An Integrated System to Support Design of Safer Batch Processes

Chunhua Zhao¹, Yukiyasu Shimada², Venkat Venkatasubramanian*¹
¹ LIPS, Purdue University, USA, ² Okayama University, JAPAN

Abstract

Batch process design is an information and knowledge intensive task. The paper discusses the needs for tools to manage and utilize various design information and knowledge to support design activity. An integrated approach, which is centered on design rationale and integrated tools, has been proposed. The details of the framework and implementation of an integrated development environment are discussed.

Keywords: batch process design, integrated development environment, design rationale, integration of tools

1. Introduction

Design has long been regarded as a “wicked problem”, in which “messy situations” characterized by uncertainty, conflict, and multiplicity have to be dealt with. The design of batch processes is no exception, with additional burdens due to regulations on product integrity and process safety. Design is the major activity in batch process lifecycle and related closely to other activities.

1.1 Design of batch processes

Batch process design starts from business requirements and feasibility analysis. Wide range of expertise and activities are involved in the design process including route selection, selection of solvents, synthesis method, thermal integration, simulation, physical properties prediction, safety and environment concerns. Design is an important component of entire plant life cycle, since the quality of design affects significantly the cost for constructing the plant, as well as operating and maintaining the plant. In the past several decades, a large amount of best practices, guidelines, methodologies, and knowledge for batch process design have been generated. These knowledge, however, are mostly ‘tribal knowledge’ which are kept on in individual engineers’ heads or paper-based documents. The authors argue that one important factor of success for wider implementation and use of these design methodologies is the availability of easy-to-use software tools that support these methodologies. Furthermore, the current process development can be considered to follow the ‘waterfall’ style of system development. It is necessary to shift to incremental and iterative styles of development in order to save time and cost and to improve design quality. With proper tool support, information and knowledge could be managed and used in a more systematic and efficient manner, and the transition of development style would be easier.

* Author to whom correspondence should be addressed: venkat@ecn.purdue.edu
1.2 Needs for tools support and types of tools
Design of software and chemical processes share many common characteristics, in spite of the apparent differences. Review of the current status of tools support for software design would shed light on the necessities for tool support in batch process design, types of tools, as well as the methodologies on construction of these tools. The tools that support software design have grown very fast in past decades in terms of amount and quality, which is easy to understand given the fact that the tools are also software. Due to the space limitation, it is not possible to present a complete review on all kinds of supporting tools. Instead, three types of tools on information management, functionality, and integrated development environment (IDE), are briefly reviewed.

The basic information in software design is source code, which is normally organized into structured text files defined by the syntax of the specific programming language. The nature of the information representation makes it relatively simple for management tasks like version management (e.g. Concurrent Version System: CVS) and build management (e.g. BuildForge®) which essentially insert tags in the source code and provide facilities to extract these tags for various purposes. Other information management tools are also available, including Requirement management (e.g. Rational RequisitePro®), test management (e.g. Rational TestManager®), etc. Rational tools are set of functionality tools to create an integrated flow of information as well as an infrastructure for the extended development team to work together supported by best practices embedded in these tools. The centerpiece of Rational approach is integrated tools and best practices. The best practices for software development, following the Rational Unified Process (Kruchten, 2000), include (1) develop iteratively; (2) manage requirements; (3) use component architecture: patterns, frameworks; (4) model visually; (5) verify quality continuously; and (6) manage change. Rational Rose® is the most widely used visual modeling tools based on unified modeling language (UML); Rational PurifyPlus® is a tool for unit testing; while Robot® for functional testing. VisualStudio.NET® from Microsoft is an example of IDE for software design. It provides an environment for managing source code, building solutions, and debugging. Most importantly it follows an open architecture and provides a platform for integrating add-ons to the environment. For example, WinCVS adds version management functionality for the IDE; and most of Rational tools are integrated with the IDE.

These tools make it possible to engineer large-scale and complex software. Some tools are simple in terms of theoretical background, nonetheless they automate some tedious tasks otherwise to be carried out manually. In our opinion, similar tools are desired for batch process design to manage information and knowledge, and to tackle the inherent complexity.

2. Proposed Approach
The ultimate goal of this work is to develop a platform for design team to share information, reuse knowledge, structure the design process and generate documents automatically. To achieve this, the proposed approach is centered on two foundations, which are discussed as follows.
2.1 Design rationale

Many design activities involve reasoning, such as the reasoning on selection of specific equipment due to cost constraint, selecting a solvent over another due to safety concerns etc. The reasoning embedded in the design can be explicitly expressed as design rationale (DR). DR plays a central role in supporting and structuring the design process. Recorded DR is an important information source for other engineering activities, such as operation, maintenance and management of change. Design rationale can also be reused to improve future process designs. Research related to design rationale has been very active during the past decade in many engineering disciplines (Regli et al., 2000).

In the domain of chemical process design, Banares-Alcantara et al. (1997) proposed to record DR and integrate it with design history. Chung et al. (1998) proposed a framework to explore design alternatives, reasons for design decisions, and constraints.

In our approach, DR is divided into two types, generic and specific. Generic DR consists of representation of best practices, guidelines, design methodologies and other design knowledge. Generic DR is process independent and is organized into a hierarchical structure. The root level of the generic DR structure contains nodes for quality, safety, cost, and delivery-related DR. Detailed categories are created under each of the nodes. For instance, the safety related DR is further divided into sub-categories of material, equipment, and human error related. Specific DR are defined as the reasoning for a specific process.

DR can be captured using different approaches (Gruber, 1990). Most widely used methodologies are either to provide electronic notebook for designers to record their design process; or to elicit DR using semi-structured text based on argument-style discussions among designers, such as IBIS, QOC, PHI, DRL (Regli, 2000). Furthermore, designers rely more and more on tools and information sources to refine the design, assess design alternatives, and uncover potential problems related to safety in the early stages of design. The designers assess design alternatives based on analysis using these tools. Thus, reasoning embedded in these functional tools are also important sources of DR. Thus, instead of answering why-questions directly (Gruber, 1990), information necessary to answer the questions using these tools, can be gathered and recorded. In other words, instead of trying to record complete explanation as specific DR as in typical DR systems, the underlying information, which can be used to answer designer’s questions, is captured. Thus, the specific DR refers to real engineering data and models, and is constructed and inferred from information. Hence, in this approach, information necessary for these tools to act upon is captured as specific DR. Facilities to record informal or semi-formal representation of DR manually are also parts of this approach. The most popular representation of DR, IBIS, is used as the representation scheme of DR. Specific DR is stored in the format of XML. The success of acquiring specific DR from various tools relies on the interoperation capability among these tools, and the open communication between IDE and these tools, which is the second foundation of this approach and will be discussed next.
2.2 Integration of tools
As discussed above, the key point in the success of the approach lies in the interoperability among various tools and the IDE. These tools can be considered as agents with specialization in different knowledge areas to assist the designers. Tools for batch process design are also categorized into types of information management, functionality, and IDE, similar as software design tools. Recipe and P&ID are two major pieces of information in batch design; and they have different representation schemes. Recipe is mainly text based, and can be divided into four levels, following ANSI/ISA-S88.01 standard. On the other hand, P&ID is a graphic representation of equipment/instruments and connections between them as pipes/signal lines. Managing this information is certainly more difficult than managing source code in software development. Nevertheless the recent progress on the data centric approach to store recipes and P&ID information in RDMS and using XML as data exchange format, alleviate the complexity. The most important requirement for information sharing is separation of data (including information and knowledge) and its presentation. Instead of simply persistence in a form of object serialization, data should be explicitly expressed in a structured format, such as RDMS or preferably XML. Furthermore, if predefined ontologies are followed in creating the structure of the information and knowledge storage, information sharing and knowledge reuse could be achieved with ease (Zhao et. al, 2003). Ontologies for process information should be created based on existing standards. Most of the existing tools can be considered as information-processing tools, which require information in pre-defined formats and generate output in pre-defined formats. Reasoning and further explanation of the results are carried out manually by designers. In order to automate the task of recording specific DR, the tools should either possess self-explanatory capability or rely on external wrappers to provide this functionality, driven by generic DR during process design.

IDE functions as a hub to integrate various tools and acts as a design platform. In order to achieve the functionality sharing, more flexible control of these tools by the IDE are required, since most of existing tools are constructed as full-fledge applications that run independently. Wrappers or mediators are built around these tools to establish the connection between IDE and specific tools if not readily available in the tools themselves. Furthermore, in order for more consistent information sharing, common representation of the process would be necessary. Petri Nets might be a good candidate for batch processes representation worthy of further investigation. This issue, however, will not be further explored in this paper. To summarize, in order to seamlessly integrate the tools with the IDE, the following features are desired (1) following common ontologies for process information which is stored in an open structure; (2) self-explanatory capability for results; and (3) controllability by the IDE.

3. Overall Architecture and Implementation Issues
A systematic architecture has been created, based on the proposed approach, for developing the integrated batch process development environment. Such a system should be feasible given the success of the IDE for software design, as well as various enabling technologies, including ontology for information and knowledge sharing based on XML; better and more powerful software component technology; information
accessing through Web; functionalities activating through Web applications; and availability of various software tools to support process design.

3.1 Overall architecture

Figure 1 Architecture of the proposed integrated development environment

The integrated, multifaceted environment consists of the following components that are divided into three layers as shown in Figure 1:

- Information and knowledge storage
- A design workspace as the principal medium for implementing design, including
  - A design rationale engine to drive the design, and record issues, answers, and arguments about the design domain
  - A communication engine for each individual tool
- Tools to assist different aspects of design

In order to reuse the specific design rationale obtained from previous designs, case-based reasoning and knowledge mining techniques could be incorporated in the IDE. Two additional components are required to achieve this:

- A catalog provides a collection of presorted design examples, and supporting reuse and case-based reasoning
- A specification component allows designers to describe some characteristics of the design. They are used to retrieve design objects from the catalog and to filter information in the specific DR.

3.2 Implementation issues

The implementation of such a system is divided into following steps: (1) define the ontologies for various kinds of information and knowledge and define XML formats accordingly; (2) create interface for the data presentation within the IDE; (3) create design rationale engine; (4) develop dispatch and record engine and establish communication between the IDE and various tools. More advanced functionalities can be further implemented given the open architecture of the IDE. Desirably, the environment should be able to evaluate current development status and provide intelligent assistant. Collaborative functionalities are also desired given the teamwork
nature of process design. Furthermore, a centralized web-based project portal is useful for better management of the progress and design information. Current implementation of the system is in its infancy and concentrates on core functionalities. Current emphasis is to integrate safety concerns with process design, as stressed in Park (2001), supported by the IDE and integrated tools. The toolset consists of, at this stage, BatchPlus® (from Aspen Tech) as simulation tool and PHASuite (Zhao, 2002) as safety analysis tool.

4. Illustrative Examples

Two simple scenarios from batch process design are used to illustrate the workflow of the environment. In first example the generic DR related to cost and to reduce batch time is considered. The guideline or best practice suggests that a buffer tank could be added to the process. Simulating the process with and without the buffer tank generates Gantt charts, and shows batch time is reduced with buffer tank. Thus the specific DR for using buffer tank is cost related. The information required for the simulation in BatchPlus® and the Gantt charts are recorded as the ‘what’ information. In another example, the generic DR for safety, and equipment safety, (over) pressure drives analysis of potential hazards related to a vessel. PHASuite found a potential vessel overpressure hazards caused by overfill of the vessel, given the operating procedure and the selected equipment. PHASuite suggests safeguard of “install independent high level alarm with instructions to prevent overfilling”. The reasoning behind the high level alarm for this vessel is recorded as specific DR. The information required for PHASuite to generate the overpressure consequence is recorded as the ‘what’ information.

5. Conclusion and Future Work

Batch process design is a complicated task involving large amount of information and knowledge, and various tools are necessary to support it. An integrated development environment framework has been proposed to center on two foundations, i.e. design rationale and integration of different types of tools. Generic DR drives the design, and specific DR are recorded as process specific reasoning behind the design from these tools. The proposed approach also provides the designer a simple and straightforward manner to access these tools, thus makes it easy to move from traditional ‘waterfall’ development style, which is sequential and linear, to an incremental and iterative style. Future work includes further implementation and testing on more complex design problems, and expanding the set of tools that are integrated in the framework.

References
Regli W. C., X. Hu, M. Atwood, and W. Sun, 2000, Engineering with Computers, 16.