Work flow view of a distributed application

by J. R. HAMSTRA
Sperry Univac
Roseville, Minnesota

INTRODUCTION

Work Flow Management is a unified set of concepts for the definition, implementation and operation of Application Systems. A companion paper to this one provides a more general treatment of the requirements motivating Work Flow Management. An Application System is a major function of the work of a computer system as perceived by the customer. Thus Application Systems often mirror the structure or work of the customer enterprise. In this paper we shall consider a credit card processing application, forming a major function of the work of the hypothetical Masterkey Credit Corporation (MCC). The portion of this application performed on the MCC distributed computer system is called the Credit-Cards Application System. MCC also uses its computer system for other applications, including personnel and payroll, financial information, and the development and maintenance of applications.

People relate to an Application System in various roles, including end user, systems analyst, programmer and operator/administrator. Figure 1 illustrates these relationships. Work Flow Management attempts to facilitate communication among these various parties including the Application System, by supporting high-level descriptions expressed in the Work Flow Definition Language.

Within an Application System certain kinds of work are performed repeatedly. This is called recurring work, as opposed to ad hoc work performed once. Recurring work, the primary concern of Work Flow Management, may be parallel (e.g. charge slips are processed concurrently in each of 10 regional data centers, and may indeed be processed concurrently within a single center) or cyclic (e.g., customers are billed monthly based in part on their previous statements) or both. Recurring work is more easily described by defining the underlying structure of the work, than by explicitly enumerating or generating the instances.

In general, a schema (pattern, diagram, schematic) provides a structured “template” of information for generating and controlling instances of complex entities. A Work Flow Schema is the description of an Application System expressed in the Work Flow Definition Language. It describes the structure of the recurring work of an Application System, with provision for the dynamic introduction of modifications and ad hoc work.

A Work Flow Schema is compiled into an internal form interpreted by the run-time system. This produces, in effect, a customized applications executive providing a complete simulated environment, including facilities for production work, as well as application modification, testing and training, and auditing and recovery. Thus the Work Flow Definition Language can be classified as a simulation language, albeit a special purpose one since the classes of simulated entities are predetermined (see Figure 2).

In the remainder of this paper we will develop the basic Work Flow concepts and show how they are represented in the Work Flow Definition Language. Being an overview, this paper must omit some components, and the descriptions of those presented are necessarily simplified. Nevertheless, it is intended to give the reader some insight into the scope, style and power of both Work Flow Management and the Work Flow Definition Language.

WORK FLOW CONCEPTS

Functional distribution

As previously suggested, the work of the Credit-Cards Application System proceeds simultaneously in 10 regional data centers. These centers are logical processing environments perceived by Work Flow Management as Applications Environments called Regional-Data-Centers. The portion of the work of Credit-Cards performed within each Regional Data Center is called a Regional-Operations Sub-Application System.

Figure 4 illustrates these relationships, using the structure notation defined in Figure 3 and the symbols for Sub-Application System and Sub-Applications Environment shown in Figure 2. Note that the relationship between Regional-Data-Center and Regional Operations is transient, since the logical work of one Regional-Operations may be moved to a different actual Regional-Data-Center if some untoward event, such as a flood, renders the first Regional-Data-Center unavailable.

We have considered the first-level decomposition of the Credit-Cards Application System. At this level, it comprises similar functions (Regional-Operations) distributed among similar processing environments (Regional-Data-Center) in a pre-determined yet alterable manner. Work Flow Management encourages and supports the functional decomposition of Application Systems and the controlled distribution of the...
functional components. This contrasts with the emphasis on load-sharing or data base distribution found in many other approaches to distributed processing, although Work Flow Management does not preclude either of these. Indeed, it requires certain forms of data distribution. MCC distributes its work by dividing the United States into 10 regions and keying both credit card and merchant identification to these regions.

Considering the requirements of credit card processing within a Regional-Operations Sub-Application System, it is reasonable to decompose them into functions associated with the cardholders serviced by that region, called Cardholder-Operations, and functions associated with the merchants serviced by that region, called Merchant-Operations. MCC derives revenue from both Cardholder-Operations (membership fees and interest) and Merchant-Operations (service charges and discounts). Considering that each Regional-Operations is a separate profit center, while cardholders may use their cards anywhere in the country (MCC has no direct international operations), we conclude that management will desire, and accountants and auditors require, separate control of inter-regional financial transactions. These functions are provided by an Inter-Regional-Operations Sub-Application System within each Regional-Operations. Figure 5 illustrates the structure of Regional Operations.

Although further decomposition of Credit-Cards into Sub-Application Systems is possible, it is not essential to this presentation and will not be pursued.

Data flow

Information processing is the work of transforming and communicating data. Availability of the data is both a necessary precondition and a major stimulus for performing this work. Thus information processing systems (organic and mechanical) are largely driven either implicitly or explicitly by data flow. Work Flow Management relies heavily on data flow to control the Application System. In the remainder of this section we consider the flow of charge information from the merchants to the cardholder accounts. This example will illustrate the key Work Flow concepts of Transaction Groups, Tasks and Queuing Points.

Merchants participating in the MCC system accumulate batches of charge slips. Each slip contains merchant and cardholder identification encoded in a suitable optical-character-recognition (OCR) font. A merchant will periodically deliver these batches to his MCC Regional-Data-Center, either directly or via his bank. The merchant is reimbursed for the total amount of these charges, less a computed discount. The amounts on each slip are OCR-encoded, then the batches are read into the computer system via an OCR-
reader. This is the first operation perceived by the Application System.

The internal form of a batch of charge slips is a Transaction Group called Sales-Inputs. A Transaction Group can be thought of as a bundle of transactions to be routed and processed together; however, its actual internal structure is considerably more complex than this, to satisfy the combined requirements of program data access and the Work Flow integrity/recovery architecture. A Transaction Group need not represent an external entity such as a batch of charge slips. Transaction Groups are the units of data flow within an Application System.

Sales-Inputs Transaction Groups are generated within the computer system by the operation of OCR-readers. Within a Merchant-Operations Sub-Application System this is represented by Optical-Character-Reader Tasks. A Task is a basic instance of work within an Application System. Tasks are the users of data, i.e., they produce, utilize and/or consume Transaction Groups. In addition to Optical-Character-Reader Tasks, the flow of Sales-Inputs Transaction Groups involves two other kinds of Tasks. The first is the crediting of the merchant accounts within Merchant-Operations, called Update-Merchant-Accounts Tasks; the second is the debiting of the cardholder accounts within Cardholder-Operations, called Update-Cardholder-Accounts Tasks.

Figure 6 illustrates the complete flow of Sales-Inputs within the Application System, using the schematic structure notation. After Update-Cardholder-Accounts Tasks the Sales-Inputs Transaction Groups are no longer necessary and they cease to be active entities within the Application System, although they are retained as archival entities for auditing and recovery purposes. We conclude this topic by noting that a Work Flow Schema contains a complete producer/consumer model of data flow within the Application System.

Commitment control

The foregoing treatment of the flow of charge information would be satisfactory were cardholders not permitted to make purchases outside their home regions. The absence of this restriction poses complications not readily resolved even if Update-Cardholder-Accounts has access to a global cardholder data base distributed among the ten Regional-Data-Centers. First, all Regional-Data-Centers may not always be available, a fact that should not hinder local processing at other centers and one that we wish not to expose to the individual Tasks running elsewhere. Second, each Regional-Operations is a separate profit center requiring a separate reckoning of inter-regional financial transactions, which would be hidden in a single distributed data base. Third, we do not wish to submit transactions to a remote region until we have some degree of confidence in the results of the processing that produced them, nor do we wish to post transactions from a remote region without similar control. We will now address these problems.

The first part of the support for remote purchases is the introduction of Remote-Purchases Transaction Groups to effect the flow of this data among the various Regional-Operations as shown in Figure 7. Each Update-Cardholder-Accounts Task can optionally produce one Remote-Purchases Transaction Group for each of the nine Regional-Operations other than its own. This is called data flow fan-out. A new kind of Task called Remote-Cardholder-Updates accepts Remote-Purchases Transaction Groups from one or more other Regional-Operations and debits the local cardholder accounts accordingly. This is called data flow fan-in. Figure 7b illustrates this arrangement for three Regional-Operations.

We now consider the support for inter-regional financial accounting of Remote-Purchases Transaction Groups. This consists of two kinds of Tasks within the Inter-Regional-Operations Sub-Application Systems, Outbound-Remote-Balancing to post outgoing Transaction Groups and Inbound-Remote-Balancing to post incoming Transaction Groups. Figure 7c illustrates the general flow of Remote-Purchases Transaction Groups.

Our treatment has regarded data flow as essentially par-
parallel and asynchronous. However, application integrity and recovery control require that certain phases in processing be synchronized. This requirement was first explicitly applied to the flow of Remote-Purchases Transaction Groups among Regional-Operations, but we have in fact relied on it throughout this presentation. Furthermore, the fact that work will proceed at different rates and with different levels of actual concurrency in different parts of the system and at different times imposes additional requirements on the synchronization of data flow. These requirements are subsumed under the general notion of controlled commitment.

Commitment (consigning, binding over for use) occurs whenever data produced by one Task is made available for use by other Tasks. This may occur when data in a shared data base is unlocked; however commitment in this manner is not very amenable to higher-level control. being necessarily synchronized to the internal operations of the Task. Transaction Groups make the flow of data between Tasks explicit. This explicit flow can best be utilized for Application control if an external agency is interposed between the relinquishing of control of a Transaction Group by one Task and the acquisition of control by another. This agency is a Queuing Point.

Queuing Points are mail boxes for Transaction Groups. They serve as brokers to acquire and dispose of Transaction Groups for Tasks, the actual users of data. Thus they serve to decouple individual Tasks from what, when and where other Tasks exist. The availability of data at a Queuing Point may be the stimulus for scheduling work, or the data may be held at Queuing Points pending some other stimulus such as time or administrator action.

We can satisfy the remaining requirements for control of the flow of Remote-Purchases Transaction Groups with appropriate Queuing Points. Within the Inter-Regional-Operations Sub-Application System of each Regional-Operations Sub-Application System we establish one Remote-Region Queuing Point for each other Regional-Operations. The Remote-Region Queuing Points accumulate Transaction Groups bound for the other Regional-Operations. Upon a suitable administrator command, Outbound-Remote-Balancing removes each Transaction Group from the selected Remote-Region Queuing Point, posts appropriate information to the Inter-Regional-Operations database, and forwards the Transaction Group to Inter-Regional-Operations within the remote Regional-Operations.

Each Inter-Regional-Operations accumulates incoming Transaction Groups at a single Inter-Regional-Transactions Queuing Point. Upon a suitable administrator command, Inbound-Remote-Balancing removes each Transaction Group from the Inter-Regional-Transactions Queuing Point, posts appropriate information to the Inter-Regional-Operations data base, and forwards the Transaction Group to the appropriate place within this Regional-Operations. In the case of Remote-Purchases Transaction Groups, this is Remote-Cardholder-Updates.

Figure 8 illustrates the complete flow of charge information into the appropriate accounts, applying the principle that all Tasks are decoupled by Queuing Points. This figure shows the power of the schematic definition concept. It shows the flow of work through the Application System in an inherently parallel manner, yet it admits of the necessary synchronization and control. The Queuing Points provide for the unification of a network communications model (data flow) with a hierarchical processing model (data transformation).

The names of the entities in Figure 8 are actually names of types of entities within the classes of entities denoted by the shapes in Figure 2. At any given time there exist multiple occurrences of entities of each named type, e.g., multiple Tasks of type Update-Cardholder-Accounts and multiple Queuing Points of type Remote-Region. Note that there might even exist multiple Application Systems of type Credit-Cards, e.g. for testing within MCC, or because a software house has applied this application to companies other than MCC. Much of the power of the Work Flow Definition Language derives from its ability to define com-
plex application structures in terms of the underlying types of entities, while associating enough information with these named types to adequately control the actual occurrences. In the next section we will show how this is done.

WORK FLOW DEFINITION LANGUAGE

Global structure

We have presented the structure of an Application System as a schematic diagram, and we will now examine its expression in the Work Flow Definition Language. The following prose description of the language is presented to explain the appendices, which are intended to convey a better understanding of the language. Appendix A summarizes the conceptual structure of the Work Flow Definition Language. The language is basically block-structured with recursive nesting of the higher-level constructs, i.e., Sub-Application Systems and Sub-Applications Environments. The representation is free-form with punctuation and indentation optional.

(Sub)-Application Systems comprise application-blocks delimited by terminal constructs. Application-blocks describe nested spheres of control of work within an Application System. Application-blocks may contain, in addition to nested higher-level constructs, the declarations of types of lower-level entities of the categories internal, external and environmental. These will be described further in subsequent sections.

(Sub)-Applications Environments comprise environment-blocks delimited by terminal constructs. Environment-blocks describe processing environments which do not directly perform any work, although they may contain Sub-Application Systems which do perform work. Environment-blocks may contain a subset of the entities contained in application-blocks, internal and external entities being excluded.

Appendix B is a sample Work Flow Schema for the portion of the Credit-Cards Application System described in the second section. While this schema conforms to the structure shown in Appendix A, it contains assertions more detailed than shown in Appendix A. It is the assertions appearing as rules or policies associated with the structural entities, that

Figure 8—Complete flow of charge information.
give much of the meaning to the schema. These assertions
apply generally or by default within the scope of the decla-
rations containing them. Many more kinds of assertions can
be made than are illustrated in the sample schema; however,
the sample should illustrate the concept.

Internal entities

The classes of internal entity types in a Work Flow
Schema are Transaction Groups, Application Modules,
Queuing Points, Clocks and Calendars. These define the
work performed internal to, and under complete control of,
the Application System.

As previously stated, Transaction Groups have extensive
internal structure, although this has been deliberately omit-
ted from the example in the interest of clarity. This structure
may be partly described in the Work Flow Schema, but is
usually completely defined in a data schema. An extensive
repertoire of control functions exists for Transaction
Groups, and a few examples are given. Initiate creates a
Transaction Group and attaches it to a Task. Terminate is
the inverse of Initiate. Export transfers control of a Trans-
action Group from a Task to a Queuing Point. Import is the
inverse of Export. Pass is an Import/process/Export se-
quence.

An Application Module is a program and related control
information defining a type of Task. Tasks may be scheduled
explicitly, or implicitly on data arrival. Assertions associated
with the Application Module define the local data environ-
ment for the Task. These assertions may also generate a
Queuing Point type of the same name.

Queuing Points are usually generated from Application
Module declarations; however, they may be explicitly de-
clared when specific routing and control functions are de-
sired. These functions include, e.g., Hold, which inhibits
the Import of data from the held Queuing Point, and Release,
which is the inverse of Hold. Normally one occurrence of
the named type of Queuing Point will be generated for each
occurrence of the containing or generating type of entity,
e.g., Application Module, Sub-Application System or Ter-
ninal Group. The occurs-clause in the Remote-Region decla-
rarion specifically controls the generation of occurrences
of Remote-Region Queuing Points.

Multiple named Clocks and Calendars may exist within a
Sub-Application System. Clocks and Calendars are entirely
synthetic. While they may represent real-world time, they
may, on the other hand, represent nothing more than logical
state in some abstract event space. Further discussion of
Clocks and Calendars is beyond the scope of this paper.

External entities

The classes of external entity types in a Work Flow
Schema are Terminal Groups, User Groups, Conversations
and Workstations. These define the work performed at the
interfaces to, and under partial control of, the Application
System. Terminal Groups are types of external interfaces to
an Application System. They manifest themselves as types
of Tasks within the Application System. Queuing Points
may also be generated for them. The information provided
in the Work Flow Schema, in conjunction with communi-
cations configuration information, can be thought of as con-
trolling a “daemon module” which defines the Task. These
functions are provided by the implementor of the computer
system, since it is generally unrealistic and undesirable to
expect the customer to implement them.

User Groups and Conversations represent, respectively,
types of external users of, and types of their interactions
with, the Application System. External users may actually
be organic (human), mechanical, or other Sub-Application
Systems. They manifest themselves as types of Tasks and
Queuing Points within the Application System, controlled
as with Terminal Groups. Conversations manifest them-
selves as Transaction Group types within the Application
System. Further discussion of User Groups and Conversa-
tions is beyond the scope of this paper.

Workstations represent locations for tracking data flow
external to the computer system, and have utility primarily
for external production control. Further discussion of Work-
stations is beyond the scope of this paper.

Environmental entities

The classes of environmental entity types in a Work Flow
Schema are Data Bases, Scheduling Classes, Resource
Budgets and Control Points. These exist primarily to resolve
the mapping of the work of an Application System onto the
supporting processing environments. Thus the actual nature
of the named entity types is largely resolved beyond the
scope of the schema.

Data Bases are identical to Transaction Groups except
that their existence is controlled external to the Application
System. The possibility that they may be shared with other
(Sub)-Application Systems is also presumed. The internal
structure of Data Bases is identical to that of Transaction
Groups.

Scheduling Classes and Resource Budgets represent, re-
spectively, the quality of service (e.g., priority, response
time) and the quantity of service (e.g., resource consump-
tion) provided for various units of work within an Applica-
tion System. Further discussion of Scheduling Classes and
Resource Budgets is beyond the scope of this paper.

Control Points are the means of exchanging Application
System control information among interested parties. Fur-
ther discussion of Control Points is beyond the scope of this
paper.

CONCLUSION

Summary

In this paper we have introduced the general notions of
schemas and Application Systems. We have presented the
basic concepts of Work Flow Management, including func-
tional distribution, data flow and commitment control. We
have shown how they apply to a hypothetical credit card
processing application. Then we have presented the Work Flow Definition Language, considering the global structure of the language and its external, internal and environmental component entities. Finally, we have described a portion of the credit card processing application in this language, illustrating many of these constructs.

Any introductory presentation of a subject must omit certain components. Chief among those omitted here are:

1. Overall application control, including events, conditions, Clocks, Calendars and Work Flow Procedures.
2. The administrators' and programmers' views of the system, insofar as they extend beyond that of the systems analysts as reflected in the schema.
3. The end users' view of the system, including interactive conversation control and display management.
4. Production control, including work performed according to a schedule rather than on demand, and the correlation of internal data flow with external data flow.
5. Integrity control, including auditing, security and recovery.
6. Application development, testing and training.

Figure 9 illustrates the overall structure of a running Work Flow Management system and its relationship to other parts of a computer system. While not all components shown are part of Work Flow Management, they are all essential to a useful computer system. Work Flow Management brings these components together to provide a unified set of tools for the support of customer applications.

ACKNOWLEDGMENTS

Many people have contributed to the ideas presented here, including the current members of the Work Flow group—Marv Beriss, Nigel Dolby, Tony Jenkins and Frank Streine. I am particularly indebted to Neville Black, who first proposed the idea of Work Flow Schemas, and to John Blasdale, who stated the requirements of credit card processing as a sample problem.

The views expressed in this paper are those of the author, and not necessarily those of Sperry Univac.

REFERENCE

APPENDIX A. Work Flow Definition Language - Conceptual Structure

Application-system

- Application-block
  - Sub-application
    - Application-module
      - Transaction-group
        - Application-module
          - Queuing-point
            - Clock
              - Calendar
  - Environment-block
    - External-entity
      - Terminal-group
        - User-group
          - Conversation
            - Workstation
    - Internal-entity
      - Control-point
        - Scheduling-class
          - Resource-budget
            - Database
APPENDIX B

This is a sample Work Flow Schema for the Credit Card scenario, as presented by J. R. Hamstra at the National Computer Conference on 6 June 1979.

The following conventions are used in this preparation:

1. Key words are written entirely in capital letters.
2. Name words are written with their first letters capitalized.
3. Hyphenated words are denoted by -.
4. Compound names are separated by ..
5. Comments are delimited by " "; end of line also terminates comments.
6. Optional constructs are delimited by [ ].
7. Any other use of punctuation is optional.

APPLICATION [SYSTEM] Credit-Cards

APPLICATIONS ENVIRONMENT Regional-Data-Center

Each Regional-Data-Center is controlled external to the Application System.

The names enumerated here are externally resolved references.

OCCURS IN New-York, Philadelphia, Atlanta, Chicago, St-Louis, Dallas, Denver, Los-Angeles, San-Francisco, Seattle

SUB-APPLICATION [SYSTEM] Regional-Operations

DATABASE Cardholder-Information END
DATABASE Merchant-Information END
DATABASE Inter-Regional-Information END

Databases are distinguished from Transaction Groups only by the fact that their existence is controlled external to the Application System. Their names are externally resolved references.

TRANSACTION GROUP Sales-Inputs END
TRANSACTION GROUP Remote-Purchases END

etc

SUB-APPLICATION [SYSTEM] Cardholder-Operations

USE Cardholder-Information
SUB-APPLICATION [SYSTEM] Cardholder-Services

etc

END [Cardholder-Services]

SUB-APPLICATION [SYSTEM] Credit-Management

etc

END [Credit-Management]

SUB-APPLICATION [SYSTEM] Cardholder-Accounts

[APPLICATION] MODULE Update-Cardholder-Accounts

These declarations will generate an implicit Update-Cardholder-Accounts Queuing Point, in addition to the Application Module.

IMPORT AND TERMINATE EACH Sales-Inputs
INITIATE AND EXPORT OPTIONAL Remote-Purchases TO EACH Remote-Region

Inter-Regional-Operations contains the Remote-Region Queuing Points.

END [Update-Cardholder-Accounts]

[APPLICATION] MODULE Remote-Cardholder-Updates

These declarations will generate an implicit Remote-Cardholder-Updates Queuing Point, in addition to the Application Module.

IMPORT AND TERMINATE EACH Remote-Purchases
APPENDIX B

END [Remote-Cardholder-Updates]

' etc
END [Cardholder-Accounts]
END [Cardholder-Operations]

USE Merchant-Information
SUB-APPLICATION [SYSTEM] Merchant-Services

' etc
END [Merchant-Services]

SUB-APPLICATION [SYSTEM] Merchant-Accounts
TERMINAL GROUP Optical-Character-Reader

INITIATE AND EXPORT Sales-Inputs TO Update-Merchant-Accounts
END [Optical-Character-Reader]
[APPLICATION] MODULE Update-Merchant-Accounts
' These declarations will generate an implicit Update-Merchant-Accounts
' Queuing Point, in addition to the Application Module.
PASS EACH Sales-Inputs TO Update-Cardholder-Accounts
END [Update-Merchant-Accounts]

' etc
END [ Merchant-Accounts]
END [Merchant-Operations]

SUB-APPLICATION [SYSTEM] Inter-Regional-Operations
USE Inter-Regional-Information

ON INITIATE HOLD Inter-Regional-Transactions AND EACH Remote-Region
QUEUING POINT Remote-Region

OCCURS IN CURRENT Regional-Operations FOR EACH OTHER Regional-Operations
ON RELEASE SCHEDULE Outbound-Remote-Balancing
END [Remote-Region]
[APPLICATION] MODULE Outbound-Remote-Balancing
PASS EACH Remote-Purchases FROM Remote-Region

TO Inter-Regional-Transactions,
THEN HOLD Remote-Region
END [Outbound-Remote-Balancing]
QUEUING POINT Inter-Regional-Transactions
ON RELEASE SCHEDULE Inbound-Remote-Balancing
END [Inter-Regional-Transactions]
[APPLICATION] MODULE Inbound-Remote-Balancing
PASS EACH Remote-Purchases FROM Inter-Regional-Transactions

TO Remote-Cardholder-Updates,
THEN HOLD Inter-Regional-Transactions
END [Inbound-Remote-Balancing]

' etc
END [Inter-Regional-Operations]
END [Regional-Operations]
END [Regional-Data-Center]
END [Credit-Cards]