ABC

Automatic Business Computing

Made in Eiffel

December 2000

Abstraction, Fbg de l'hopital 18, 2000 Neuchâtel, Suisse.
Tél: +41 725 04 93 E-mail: abstraction@access.ch

Content

Introduction

ABC is made to minimize the administration of small companies. Of course administration has to exist if the company wants to stay alive. Meanwhile, administration is time and energy consuming. Especially for people who wish to have an accurate situation about pending business, cash availability, etc., but whose hobby is not paperwork.

The company has to write invoices, orders, price lists and pay suppliers. These are the basic documents. There are many others. Their number and layout depend on the specific activity, context and size of the company. They have to be created and checked to avoid mistakes which could be expensive. Moreover, these actions have to be translated into accounting records. Debtors and creditors have to be constantly tracked, and don't forget the sales tax administration.

On one hand, many document are partially or totally linked to each other. On the other hand, most of the accounting records are direct consequences of business documents. For example, a supplier invoice is, if everything goes well, the supplier version of an order. Of course, an exceptional discount could be granted in between. Of course, the invoice should be paid but not too early. Therefore, this invoice becomes automatically a debt. That debt shouldn't be forgotten at expiration time. What a big job.

The first goal of ABC is **to reduce all these activities to the decision steps**. The first decision, in the previous example, is to decide to order something from the supplier and to fix the quantities for each item. The second step is at receipt of the supplier invoice. The user has to approve it according to what he really has received. With ABC, that process is realised by creating automatically a clone of the supplier invoice from the order to which the invoice refers and which should by checked to perform the approval decision. The user could perhaps fix the delay for payment. That's all. The debt will be automatically recorded in a credit account (and the company's balance updated). The analytic dispatching will also be made. Later, the payment order will be automatically produced at the correct expiration date and then printed or electronically sent as appropriate. At payment time, the transaction from supplier credit account will automatically be made to the debit account specified. In summary, it leaves only two actions to the user: ordering and approving. Is it possible to be any easier?

The **second goal is to minimise errors**. To achieve this, ABC minimises the use of the keyboard. Most manipulations are made by drag & drop. E.g. to create a customer invoice, the user just opens the appropriate editor and then drops the customer address in the recipient hole or field. He does the same with one or more products. Of course, he has to specify the appropriate quantities. The invoice is ready. Shorter than ever. Suppose two customers should receive the same or similar invoice. The user creates the first as described previously. The second one will be produced by dropping the new address in the address hole and then the first invoice in the model hole of the editor. That's it. The system will fill in the document. Of course, if both invoices are similar but not exactly the same, the user has to modify the differences, like adjusting the quantities sold.

The **third goal is flexibility**. First of all, the system can be adapted to all languages (except for Japanese and other special characters). It accepts several text processors to publish printed or sent versions of the documents. Also, individual and company preferences are available. Further, all the document models and their behaviours are defined according to the company procedures. ABC looks like a **chameleon**. Following the same idea, terminal accounts, products and addresses can be simultaneously classified in many ways. Such classification can be permanent or just of action period. These partitions can also be made according to several points of view. The account manager has not the overlook on addresses as the salesman. The first will classify them according to solvability. The second will see different clients groups according to potentiality. Another classification could be a price discount structure. Sales tax balance is nothing more than a reduced view over specific accounts.

Next, ABC have the notion of project and personal or partner folder. All documents relative to a project can be grouped in a single folder. This allows to manage and evaluate the project performances in an easy way. The partner or personal folder allows to evaluate an drive the relation the enterprise have with either a customer or a supplier. This folder is the basic tool for liberal activities such as lawyer, physician, etc. The notion of tutor allows to separate the subject of action from the account. This allows the physician to act on a patient and to address its invoices to an insurance.

Features

ABC may

- Generate any kind of document useful for business or manufacturing.
- Produce automatically all accounting records on the basis of events happening to document specimens.
- Trigger event according to time.
- Manage project documents.
- Manage personal document sets .
- Communicate with the world by mail, fax, and post.
- Exchange data with other applications
- Launch external applications.
- Trace events defined as important

Install

The software is delivered in an archive file (zip or tar.gz) from the net or on a CDRom. To install it from an archive, extract files into a temporary directory.

Then read the README. This text gives precise instructions function of the platform and the version.

If this is an update, make first a full backup of previous data and distribution.

For a new installation, the system automatically offers a free trial period. Examples are supplied with the distribution. They can be copied in the data space. This allows a first contact with the system to better understand its behaviour. Indeed, ABC allows very deep adaptation, but this is only possible once you have thoroughly understood the way it works. Specifically, definition of documents models and behaviour is not straightforward. Therefore, it is then a good idea to practice with the examples and to modify them in order to study their mechanism before starting with the definition of the company real models. The installation kit offers several predefined models. It is possible to install several one sentimentally or simultaneously. By installing very different model such than ‘sale’ and ‘association’ it is possible to understand how much the product is adaptable to your situation.

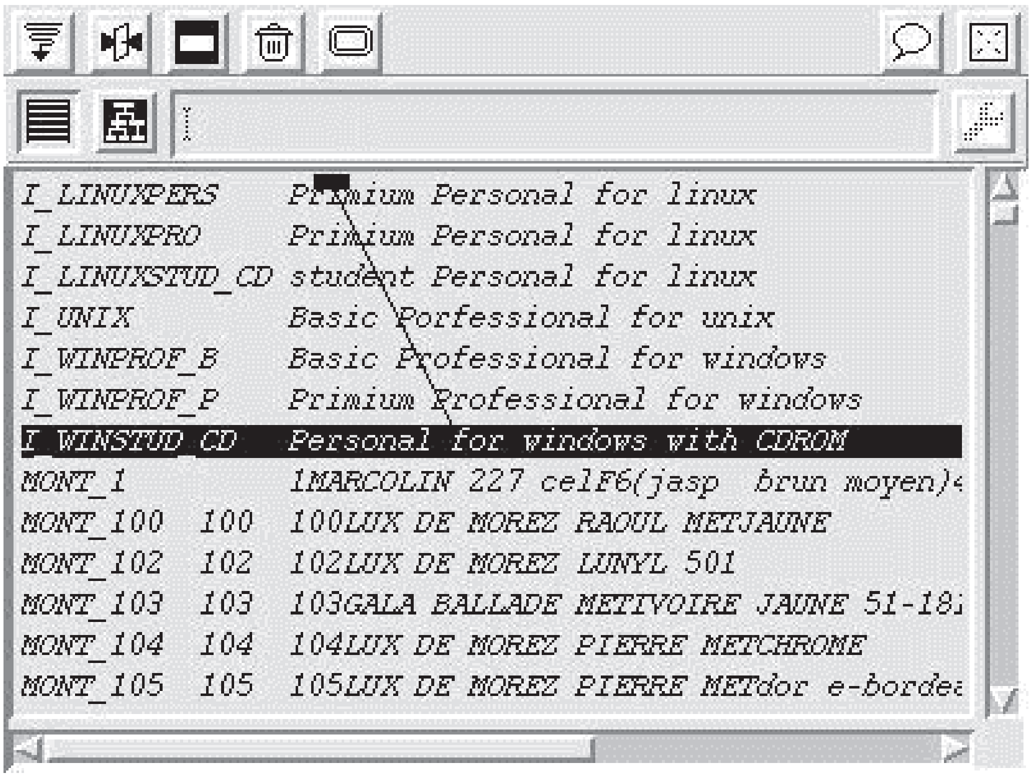
It is appropriate to create a specific installation directory. We suggest ‘C:\abc’ for Windows and ‘/usr/local/abc’ for Linux or other UNIX. Over, if you wish to install more than one model, then create for each an extra sub-directory. The you can uninstall useless version supply by removing the corresponding directory. ABC does not copy any thins outside of two directories one for the data and one for the tools. These two directory are normally under C:\abc for Windows and in your working directory for Linux.

Now the system is ready for trainning. When you are ready to swtich to a real use, then refere to the section called ‘‘real startup’’ bellow.

Usage

Drag & drop

The graphical interface is widely based on the drag and drop mechanism. This metaphor suggests the idea that an object is dragged from one source and that this object is then given to a receiver somewhere else, either in the same window or in another one. More generally, the source does not really give the object but just an access, a reference to it. Once the dropped object has been accepted by the receiver , it can be worked on. At the same time, the source will always know about that object. For instance, if you drag an object from a catalogue into an editor, the catalogue will still include the dragged item.



```
I_LINUXPERS    Primium Personal for linux
I_LINUXPRO     Primium Personal for linux
I_LINUXSTUD_CD student Personal for linux
I_UNIX         Basic Porfessional for unix
I_WINPROF_B    Basic Professional for windows
I_WINPROF_P    Primium Professional for windows
I_WINSTUD_CD   Personal for windows with CDROM
MONT_1         1MARCOLIN 227 celF6(jasp  brun moyen)
MONT_100  100  100LUX DE MOREZ RAOUL METJAUNE
MONT_102  102  102LUX DE MOREZ LUNYL 501
MONT_103  103  103GALA BALLADE METIVOIRE JAUNE 51-18
MONT_104  104  104LUX DE MOREZ PIERRE METCHROME
MONT_105  105  105LUX DE MOREZ PIERRE METdor e-borde
```

Graphically, this mechanism is represented by a cursor which form symbolises the attached object. A line links the source to the cursor, showing the sharing mechanism between source and receiver. Typically, the receiver is a button which form suggests that it can take care of the dragged item. But, sometimes, as a short-cut, a field or a scrolling list can also accept a dragged object.

You start the mechanism by pointing the cursor on a source and clicking the right mouse button. Then you point the cursor on a receiver and click the right button to drop the item. If you drop on an appropriate receiver, the process will terminate successfully. Otherwise the item falls through and nothing more happens. To cancel the drag operation, just press an other button.

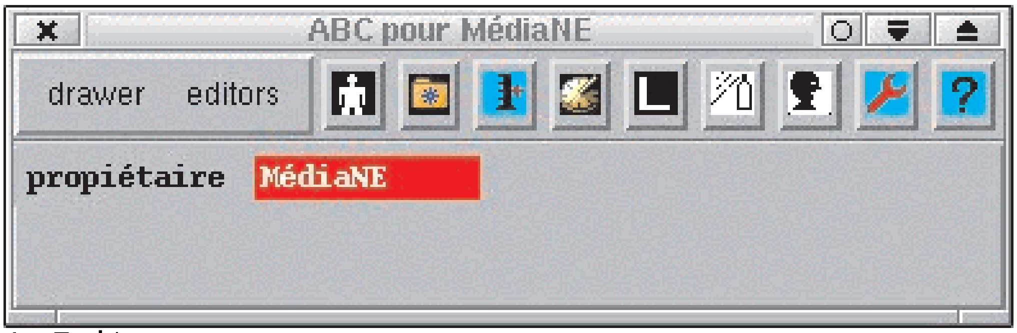
This mechanism is not very common yet but allows very powerful compacted and expendable interfaces. Indeed, it is not necessary to build mechanisms to allow each source to communicate to each receiver. This way combinatory explosion in the interface is avoided.

Some holes are access point to a tool and loader for that tool at the same time. For example the button **../image/product.jpg: Fichier ou répertoire inexistant** in the above window is an access and loader button. It accepts a product symbolized by a simple flat rectangle like on the cursor in this windows picture. On reception of such a object, it loads the product editor and then popup the window editor as necessary. But, if the same button is pressed with the select mouse button, the product editor is, if not currently exposed, popped up. We call such a button an access-hole.

Start-up

At this point, the system should be loaded with all data and models needed. This is the case if you used the supplied examples or if you have already loaded the system with your company imported data and models. These loading procedures are explained in the maintenance chapter.

At start-up ABC always presents the following control panel.



1. Exit
2. Help
3. Document menu
4. Diary
5. Stock motion Diary
6. Product caltalogue
7. Addresses catalogue
8. Save graphic configuration
9. presonal preferences
10. Interface translator
11. maintenance access

All windows have the exit button in the upper right corner. This rule also applies to the main control panel, where a confirmation is required before quitting completely. In all the other cases, the window disappears immediately without asking for confirmation even if an editing session is in progress. In that case, any modification done is lost. As the amount of work which can be lost is always small, this system is better than the fastidious process of having to confirm everything.

On the left of the exit button, there always is a contextual help button.

The "document'' menu gives access to the lists and editors for each document model. The number of models is not limited and is expendable at any time. The next three buttons go to the main catalogues : products, addresses and log.

If the same job is done repeatedly or if a job has to be interrupted, the user can save a certain working configuration. This is done by simply pressing the "configure'' button at the bottom of the control panel. The last two buttons are reserved for maintenance operations. They will be thoroughly described later. Just to mention their purpose: the "translate" one allows you to adapt all the interface labels to the user needs and language. The second one gives access to the system maintenance facilities. However this button is reserved for privileged users. People who don't have the appropriate privilege will not see that button.

Catalogues

../image/catalogue-en.jpg: Fichier ou répertoire inexistant

A catalogue has two tool bars. The top one allows you to search into the text window, print, filter the content or call the editor. The next two buttons allow you to suppress an element or to rename another.

The second tool bar allows to select the presentation mode. The first one is a sequential mode. The second one is a hierarchical mode. In that mode, one hierarchy must be chosen among those defined. As many partitions as wished can be defined for each catalogue. In particular, it is possible to classify only a part of the whole catalogue. The remaining elements are automatically put in a set called ‘‘unclassified’’. This allows, for example, to build a partial partition in the terminal accounts catalogue including only the accounts concerning the sales tax in order to process the cyclic taxes balance.

In a catalogue, each item is represented by its identity and a short description depending on the type of these elements. Trough a personal preference, it is possible to suppress the display of identities.

To choose a hierarchy, type its name in the text field on the tool bar. Or type directly a "enter" in that field: a choice list will pop up with the different hierarchies.

The last button allows you to open the hierarchy editor to create and modify the herarchies belonging to a specific catalogue.

Any element in a catalogue can be dragged from the scrolling list in order to be dropped somewhere else, most of the time in a document.

When a hierarchy is shown, only the top level is displayed. The sign @ next to a partition name means it’s a closed (rolled-up) partition, the sign ‘‘-’’ shows an open one. Selecting a partition will open or close it. If the ‘‘ctrl’’ key is pressed at the same time the partition is open recursively. Each mechanism works also when selecting multiple lines.

Editor

../image/editeur-en.ag.jpg: Fichier ou répertoire inexistant

All editors are equipped with a standard tool bar. The first button allows you to load the item to edit. The button shape shows which kind of object it will accept. The second button is the print/filter , followed by the eraser. The following field shows the item name. It also allows you to load an existing item by typing its name and then "enter". Next to that field is the button to save the editing session. The following button allows you to edit a comment associated with the targeted item.

The following hole button looks exactly like the first one, but has a different function. It allows you to load values from a compatible item. However the loaded model does not become the target of the editor. Thus it is possible to define partially a new item and then complete the definition by loading a model. All the not yet defined fields will be filled with the one from the model.

Some editors may have read-only fields, like the products editor where the sale price field is locked. This field is calculated from the sale-function.

Product

Products can be pure charge or pure service or standard product. A standard product is defined by an input price in a given currency and either by a sale function (depreciated) or a output price. A pure charge have only an input price and can not be referred into a output document. A pure service have only an output price and may no be included in a input document. When un produict is deleted from the catalogue, all it’s aliases are also removed. (This contrast with most of the file system where short cuts and symbolic links stay in place when theire target was removed and then point to nothing!). Also, the barcode reader, when it fail down to a unknown code, start a procedure to introduce it as a new product or an alias if appropriated.

The product catalogue has a special tool allowing to add and remove aliases.

Like any item in a catalogue, a product has an unique identifier (identity). This can be a barcode. Over the main identity a product can be referred by aliases. This allows to have multiple vendors labelling a single product with their own code.

Addresses

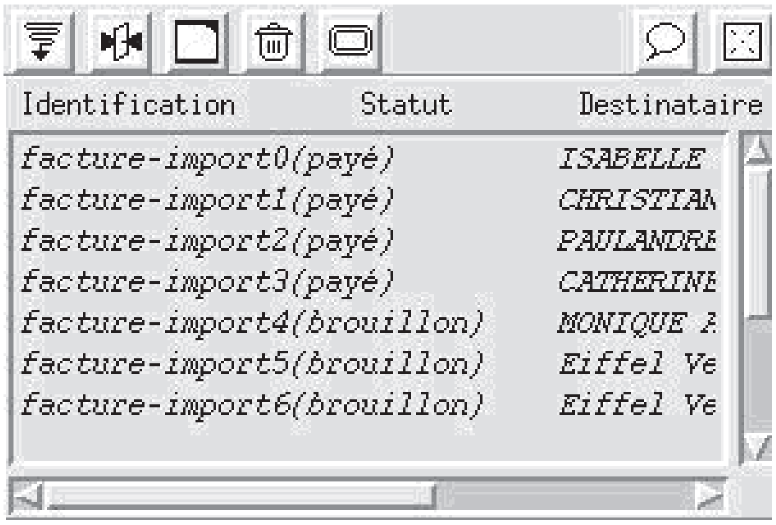
An address define as well a supplier as a customer or a worker. A address may represent a sale man of a big company. It is an important contact, but from a financial point of view it does not exist. Only its company is relevant for that. It is the same for the patient in an hospital. He his important for the hospital activity, but for the finances, only the insurance is relevant. The tutor is defined by draging in an address from the address catalogue. The tutor has in its member list all the address for which he is responsible. Disabling the tutor flag in a tutor record will temporary disable that mechanism but the list still appears.

Documents

A list is dedicated to each model. To each list, an editor is associated. A document list is equipped with the first catalogue tool bar.

The upper part of the editor concern the actions applicable to the document. That section is only active when the target document is correctly completed. As the edition session was saved, the action menu can be pulled down. The first action is necessary to close the document

.



```
Identification              Statut       Destinataire
facture-import0(payé)                    ISABELLE
facture-import1(payé)                    CHRISTIAN
facture-import2(payé)                    PAULANDRE
facture-import3(payé)                    CATHERINE
facture-import4(brouillon)               MONIQUE A
facture-import5(brouillon)               Eiffel Ve
facture-import6(brouillon)               Eiffel Ve
```

That means that it is no longer editable. The date appears in the read-only dedicated field. The next action depends on the model designer. This design should reflect the procedural complexity of the company.

../image/editeur-document-en.ag.jpg: Fichier ou répertoire inexistant

In a simple case, the close action can be the first one of a sequence which links all the others up to a unique final state. For example, imagine a simple letter. When the author closes such a letter is automatically signed and sent by post and only by that method. Alternatively, the designer can specify that the send operation is not automatic. The user has to decide for each such document which transmission channel is used.

The next editor section is reserved to the text fields specific to the document model. Some fields can be mandatory. For example, the delay in an invoice. This field can, under some conditions (see the system preferences), be used to drive the automatic resent operations.

The more typical business documents have a product list. For them, the editor has a supplementary section. This section has first a line editor area followed by the list of items. In the line editor area, a product hole allows you to drop a product. That drop will automatically file in all the fields except the quantity one. The user must file in this field and then, if he agree with the standard price, he just needs to save the line which wiil then be transferred to the list. Of course, the user can type in all the field. In this case, the system will accept the user input if it recognizes the product. The user can also reload any line from the list to correct it. He just has to double-click on.

At the bottom of that section, it is the total the total duties and then the full total. The currency is also displayed and can be changed at any time. Depending of the model designer, the duties are automatically included or the document is in duty free mode or else the user can choose duty mode. We have to mention here that a model for suppliers would be by default in the currency of the first product included. In contrast, the customer document will by default be evaluated in local currency (see reference currency).

> *Note: A document targeted to a customer can be create by loading (in the model hole) the data from a supplier document. The system evaluates the prices according to the input or output sales rules.*

A document list shows the state of each document. This allows it to quickly see the pending ones. All the automatic action which have to be done. But we should remember that any action to be automatically done after a document was in a certain state for a given delay could be realized by an ‘‘wait’’ action introduced in the sequence of actions which leads to that state. If the document state is still the same after the given delay, then the timeout transition will run. That transition includes an action to be performed (that can be the null action). If that action succeeds, the document will change its state. For an invoice, the action can be automatically sent a recall. We should mention that the text of that recall can be deferred from the original version through the selection of another filter than the one used to produce the original version. In fact, the system select in the set of filter the one which extension name matchs the document current state name. Thus, if they are only to version of interest for an invoice, the document designer will create a standard version and a recall one. The standard version has an unqualified name ‘‘invoice’’ and will be used for any document state except the ‘‘wait’’ state. For that state, a version named ‘‘invoice.wait’’ will be used. Of course, the recall version will be the same as the standard one plus some fixed formula recalling the debtor to its engagement.

That process can be cascaded. We can imagine two or three recalls, followed by the automatic generation of a document send to the specialized recovering office. By combining wait states, simple and sequences of actions, including automatic generation of sub-document, offer the opportunity to cover flexible and complex process.

Back to the order example. The company order some goods to a third party. Anorder form is edited. This is quickly and safely done when the system knows every thing except the quantities. Fixing these amounts are the only thing the operator must take care of and, perhaps, an exceptional discount given by the supplier!

That document is automatically faxed for example. As the goods arrive, that order form will be forced to ‘‘received’’ state. This produces the automatic generation of a supplier invoice in the ‘‘draft’’ state. That invoice, can be modified, in the measure that some differences appears, and have be approved, between the order and the invoice.

At the effective reception of a supplier invoice, a simple check will suffice, and if everything is satisfactory, one will close the document which will push the document in the state ‘‘wait_for_payment’’. The payment action will be started at right time through the automatic generation of a payment form. Of course, at each states, the accounting entries will be implicitly performed.

For examples, at approbation time, the creditor account and the set of input VAT will be charged with the appropriated values.

The analytic dispatching of buying, could be done at the reception time or at real reception time of the supplier invoice. This is a question of choice. Let’s go to the accounting section for discussion about the accounting actions and the management of the VAT.

The maintenance section will discuss detail of all the possible actions and their uses. It is also useful to print the definition of the supplied models and then to understand them. These model implement the previously discussed mechanisms.

Some special documents can be equipped with a generator/controller filter. This filter controls the data included in the document specimen and generates new values to be inserted back in the edited specimen. This filter operates when the user presses the save button. The sequence is

1. call the filter,
2. replace values included in the specimen by the one generated by the filter.
3. Then the document completion and correctness test is executed and if successful the document specimen is saved.

This technic is explained in the salary report example included in the distribution. One may use such technic in documents specific to a profession. How to implement such a document is discussed in the maintenance section.

Accounting

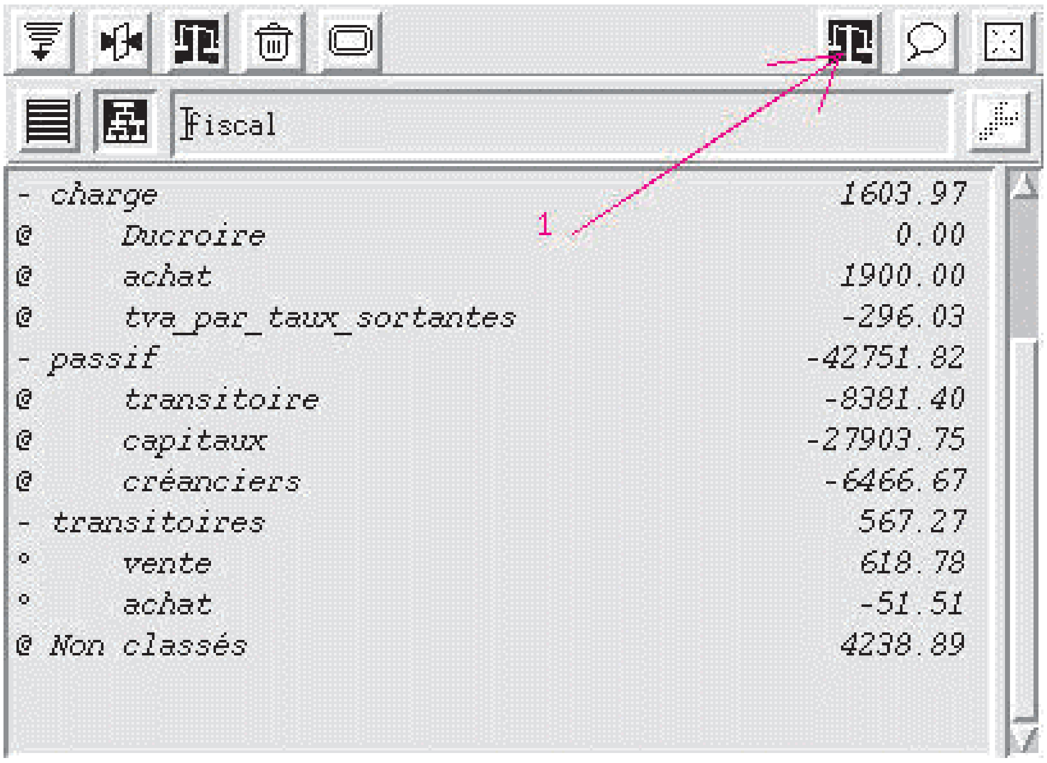
The accounting catalogue shows the list of existing terminal accounts. Terminal account stands for an account which can be directly use as origin or destination in a journal entry. In the accounting catalogue, such an account is presented by its identity and a comment if defined. Such a comment may start with a number like ‘‘1001 cash...’’. If the identity is an natural name like ‘‘cash’’ this leads to ‘‘cash 1001 cash’’. But one may suppress the identifier trough a option of the personal preference file. Finally you can adjust your naming strategy in function of your reporting constraints and your internal usage. But don’t forget that when an accounting action is not automatic the system use the account identifier to guide the user and not the comment. For example for a payment action to chose between cash, credit-card or bank. If the corresponding account identifier are : 1001, 1004, 1002, your accounting chef is happy but real user may be confused!

As for other catalogues, as many hierarchy as wished on the set of terminal accounts. This is the way to establish accounting plans. The global accounts are the parts of partition system defined by each hierarchy. It is thus possible to have several point of view on a single reality as long as we think an accounting system may represent the reality!

As for privileged users, a hierarchy over accounts shows the balance for each regardless of their are terminal or global. Looking at the top of such a hierarchy, made following the usual standard, we have an instantaneous view of the company balance (of course, the closure entry excepted).

At least, any system should include two hierarchies. The first presents the legal accounting, the second presenting the duty account balance at least in VAT context. Caution, all presentations are equivalent and correct when all the terminal accounts are classed (that means that the automatic part which includes all unclassified elements is empty).

However one may add several analytic view points. When a new product is defined, two specific accounts are also created for the analytic accounting. These accounts are normally automatically created following a naming rule defined in the system preference file. The accounting action generated through the document state transition allows it to maintain that analytic accounting perfectly accurate in a perfectly transparent way.



```
- charge                          1603.97
@     Ducroire                       0.00
@     achat                       1900.00
@     tva_par_taux_sortantes      -296.03
- passif                        -42751.82
@     transitoire                -8381.40
@     capitaux                  -27903.75
@     créanciers                 -6466.67
- transitoires                     567.27
°     vente                        618.78
°     achat                        -51.51
@ Non classés                     4238.89
```

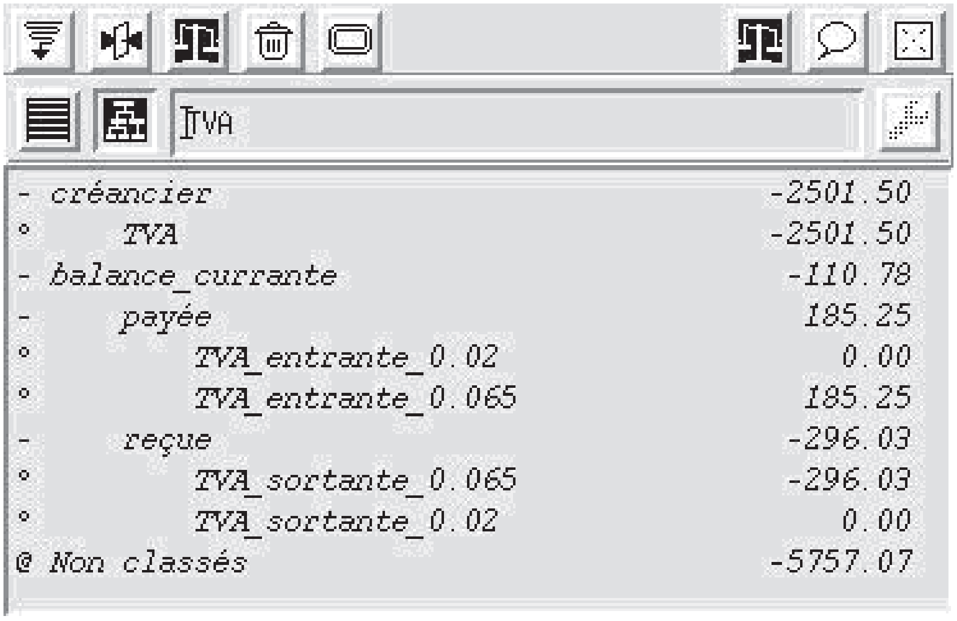
The accounting catalogue has, in its tool bar, a supplementary loadable access button(1) against the other catalogue. This allows it to visualise any terminal account.

Last but not least, the next button allow to publish transactions made for a given period on a set off accounts. This set is defined by the hierarchy presently displayed. All the visible terminal accounts are published. If, on the publisher dialogue the switch "global" is pressed, the group accounts are also published as if they where single terminal accounts. That means they include all the transactions from all included (recursivelly) single accounts.

Duty

Amount the accounting actions, they are those which allows it to register the input and output duty flows. In the definition of a product, we must introduce the specific duty level. That value is used by the duty accounting action. (See the action list in the modelling section).

The system automatically builds the input and output account for each duty level. Thus the duty flows are registered on separated accounts by level. If there are 5 duty levels then the system will have 10 accounts. One may then build a hierarchy with these terminal accounts in order to dispose of necessary data to fill-in the duty administration form. The following figure show such a structure.



```
- créancier                          -2501.50
°     TVA                            -2501.50
- balance_currante                    -110.78
-     payée                            185.25
°         TVA_entrante_0.02              0.00
°         TVA_entrante_0.065           185.25
-     reçue                           -296.03
°         TVA_sortante_0.065          -296.03
°         TVA_sortante_0.02              0.00
@ Non classés                        -5757.07
```

Stock management

At the definition time of a product, one may define the current stock level. Two other fields allows the user to define the lower an upper stock boundaries. This enables the user to define a document model which effects the automatic purchasing process.

The following conditions shall be satisfied to operate this mechanism:

Some documents shall have an automaton which call the actions ‘‘stock’’ and ‘‘unstock’’, an order to maintain the current stock level.

A document that we will name ‘‘order model’’, is defined. It contains a set of products which should be ordered together and from the same supplier. The automate of that model is designed in order to generate an order form. The designer may force the execution of the first transition of that order form. That transition may close the order and then send it to the supplier. But, it is safer to leave that order in draft state. Then one may check it, make some adjustment, if appropriate, and then send it.

Project management

The tool box allows the define new projects. This is simply a name and a comment. Then a new folder appears in the project folder. A project folder may receive all document of any model attached to the project. That can be specified manually by dropping a document in the project folder or automatically by the link mechanism. The project folder is always listed in chronological order. The link mechanism works as follow. If a document is in a project and is used to create an other one this one become member of the project. One should remember here that they are two way the create a document on the basis of an other. The first one is dropping the model the the model hole of a document editor. The second is by an automate action which generate the next document (see generate_action).

Therefore, a company which is project driven should for the link mechanism. This is made in the pattern definition tool. Of course, in that case, a elementary document, which does not require a link, must be inserted first in the project folder. This will act as project root.

The project folder viewer looks like an homogenous document viewer. But, in its selector it has one line more which allows to hide some document model. Typically if one mask all document but supplier invoice and customer invoice then one see the project balance.

Personal folder

The address editor has an button which give an access to the personal folder of the editor target. This folder looks like a project folder. It differ in three main points. The first point is that all the document concern the same address. The second is that it purely automated. No action is needed to define such a folder. The third is that such a folder may not be limited in time. As long as the address exist, the folder maid be produced. As the address is removed the folder disappears. Of course, a view of that folder can be limited to a period.

Employee management

A document called salary report can be created in order to manage the employee salaries. This must be a supplier document. The enterprise pays for the worker’s service with no VAT but with social charges. These are specific to each country. But, generally a percentage is paid to the worker, the remainder should be paid to some dedicated organisation immediately or later.

A example of such a document is presented in the distribution. It was build according to the some suisse rules. It is equipped with a generation filter. The user has only to introduce the line of product called ‘‘hours_of_work’’ with the appropriate quantity. The generator filter produces extra lines for social contribution. That generator is described in the maintenance section. The supplied example automatically sends the payment form. At the same time the social accounts are charged.

Trace

Some time it’s useful to keep trace of magor events. here they are two main solution. The first is achived by using output and system call in the automaton associated with a document. The output action allow to export a set a data from the document in current state. The second is call an external program which could be an call to fill in a data base record with the exported data from the output action. The second solution is to use the dedicated action called trace. That action like an output action allows to select a subset of date from the document current state. Then that set of data is stored in the comment associated to any document. (see action descrition for details).

Specialized stations

The system is not actually a true multi-users in the sens that many users could work on every thing in the same time (with the usual conflicts and roll backs). But on may organize a hierarchy of stations which specialized substations. This can be achieved by using appropriated automatic export actions. One will explain the mechanism around the idea of cash point.

First to create a substation, one must copy the data directory of the upper station to a dedicated directory. Then on define a communication channel. This is made by two definitions in the common preferences file (sys_prf.txt) of the new directory. These definitions are:

export_host: "localhost"

export_port: 10000

Where Localhost is the ip name of the upper station (See your network manager)

and export_port is the tcp port use for that communication channel.

In the upper station the import port must be defined as :

import_port: 10000

With that definition, the upper station will automatically import any export from any substation. For example, if the cash point substation send a copy of the last sale document report with the mention (event) "done", then the upper station will register this sale. In fact, the upper station checks and translates the received document and then applies the transition labelled "done". The actions associated with that transition allows to manage the stocks and the accounting.

That technic allows to have a upper station kept informed of all events from itself and its substations in real time. By contrast the substation are not informed about the events in other stations. The simplest solution to upgrade the substation is to stop them and then to copy in their data area the file business.dmp from the upper station. This may be performed at the closing time of the substations or before the start up.

The system take care of the network trouble. No transmission is lost.

Maintenance

../image/maintenance-en.jpg: Fichier ou répertoire inexistant

Currency and accounts

These three collections are only accessible for the maintenance for safety reasons. In fact, the managers are the only peoples who can modify the currency rates, add and suppress accounts and modify sales functions. It is not difficult to use the corresponding editors. However we must mention now that these collections are also visible to all users but only in read only mode.

Sale-functions

Sale function are tools which allow to compute an output price from a standard input price. The basic parameters are the discount from the supplier, the raw unit margin. Then a quantity step function allow to modify the price according to the sold quantity. But one may also modify the price with a factor associated to some persons. This is made through a partition associated with the address book. The title of the part which include the targeted person may define a price modifier. Two forms are possible to define such a modifier. The first form is ’*xx’. This give a direct factor. The seocnd form ’-XX’ give a discount in %. Example: good_clients-10 or good_clients*0.90 both forms gives 10% of discount.

Next, the last modifier available is either an external function call or an symbolic expression. An external call is defined as just a name of an accessible executable. An expression is introdiced by an equal ‘=’ sign. An expression is made of symbolic variable and symbolic operands. The operator are the classical arithmetic operators +,-,/,* and the exponent ^. The operator also include to validation operator: & , ¬ . The first one gives 1 is the its right operand is greather than 0. Otherwise it gives 0. The second operator gives the inverse result.

The accessible symbolic variable are :

- SP: standard price
- QF: the quantity factor
- PF: the personal factor
- T: an integer representing the today's date. This integer is equal to the number of second from the 1/01/1970 (unix zero time)
- and all the product options fields which have a numeric value.

The system preferences may define one sale function as a default one. Thus any product which define only a input price and a output price are also subject to that default function.

Example: Suppose an optional field named "discount" was defined for product descriptions. If a value is filled in that field this is the current price and that price exclude any other form of discount. The appropriated expression should be:

```
=FP*¬discount + discount/SP
```

The first term equal zero if a discount is defined otherwise it is equal to the person factor. The second term is the ratio between the current price et the standard price. Then if the standard price is 10 et the discounted price is 5, the computed price will be:

```
 p = 10 * (0 + 5/10) = 5
```

Model

../image/pattern_editor-en.jpg: Fichier ou répertoire inexistant

Any document is edited starting from a model. Each model must be defined by privileged users. The model definition is divided into three sections. In the first model, one should choose the basic class for the document, then the specific text fields and then the associated automaton. There are currently four basic recognized : letter, supplier, customer and payment. The letter class is the simplest one. This type includes only text fields. The customer and supplier types both have a product list and some notions of value. The payment class is used for after sale and by action. The most obvious use is to create payment forms which explain the name (but does not justify it).

Each document has an identification which is also called a reference. The identification is unique for a given model. Two models are available. The pure continue numeric one, the numeric continue by year. Also, that numeration can be prefixed. However, we must notice, that the reference system should not change as soon as the corresponding folder is no longer empty. This is for external conference reasons. Of course, this reference plays no internal roles but they traditionally play an important role in external communications. That reason is important enough to maintain this conherence constraint.

../image/pattern-1-en.ag.jpg: Fichier ou répertoire inexistant

../image/pattern_editor_business-en.jpg: Fichier ou répertoire inexistant

The business customer and supplier documents include a list of products concerned by a transaction. Each line includes a quantity and a unit price, etc. Such a document has a value and often an associated duty amount. The document value plus the duty total give the total value also called full total.

For such a document, it is necessary to specify the duty mode. Either, the mode must be defined by the user at exemplar edition time, or the model force the mode. Typically, one may define an invoice for export in such a way that the document is always in duty-free mode. In that section, one can force another currency other than the reference one. We must mention here that the reference currency for a sale is the local currency (see the currency section). In contrast, the reference currency for a supplier document is the one of the first referred product. We assume that normally a supplier sells all its product in a unique currency.

The next step consists of defining the specific text fields. There are three type of fields: text, integer or real. Real or integer can be an array of values (1..N). Each text field can be a single line or multi line. A multi line is defined by a number of visible lines greater than 1. To make a field unlimited in length the scroll button must be pressed. A mandatory field is marked by pressing the mandatory button.

The field list shows the optional field by including them in square backed. A multi line field indicates the number of visible lines. If followed by an @ character then that field is unlimited in length. To modify the definition of a field, it is sufficient to select it.

../image/pattern_editor-en-2.jpg: Fichier ou répertoire inexistant

Then cames the behaviour definition. For example, an invoice can have the following states: draft, approved, sent, recall and played. In each state, several recognised events can be associated. For example an invoice in a sent state should accept two events: pay and delay expired. The document has thus a behaviour in function of its states and recognised events.

../image/state-diagram-en.ag.jpg: Fichier ou répertoire inexistant

The actions are built from the following possible action classes.

-
- Close: freeze the document content. This is the first possible action on a document except an action called *sequence* which first element is precisely an closure action. At closure time, the document is dated.
- Sign: This adds the current users signatures. For a complete automatic signature, a graphic file, the word processor (*see filtering section*) can understand, must be present in the directory called ‘‘signature’’ under the data directory. Of course, one may prefer a manuscript signature on the printed document. However the electronic signature is recommended for fax version which must be signed and then directly send by modem.
- Send: this action sends either automatically or on manually confirmation a version of the document by post and or by fax and or by E-mail. The page layout and the distribution channels are determined by definition in the system preference file and a script specific on each platform. These scripts can be modified by the system manager of the platform to adapt to the software and hardware context.
- Sale and buy accounting: these actions allows it to register a product motion from a buy (sale) account to a creditor (debtor) matching the supplier (customer; which could be ‘‘anonym’’).
- Analytic accounting: these operations are complementary to the previous ones. They allow the dispatching of buy and sale items on specific accounts. If the system preference file allows the automatic creation of a product account, these accounts are created as soon as necessary by the presently discussed actions. Otherwise, the user must manually define the products accounts at the product definition time or, as a last solution, but not the best, the analytic accounting capability of the system cannot be used.

- Duty accounting: these actions manage, either in input and output, the dispatching of the duty by VAT level. These actions are indispensable for any company assigned to the VAT in order to be able to do in two minutes, the periodic report.
- Payment: a set of actions to manage the payment either of supplier and customers. Some are intended for full payment and other for partial payment. Payment can be performed in many ways such as post, banks, credit cards or cash. The definition of each choice is made by dragging account from any sources, but obviously from the accounting catalogue. If no way is define, the user will be call to select the account at execution time. At definition time of partial payment action, it is also possible to define the payment ratio ([0..1]). In that case the action becomes automatic and can be included in a sequence action. This technique is for example useful in salary processing or in a production process where some inputs are dispatched on resulting components.
- Wait: that action type starts an event after a certain delay has expired, if the document is in the appropriate state at that time. For example, sending a recall of invoice if the invoice is still in ‘‘sent’’ state means that the invoice has not been paid. The automatic action is defined by a transition dragged from the possible transition list for the target starting state. If the document is in that state the action is performed and the document moves to the new state defined by the transition. Of course, a wait action only accept a transition when the action is perfectly automatic. For example, a wait action cannot accept a send action in manual mode.

  A delay can be specified in several ways. The first and easiest one is to set a fix number of days. The second is by referring a existing field in the document pattern. This field must be of type "delay". Last, but not least, on may specify partially a date. The unspecified parts being replaced by a character ‘*’ (e.g. 1/*/* for the first of the mouth). This notation can also be used in the a delay field in the document.
- Sequence: this last action allows it to chain actions. It only accepts actions in automatic mode. It grandly simplify the automaton. For example, a payment form can have only one initial state, one transition and one finial state. The transition has a sequential action composed of a closure, a sent and a payment-accounting.
- Generate: this action creates a new document and optionally starts a transition existing in the initial state of that document. We go back to the previous example. When a supplier invoice arrives at expiration time, a payment form is created and the unique transition of that document is executed. This closes the cycle of action attached to the management of supplier accounts. Thus, the last action really made by the user is to approve the invoice. (We think that it’s better not to automatised that step!)
- Output: this action allows to export a set data value from a document. A template like for document presentation is used to define which and where values are to be inserted in the output file. It exactly works like template defined in the html filters directory for presentation. The output template should be placed in the bc filters sub-directory. Some example are present in the distribution.
- Export: this action allows to export all or parts fo the system data. The fragments are the same as those defined in the import/export section. A extra field separator, like a charactere ’;’, may be defined.
- System call: allows to call an external program. To start an other application, to make quality check or to produce data to be introduced back to the document with a input action (see next). In particulary, if the system call does not terminate with a success status, the calling transition will abort and the system message will be displayed to the user.

Input: that action allows to read data from an external file. The text of that file must match the grammar define in the chapter about filters int the section about external data. That action complete the set of action to interact with the external world.

- Import: that action allows to retrieve data written in a text file conforming to the import grammar. Caution, that text can not include field separator like the one defined in an export action.

Trace: that action allows to record data about a event. First, trace starts like an output action by producing a specimen from a template like. It also the same process as the one use to produce a printable version of a document. But, in contrast with the output action, trace does not copy the specimen text in a file. It record re resulting text in a dedicated section of the comment associated with each document. More, a trace action may define a fields which should be define at the moment when a trace action is initiated. The trace action may also clear such field automatically. This force the user to fill in that field for every traced event.

Processing a document always starts by an initial state. We will call it ‘‘draft’’ for facility. For example, an order form, may have the other states called: signed, waiting, resent, received. Such a document moves from the draft state to the signed state after a sequence of actions: close and then sign. The transition from signed to waiting forces the action sequence: send and wait. From that state to ways are possible. The first one corresponds to the expected behaviour, where the delivery happens in the expected delay. The second happens when the delivery is late. In the first case, the user will move the document to the received state at the time the goods arrive. In the second case, the system will automatically start the recall action which will lead the a resending of the document. We must mention here that composition of the image (paper, fax, E-mail) of a document depends on its current state. Thus the resent version can contain some extra text clearly stating that status. On the other hand, the action which leads to the received state can be to generate a supplier-invoice.

When defining a sequence of action, it is important to consider the failing risk for each action. For example, a send action can always fail because that the associated recipient does not have a valid address for the required media. A sequence such as <wait, send> is bad. Indeed, the wait action is queued even if the transmission fails. This is meaningless, and leads to queue two *wait* process in the queue for the document. The second one will trap one an error while the first one already process the transition. Such situations can lead to some tricky errors or simply to unuseful error messages which disturb the users. In contrast, the sequence <*send, wai*t> is safe because the wait is only queued if the send succeed.

A sequence of action can only include action in automatic mode. For instance the payment of a supplier can normally be included in a sequence because it is possible to fix the debited account. In contrast, a payment from a customer, may happen by several ways: banks, cash, etc. In that case, the user should chose manually the right one at execution time. However it is also possible to choose another design in which we have separate events for different payment method. This solution allows the possibylity of creating different sequences of each payment channel which are then full determined. One should also remember that the first action on a document must be a closure. Thus a closure is necessarily in first position of a sequence. Note that a closure can fail. Indeed, it is possible to close a document only if it can be considered as complete.

The definition of the behaviour is made through the bottom section of the pattern editor. On the left hand corner is the list of existing states. On the right corner is the possible events (transitions) for the selected state. Over these two last, it is the transition editor. That editor is loaded by double clicking an event in the right corner list.

There is a sequence of four fields in the transition editor.

1. The starting state: this is either an existing state or a new one to be created. It is possible to fill this field by dragging an existing state.
2. The event: this is a label giving the name of the event starting the transition. The events names are also used in the action list of the document editor. Thus names should be chosen carefully.
3. The action: defines the action to be executed for that event. This field could be manually edited, but it's much easier to drag it from the list of action patterns which are available from the tool bar. As soon as an action has been selected, it is possible to drag it to an action editor hole. The loaded editor allows it to supply the parameters. The action editors have configuration depending of the action type.
4. The destination: defines the final state reached when the action is complete successfully.

Down to the transition editor, a field defining the document starting state. This field must contain an existing state, normally not a terminal one!

../image/pattern-3-en.ag.jpg: Fichier ou répertoire inexistant

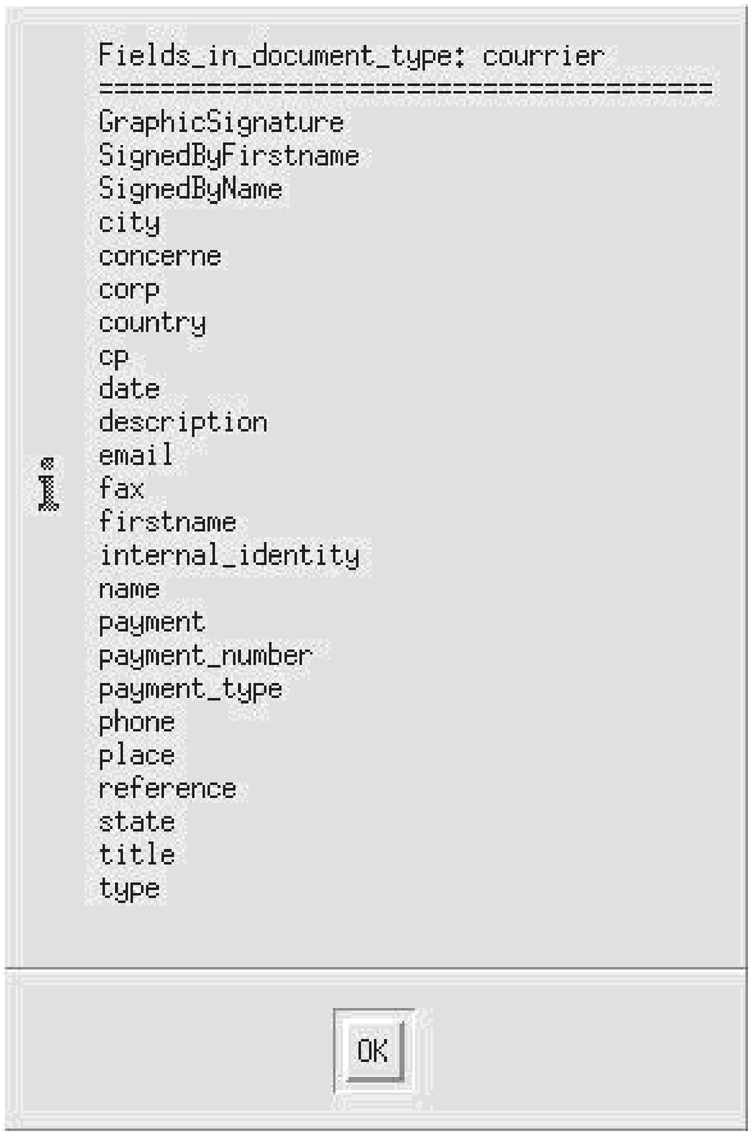
Any action editor has first a list to define of the allowed users. If this section is empty, that means every body may perform it. To fill-in the list it is sufficient to drag & drop people from the recognised user's list.

The accounting actions generally require the definition of several possible accounts either as a source or destination. These lists are also defined by dragging accounts from some account suppliers but generally from the accounting catalogue. Remember that accounting action are only automatic if the list of possible accounts include only one item. Remember too that the second term of such a transaction is defined in function of the action sub-type and of some definition in the system preferences. It is imperative that these definitions are made in the system preference in order to provide a common accounting method for all users.

To edit a wait action a transition is required. This transition must came from the present model. To drag such a transition, one selects the starting state, which is normally the final state of the transition currently under construction, then one drags the a transition (event) from the set of possible transition displayed. Do not double-click the wished transition because this will change the transition editor target.

Filter

Before sending an ABC document image, it must be in a format compatible with the selected transmission channel, paper, fax, E-mail and with the presentation fashions. The common tools to process text layouts are called text processors. The system uses standard text processor available on the market to produce the textual versions in function of the document state. Several filters must be prepare for each model able to generate the image sent to the recipient. A filter is nothing more than a generic version of the document model for a given state. A generic version has all the attributes of the expected image but the true values are replaced by substitution symbols. The symbol names for a model are visible by pressing the button .



```
Fields_in_document_type: courrier
=========================================
GraphicSignature
SignedByFirstname
SignedByName
city
concerne
corp
country
cp
date
description
email
fax
firstname
internal_identity
name
payment
payment_number
payment_type
phone
place
reference
state
title
type
```

Suppose, a model named letter was defined. This includes only a recipient and two mandatory fields named subject and body. The generic version includes, over the fixed parts such as the logo, header and footer, the necessary symbols for a post transport. The needed symbols are : firstname, city, mail box, country. Remember that there is no obligation to use all of them. For local letters the country symbol can be forgotten. This depends on the model used. Either different models are defined for different destinations or only one model is defined with use address fields in excess in order to ensure completeness. In the latest case, it is also possible to play with some possible conditional assembly capability of the word processor to optimize the layout.

Then a date symbol must be placed. It is recommended, in general to use the ABC system date and not the one from the word processor. Indeed, when the document is closed the date is automatically dated. It is normally sent immediately and the delays start from that date. These delays are then managed by the system. The date recorded on the paper will then be the same as the one used internally by the system. If you should send a new copy ten days later, the date should not be the date of the resent but the one of the document closure. The only reason to add the publication date is, when you send a recall, you add the internal system reference date, the current date stating clearly that is just the date of the recall.

Then came the symbols for the subject and the body equipped with all the useful typographical attributes. Remember that the body can run over several paragraphs even if the generic version will only use a very small symbol instead.

It is a good practice to add at least the name of the signor and, depending of the company style, the firstname. These fields are named : ‘‘SignedByName’’ and ‘‘SignedByFristname’’.

It is also possible to add the electronic signature which symbol is : ‘‘graphicSignature’’. That way is perhaps not the best for the paper version in which case one may prefer the handmade one, but is recommended for fax. Remember that this only works for persons authorized to sign and if a graphic file matching the user identity is present in the ‘‘signature’’ data sub-directory then the graphic format is understood by the word processor.

The list of symbols presented by the model editor only presents the logical name of each substitution symbol. In the text of the generic version, each symbol must be surrounded by a prefix and a postfix in such a way that the symbol cannot be confused with an ordinary word. These elements are defined in the system preference file. In the supplied examples for the latex[1] processor, the prefix is ‘‘+§’’ and the postfix is ‘‘§+’’. The date symbol should then be written in the filter as following : ‘‘+§date§+’’ without the bracket. These suffixes must be chosen carefully according to the word processor used.

The description of the product lines must also be marked in a special way. For the latex processor, in the supplied examples, the generic description for such a line is prefixed ‘‘\$\$\$’’.

Caution, numeric text fields with multiple values produces a set of variables named according to the rule as following. If the field named ‘‘X’’ has four values, the substitution variables are: ‘‘X.1’’, ‘‘X.2’’,‘‘X.3’’, ‘‘X.4’’.

The filter files must be placed in a directory in conformance to the definition given in the system preference file. The first definition gives the basic directory. This directory must include a sub directory at the name of each used processor in function of the transmission channel. The system knows 3 channels: post, E-mail, fax and a pseudo channel for the paper archives.

The preceding text may cause consternation for people not really aware of all software problems. It is true that ABC is intended for company managers and not for ‘‘gourou’s’’ of ‘‘C++’’ and the like software traditions. To solve that dilemma there are two solutions. The first one consist of copying the supplied examples and to modify them as lithle as possible in order that these examples will work with your company logo and your address instead of our references. The second solution is to understand that the modelling process must be done only once at start-up and may be changed after a few years because company procedures change or for a change of look. In that case, the effective manager may call a subcontractor who knows all the specialities of word processors and who may also produce the appropriated models.

Input text grammar

ABC understand the format:

identitfier ‘:’ value

The *identifier* is either the name of a text field or the word ‘‘line_’’ directly followed by a number meaning a line number the document product list. For example: ‘‘net’’ is the name of a field

[1]Latex is a text formater available on every platform. It is academical free product. Original design called TeX by Donald Knuth for its proper need. We recommand to use a tools such as Lyx the prepare trivial layout for TeX and latex.

included in the form, but ‘‘line_2’’ means the second line of the product list. For a text field, the value will replace the text in the corresponding from entry. For product line, the rules are as follows.

If the number is bigger than the current list count then a attempt is made to append a new entry to the list. Otherwise, the corresponding line is modified if possible.

The value for a line is a variable list of arguments. The first one gives a quantity. To modify an existing line that argument is enough. The next one is the product identity. This must of course exist in the system. To create a new line these two arguments are sufficient. But a third argument can follow. This will be used as a forced unit price. If a fourth argument is present it will overwrite the default modifier which value is 1. For recall the total price equation is : TP = UP*Q*modifier.

The special label: ABC_INPUT_ACTION_ERROR_MESSAGE is reserver that and error occurre somewhere. This cancel the input action process and then the document transition. The value of that label is returned to the user.

Here is an example for a salary report:

```
$$$ q=+$quantity$+ ; pu=+$price$+
print "net:", q*pu*(1-(0.0505+0.015)), "\n"
print "line_100: ", q*pu, " AVS_PATRON" , "\n"
print "line_101: ", q*pu, " AVS_EMPLOYE" , "\n"
print "line_102: ", q*pu, " AC_PATRON" , "\n"
print "line_103: ", q*pu, " AC_EMPLOYE" , "\n"
halt
```

The first line is substituted by ABC. A copy of such a line is produced for each line existing in the current document specimen. This generator filter expects that the last line represents the worked hours. It then produces four lines with virtual products which are in fact social contributions. These lines will be add to the end of list. Also the text field called ‘‘net’’ will be filled with the value *q*pu*(1-(0.0505+0.015)).* The last program line instructs bc to terminate.

Users

To define the list of users, it is sufficient to drag it from the address list. If the identity does not match the loging name of the user on the current platform, that difference should be corrected. The first way is to rename the address identity. The second is to change the loging name. The third one is to rename the user identification in the user list.

> *Note: For system like Windows9X which can not manage multiple users, the only meaningful usage of users list is to define a pseudo user representing the company. That pseudo user will be attached to that action which should only be performed after a timeout occurred in the system. To turn around that protection under Windows9X, one need only to define an environment variable ‘‘USER = root’’.*

It is one implicit user who has all privileges. This allows that user to perform actions which do not work automatically. This is the platform system administrator (root under UNIX or Linux).

Import Export

Import of data

The system has the capability to import data from external text files. One can actually import data about addresses, products, sale functions, currencies and transactions. The import function has two goals. The first is importing initial data coming from a running system. The second is to allow an easy upgrade of some externally supplied data. This can typically be the case for supplier product lists. A preference parameter is specified which defines if overloading data happens silently or if the user has to confirm any replacement. The default is to require confirmation.

The grammar joint in the appendices defines the exact syntax for the importable texts. An example is also supplied in the example directory.

Data export

The export process also achieves two goals. The first one is to produce a text allowing an easy elaboration of price lists. The second is to allows it to switch between major system releases which could imply a change in the stored structures. This is a provision for the future of an evaluative system.

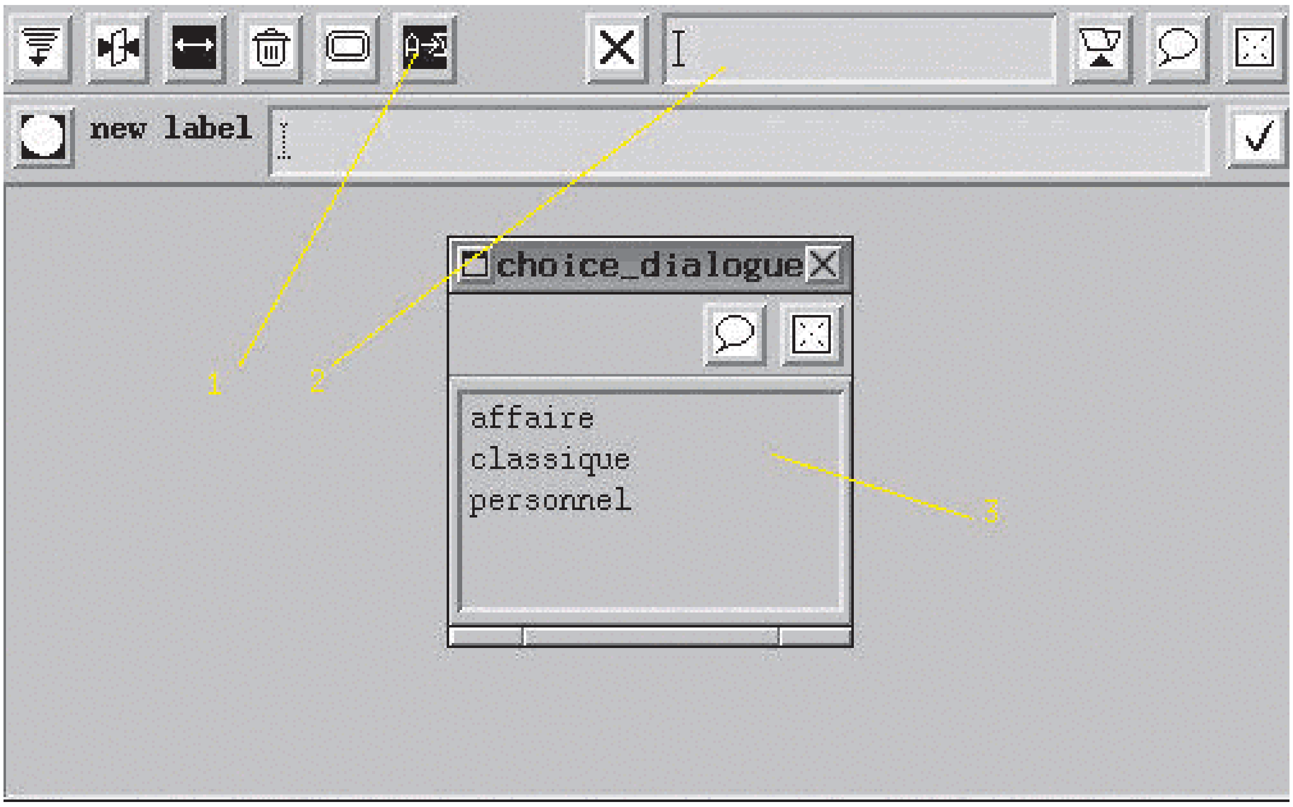
Diary purge

It is normal to regularly purge some transactions to close an accounting period. Often, one produce a company balance at that time. That operation done, the past diary records are considered as frozen. The corresponding records have no more interest for the current management. Also, a huge list slows down some parts of the system.

It is then useful to purge these old records. The diary purge panel allows that operation. But, in contrast with main system, it is always possible to load back purged records through an import of the purged text.

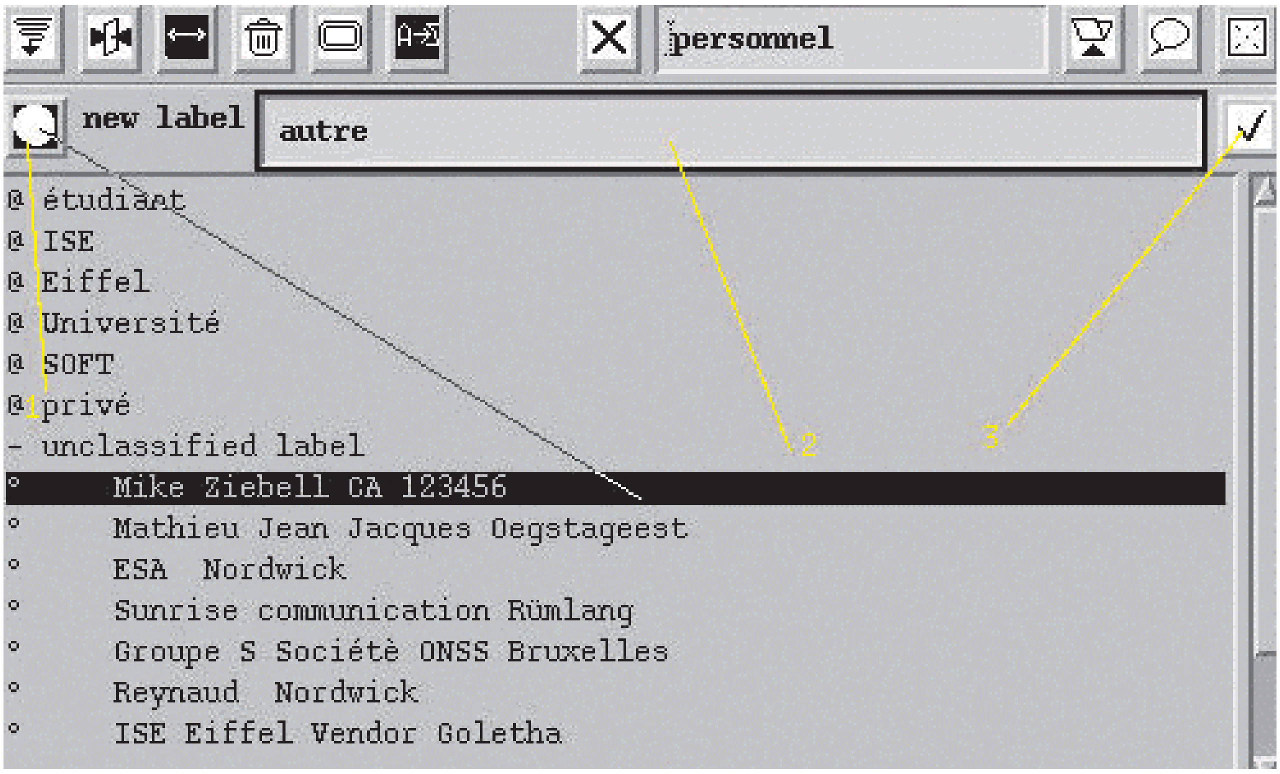
The import-export process could be eventually be used for simulation exercises. In that case, it is *imperative* to work in an environment completely separated from the true data.

Hierarchy editor



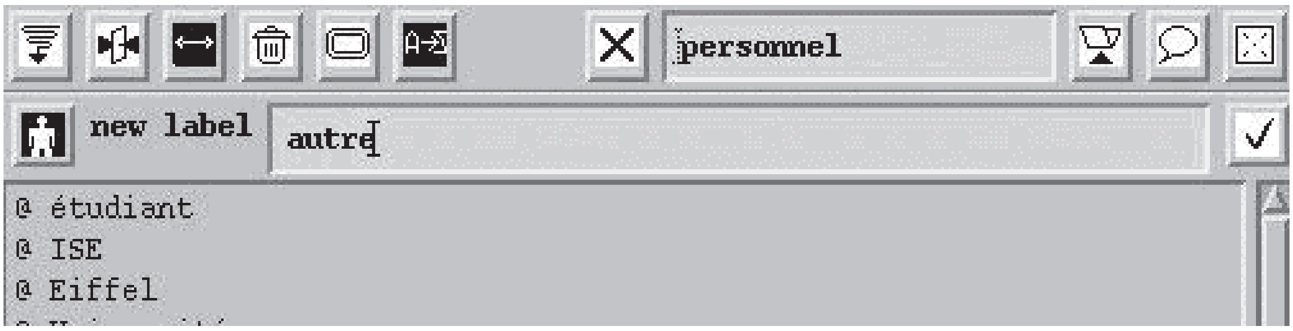
The number of hierarchical partitions on a catalogue. As an new partition is defined by its name, the bottom window show a single part named "unclassified. That part include all elements. As new parts are defined, that special part decrease until it became empty at witch that part disappears.

A new part must include at least one element. Thus one drag one or more items from the catalogue or from the unclassified partition in the round hole in the new part line editor. Then one gives a name for that new part.



```
personnel
new label   autre
@ étudiant
@ ISE
@ Eiffel
@ Université
@ SOFT
@ privé
- unclassified label
°     Mike Ziebell CA 123456
°     Mathieu Jean Jacques Oegstageest
°     ESA  Nordwick
°     Sunrise communication Rümlang
°     Groupe S Société ONSS Bruxelles
°     Reynaud  Nordwick
°     ISE Eiffel Vendor Goletha
```

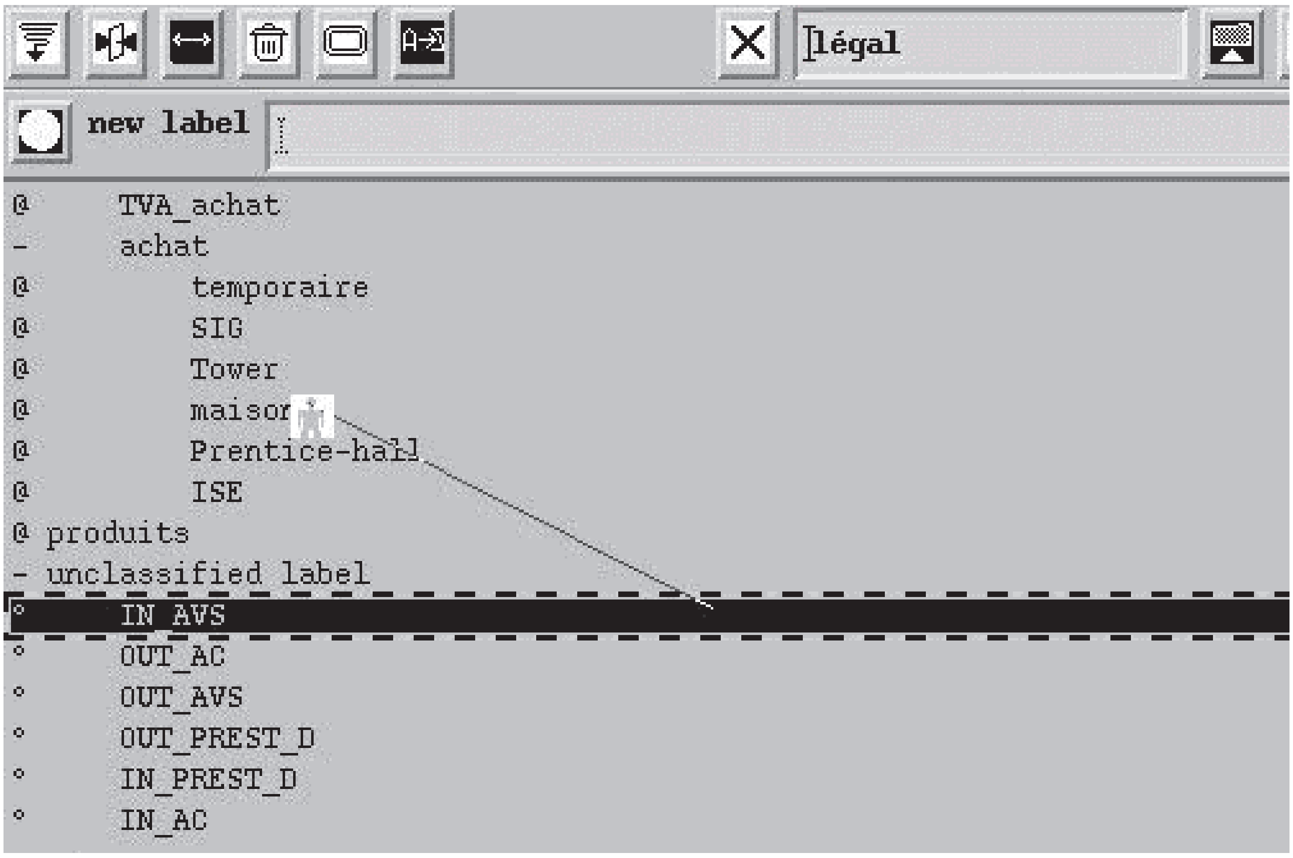
The new part appears in the bottom scrolling list at bottom level. The round hole change its shape as soon as it receive a first element. If this element is a catalogue item then the hole became the same as the hole for such an item. If the first element is a part, then the hole take the shape of a part. This show clearly that it is not allowed to build mixed part including items and parts.



```
personnel
new label   autre
@ étudiant
@ ISE
@ Eiffel
```

Then it is possible to add upper levels. By dragging one or more parts from the bottom list in the item editor, one may define an new part including subparts.

Finally, it is possible to reorganize the structure by drag & drop inside the bottom scrolling list.



```
@     TVA_achat
-     achat
@         temporaire
@         SIG
@         Tower
@         maison
@         Prentice-hall
@         ISE
@ produits
- unclassified label
°     IN_AVS
°     OUT_AC
°     OUT_AVS
°     OUT_PREST_D
°     IN_PREST_D
°     IN_AC
```

Hardware

A Barcode readers can be connected to a ABC station. To support such an option, a definition in the system preference (sys_prf.txt) must be added. This definition as the form:

barcode_reader: device_def.

If the barcode reader is connected on the keyboard, the device_def is equal to "stdin". Else the device_def must express the name of the serial device on which it is connected. (com1, com2 on windows) (dev/ttyS0,... for unix).

The barcode input will be trapped by the line editor of the document which has the input focus. The barcode value goes to the product field , the quantity field is automatically set to 1 and the line is valided. This can be used for cash point station working in the supermarked style.

We recall that any product accepts aliasses. Thia allows, amount other, to recognise products made ans labelled by several vendors.

System administrator privileges

The system administrator may do some action which should be prohibited for any other users. Mainly, to type of action can be performed in such context. First, it may delete any item from any collection. The process is the same as usual, one drag&drop the item on the trash. But here, the system will never complains.

Second, the administrator may force an other state for any document. The rational for that it that an mistake in the design of the associated document type, prevent to follow the reality. One may correct that mistake by a correction of this automaton. But, that change will only take effect for new specimens. Thus, for an old blocked specimen, a good solution is to force it in a terminal state such that ‘cancelled’. That solution does not lead to holes in the reference stream. Of course, it is recommended to provide such a state for any document type.

System parameters

Interface translation

Each user can choose their own interface dictionary. That dictionary contains all the written interface elements. But each user can also modify their its own dictionary. This is made by calling the dedicated window by pressing the button .

../image/translator.jpg: Argument non valide

The load button start the mechanism of selection. A cross cursor appears. Then moving the mousse over the target graphical element and pressing again the select button of the mousse loads the editor. This work in two modes. With the generic button activated, the translation will apply to every items which identifier matches the generic identity (in the above example: *doc_id_lb*). Otherwise, the translation only applies the selected element specified by the full identity (in the above example: *TOP_lisence_wind.box.doc_id_lb).* Of course, a specific translation overwrote a generic one.

It is the possible to decide that all buttons is labelled ‘‘quit’’ but the main panel one is specially named : ‘‘Exit from ABC!’’.

When the new name is written in the editor field, taping a ↵ force the new name in the interface and in the dictionary. But, the generic effect will only take place for the next session.

Preferences

At start-up, the system loads the user preferences, then the system preferences, also called common preferences. In that order, common preferences overide the user’s one. If a user has no personal preferences, then the system loads default values comming from resources directory. The personal preferences file is looked for in the home directory under the name: ‘‘preferences.txt’’. It is also possible to define an alternate personal preference file through an option in the command line. The syntax is ‘‘-preferences=*non_fichier*’’.

> *This is the only solution for Windows9X which do not know about home directory.*

Reasonable examples are provided for individual preferences common one in the delivery.

System preferences are only accessible to managers. System preferences are grouped by categories. One should reply that address and product records can be extended by optional fields. Nad, in the case of product records, when such a options fields os numeric, il can be used to modify the standard price.

External commands.

ABC must request services from the operating system. These commands are all implemented through script files. This allows an easy adaptation to the local environment and allows some extra adaptation. The system is delivered with an operational version of each command. They are listed in the appendices which comments when appropriated.

Real Startup

We assume that you will start from the example you are using right now. It should be a mixture of the original example and of your own data. We will describe here a procedure keep the useful things and introduce the real accounting figures.

First

Suppress any addresses and products from the example

Suppress any unusefull document patterns and currencies.

Purge the journal up to ‘‘tomorrow’’.

Do the same with the document. Press the remove button only if you do not want to restart from particular initial reference numbers.

After that big cleanup, export all in a text file. Then quit the application and suppress the file named ‘‘business.dmp’’ in your data directory. Then we have to prepare the real accounting data.

If you have an accounting program, it should be capable of producing a file with the balance values. The format of that file will inform you on the positive and negative accounts (active = +; passive = -).

These date must be converted in a format that ABC may understand. If your are lucky, one of the small conversion program included in ‘‘resources/tools’’ can do the job for you. Foe example if you use ‘‘winway’’, then, in a terminal, use the command:

```
perl winway your_winway_report_file > accounts.txt
```

```
(the character ‘>’ here means that the output must be writen in a file)
```

If you are less lucky, you must prepare manually that source text file. You can do that by comparison with the original file which was used the create the training environment. This file called ‘‘import.txt’’ should be in your data directory, or under ‘‘resources/examples/*/’’.

Then you have to edit the export file produced by ABC. In that file you will replace the account section by the one you just prepare. Then suppress the transaction section. You may also suppress some more useless addresses or product, but in this context you have to suppress in the same time any references to the deleted items in corresponding hierachies.

When the cleanup complete, you can re-import the data. You may do that from the main ABC program. But it seem more comfortable to use the specific import tool. From a terminal use the command:

```
<path>/import_abc <your modified export file> <your data directory>.
```

```
The <path> is the same as for the main program.
```

The import program does import nothing is their are some mistake in the source text. In that case the import program produce only error message to help the correction of the source text. When the import process succeed, you can restart the main program. Then check all your data and specially the accounting values. I your are no satisfied, the exit and delete again that ‘‘business.dmp’’ file. Correct your source text file and re-import one’s more.

Appendix

File Specification.

At some place the user, specially a system user, has to specify some file names and paths. Of cours absolute path are allowed but are dangerous as soon as some change happen in the machine configuration. Thus, some relative and symbolic method have bee added.

A file path may be defined by:

- a environment variable. The syntaxe is ‘$’ followed by the variable name.
- A reference to the directory of the data file is made by either ‘+’ or ‘$d’.
- A reference to the resources directory is made by ‘&’.
- A reference to the temporary file directory is made by ‘§’.

Examples:

To create a temporary output file:

```
§/output.tmp
```

To produce a result file in the same directory as the opened data file:

```
+/result.txt
```

Note: the character ‘/’ is the directory separator for Unix (Linux). The character ‘\’ is the one use by DOS and then Windows. But, ABC automatically translate that parameter of the specification accordingly to the runnig platform.

Models

Here, a few examples of document patterns. These examples have been produce by pressing the button ‘‘Printing/filtering’’ in the model editor.

```
LETTER

state_control:
    start_state: brouillon
    états:
        brouillon:
            fermer:
                action: CLOSE_ACTION
                destination: pret
        pret:
            envoyer:
                action: SEND_ACTION
                    mail: False
                    poste: False
                    fax: False
                    or_mode: False
                    automatic: False
                    copies: 0
                destination: fait
        archive:
        fait:

private_type: lettre
```

Lettre

```
SUPPLIER_DOC

state_control:
    start_state: brouillon
    états:
        brouillon:
            accepte:
                action: COMPOSITE_ACTION
                    sequences:
                        CLOSE_ACTION
                        SEND_ACTION
                            mail: False
                            poste: True
                            fax: False
                            or_mode: False
                            automatic: True
                            copies: 1
                        ANALYTIC_BUY_ACTION
                        IN_DUTY_ACTION
                        BUY_ACCOUNTING_ACTION
                        WAIT_ACTION
                            start_state: acceptée
                            event: expire
                            delai: 10
                destination: acceptée
        archive:
        payée:
        acceptée:
            expire:
                action: COMPOSITE_ACTION
                    sequences:
                        GENERATE_ACTION
                            generated_document: paiement
                        PAYMENT_CREANCIER_ACTION
                destination: payée

private_type: facture_fournisseur
duty_free: False
fix_duty_mode: False
currency_rate: 0
```

Supplier_invoice

Import Grammar

```
IMPORT_TOP :[CURRENCY_LIST_B] [ACCOUNT_LIST_B] [SALE_FUNCTION_LIST_B]
[ADDRESS_LIST_B] [PRODUCT_LIST_B] [FOLDER_LIST_B] [TRANSACTION_LIST_B] END
CURRENCY_LIST_B : CURRENCIES [CURRENCY_LIST]
CURRENCY_LIST :   CURRENCY_DESCR ..
CURRENCY_DESCR :  RECORD_LABEL : CURRENCY_SYMBOLE NUMERIC_VALUE NUMERIC_VALUE
[COMMENT_DESCR]
RECORD_LABEL :    BOUNDED_IDENTIFIER | SIMPLE_IDENTIFIER | QUOTED_IDENTIFIER
CURRENCY_SYMBOLE :SYMBOL | CURRENCY_SPECIAL
COMMENT_DESCR :   COMMENT = DESCRIPTION
DESCRIPTION :     BOUNDED_STRING | QUOTED_STRING | VALUE_IDENTIFIER
ACCOUNT_LIST_B :  ACCOUNTS [ACCOUNT_LIST] [PARTITION_LIST_B]
ACCOUNT_LIST :    ACCOUNT_TYPE ..
ACCOUNT_TYPE :    POSITVE_ACCOUNT_DESCR | NEGATIVE_ACCOUNT_DESCR |
POSITIVE_ONLY_ACCOUNT_DESCR | NEGATIVE_ONLY_ACCOUNT_DESCR
POSITVE_ACCOUNT_DESCR : RECORD_LABEL : + [NUMERIC_VALUE] [COMMENT_DESCR]
NEGATIVE_ACCOUNT_DESCR :RECORD_LABEL : - [NUMERIC_VALUE] [COMMENT_DESCR]
POSITIVE_ONLY_ACCOUNT_DESCR : RECORD_LABEL : >= [NUMERIC_VALUE] [COMMENT_DESCR]
NEGATIVE_ONLY_ACCOUNT_DESCR : RECORD_LABEL : <= [NUMERIC_VALUE] [COMMENT_DESCR]
PARTITION_LIST_B :PARTITIONS [PARTITION_LIST]
PARTITION_LIST :  SUBPARTITION_TYPE ..
SUBPARTITION_TYPE :     PARTITION_DESCR | TERMINAL_PARTITION_DESCR
PARTITION_DESCR : ITEM_LABEL : { PARTITION_LIST }
ITEM_LABEL :QUOTED_STRING | VALUE_IDENTIFIER | BOUNDED_STRING
TERMINAL_PARTITION_DESCR :    ITEM_LABEL : [ MEMBER_LIST ]
MEMBER_LIST :     DESCRIPTION ","  ..
SALE_FUNCTION_LIST_B :  SALE_FUNCTIONS [SALE_FUNCTION_LIST]
SALE_FUNCTION_LIST :    SALE_FUNCTION_SUBTYPE ..
SALE_FUNCTION_SUBTYPE : DISCOUNT_SALE_F_DESCR | COMPLEX_SALE_F_DESCR
DISCOUNT_SALE_F_DESCR : RECORD_LABEL : NUMERIC_VALUE NUMERIC_VALUE
[NUMERIC_VALUE] [COMMENT_DESCR]
COMPLEX_SALE_F_DESCR :  RECORD_LABEL : DESCRIPTION
ADDRESS_LIST_B :  ADDRESSES [ADDRESS_LIST] [PARTITION_LIST_B]
ADDRESS_LIST :    ADDRESS_DESCR ..
ADDRESS_DESCR :   RECORD_LABEL : ADDRESS_TYPE ADDRESS_IDENT_DESCR
POST_ADDRESS_DESCR [PHONE_NUMBER] [FAX_NUMBER] [EMAIL] [MEMBER_LIST_B]
[OPTION_LIST_B] [COMMENT_DESCR]
ADDRESS_TYPE :    RESERVED_ADDRESS_SHORTENING | RESERVED_ADDRESS_WORD |
BOUNDED_ADDRESS_TITLE
ADDRESS_IDENT_DESCR :   DESCRIPTION DESCRIPTION DESCRIPTION DESCRIPTION
POST_ADDRESS_DESCR :    DESCRIPTION DESCRIPTION DESCRIPTION DESCRIPTION
MEMBER_LIST_B :   [ MEMBER_LIST ]
OPTION_LIST_B :   OPTIONS = [ OPTION_LIST ]
OPTION_LIST :     OPTION_DESCR ","  ..
OPTION_DESCR :    DESCRIPTION : DESCRIPTION
PRODUCT_LIST_B :  PRODUCTS [PRODUCT_LIST] [PARTITION_LIST_B]
PRODUCT_LIST :    PRODUCT_DESCR ..
PRODUCT_DESCR :   RECORD_LABEL : [CURRENCY_SYMBOLE] NUMERIC_VALUE DESCRIPTION
SALE_FUNCTION_TYPES NUMERIC_VALUE NUMERIC_VALUE [NUMERIC_VALUE] [NUMERIC_VALUE]
[NUMERIC_VALUE] DESCRIPTION DESCRIPTION [COMMENT_DESCR]
SALE_FUNCTION_TYPES :   SALE_FUNCTION_ID | SALE_FUNCTION_VALUE
FOLDER_LIST_B :   FOLDERS [FOLDER_LIST]
FOLDER_LIST :     FOLDER_CONTENT ..
FOLDER_CONTENT :  FOLDER_TYPE [ [DOCUMENT_LIST] ]
FOLDER_TYPE :     CUSTOMER_DESCR | SUPPLIER_DESCR | LETTER_DESCR | PAYMENT_DESCR
CUSTOMER_DESCR :  RECORD_LABEL : SALE_DOC REFERENCE_TYPE [COMMENT_DESCR]
REFERENCE_TYPE :  YEAR_NUMERIC_REF_DESCR | NUMERIC_REF_DESCR
YEAR_NUMERIC_REF_DESCR :YEAR_NUMERIC_REF DESCRIPTION
NUMERIC_REF_DESCR :     NUMERIC_REF DESCRIPTION
SUPPLIER_DESCR :  RECORD_LABEL : SUPPLIER_DOC REFERENCE_TYPE [COMMENT_DESCR]
LETTER_DESCR :    RECORD_LABEL : LETTER REFERENCE_TYPE [COMMENT_DESCR]
PAYMENT_DESCR :   RECORD_LABEL : PAYMENT REFERENCE_TYPE [COMMENT_DESCR]
DOCUMENT_LIST :   DOCUMENT_DESCR ..
DOCUMENT_DESCR :  RECORD_LABEL : DESCRIPTION [MEMBER_LIST_B] DESCRIPTION
DOCUMENT_BODY
DOCUMENT_BODY :   LIVE_DOCUMENT | ARCHIVE_DOCUMENT
LIVE_DOCUMENT :   [DATE_TEXT] TAGGED_TEXT_LIST_B [PRODUCT_DOC_DESCR]
TAGGED_TEXT_LIST_B :    [ [TAGGED_TEXT_LIST] ]
TAGGED_TEXT_LIST :TAGGED_TEXT_DESCR ","  ..
TAGGED_TEXT_DESCR :     DESCRIPTION : DESCRIPTION
PRODUCT_DOC_DESCR :     DOCUMENT_VALUE_SUMMERY DOC_LINE_LIST_B [NUMERIC_VALUE]
DOCUMENT_VALUE_SUMMERY :CURRENCY_SYMBOLE NUMERIC_VALUE NUMERIC_VALUE
[NUMERIC_VALUE]
DOC_LINE_LIST_B : [ DOC_LINE_LIST ]
```

Commands

..common/user_command.aw: Argument non valide..common/user_command.aw: Fichier ou répertoire inexistant