

Ergon Workflow Tool White Paper



Version 1.1, August 14, 2002
Andreas Fleischmann

Content

1.	Introduction	3
1.1	Scope of this document	3
2.	Business considerations	3
2.1	The workflow philosophy	3
	The glue between “who”, “how” and “when”	3
2.2	Make or buy?	4
2.3	Usages and future development	4
3.	Technical features overview	5
3.1	Architecture.	5
	Basic architecture	5
	Modelling environment	5
	Runtime environment	5
3.2	Modelling features.	5
	Modelling entities	5
	Task properties	5
	Flow logic.	6
	Control structures	6
	Data flow	6
	Planning.	7
	Workflow Editor	8

1. Introduction

1.1 Scope of this document

This document gives a short description of the Ergon Workflow Tool. It describes the basic workflow philosophy that went into the design of the Workflow Tool. It lists the reasons why Ergon choose to develop a Tool by itself instead of buying one. Finally, it explains the architecture of the Workflow Tool and the modelling features that can be used to model business processes.

This document is not a developers guide. If you need more details of how the Workflow Tool can be used, please refer to the developers guide [WF-DG].

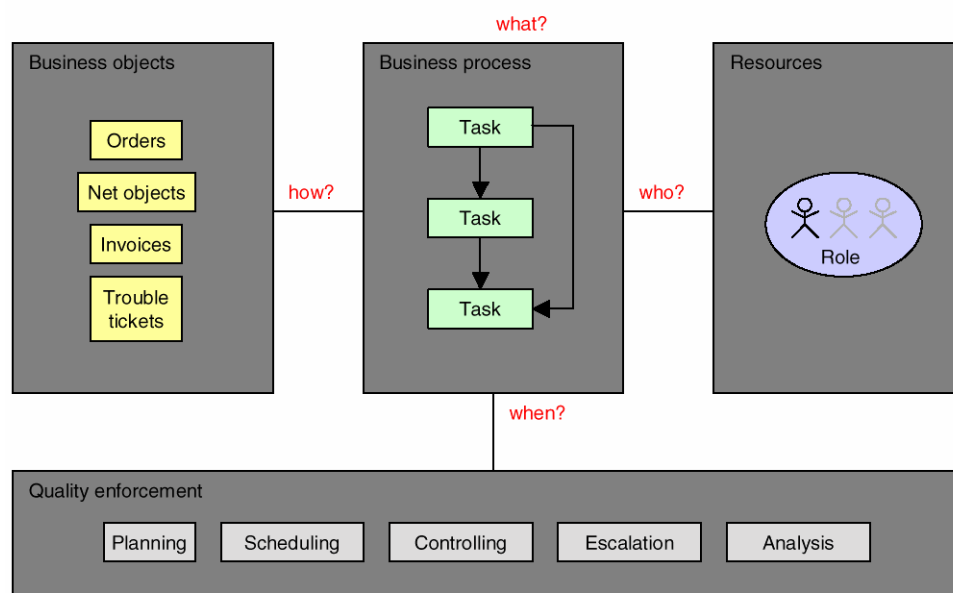
2. Business considerations

2.1 The workflow philosophy

The glue between “who”, “how” and “when”

A workflow management tool controls the execution of any predefined business processes. A workflow managed business process embodies the central business rules and provides the glue that ties business objects (like orders or invoices) and organisational resources (like employees and their predefined roles) together.

A workflow process consists of several tasks that have to be executed in a controlled and managed way. It defines “what” (the tasks”) has to be executed “how” (the business objects or application modules) by “whom” (the resources like roles and employees) and “when” (defined by the quality enforced dates, schedules, and plans).



Using a workflow management tool for defining and executing business processes has several advantages:

- Business process models serve as an executable documentation of business rules.
- Since tasks are linked with business objects, a user doesn't have to know which application to use in order to work on a task. He just opens the task he receives in his tasklist.
- In-time execution of tasks can be enforced by checking overdue tasks and initiate any warning or escalation procedure that is appropriate.
- Since tasks are addressed to roles instead of individual users, the users might change without affecting running processes.
- Mass production of services is enforced in a predefined way that gives maximum benefits in terms of the learning curve.

2.2 Make or buy?

At the time when the need for a workflow infrastructure arose in Ergon's projects, the central question of whether to use an "of the shelf" workflow product or to develop such a tool by our own had to be answered. Ergon decided for a custom development because of the following reasons:

- **Experience:** At the time of decision, Ergon employed engineers with several years of experience in the workflow business, and therefore knew the tool market and the features as well as the limitations of these tools.
- **Flexibility:** We wanted the maximum of flexibility in modelling and runtime integration.
- **Planning:** The feature of processes with planned tasks and editable gantt diagrams was not found in any tool at that time.
- **JTT integration:** Most of our customer client/server applications are based on *JavaTableTool* [JTT-SD], a generic database application with a interactive frontend based on Java. We wanted a seamless integration of our workflow tool with this environment.
- **Role model:** Ergon has designed and developed a hierarchical role model that can be used for database permission modelling, role oriented GUI parametrization, and also workflow task addressing. We wanted to share the same role model in all of these components.
- **Eval:** Eval is a interpreted programming language developed by Ergon, and is used in our projects for scripting, but also for developing business logic. The usage of Eval for all server side workflow programming purposes provides perfect business logic leverage in these cases.

2.3 Usages and future development

The Workflow Tool has been successfully used in various productive applications that Ergon developed for their customers. It is going to be continuously improved and extended based on customer requirements. It more and more becomes a central component of the Ergon toolkit and component set.

3. Technical features overview

3.1 Architecture

When talking about the architecture of the Workflow Tool, one has to distinguish between the modelling and the runtime environment. The modelling environment is used to design and model the workflow processes, whereas the runtime environment is used to execute these processes.

Basic architecture

Basically, the Workflow Tool consists of a relational database and various programs and components that provide the basic services of the Workflow Tool. It is currently implemented as a pluggable module that can be attached to applications.

Modelling environment

During the modelling phase, the workflow modeller first designs the process. He then creates the appropriate database objects like process, task and transition records in the database. He can choose whether he wants to use any database development tool to create the database records, or he can use the graphical workflow designer component that brings process modelling to a more abstract level. Additionally, the designer or application developer might write the necessary program code for task and action handlers by using the development environment of his choice.

Runtime environment

The runtime environment consists of a server and a client part. The basic server functionality of the Workflow Tool is encapsulated as a module that can be imported by the application server. The Workflow Tool's client components, such as the tasklist, use the client/server communication infrastructure of the host application to gain access to the server functionality. The tasklist itself gets loaded as a part of the host application's client. When the user wants to open a task, the tasklist brings up the application's client window that was registered for this particular task.

There is an additional component at the server side, called the "Task Daemon". It is a standalone process that runs in the background and periodically checks for timed events that must be notified to running processes, such as tasks that have a defined timeout period.

3.2 Modelling features

Modelling entities

A *process* is the toplevel structure in the Workflow Tool. It consists of *tasks* that represent single pieces of work to be done manually by persons or automatically by background programs. Tasks can be grouped into *blocks* to represent logical structures within the process. Predefined Processes can be referenced by other Processes as *sub-processes*.

Task properties

Tasks are the most important modelling entities. They represent the "atoms" of work and appear in the tasklists of the users, much like emails in an email client.

Task addressing uses logical *roles* to which individual users can be assigned. The advantage of this indirection is that changes of individuals don't affect running processes, because these changes are encapsulated by the roles. Roles can be hierarchically structured, which allows to build complex addressing schemes.

Tasks can be manually or automatically executable. Manual tasks appear in the tasklists of the assigned users, and are executed by simply clicking on them. The tasklist will then open the application window that was registered for this particular task, and that was specially programmed to interoperate with this task (such as querying runtime parameters from the task, and writing information back to the task upon closing the application window).

Task actions can be defined for each task and are presented to the user of a task. An example of such task actions is a task called "Approve credit request" that is sent to some person who has to decide whether a certain credit request is to be confirmed. If the modeller wants to give him the two options "Confirm" and "Request", the modeller would define these two options as task actions. When the user performs one of these actions, the corresponding event will be sent to the task, and logic will go on as defined.

The modeller can define exit conditions for each task to ensure that tasks are only finished if certain conditions are respected, and therefore ensuring overall quality.

Flow logic

Transitions between tasks basically define the order in which the tasks are executed. They can contain logical conditions. Whenever a task finishes, all transitions that evaluate to true are executed, which in turn activates all tasks to which these transitions lead.

Control structures

It becomes obvious from the description of the flow logic that conditional transitions can be used to build conditional branches. If more than one subsequent transition is executed, then more than one task will be activated at the same time, and parallelism comes into play. Multiple parallel threads of execution must be resynchronized at the end of a block.

Loops can be modelled by using blocks with exit conditions. As long as the exit conditions evaluate to false, the block can not be exited and will restart.

Data flow

Running processes need runtime information for different reasons. Modellers want to formulate conditional expressions like transition conditions or exit conditions in the form of "credit amount greater than one million". Furthermore, this kind of information is also needed by task worker application, which must automatically display for example the current order or information about the customer to which a process belongs.

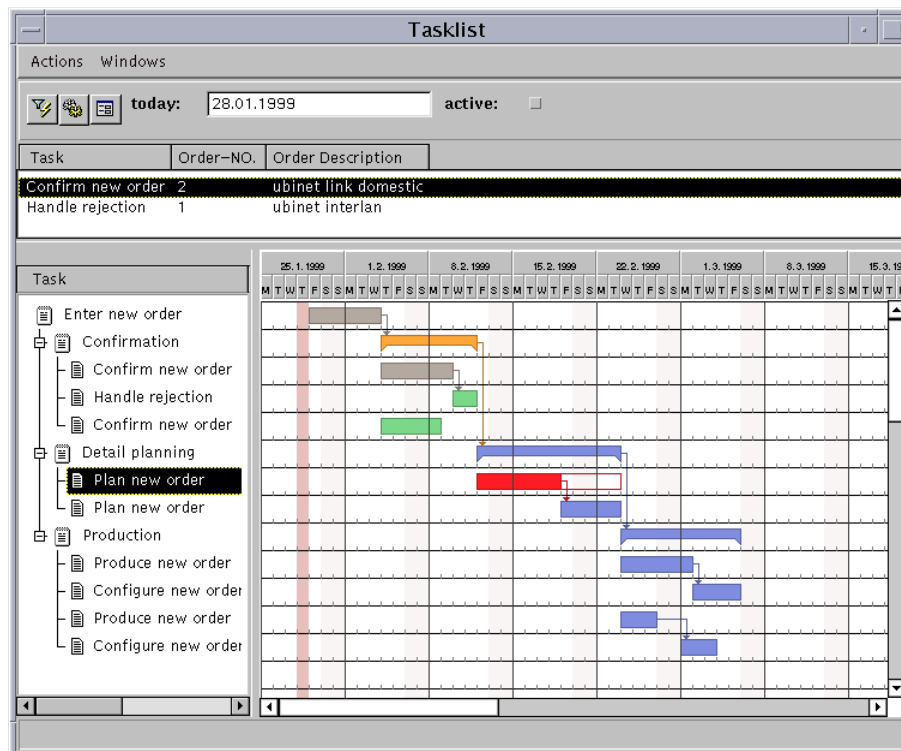
All this runtime information is maintained by a running process using a data container. The data container contains named variables along with their values and can be accessed by conditional expressions, callback functions and task worker applications.

Planning

One feature of Ergon's Workflow Tool is really outstanding compared to most of the other tools. While all tools provide a special process view that shows the *history* of a process (who did what when?), our Workflow Tool shows the planned *future* as well. For this to work, the following two definitions are necessary:

- The standard or average execution duration for every task must be defined
- The default outgoing transition for every task must be defined

Using this information, the Workflow Tool can create execution plans for the future of the process, and represent them as a gantt diagram known from typical project management tools. This is a powerful feature, because team managers can predict the future workload on their teams, and can even change the planned execution of tasks by interactively edit the gantt diagram.



The picture above shows a typical tasklist in which a user receives the tasks he has to work on (the upper part of the window). The lower part of the window shows a graphical outline of a single process, where all tasks are arranged in a "Microsoft Project" like style that allows the user to expand or collapse block tasks, change the duration of individual tasks, and visualise the timing and scheduling information throughout the entire process.

The different color of the task represent the different states of the tasks. The green ones are those that are currently ready for execution, while the blue ones are planned.

Workflow Editor

The Workflow Tool contains a graphical workflow editor that can be used to design and model workflow processes. The user can draw the workflow process like in any other flow chart diagrammer. Additionally, he can click on a task or a transition and gets a view in which he can edit the properties of this particular workflow component.

