# An Architecture and Prototype Implementation of a System for Individualised Workflows in Medical Information Systems

**Joxan Jaffar*** National University of Singapore *joxan@iscs.nus.sg*

**Michael J. Maher*** Griffith University Brisbane, Australia *M.Maher@cit.gu.edu.au*

**Gustaf Neumann*** University of Essen Essen, Germany *Gustaf.Neumann@uni-essen.de*

## Abstract

*This paper is a report from a project that tried to design and implement a highly flexible workflow management system in the area of patient care delivery. Within eight weeks the authors developed a formal model for the specification of medical directives, designed and implemented a coordination system (based on the IBM prototype* ObjChart*) with multi-user capabilities, and developed graphical user agents for the most important classes of users of the system (medical doctors, nurses and nurse technicians). This rapid development was possible not least due to the modular design of the system, which is made up of highly specialised, declarative and orthogonal components.*

## 1 Introduction

This paper presents an approach and an implementation of a workflow management system in the area of medical information systems. While the scope of possible applications of workflow systems in the area of medical information systems in general is very wide and can cover many different application areas ranging from admission of a patient to discharge, this paper concentrates solely on the area of patient care. In particular, we designed and implemented a workflow system (called Diabetes Manager) for a hospital department devoted to the treatment of diabetes. As in other areas of care, one of the particular problems of diabetes care is that the treatment is often affected significantly by other medical problems that a patient suffers. This implies that in this application area there is a strong need to tailor all medical and application aspects according to the needs of each particular patient, leading to patient centered care and patient centered workflows for the delivery of the treatments.

The primary goals of the application were:

- *Strong reliability:*
  The users of the system (medical doctors, nurses, technicians, administrators) are typical end users without computer science background. The system must be simple to use, it must be as transparent as possible to avoid mistakes, the system must be understandable and predictable and provide various means of reporting and logging to achieve this.

- *Flexibility:*
  The users of the system should be empowered to alter the system within their responsibilities, within the globally enforced policies of the hospital. Technically this means that the system must be able to alter the general workflow (processes), the coordination mechanisms (scheduling and transfer of duties) and the medications (prescriptions, application semantics) on the fly, without disturbing the other operations of the system.

- *Well-structuredness:*
  Both we, the developers of the system, and

* This work was performed as part of a project at the IBM T.J. Watson Research Laboratory, Hawthorne, NY, USA. At the time of the project all three authors were employees of IBM.

† This paper was published in the Proceedings of the Thirty-Second Hawaii International Conference on System Sciences HICSS-32; IEEE Computer Society Press, January 1999.

the involved project members from the hospital had a high interest in a clear structure of the system, which was built with well defined system components and modelling techniques. The goal was to unify the modelling and the implementation process as far as possible, and to view the system as an "executable specification": altering the model should result immediately in a different behaviour of the system.

- *Formalisation of the application semantics:* By providing an infrastructure through our workflow management system the hospital had, for the first time, the chance to formalise the patient case delivery model. This made it possible to analyse carefully the application model (e.g. the directives model of the medical doctors (see Section 2.1), the assignment model for nurse stations (Section 3), etc.) and to detect opportunities for improvement.

- *Clear organisation of application data:* For the practical use of the system it was necessary to provide the patient data in an easy assessable manner with graphical charts where appropriate. The primary metaphor we used for reporting of patient data was the *patient chart* which serves as the single place to look up patient information.

After providing this background information the next section (1.1) will introduce the basic terminology and will position our view of workflow management in relation to other approaches provided by the literature. Section 1.2 will apply the general concept of workflow management to patient care. Section 2 will discuss our approach to how the treatments of the patient (that will result in tasks to be performed, for instance, by nurses) can be specified by medical doctors. The directives that are specified by the doctors are generators of basic activities that have to be executed at some specified time (we will call these "workitems"). Section 3 discusses how the identified workitems are distributed among the workers in the system, realised by the hospital staff and recorded in the system. After that, section 4 describes the architecture and the components of the system. The summary in section 5 concludes the paper.

### 1.1 Workflow Management Systems

The Workflow Management Coalition [6] defines a workflow management system as "*a system that completely defines, manages and executes 'workflows' through the execution of software ...*". A workflow management system therefore has to provide support for the definition, management and execution of workflows:

- The *definition* of a workflow is realised through build-time functions that are concerned with the definition and the modelling of a workflow process and its constituent activities.

- The *management* of a workflow is realised through run-time control functions concerned with launching, dispatching and sequencing of basic activities as well as management functions for logging, monitoring and access control.

- Finally, for the actual *execution* of a workflow, run-time interactions with human users and other IT applications are required, which are controlled by the workflow management components.

The two latter items are frequently defined by the workflow enactment software [6] that interprets the process description and controls the instantiation of work processes and sequencing of basic activities. It adds basic workitems to the users worklists and invokes application tools as necessary. The support that the workflow enactment system can provide in principle ranges from (a) passive guidance (e.g. compilation of worklists) over (b) active guidance (notification services and push technology such as messaging, highlighting etc.) to (c) enforcement of flows and (d) the full automatic execution of (sub-)processes.

The development of a workflow management system requires a deep understanding of the application area. Since a workflow management system requires the application area to be specified by formally described tasks, the development of

the system provides an excellent opportunity to reengineer the tasks during modelling. Consequently there is a high interest in this area from business process reengineering [2]. In praxis this reengineering requires flexibility from a workflow system that most current workflow systems still do not provide [4]. For our application this was no problem, since we developed the system from scratch and implemented the flexibility that we needed.

The classical definitions of workflow management systems are mostly concerned with the definition of the work which has to be achieved (the objectives), and how these workflows can be handled by the system. However, when defining a real-world workflow system it is not sufficient to concentrate on the objectives, it is also necessary to model the users (subjects) of the system and the way the work is to be distributed and coordinated among the subjects. As it is not very practical to model the individual subjects, roles can be defined that share common characteristics of individuals. The definition of roles eases the management of the rights and the duties of individuals and allows, at the same time, the separation of duties [12].

The individuals of a workflow system will use the system as some kind of collaboration tool that allows to distribute and to coordinate work in a regulated manner. The aspect of collaboration raises some evidence that workflow management systems are actually a particular instance of computer supported cooperative work (CSCW[1] [7]). McCarthy and Bluestein [8] define a workflow management system "*as a proactive system which manages the flow of work among participants according to a defined procedure consisting of a number of tasks. It coordinates user and system participants, together with the appropriate data resources which may be accessible directly by the system or off-line, to achieve defined objectives by set deadlines*".

[1]The term CSCW was coined by Irene Greif and Paul Cashman in 1984 as a marketing idiom, for a vision of integrated office IT support – "...A shorthand way of referring to a set of concerns about supporting multiple individuals working together with computer systems." [1]

We will use in the rest of the paper the following terms and definitions: A *workitem* is a basic activity (or task) to be performed at a certain time by an agent (e.g. an individual user in a role or a program) on a work-object. Workitems are assigned to agents by a *assignment policy* which might be manual. Every agent has a *worklist*, which is a list of workitems assigned to this agent. These workitems are scheduled to be performed by this agent. The worklist of an agent is similar to an incoming mailbox, but contains time stamps that signal when the work is to be done. In general, the workitems in the worklist are transferable among agents.

The definition of a *workflow* is concerned about both the generation of workitems and the execution of workitems. The *generation* of workitems is performed by agents in certain roles, either in terms of basic workitems or in terms of generators that emit workitems during a specified period in specified ways. When an agent picks a certain workitem (which is due) and performs the task specified by the workitem we say that the workitem is *executed*. *Workflow management* is concerned with the controlled generation and dispatch of workitems and with the coordination and controlled distribution of workitems among the agents in the system to achieve flexible collaboration.

### 1.2 Workflow in the Area of Patient Care Delivery

The workflow management system that we implemented combines the aspects addressed above. On one hand we developed a new formal model for the specification of the work to be done (up to a certain degree combined with business process reengineering) that led us to the medical directive model discussed in Section 2. Since these directives (generators for workitems) are specified and entered into the system by multiple medical doctors, and since these directives differ from patient to patient and change regularly, it becomes clear that the transparency of the system state is very important.

Workitems should not appear on a worklist out of nowhere. For every workitem an agent

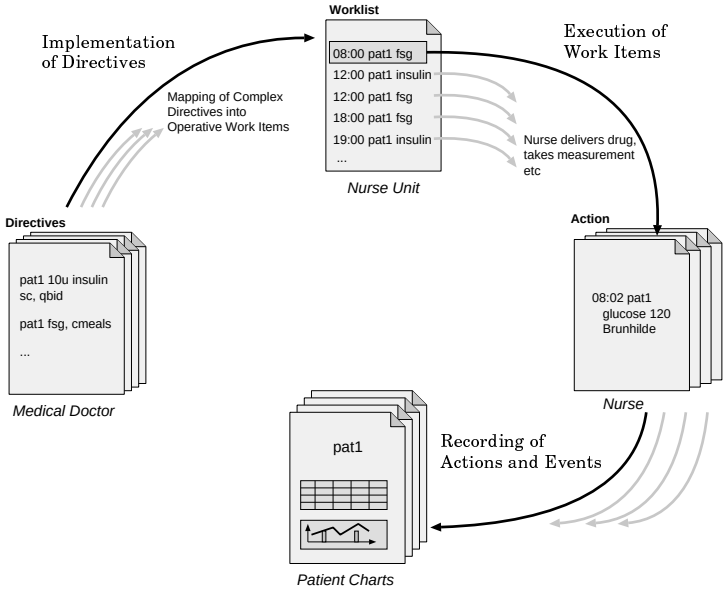


Figure 1: Basic Concepts and Data Flow

should be able to trace to the origins of the workitem in order to obtain some context and understanding about the intentions. It is critical for a care delivery workflow system to define a clear collaboration model with a shared workspace awareness (see e.g. [13]), which allows flexible rescheduling.

It is quite hard to categorise the type of workflow system that is needed here: some of the directives are of an ad-hoc nature, others are similar to an industrial production workflow with strictly pre-structured processes, and some interactions are quite knowledge intense. Furthermore, the scheduling and the execution of workitems requires flexibility: the rules for how to distribute the work between nurses differ sometimes between nurse stations in a single clinic, there might be different models for shift changes, and many workitems require specific skills from the nurses or technicians.

Consequently the emphasis of our system was to provide the highest possible flexibility for the specification and the distribution of workitems, and the control mechanisms of their execution. For the specification of workitems we developed a formalised language, based on the idioms of the doctors, which we describe in the following section.

## 2 Generation of Workitems

The generation of workitems is initiated by a doctor through the introduction or modification of a medical directive. Such directives are specific to a particular patient, have a well-defined lifetime, and address only issues relevant to patient care. The treatment a patient receives is represented by the set of all directives concerning the patient. The patient is cared for by a unit of nurses (which might represent a ward, floor or wing of the hospital, depending on the administrative structure). The system generates workitems from these directives, in the process incorporating further information from the nurse-unit. In particular, each nurse-unit may assign responsibility for executing a workitem to a specific nurse. The worklist for a nurse-unit consists of all such workitems for all patients in the care of the nurse-unit. Upon completion of a workitem activity, the nurse must record the results of the action as part of the execution of the workitem. Ultimately, all information related to workitems and directives is recorded in the patient chart, which is accessible to the doctor, and which

can be the basis for modification of directives (see Figure 1).

### 2.1 Directives

Directives indirectly specify workitems. Some combinations of workitems are not permitted (without further consideration) for medical or hospital policy reasons. Examples are drugs that interact, a drug that the patient is allergic to, and lab tests that have been performed recently. If a directive, in combination with existing directives for the patient, would introduce such impermissible combinations a warning is issued to the doctor when the directive is introduced.

Directives are divided into classes: *Medication* directives order a course of medical treatment, for example, treatment with insulin. *Measurement* directives order a series of measurements from the patient, for example, blood glucose levels. *Informational* directives state current information about the patient such as condition, diagnosis, allergies.

Orthogonal to the classes of directives is the type of directives, which is either *initial* (for a fresh directive), *delta* (a temporary override of an existing directive, possibly for a time range in the future), or *supersede* (a permanent and immediate override of a non-delta directive).

An *initial directive* $i$ is used to initiate a medical treatment, a course of measurements or to state the initial information on a patient. A *supersede directive* $\sigma$ is used to override an existing treatment, course of measurements or information statement and replace it with a different, but similar, treatment (for example, a drug treatment with a different dosage but otherwise unchanged). A *delta directive* $\delta$ is used to provide a temporary modification of a drug treatment or course of measurements (for example, to modify the treatment on the day that surgery is scheduled). By the nature of informational directives, a temporary modification is meaningless, and so deltas of these directives are not permitted (see Table 1).

For initial and delta directives it is possible to specify that this directive will become effective at some future time. A supersede directive always takes effect immediately. A supersede directive overrides any delta directives which are (or will be) in effect following the time the supersede directive is installed.

An initial directive, which is modified over time by some delta or supersede directives, is called a *compound directive* and might look like:

```
1. Initial on Day 0:    Day 0-14 insulin sc tid 5
2. Delta on Day 2:      Day 3-5             qid 7
3. Supersede on Day 5:  Day 5-19            bid 6
4. Delta on Day 6:      Day 7               tid 9
5. Supersede on Day 10: Day 10-24           qid 8
6. Delta on Day 10:     Day 14-15           am 12
```

In this example, each directive (1 to 6) contains a directive type (such as "initial") and time of its introduction (e.g. Day 0), a lifetime (e.g. Day 0 to 14), a drug and method of delivery (e.g. insulin, delivered subcutaneously), a frequency code or an explicit time (such as tid, which means three times a day), and size of the dosage (e.g. 5 units). The above compound directive exists in this form only on day 10 or later (see the time of introduction). Since the supersede directive (5) overrides all earlier directives, the compound directive is interpreted as following on day 10:

```
Day 10-24:     insulin sc qid (8 units) EXCEPT for
Day 14 and 15: insulin sc am (12 units)
```

A diagram for the above compound directive, relating the time a directive is in effect with the introduction of other directives, is given in Figure 2. The arrows indicate the original lifetime of each directive (including the day of the upper bound), and it is noted on the diagram whether the directive is an initial, a delta or a supersede directive. Each directive is in effect (for the purpose of generating workitems) only when it is not covered from below by another directive.

There are several restrictions, akin to integrity constraints, that must be satisfied in our model of directives and are enforced by the system.

1. Delta and supersede are modifying directives and must be made in reference to a base directive, which is the directive they modify.

| | | *Medication* | *Measurement* | *Informational* | *Future* |
|---|---|---|---|---|---|
| *supersede* | $\sigma$ | • | • | • | |
| *delta* | $\delta$ | • | • | | • |
| *initial* | $i$ | • | • | • | • |

Table 1: Types and Classes of Directives

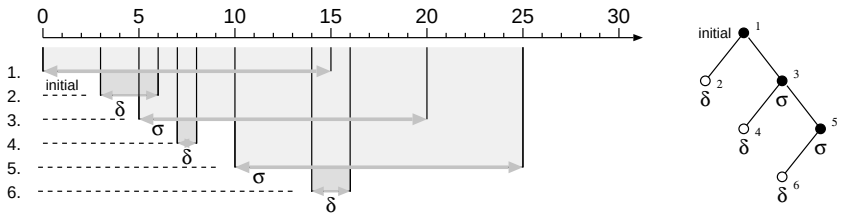


Figure 2: Time of Effect Summary for a Compound Directive

2. The base directive must be an initial or supersede directive, modifications of deltas are not allowed.

3. When a directive supersedes a base directive, it supersedes as well all its delta directives.

4. A directive may be superseded only once, since the superseding directive completely annihilates the superseded directives.

5. Every directive may be cancelled, which means that the cancelled directive and all other directives that modify it, are deleted from the database of active directives.

As a result of these restrictions on the use of the different types of directive, the directives in a compound directive form a linear tree-like structure (see Figure 2). All the nodes are directives. The root is an initial directive, and each initial or supersede directive has children corresponding to the several delta directives (left branches) and at most one supersede directive modifying it (right branches). The left branches are at most one level deep whereas the right ones can be arbitrarily deep. Modifications to the compound directive can only be made by modifying the most recent uncancelled initial or supersede directive, the rightmost node of the tree.

All directives, whether active, superseded or cancelled are retained by the system. They provide a basis for understanding the generation of workitems at earlier times, for analysing the effects of policies, and for refining decision support subsystems.

Figure 3 shows the view of a patient's directives which is provided to the doctor. The first pane gives an overview of the patient's medical treatment. Focussing on a single compound directive produces the history of that compound directive in the second pane and focussing on a component directive will show further details in the third pane. Icons at the top of the display provide easy access to creation, modification and cancellation of directives, and to patient information. The doctor is also able to view the workitems that a directive generates.

### 2.2 Workitems

Workitems denote activities to be performed, usually, by a nurse or a nurse technician at a specified time. Workitems are volatile in our system: The directives imply and therefore generate (emit) workitems, and patient charts record executed workitems. A display of workitems is generated, upon request, from this data for a specified time interval but there is no individual persistent record corresponding to a workitem. When we refer to the generation of workitems, we really

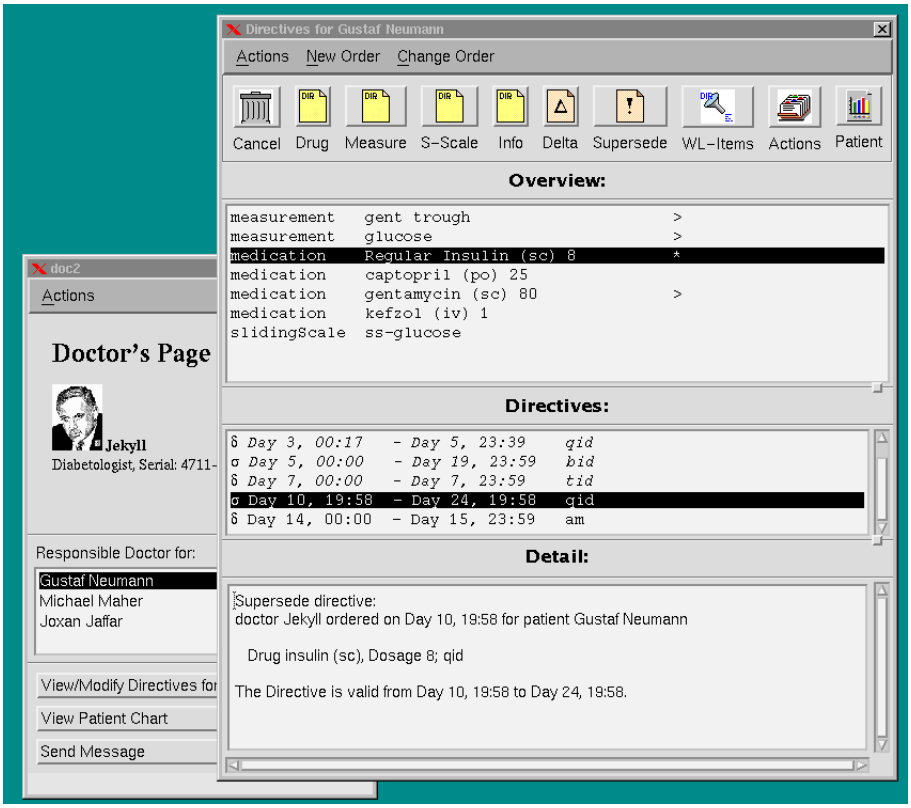


Figure 3: Directives Display for Medical Doctors

refer to the enumeration of actions as implied by the directives.

The system uses directives to generate workitems for a given time interval (e.g. for a shift in a nurse-unit, see Figure 4). A significant technical difficulty is the determination of which directives are in effect in the course of the time interval, and for what period. Supersedings, temporary overrides, varying lifetimes and cancellations can combine to make the meaning of compound directives arbitrarily complex. This issue applies equally to the doctors, who must understand fully the effect of compound directives. The restrictions put in place on the use of different types of directive directly address this problem. The resulting tree-like structure of compound directives is straightforward to understand since only the deepest interior node and its children are possibly in effect.

The system has standard conversions from frequency codes (e.g. *bid*) to specific times of the day (e.g. 12:00 and 19:00). These are used, with the information about periods for which different directives in a compound directive are in effect, to generate a set of times at which the activity specified by the compound directive should be undertaken on the patient. A workitem is generated for each of these times. Some data in a workitem, such as the patient and the basic activity, is common to all workitems generated from a single compound directive. Others, such as the dose of a drug, depend on the specific directive (in the compound directive) from which the time was generated. Generated workitems are then passed to the appropriate nurse-unit, which may add more information to each workitem.

A nurse-unit receives the workitems corresponding to all patients in its care. This worklist can be viewed by a nurse (see Figure 4). The first pane provides a summary of the worklist. Workitem summaries are shown in bold, normal or italic fonts, depending upon whether the workitem is

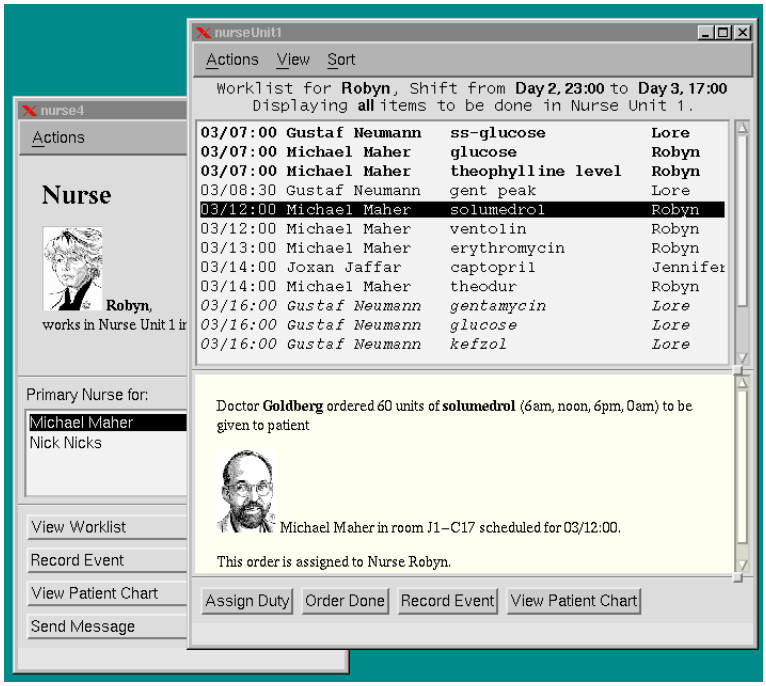


Figure 4: Worklist Display for a Nurse in Nurse-Unit 1

overdue, should be executed in the near future, or in the distant future. Focussing on a line displays the details of that workitem in the second pane. The display has buttons which allow the nurse to review patient chart information and to perform or record actions related to the execution of workitems.

## 3 Execution of Workitems

Each directive refers to a single patient. Workitems generated from a directive are associated with the nurse-unit that cares for the patient. They may be further assigned to individual nurses. These assigned workitems make up the nurse's worklist. The nurse-unit has a worklist of all workitems for all patients in the care of the nurse-unit.

Many factors determine the assignment of workitems to nurses. In the first place a nurse may only be assigned workitems in the nurse-unit to which he/she is assigned. A second factor is the possible assignment of primary responsibility for each patient to a nurse in each shift. The third factor is the default assignment policy of the nurse-unit. The system currently supports three predefined policies, which reflect existing assignment policies at the hospital.

**by nurse**
: Each workitem for a patient is assigned to the nurse who has primary responsibility for that patient.

**nurse with tech**
: Each workitem for a patient, except for certain simple measurements, is assigned to the nurse who has primary responsibility for that patient. The simple measurements are assigned to a nurse technician (or a group of nurse technicians).

**anarchy**
: There is no default assignment; all workitems are initially unassigned.

In addition to these default assignments, nurses may manually alter the assignment of workitems. Thus, in a nurse-unit under the "anarchy" default assignment policy the nurses cooperate to assign the workitems. In all nurse-units, manual assignment provides an important degree of flexibility in

the distribution of work. For example, it can be used to cope with mid-shift changes in staff that occur when someone becomes ill.

Workitem assignment, in addition to being a convenience for nurses, has a limited safety feature. When combined with a proper use protocol, workitem assignment provides a (weak) form of locking on workitems; it minimises the possibility of two nurses separately performing the same workitem. Suppose nurses are required to assign a workitem to themselves *before* they execute it, only reassign another nurse's workitems when that nurse is aware of the reassignment and only perform actions done for workitems assigned to them. Then the nurse assigned to the workitem is the presumed performer of the workitem, and another may take over this presumption only with the consent of the original nurse. The end result is an unambiguous determination of who shall perform each workitem, so that no workitem is performed twice.

Alternatively, if reassignment is more laissez-faire but nurses are required to check the workitem assignment at the patient's bedside immediately before performing the work, and to report the result of this action before leaving the patient then, again, duplicate performance of a workitem is avoided.

Speaking generally, a workitem is executed when the corresponding activity has been performed. From the view of the system, a workitem is executed when data associated with the execution has been entered. The data may range from simply recording that the activity was performed, to the results of measurements and extra comments by the nurse. This information is recorded in the appropriate patient chart. Once a workitem is executed, it disappears from the nurse's worklist.

Although the most common role of nurses is to execute workitems, there are some situations where a nurse can initiate a medical treatment workflow. In emergency situations, when a doctor is not available to issue a directive, a nurse may initiate medical treatment. In such cases, the nurse's actions must be counter-signed by a doctor at some later time. This appears as a workitem for the doctor to execute. In other circumstances, a nurse's action may be non-medical but nevertheless significant. A common case is the treatment of a hypoglycemic incident with a glass of orange juice. Reports of these actions are recorded in the patient chart.

The nurse's display (Figure 4) provides buttons to reassign a workitem, record that a workitem has been executed, and record an event of medical interest. Thus the single screen provides all the functionality required in the execution of workitems.

## 4 Architecture and Components of the Diabetes Manager

The implementation of the Diabetes Manager follows an open, programming language-independent architecture (see Figure 5). In this system multiple processes, that are called *external agents*, are connected to a server, the *Workflow and Communication Manager* DM, which controls and coordinates the messages originating from and targeted at external agents. DM is able to process asynchronous messages, computations are event driven.

The *external agents* are either independent subsystems for specialised tasks (such as reasoning or decision support) or for user agents that define the interaction of a user (similar to an avatar). In general, the external agents can be implemented in any programming language. The current prototype uses $Wa f^e$ [10] for the implementation of the user agents and time triggering tasks and CLP($\mathcal{R}$) [9] for reasoning and decision support.

DM and the external agents are implemented as separate processes which communicate via sockets and TCP/IP. In general these processes can execute on different hardware and OS platforms. It is important to note that while the implementation of the external agents does not matter, the application level communication protocol does. The protocol was specified in a way such that messages can be "inspected" by DM to grab the basic semantics (origin, destination, class of message, etc.) in order to direct it to the right OBJCHART-

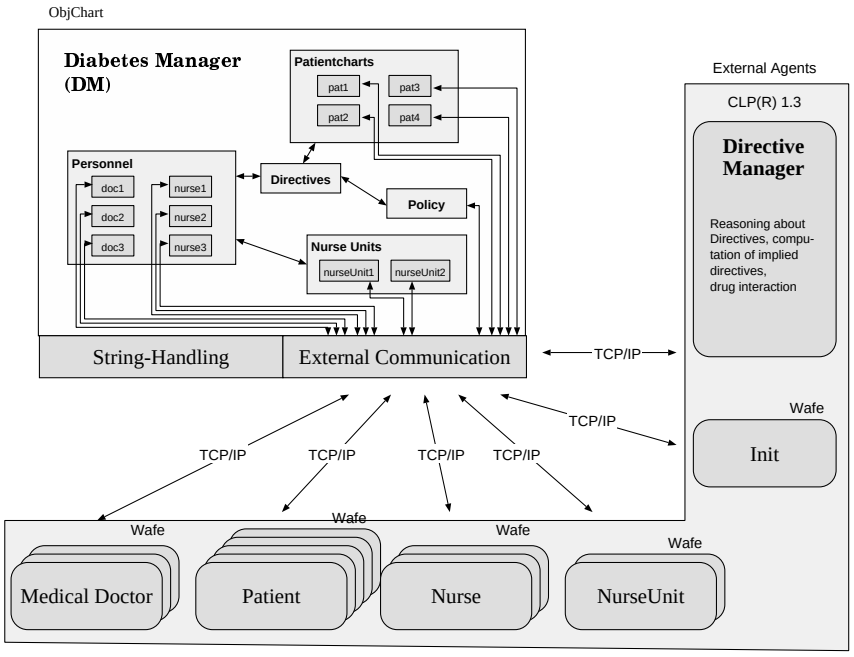


Figure 5: System Overview and Communication with external Agents

object (see Section 4.1).
This open architecture has the following advantages:

- Since each of the external agents and DM are separate programs they can either execute stand-alone or in concert. All communication between agents and DM is performed via standard input/output in the form of readable text messages.
- As a consequence all system components can be developed, debugged and extended separately; test data can be easily recorded and replayed.
- For the running system it is not necessary to restart DM when an external agent is modified, extended, etc.
- Several external agents do not require permanent interaction with DM (for example, patient charts). In general, it is possible to down-load several interesting data sets to a stand-alone machine which is not permanently connected to the network (e.g. to a notebook PC) and use the agents there in browsing mode.

### 4.1 Workflow and Communication Manager

The workflow and communication manager DM is implemented in ObjChart [3]. ObjChart is a visual prototyping environment supporting an object oriented design and development methodology, which is strongly influenced by David Harel's Statecharts [5]. The basic language supported by ObjChart is declarative and can be extended by users via C++ methods.

ObjChart allows to create objects and to define their behaviour through its graphical user interface. The system allows to send messages to these objects which might alter their states and send in turn messages to other objects in the system. The behaviour of an object in ObjChart is specified via a finite state machine which reacts to user-defined messages by storing information in its local attributes, calling methods or sending messages to other objects. This way ObjChart environment provides support for both, the high-level object design and the execution of the system that can be observed graphically (e.g. via message scenario diagrams).

The implementation of DM consists of several

ObjChart objects, which are hierarchically organised. As indicated in Figure 5, we implemented objects for the personnel (including an object for each individual doctor and nurse), an object for each nurse-unit, and a patient chart object for each patient. In addition to these objects corresponding to each user of the system, there is an object handling the assignment of people (such as nurses to nurse units, patients to nurses, etc.), an object embodying policy decisions which might lead to a directive being rejected, an object which handles all communication with external agents, and an object for initiating execution of the system.

The purpose of these ObjChart-objects is to know which other objects should be notified in which situation (state) and which actions must be recorded. An ObjChart-object might be for internal purposes or might correspond to an external agent. For every external agent exists one corresponding ObjChart-object. When an ObjChart-object receives a message it might send this message to an external agent as well. Similarly when an external agent sends a message to Dm, this message will be delivered to the corresponding ObjChart-object that decides what to do with this message. All control of message delivery is implemented in Dm. This allows logging, access control etc. at a single place.

In general, in a larger scale system, it is possible to connect multiple communication managers in a similar way as external agents. This allows for example decentralisation on a per-department level.

### 4.2 The Directive Manager

The directives manager performs reasoning tasks, chiefly about directives. It is the heart of the implementation of our model of directives and workitems (described in Section 2). It enforces the restrictions that produce the tree-like structure of compound directives and uses this structure to generate workitems, on request. When a directive is initially issued, the directives manager first silently generates the resulting workitems and tests them for impermissible combinations, such as drug interactions, before the directive is admitted into the system.

It also functions as an active database. Not only are all updates to patient data stored, but the consequences of the update are computed, and broadcast to the relevant screens that are open. As a result, the effect of a supersede, for example, that is issued by a doctor is immediately visible on nurses' worklists.

The directives manager also manages information on the assignment of nurses and doctors to patients, and the default assignment policy of nurse-units.

The directives manager is implemented in CLP($\mathcal{R}$) [9]. CLP($\mathcal{R}$) is a rule-based, declarative language with inbuilt facilities for reasoning about arithmetic constraints. One of its strengths is the ability to rapidly model and to efficiently reason about complex systems.

### 4.3 User Interface Agents

All user interfaces are built in $W_a f^e$ [10]. $W_a f^e$ is a graphical user interface scripting environment based on the X Toolkit from MIT, various X Toolkit compliant widget sets (such as OSF/Motif 1.2 or 2.0 from the Open Software Foundation) and the embeddable programming language Tcl. $W_a f^e$ was a convenient choice since it provides various built-in means to act as a frontend for programs in arbitrary programming languages, supports rapid prototyping, and interfaces to commercial widgets sets.

For all basic roles of the Diabetes Manager we developed separate $W_a f^e$ scripts. The scripts can be tailored easily for individual users to provide, for example, short-cuts. Altering such a script does not affect other parts of the system. The sizes of the scripts range from 40 lines (for the nurse script, left window in Figure 4) to about 800 lines (various support windows for the medical doctor, see Figure 3).

## 5 Summary and Conclusions

In this paper we presented our experiences from the Diabetes Manager project which was a joint project between IBM Research and a Massachusetts hospital. The main emphasis of the system was

to provide a workflow management system for patient centered care delivery and to provide a scalable architecture for the integration of larger parts of the hospital.

The whole system was designed and implemented in a very short time by using three high-level components: (a) ObjChart as a coordination and communication manager, (b) CLP($\mathcal{R}$) as a logic-based deductive system for reasoning purposes and (c) *Wafe* for graphical user interface and frontend capabilities. We achieved this by using the declarativeness of the language for both the semantics (CLP($\mathcal{R}$)) and workflow (ObjChart).

## Acknowledgements

We want to thank the medical staff in the Massachusetts hospital who supported our work substantially. In particular we want to express our thanks to MD Howard Goldberg who was heavily involved in our initial design and who served as our medical expert in the development.

## References

[1] L. Bannon, K. Schmidt: "*CSCW: Four Characters in Search of a Context*", Proceedings of the First European Conference on Computer-Supported Cooperative Work (EC-CSCW '89), London, 1989.

[2] T.H. Davenport: "*Process Innovation: Reengineering Work through Information Technology*", Harvard Business School Press, Boston, MA, 1993.

[3] D. Gangopadhyay, S. Mitra: "*ObjChart: Tangible Specification of Reactive Object Behavior*", in : Proceedings of the European Conference on Object Oriented Programming, 1993, Lecture Notes in Computer Science 707, pages 432–457.

[4] T. Goesmann, K. Just-Hahn, T. Löffeler, R. Rolles: "*Flexibilität als Ziel beim Einsatz von Workflow- Management-Systemen*", in: E. Ortner (ed): Proceedings des EMISA-Fachgruppentreffens 1997: Workflow-Management-Systeme im Spannungsfeld einer Organisation. Darmstadt, 1997.

[5] D. Harel: "*Statecharts: A Visual Formalism for Complex Systems*", Science of Computer Programming 8, 1987, pages 231–274.

[6] D. Hollingsworth: "*Workflow Management Coalition: The Workflow Reference Model*", Document Number TC00-1003, Issue 1.1, 29. Nov 1994. `http://www.aiim.org/wfmc/DOCS/refmodel/rmv1-16.html`

[7] J. Grudin: "*CSCW Introduction*", Communications of the ACM, Vol. 34, No. 12, 1991, pages 31–34.

[8] J.C. McCarthy, W.M. Bluestein: "*The Computing Strategy Report: Workflow's Progress*", Forrester Research Inc., Cambridge, MA, 1991.

[9] J. Jaffar, S. Michaylov, P.J. Stuckey, R.H.C. Yap: "*The CLP(R) Language and System*", ACM Transactions on Programming Languages and Systems 14(3), 1992, pages 339–395.

[10] G. Neumann, S. Nusser: "*Wafe – An X Toolkit Based Frontend for Application Programs in Various Programming Languages*", USENIX Winter 1993 Technical Conference, San Diego, California 1993, also as: `http://nestroy.wi-inf.uni-essen.de/wafe/wafe-usenix93/`

[11] J. Ousterhout: "*Tcl: An embeddable Command Language*", Proceedings of the 1990 Winter USENIX Conference, 1990.

[12] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, C.E. Youman: "*Role-Based Access Control Models*", IEEE Computer, Volume 29, Number 2, February 1996, pages 38–47.

[13] J. Schlichter, M. Koch, M. Bürger: "*Workspace Awareness for Distributed Teams*", in: W. Conen, G. Neumann (eds.), "*Coordination Technology for Collaborative Applications*", Springer Verlag, Heidelberg 1998.