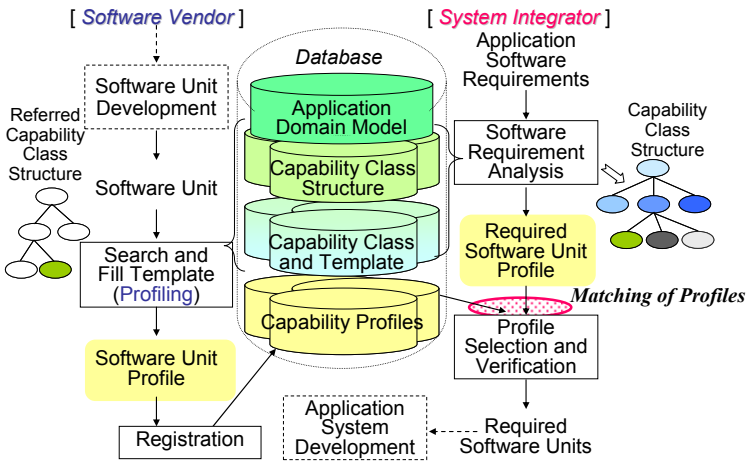

Fig. 5. Conceptual Interoperability Framework.

4.2 Matching Procedure of Capability Profiles

A Matcher is used to match a required capability profile and an existing capability profile for a software unit. A Matcher makes use of the reference CCS names and related information from the two inputted capability profiles in order to determine if these profiles are based on a common MDM and common set of MDDs. When these profiles are based on a common MDM and common set of MDDs, a Matcher can evaluate the existence of a functional correspondence between these profiles.

The matching processes in the matcher are shown in Fig. 6. The first step is to extract the reference MDM IDs from the inputted capability profiles and compare these MDM IDs. If they are not the same, then the matcher reports that a comparison of the inputted profiles cannot be made. If these MDM IDs are the same, the second step is to extract the capability definition formats from the inputted profiles and compare these formats. If these formats are the not the same, the MDDs in the capability definitions are converted to a single capability definition format by means external to the matcher. If these formats are the same, no conversion is made. The third step is to extract the sets of MDDs contained in the capability definitions for these profiles. The fourth step

is to compare them to determine the existence of functional correspondence between the profiles. At the last step, the matcher reports the matching level of the capability profile relative to the required profile[5][8].
When comparing the contents of two target capability profiles, the matching level generated by a Matcher is assumed be one of Complete Match, All Mandatory Match, Some Mandatory Match or No Mandatory Match. Complete Match means that both sets of manufacturing functions are fully equivalent in terms of both the MDD objects being equivalent and the time ordering of these objects being equivalent. All Mandatory Match means that all the mandatory functions in the required capability profile are completely matched with a corresponding set of manufacturing functions referenced in the MSU capability profile. Some Mandatory Match means that the required capability profile is matched partially by the MSU capability profile. No Mandatory Match means that none of the mandatory functions referenced in the required capability profile match the functions referenced in the MSU capability profile.

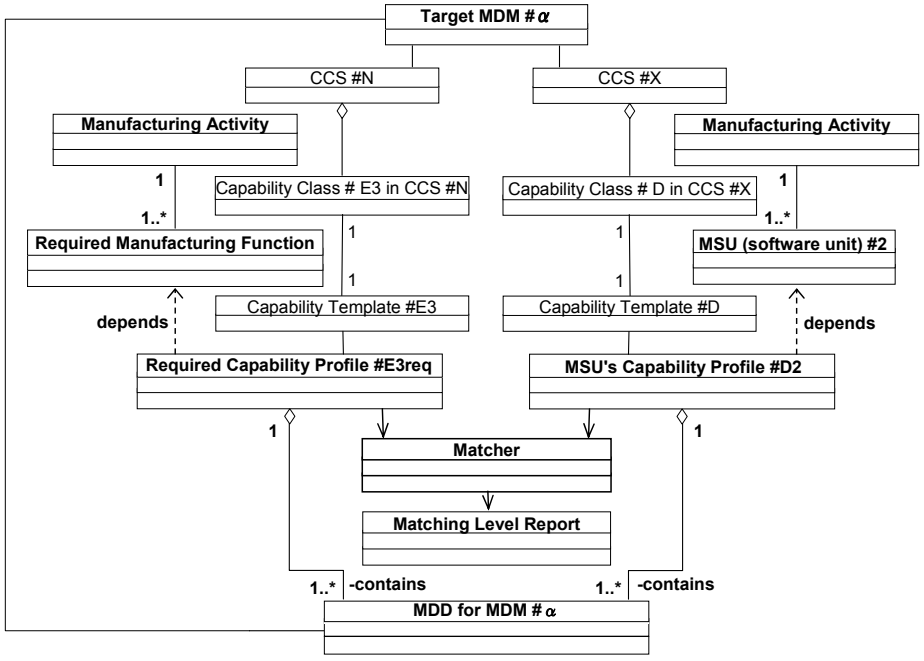

Fig. 6. Matching for software capability profiles.

5 Trial Implementation of Capability Profile Matcher

Based on the capability matching procedure mentioned above, a capability profile matcher is developed. The prototype system is performed in Widows 2000/Windows XP/Windows Vista .Net Framework 2.0 in C#. The system is based on the ISO 16100 framework[1][2][3][4][5] of the software capability profiling for interoperability. The prototype system consists of following functions as shown in Fig. 7.:

- For CCS (Capability Class Structure)
  - Create / register a new CCS based on an activity tree for a particular application
  - Edit / register a capability class for an activity
  - Delete a capability class for an activity
- For Capability Template
  - Create / register a Capability Template for a capability class based on the formal structure
  - Edit / register a Capability Template based on the particular activity
  - Delete a Capability Template
- Creating a capability profile based on a Capability Template
  - Create / register a new capability profile based on the particular Capability Template
  - Edit / register / a capability profile
  - Delete a capability profile
- Searching a proper Capability Template and return the search result with the detail report
- Matching a user required capability with a set of existing capabilities and return the matching result with the detail report

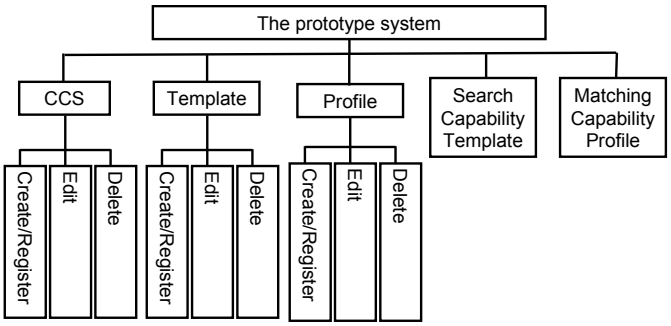

Fig. 7. The functions of the prototype system.

Fig. 8. shows the screen for editing a capability profile. When the profile template ID and its name is input, the tree-style schema of this profile template will appear on the left part of the screen. Profile information is shown on the right part of the screen. From top to bottom, there are the head part (Profile Type, Package Type, Version, Profile ID, etc.) of the profile, the common part of the profile and the specific part (Operations. Exchanged information, Resources and Constraints, etc.) of the profile. Before storing/registering the profile into the database, a conformance test is done automatically. The XML format file for profile is stored in the database. Also any XML file in profile style is presented in a tree on the screen.

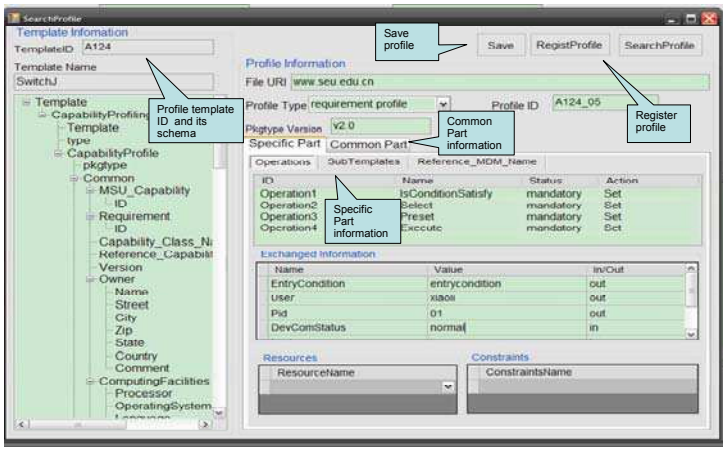

Fig. 8. Editing a capability profile.

Matching on the prototype matcher is shown in Fig. 9. This prototype matcher can match two profiles which come from different Capability Templates. Two profiles may have different schema. The prerequisites for the matcher are that these two profiles are in the same application area: same MDM, and using the same MDD dictionary during the filling in of values. After giving the required capability profile (Source profile) and one existing capability profile (Target profile), the matching starts. During the matching, the matching process is shown in the middle of the screen. On the left part of the screen is the Source profile, and on the right part of the screen is the Target Profile. They are both in a tree style. The red color stands for the matched terms. Each matching step (matching point in each tree and its matching action) is shown below the two profiles' trees. A matching speed adjuster called "interval" is used to adjust the interval for each matching step in order to watch the process step more clearly.

The matching for the head part and the common part in two profiles are simple string comparisons, however, the matching for a specific part is more sophisticated. It follows the following procedures:

1. Compare each operation in 'Source Profile' with each operation in 'Target Profile' until an operation is matched;
2. For each operation, compare each element of the exchanged information in 'Source Profile' with each element of the exchanged information in 'Target Profile' until an element is matched;
3. For each operation, compare each element of resources in 'Source Profile' with each element of resources in 'Target Profile' until an element is matched;
4. For each operation, compare each element of constraints in 'Source Profile' with each element of constrains in 'Target Profile' until an element is matched;

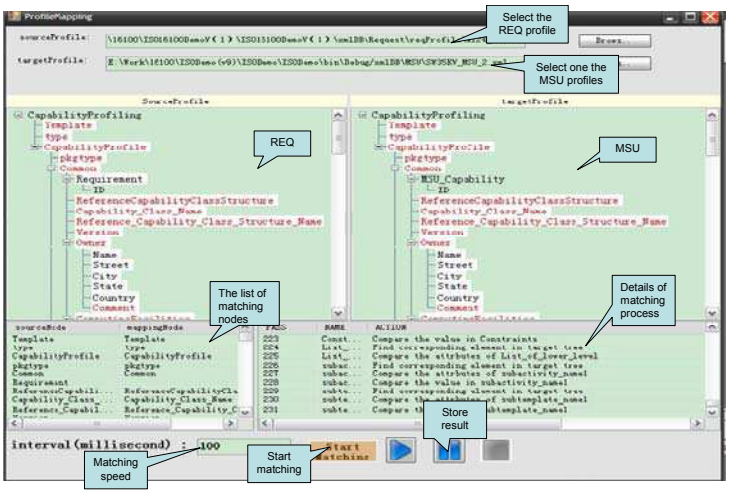

Fig. 9. Matching a capability profile.

6 Conclusions

ISO16100 provides the standardized framework for software interoperability. In other words it provides a methodology to search for a good and proper manufacturing software unit and to show the precise capability of this software unit. In ISO16100, the interoperability and assurance of software units can be managed through their capability profiles which describe their capabilities that are associated with the aspects of functionality, interface and structure of the software unit. At present, ISO1600 consists of six parts. Even though ISO16100 is developed for manufacturing software, the concepts and methods which are proposed in ISO16100 are generally applicable to software interoperability and assurance. MDM, MDD, CCS and Capability Template are basic tools and the capability profile matcher is a most useful tool in ISO16100 framework. Trial implementation of the matcher shows the practical usefulness of the methodology. When the proposed environments are available, software interoperability and assurance will be enhanced on a grand scale.

Acknowledgments. The authors thank ISO/TC 184/SC 5/ WG 4 members for fruitful discussions and their useful effort to complete international standards. The author is also grateful to Dr. U. Graefe, the previous convenor of ISO/TC 184/SC 5/ WG 4 for his helpful assistance with the writing of this paper in English.

References

1. ISO 16100-1:2009 Industrial automation systems and integration -- Manufacturing software capability profiling for interoperability -- Part 1: Framework (2009)
2. ISO 16100-2:2003 Industrial automation systems and integration -- Manufacturing software capability profiling for interoperability -- Part 2: Profiling methodology (2003)
3. ISO 16100-3:2005 Industrial automation systems and integration -- Manufacturing software capability profiling for interoperability -- Part 3: Interface services, protocols and capability templates (2005)
4. ISO 16100-4:2006 Industrial automation systems and integration -- Manufacturing software capability profiling for interoperability -- Part 4: Conformance test methods, criteria and reports (2006)
5. ISO 16100-5:2009 Industrial automation systems and integration -- Manufacturing software capability profiling for interoperability -- Part 5: Methodology for profile matching using multiple capability class (2009)
6. ISO/CD 16100-6 Industrial automation systems and integration -- Manufacturing software capability profiling for interoperability -- Part 6: Interface services, protocols for matching profiles based on multiple capability class structure (2010)
7. Matsuda, M., Arai, E., Nakano, N.: Capability Profiling for Interoperability of Manufacturing Software. The CIRP - Journal of Manufacturing Systems, Vol. 34, No. 1, pp. 63-70 (2005)
8. Yu, W., Matsuda, M., Wang, Q.: Enhancing Interoperability of Manufacturing Software Units Using Capability Profiling」 Enterprise Interoperability. IFIP the new Challenges and Approaches, p.451-460, Springer (2007)