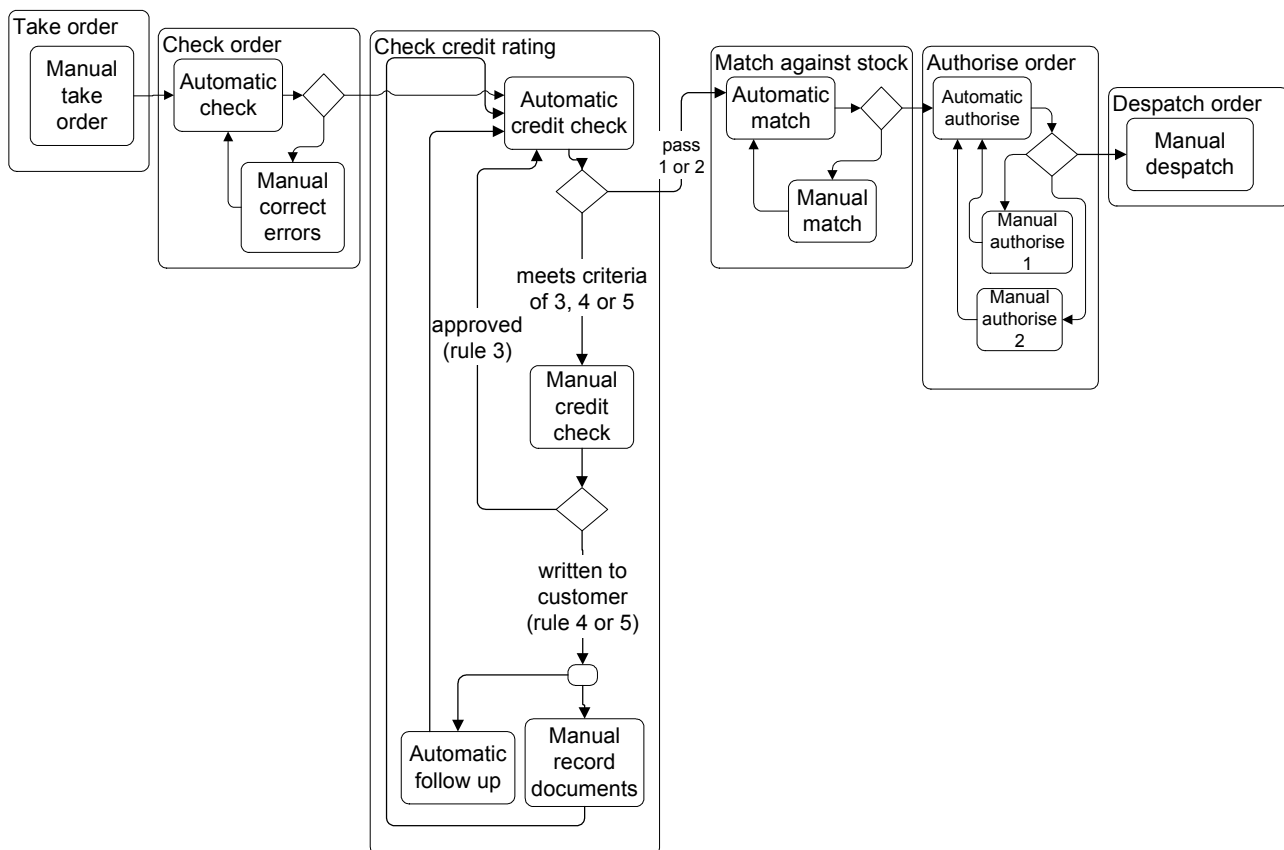


make work make sense

IFW and 'classic' workflow compared

I do not want to delve into the mathematical theory underpinning workflow management systems. I shall however take as my example 'generic external workflow system' not any particular proprietary product but a 'model' as discussed in the theoretical literature (eg van der Aalst 1997⁹). The main comparative features will be as applied to business applications and implementation projects.

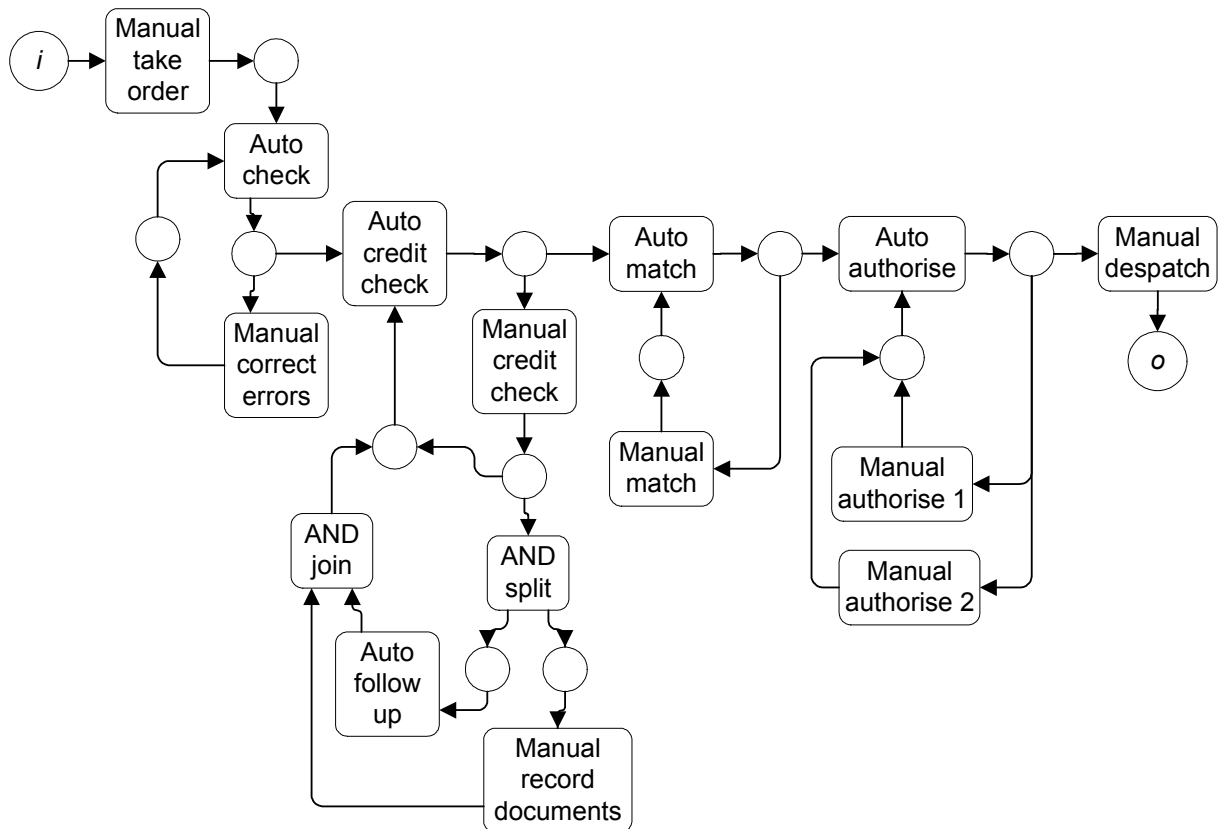
In pure workflow terms it is generally possible to translate the two paradigms into each other. The main difference between classic workflow diagrams and an IFW process diagram is that classic workflow doesn't have 'subprocesses'. Since in an implemented workflow solution a subprocess is ultimately only a 'container' for one or more tasks, this should not matter - in an *implemented* solution.



⁹ WMP van der Aalst, The application of Petri nets to workflow management, Eindhoven University of Technology, 1997.

Our completed 'Order process' in IFW format (and simplified BPMN notation) would appear something like the diagram above.

The same process expressed in 'classic' workflow notation would appear something like this:



The point is not that one is 'right' and the other is 'wrong'. As an implementation the classic schema is just as 'right', especially if the automatic and manual tasks do exactly the same in both cases.

But the classic schema is normally (although not necessarily) associated with a context where 'manual' tasks are the focus of attention, and are either 'truly manual', or tasks associated with separate administration systems. The automatic tasks are normally concerned with routing, and therefore route the work (request) automatically by the application of coded rules. Those rules typically only have a limited amount of data available to them.

There is something 'producer-centric' about this paradigm, as the accent is on getting the most out of human resources, and limiting the overhead of routing between them.

But, as I said, this is not a necessary feature of the 'classic' workflow schema. One could imagine a workflow system

make work make sense

configured in the classic format, but with no linked 'administration' system. And then more and more functionality and data storage is built into that workflow system so that in the end the workflow system and the administration system are one. The result would be an IFW system, even though it originated as a workflow 'skeleton' designed in the classic format.

If this were done, it would have been without the 'subprocess' construct. What then is the point of the 'subprocess'?

The subprocess does not serve a crucial system-architectural function in the implemented solution. The solution will work without it.

But it serves a business-architectural function. This is best brought out by the following observations, which are all different aspects of the same thing:

- 1 The subprocess is concerned with the *what* rather than the *how*. At the subprocess level what is important is that a transition has occurred between one business status and another. How it happened is not relevant. This accords with the customer-centric viewpoint. The customer is not interested in how his claim got approved, just that it did get approved.
- 2 Continuing the theme of the *what* versus the *how*, a process model which stops at the subprocess level is almost by definition a model of the business at the logical level.
- 3 The subprocess level can be important for measurement. Measurement at subprocess level can be both 'natural' and familiar. Yes it might be possible and important to count the number of cases waiting for initial assessment, waiting for medical evidence requested by the underwriter, or waiting for a final underwriting decision, but it is also important to provide subtotals at subprocess level: how many cases are 'in underwriting'?
- 4 In terms of solution design the subprocess level very often corresponds to particular domains of data and functionality – which span across the component tasks.

This partitioning is not as evident from a 'classic' workflow schema. In the example above the only indications that 'Auto authorise', 'Manual authorise 1' and 'Manual authorise 2' are linked are (i) the word 'authorise'; and (ii) the fact that they are in a loop 'controlled' by 'Auto authorise'. Reason (i) is completely arbitrary. Reason (ii) is also fairly

arbitrary, in that 'Manual authorise 1' and 'Manual authorise 2' (in either classic or IFW schemas) do not *have* to route back through 'Auto authorise' to make 'Auto authorise' the only entry and exit point. Both manual authorisation activities could quite legitimately route direct to 'Manual despatch'. If so this would link both 'Manual authorise 1' and 'Manual authorise 2' closer to 'Manual despatch' than to each other.

But as soon as one considers the business content the link between (say) 'Auto authorise', 'Manual authorise 1' and 'Manual authorise 2' is evident. The sort of data items processed automatically in 'Auto authorise' (cash amount of order; customer's outstanding credit; etc) will typically be the ones needing to be displayed and considered 'manually' by 'Manual authorise 1' and 'Manual authorise 2'.

I do not want to labour the point. A key claim of the approach presented in this book is that it keeps the business model and the solution model aligned because they are one and the same model. The subprocess concept and construct is an important factor in that alignment – which is effectively the alignment between *what* and *how*.

This should come a lot clearer in the context of a practical case study.