Service Oriented Platform Design for Collaborative Engineering Data Analysis*

ISSM Paper: MC-P-235

Jun-Rong Lee, Shi-Chung Chang**, Fang-Hsiang Su, Chih-Min Fan
r94725027@ntu.edu.tw, scchang@cc.ee.ntu.edu.tw, b90502052@ntu.edu.tw, cmfan@saturn.yzu.edu.tw

**Correspondence author: Shi-Chung Chang, Rm. 245, Electrical Engineering Dept., National Taiwan University, Taipei, Taiwan 10617;
e-mail: scchang@cc.ee.ntu.edu.tw Phone: +886-2-2362-5187 Fax: +886-2-2363-8247

Abstract - Data and knowledge sharing for seamless and innovative engineering collaboration is critical to quick yield ramping. This paper presents the developments of a pragmatic methodology and a platform design for quick, flexible and collaborative composition and provision of engineering data analysis (EDA) services. The methodology combines Markov chain-based EDA procedure modeling and knowledge extraction with an identification method of re-usable service components from the EDA procedures. The platform adopts a service oriented architecture and web service technology to facilitate EDA service management and sharing. The design and developments are conveyed and supported by using a legacy EDA system and its usage data.

I. Intra-fab Engineering Collaboration

Rapid development of semiconductor products with high manufacturability and profitability has posed stringent challenges to business and engineering integration. In foundry services, the integration of engineering services for intra-fab collaboration is a corner-stone to the larger scope of engineering chain collaboration (ECC) among design and manufacturing for quick time-to-market with high yield [SGC04, ITR05]. How to exploit advanced information technology and design a platform to enable collaborative integration is key to foundry services in the era of sub-wavelength process technology.

Data and knowledge sharing is fundamental to ECC, which should enable engineers to leverage on each others' specialty, to produce quality engineering work and to nurture creative ideas. Tapscott and Williams gave a few renown successes of integrated collaboration in the design and production of motorcycles, airplanes and automobiles [Chapter 8, TaA06]. In semiconductor fabrication, Fan, Chang and Chang [FCC05] proposed a framework for integrated management of heterogeneous engineering data, information and knowledge among various organizational entities. Morinaga et al. of Toshiba Co. [MKI06] exploited web-based and service-oriented technologies to develop, under an IDM environment, a platform for realizing collaborative engineering data flows between design and manufacturing. They reported 3-20 times reduction of information linkage turn-around-time and cost reduction from software reuse. However, problem-solving knowledge sharing and collaboration process analysis are yet to be explicitly addressed.

II. Challenges: EDA as Conveyor Problem

This paper aims at pragmatic methodology and platform designs for quick, flexible and collaborative composition and provision of services to share fab engineering data and knowledge. Specifically, engineering data analysis (EDA) serves as the conveyer problem domain. There are three challenges:

C1) Identification and modeling of engineers’ requirements for and application processes of EDA

EDA procedures for problem solving are situation dependent and vary among individual engineers. How to systematically model and extract engineers’ problem solving requirements and EDA procedures is therefore a very challenging knowledge engineering task. It is further complicated by how the extracted knowledge can be presented to engineers of different background as a guidance to problem solving.

C2) Identification of EDA service components (SCs) from problem solving processes

To substantiate their EDA procedures, engineers have many EDA functions/tools to adopt and use in their computer systems. Some of the functions may be used frequently but only part of the functionality is actually needed while some separate functions are very often jointly used. How should these functions be aggregated or segregated into components of proper granularity that can be efficiently reused and flexibly composed into various EDA procedures?

C3) design of a management platform for EDA service discovery, composition and provision

Engineering collaboration needs a platform that exports legacy functions and knowledge to services, and provides individual engineers with service composition and service management capabilities. Through the platform, engineers of different background may consume existing or compose new EDA services through a familiar GUI. Engineers should also be able to easily document their knowledge of successful problem solving procedures. Such knowledge should be effectively shared with others through a search and discovery mechanism of the platform. For system developers, the platform should have a scalable architecture with high reusability of functionality, software and services.

In view of the three challenges, we propose an EDA service system structure as depicted in the schematic diagram of Figure 1, where the notion of service oriented architecture (SOA) [YBS06] is exploited. In this paper, we shall first describe our design of engineering procedure modeling and extraction that addresses challenge C1. Based on the model, we shall then discuss

---

*This work was supported in part by the National Science Council, Taiwan, ROC, under grant NSC 95-2221-E-002-346-MY3.
how we perform reverse engineering over usage data of legacy EDA functions to identify EDA service components and flow sequences among them (C2). Finally, we shall present our ideas on design and implementation of a collaboration platform for EDA service composition and management (C3).

III. Procedure Modeling and Extraction

An EDA procedure consists of a sequence of analysis steps, where a step may be the application of an EDA function or an analysis decision by an engineer. In each step, there involves data/information query, processing and presentation, which belong to information services that are needed to realize individual EDA steps. Our procedure modeling is focused on modeling the application sequences of EDA functions of based on empirical usage data of using a legacy EDA system. Once modeling and extraction methods are available, futuristic EDA service requirements will be developed by extending the methods through interviews with veteran engineers who are working on advanced process technologies and next generation fabs.

In identifying engineers’ application sequences of EDA functions to problem solving, we first adopt Markov models of various orders to represent EDA function sequences. We then design an algorithm to extract, from engineers’ usage data, the relative frequency information of applying a specific EDA function after a given sequence of EDA functions. Key ideas and results are as follows.

III.1 Procedure modeling by Markov models

The motivation that we consider Markov models is that the application of an EDA function at step \( n+m \) is affected by the functions used at steps \( m \) to \( m+n-1 \). If we consider the application of an EDA function as a “state,” the dependency of a state to the \( n \) previous states can then be modeled as an order-\( n \) Markov chain. The transition probability from an \( n \)-state sequence to a next state corresponds to relative frequency of adopting a specific EDA function after a sequence of \( n \) EDA functions. Such a modeling concept matches practitioners’ intuition about EDA procedure for yield analysis. Figure 2 depicts an exemplary analysis flow in an order-1 Markov chain.

III.2 Procedural knowledge extraction

On top of the Markov models, we then exploit empirical EDA function usage data to extract the specific states (EDA functions) and state transition probabilities (relative usage frequencies) of EDA function flow sequences, which we consider as the procedural knowledge. Figure 3 gives an exemplary usage data of one user, where there are function start times and dates. Exploiting the timing information, one may find the sequence of applying these functions. In this example, function A is the starting one followed by function B. After applying functions A and B, the next function can be B, C or D. Function C is followed by function D. The sequences are depicted in Figure 4. It is well recognized that EDA procedures vary among engineers, not only among those of different levels of experience or background but also among individuals at a similar level.

EDA function usage data of multiple users at the same level is therefore used to extract the sequence-dependent relative frequencies of next functions in an order-\( n \) Markov chain model. Given an \( n \)-step sequence, there can be many next functions, e.g., B, C, and D after the 1-step sequence of B in Figure 4. The relative frequency of using C next can be calculated by dividing the total number of B-C sequence with the total number of B-X sequences in the sample usage data, where X is any EDA function. To extract procedures common to many engineers, we set a threshold for each given sequence to filter out infrequently used next functions for the state; namely, if the relative frequency of using a next function is lower than the threshold, this next function will not be included as part of the EDA procedural knowledge with the given sequence. Based on such ideas, a Markov chain-based procedural knowledge extraction (MCPKE) algorithm is developed.

III.3 Validation and applications

With input of some a priori knowledge about standard operation procedures for yield analysis, the MCPKE algorithm is applied to EDA usage data of more than two hundred engineers over a three months period. Veteran engineers were invited to review the extraction results. The Markov chain-based procedural knowledge models indeed capture many key procedural features and reveal many phenomena for further exploration. Figure 2 depicts an example of a partial EDA function flow derived from fab data.

The modeling and extraction methods automate EDA procedural knowledge extraction. At the current stage, it at least helps document common practices of EDA users if not the golden practice. On the user side, the documented EDA procedures can then be used as i) a reference foundation for communications and collaborations among engineers, and ii) training guidance for rookie engineers. On the system developer side, it may serve as iii) a basis for aggregation or segregation of functions into reusable SCs by examining the sequence dependency among functions over various procedures, which we shall elaborate in the next section.

IV. Reverse Identification of Service Components

With EDA procedures extracted from engineers’ usage data, we now adopt the method of reverse identification [WXZ05] to identify EDA function sequences of strong dependency, high reuse value and/or good reuse performance. Such function sequences will then be put respectively as SCs to support effective reuse of EDA function software. We propose four steps to identify EDA SCs:

Step1: Legacy function usage classification

Each EDA function has at least one purpose for yield analysis. EDA functions are classified into different function categories according to their individual purposes. One function may belong to more than one function categories.

Step2: Function to service category mapping

This step maps EDA functions to services because users understand what kind of EDA services they need but may not have a good grasp of the functionality of
every EDA function. However, the granularity of SCs that make of services affects the flexibility of service composition, reusability of SCs, and efficiency of service delivery. Instead of finding an optimal granularity, we adopt a heuristic that maps EDA function categories to EDA service categories based on the extracted procedure models and only considers aggregation among functions in the same service category.

**Step 3: Aggregation in a service category**

Within a service category, functions may be aggregated based on their procedural dependency extracted from empirical data. By analyzing EDA procedures extracted from fab data, we identify two possible types of aggregation and illustrate the ideas in Figures 5(a) and (c). Figure 5(a) is an aggregation of functions in a frequently used and strongly dependent function sequence into a SC. For this case, we define the strength of sequential dependency between functions \(i\) and \(j\), \(S_{ij}\), as

\[
P_j = \text{prob}(\text{next function } j | \text{current function } i)
\]

\[
S_{ij} = C_{ij} \cdot P_j,
\]

where \(A \cdot A\)

\[
C_{ij} = \begin{cases} 
1, & \text{if function } i \text{ and } j \text{ in the same category} \\
0, & \text{if function } i \text{ and } j \text{ in different categories}
\end{cases}
\]

high \(S_{ij}\) value indicates that functions \(i\) and \(j\) could be aggregated into the same service component. Figure 5(c) illustrates a case that functions \(C\), \(D\) and \(F\) are all used between functions \(B\) and \(X\). There may very likely be redundancy among them. Further analysis can be used to determine their similarity and opportunity for aggregation into one SC.

**Step 4: Segregation of a legacy function**

Our object-oriented system analysis of various EDA functions in a legacy system shows that duplications of code exist among various functions, which could be segregated into small and reusable SCs. Figure 5(b) illustrates the segregation of functions into commonly used sub-functions; each of these sub-functions can then be re-code into an independent SC. Limited by time and resources, we currently consider EDA function segregation a low priority and treat each EDA function as the smallest unit for composing a SC.

Such efforts for SC identification allows EDA system developers to shorten EDA service development time, reduce cost, and make SC easily comprehensible to users while retaining good flexibility for service composition.

**V. Service Composition and Management Platform**

Once EDA SCs are defined, we design and develop a EDA service management platform for service discovery, composition and provision. Figure 6 gives the platform architecture. In the platform, we implement SCs identified from legacy systems by using Web Service (WS) technology. The legacy EDA system we study is tightly coupled with GUI and can not be deployed as a WS directly. We develop a WS wrapper for each legacy EDA functions. A wrapper can get input and output from the legacy function it wraps without using the legacy GUI. Hence a legacy EDA function can act as a background program through the wrapper while the WS communicates with the wrapper. The legacy functions remain as the actual service providers. Such an implementation allows scalability and easy addition of new SCs.

To facilitate EDA knowledge and system service sharing and resource reusability, we design a vocabulary hierarchy, service descriptions and a service directory. The vocabulary hierarchy is designed based on EDA function input and output items and their relation to wafer lot, process, product and parameters. This vocabulary hierarchy provides a systematic way and standardized vocabularies to describe EDA services and interfaces. Engineering knowledge can then be shared through standardized vocabularies for knowledge description, and services and their requirements can be effectively communicated via service descriptions. Through them, EDA developers can also effectively search for, understand and reuse available SCs and services.

In the EDA service management platform, a service GUI is designed that supports drag-and-drop EDA service composition. The drag-and-drop composition is based on BPEL technology [Ecl07], which is used to orchestrate existed web services. The drag-and-drop EDA composition interface shown in Figure 7 enables engineers to flexibly compose their own analysis flows based on the available EDA SCs, which can then be automatically documented for future reference, performance analysis and sharing with others.

**VI. Conclusions**

In this paper, we consider EDA as a problem domain. We have developed a pragmatic methodology that combines Markov chain-based EDA procedure modeling and knowledge extraction with an identification method of re-usable service components from the extracted EDA procedures. We have also designed a EDA service management platform architecture that adopts a service oriented architecture and the web service technology. The methodology development and platform design together will allow us to establish an environment for quick, flexible and collaborative composition and provision of EDA services.

**References**


[WXZ05] Z.J. Wang, X.F. Xu, D.C. Zhan, “A Survey of


Figure 1 Schematic EDA System Structure

<table>
<thead>
<tr>
<th>Function name</th>
<th>Function start time</th>
<th>Function start date</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2006/10/31 12:57</td>
<td>2006/10/31</td>
</tr>
<tr>
<td>B</td>
<td>2006/10/31 13:28</td>
<td>2006/10/31</td>
</tr>
<tr>
<td>C</td>
<td>2006/10/31 13:00</td>
<td>2006/10/31</td>
</tr>
<tr>
<td>D</td>
<td>2006/10/31 13:01</td>
<td>2006/10/31</td>
</tr>
<tr>
<td>E</td>
<td>2006/10/31 13:12</td>
<td>2006/10/31</td>
</tr>
<tr>
<td>F</td>
<td>2006/10/31 12:15</td>
<td>2006/10/31</td>
</tr>
<tr>
<td>G</td>
<td>2006/10/31 13:16</td>
<td>2006/10/31</td>
</tr>
<tr>
<td>H</td>
<td>2006/10/31 13:17</td>
<td>2006/10/31</td>
</tr>
<tr>
<td>I</td>
<td>2006/10/31 13:21</td>
<td>2006/10/31</td>
</tr>
</tbody>
</table>

Figure 2 Exemplary EDA Function Flow Extracted from Usage Data

Figure 3 Example of EDA Function Usage Data of a User

Figure 4 An Exemplary Markov Chain Model

Figure 5 EDA Service Component Identification

Figure 6 EDA Service Management Platform

Figure 7 Drag-n-drop EDA Service Composition