A DYNAMICALLY RECONFIGURABLE DATA STREAM PROCESSING SYSTEM

J.M. Nogiec[#], K.Trombly-Freytag*, FNAL, Batavia, IL 60510, USA

Abstract
This paper describes a component-based framework for data stream processing that allows for configuration, tailoring, and runtime system reconfiguration. The system's architecture is based on a pipes and filters pattern, where data is passed through routes between components. A network of pipes and filters can be dynamically reconfigured in response to a preplanned sequence of processing steps, operator intervention, or a change in one or more data streams. This framework provides several mechanisms supporting dynamic reconfiguration and can be used to build static data stream processing applications such as monitoring or data acquisition systems, as well as self-adjusting systems that can adapt their processing algorithm, presentation layer, or data persistency layer in response to changes in input data streams.

INTRODUCTION

Software developers frequently face the challenge of quickly developing an application of high quality. At the same time they are expected to reduce the costs of maintenance and improve reliability. One of the ways to achieve these goals is to use a technology that offers a high level of reuse such as component-based development. In this technology, applications are assembled from previously developed components [1]. The partitioning of the processes of programming and application building achieved with components allows for increased reuse and rapid development. The separation of concerns between programmers and application developers results in a clear distinction between the application structure and programming details, which in turn improves maintainability.

Another challenge faced by software developers is the fulfilment of functional requirements, especially when systems may be required to perform different functions depending on changing environment or requirements. This cannot always be satisfied with writing new applications or modifying existing code. An alternative to this old and costly approach is to build into the systems an ability to evolve or change over time. Component based technology provides this ability by creating a foundation for both static and dynamic reconfiguration of the system, where components can be exchanged, removed or added to the system either at configuration time or at runtime [2]. In recent years, there has been an increased interest in dynamic reconfiguration techniques, including component based reconfigurable systems [2][3][4][5].

The data sequences produced in real-time by such utilities as stock tickers, news feeds or data acquisition systems are known as data streams and the applications which such process continuous data flows are known as data stream processing systems. Many applications utilize data streams to pass information between components, processes and subsystems [1][6][7]. Data stream systems can be easily modelled using components as dataflow systems. The resulting architecture offers low coupling between components, upgradeability, scalability, and extensibility[8].

The Extensible Measurement System (EMS) framework developed at Fermilab is an exercise in the component-based technology, allowing for experimentation with dynamic reconfiguration in a data flow system.

DYNAMISM IN SOFTWARE SYSTEMS

Software systems can be developed as highly specialized single purpose systems, but increasingly often software systems are required to adapt to changes in their environment, new user requirements or conditions that were difficult or impossible to predict at design time. To deal with these and similar problems, a dynamic system that is more flexible and adaptable can be designed.

One can begin introducing flexibility to an application by adding the ability to configure it prior to running. Thus a framework capable of creating many applications that form a product line based on a common architecture can be developed. The next step in flexibility is to allow for some aspects of an application, such as user interface preferences, to be modified at runtime, which is sometimes referred to as tailoring [9]. Finally, an application can be changed, or change itself, at runtime, which is referred to as dynamic reconfiguration.

Dynamic reconfiguration can be characterized as either open or closed based on whether changes to the system can be predicted.. At runtime, a closed dynamic system performs changes that were predicted at build time, whereas an open dynamic system is capable of performing changes that were not anticipated at build time.

Dynamic reconfiguration is the only acceptable solution in the case when continuity of operation and preservation of the application's state and presentation aspects are required. Reconfiguration can also be used to dynamically extend base application functionality with services not

[#] nogiec@fnal.gov
* kfreytag@fnal.gov

pertaining directly to the application's core function (such as enhanced debugging when problems are detected). Moreover, it can happen autonomously, that is, without the user's intervention.

In the case of a dataflow system, reconfiguration can include the following:

- Modification of the data flow topology
- Activation/deactivation of data sources or data processing paths
- Addition/removal/exchange of data processing components
- Modification of components' properties
- Activation/deactivation of various tasks (activities, computations, subsystems, etc.).

EMS FRAMEWORK

EMS is a component-based framework designed to build data acquisition or data processing applications [10]. Its architecture is based on a pipes and filters pattern, where data is passed through routes between components. EMS provides for applications to be built through assembly of components that then form a network of processing elements. The topology (connections between components) is defined separately for the data, control, debug, exception, and property types of communication. The initial configuration of an application is described using a specialized configuration language that is based on XML [11][12]. The purpose of this configuration language is to specify components and their interactions, and it includes descriptions of components, routes, and initial control signals.

Components

Components are the fundamental building blocks of applications. Each component has a number of properties, which can be examined and/or set externally by other components. One such property is the component's state, which can be manipulated by sending control events to the component. In response to control events, components perform requested actions and eventually change their states. Some components are purely data driven, and while in the running state, their actions depend solely on the received data. These components process data and add, substitute, and/or remove named data items from a data stream. They can also manipulate data streams by buffering data, compressing or decompressing individual streams, and combining, splitting, or synchronizing multiple data streams. Components, depending on their functionality, range from a highly specialized to fairly universal ones. EMS consists of configurable general-purpose components for manipulating streams, visualizing data, persisting data, and reading data from various standard data sources, and many application specific components, such as DSP, scripting, or instrumentation-specific components [13].

Events

Components can be recipients and/or sources of events. A single component can be a recipient of one type of event and a source of another. There exist also translator components that can translate between different types of events and serve as junction points for flows of different events. Typical data processing components are recipients of data and control events and sources of data, exception, and debug events. A typical control component is the recipient of control events and source of control, property, exception, and debug events. Data acquisition components serve as data event sources, with data events being collections of named data items. Data persistence and visualization components provide the output of the system, and are typically destinations of data, control, exception and debug events, while being the source of only exception and debug events.

Routes

Routes are unidirectional connections between components. They are specified separately for each type of event defined in the framework; that is the data, control, debug, exception, and property events. Both multicast and broadcast communications can be implemented via these routes, and they can be inspected graphically with the help of a configuration tool [13].

RECONFIGURATION TECHNIQUES

There are several techniques that could be employed to alter the behaviour of an EMS-based system:

- Modification of component properties
- Modification of data paths
- Activation/deactivation of components
- Use of gate components
- Addition/removal of components
- Colouring of data
- Use of source routing.

Modifiable Component Properties

A component's functionality can be altered via changes to the component's properties, either by the user (tailoring) or the system itself (adaptation). Properties can "fine-tune" the actual processing performed by a component, possibly including a complete change of the algorithm within the component [11][12].

Modifiable Data Paths

Changes to the routing can cause the algorithm of the application to change dramatically. This could, for example, supply data for debugging purposes or provide a shuttle around parts of the system that are currently not working to alternative sub-systems.

Activation/Deactivation of Components

Each component can be brought to the inactive state by sending a special control event to it, in which state it will ignore all incoming data.

Gate Components

The data paths to a component can be opened or closed using specialized gate components. All components downstream from the gateway will cease to receive data.

Colouring of Data

The data colouring method is modelled on the decorator pattern, where data driven components act only on the "interesting" data. Data is marked (using additional data items as markers), and each set of uniquely marked data is processed by a different set of components on their path through the system.

Adding/Removing Components

Components can be added, removed or replaced by other components in order to directly alter the data processing. Custom loaders can be employed to dynamically load code from various available code repositories.

Source Routing

Source routing allows a component to send its results to a dynamically chosen component, rather than, or in addition to, those components connected to it via configured data routes. The addresses of recipients can be redefined at runtime as properties of the source component.

RECONFIGURATION PROCESS

Data stream processing systems are naturally modelled as dataflow systems, which in turn can be easily implemented using component-based technology. In such systems the interdependencies among data processing steps and their boundaries are explicitly defined, thus providing a transparent structure of the system, and therefore simplifying the implementation of dynamic reconfiguration.

EMS was designed to implement this class of systems. In EMS, where components are stateful and their dependencies are explicitly defined, recognizing and verifying a valid reconfiguration state is easily achievable. Because the state of each component is known, reconfiguration strategies do not have to deal with the state of the entire system, but rather only with the state at the specific point of reconfiguration. This allows for quicker reconfiguration, as well as less disruption of the entire system.

Specialized Reconfiguration Components

The component-based architecture of EMS allows for easy separation of the application's functional and non-functional aspects, specifically including the aspect of reconfiguration. One can introduce a set of dedicated reconfiguration components to an application that will manipulate the topology, act on the states of various components, or load and start new components. This approach separates the reconfiguration specific concerns from the functional code.

While the EMS offers reconfiguration capabilities, it doesn't enforce any specific reconfiguration strategies or techniques. Various reconfiguration strategies can be implemented, and decisions on the type of strategy to use are left to the application designer. This allows each application to choose the reconfiguration strategy most applicable to its particular set of requirements and quality of service measures.

Reconfiguration with Scripting

The scripting capabilities of EMS can be used for controlling the application and the sequencing of its actions. It allows for rapid modifications to the behaviour of the system and rapid development of specialized applications [13]. In addition to these typical scripting uses, one can employ this mechanism to dynamically tailor or reconfigure a running application. One can use scripting to program known sequences of actions and corresponding changes of the configuration. At runtime, the system will go through the pre-programmed configurations and perform the requested computations or data acquisitions for each of them.

Scripts can directly inspect and modify the properties of components, manipulate the components' states, and add or remove components and routes from the application. Property manipulations and control requests are sent as events to the appropriate recipient components. When applying scripts to reconfiguration, one also guarantees the desired separation of concerns between the functional code and the code responsible for configuration. A script contains only control and property operations that stimulate, sequence and control the application, rather than containing any data processing code.

The use of an interpreted scripting language indicates there is no imposition of compile-time linking of components and therefore allows for the dynamic specification of components at runtime. Conditional structures can be used to reconfigure the application according to the runtime environment. Therefore, an interpreted scripting language is an easy to use and to maintain reconfiguration tool, well suited to closed dynamic systems.

CONCLUSIONS

The demand for dynamicity in software systems is one of the key challenges facing developers today. Component-based systems form a good foundation for building configurable and dynamically reconfigurable systems. At the same time, a configuration-based framework provides a solid basis from which to describe the organization of an application. Both allow for building dynamic applications capable of keeping up with changing requirements and demands. Data stream processing systems, due to their lack of cyclic dependencies, can be modelled with pipeline architecture, and when implemented with components are well suited to dynamic reconfiguration.

EMS is a configuration-driven component-based system, which offers both static configuration and dynamic reconfiguration capabilities and is suitable for building data stream processing applications. The EMS framework provides the mechanisms needed to accomplish dynamic reconfiguration, whereas the actual dynamic reconfiguration can be directed by the interpreted scripting language or by specialized reconfiguration components. The reconfiguration can be: a) a result of the adaptation process, in which the application itself responds to the changing environment or changes in data, b) a result of an operator-initiated transformation, or c) a part of the normal lifecycle of an application, wherein different processing is needed at different phases.

EMS, through its flexible and powerful reconfiguration mechanisms proved to be a useful test-bed in the area of coordination, adaptation, dynamic reconfiguration and data flow processing. It has already has already been used to build production quality systems that have been successfully deployed [11][12].

ACKNOWLEDGEMENTS

The author would like to thank his colleagues Dana Walbridge, Sergey Kotelnikov, and Gene Desavouret for their contributions to the magnetic measurement systems built with the EMS framework.

REFERENCES

[1] A. Lui, M. Grigg, T. Au, M. Owen, "Component Based Application Framework for Systems Utilising the Streaming Data Passing Semantic," 37th International Conference on Technology of Object-Oriented Languages and Systems, Sydney, 2000.

[2] T. Batista, N. Rodriguez, "Dynamic Reconfiguration of Component-based Applications," Proceedings of the International Symposium on Software Engineering for Parallel and Distributed Systems, June 2000, pp 32-39.

[3] A. Rasche, A. Polze, "Configuration and Dynamic Reconfiguration of Component-based Applications with Microsoft .NET," Proceedings of the Sixth International Symposium on Object-Oriented Real-Time Distributed Computing, 2003.

[4] K. Hawick, H. James, P. Coddington, "A Reconfigurable Component-based Problem Solving Environment," Proceedings of the 34th Hawaii International Conference on System Sciences, Hawaii, 2001.

[5] A. Ketfi, N. Belkhatir, "Open Framework for the Dynamic Reconfiguration of Component-based Software," SERP'04, Las Vegas, 2004.

[6] D. Abadi et al., "Aurora: a new model and architecture for data stream management," The VLDB Journal, August 2003.

[7] The STREAM Group, "STREAM: The Stanford Data Manager," IEEE Data Engineering Bulletin, Vol. 26 No. 1, March 2003.

[8] J. Guo, S. Edwards, D. Borojevic, "Implementing Dataflow-based Control Software for Power Electronics Systems," CPES Seminar 2003, pp. 429-434, Blacksburg, VA, 2003.

[9] I. Mejuev, A. Kumagai, E. Kadokura, "Tailorable Software Architectures in the Accelerator Control system Environment," ACAT2000, Batavia, 2000.

[10] J.M. Nogiec, J. Sim, K. Trombly-Freytag, D. Walbridge, "EMS: A Framework for Data Acquisition and Analysis," ACAT2000, Batavia, 2000.

[11] J.M. Nogiec et al., "An XML Driven Framework for Test Control and Data Analysis," ICALEPCS 2001, San Jose, 2001.

[12] J.M. Nogiec et al., "A Flexible and Configurable System to Test Accelerator Magnets," PAC'01, Chicago, 2001.

[13] J.M. Nogiec, E. Desavouret, S. Kotelnikov, K.Trombly-Freytag, and D. Walbridge, "Configuring Systems from Components: The EMS Approach," ACAT'03, Tsukuba, 2003.