Design Patterns and Frameworks Dipl.-Inf. Steffen Zschaler INF 2097 `http://st.inf.tu-dresden.de/teaching/dpf`	**Exercise Sheet No. 11** Software Engineering Group Institute for Software and Multimedia Technology Department of Computer Science Technische Universität Dresden 01062 Dresden

Patterns in Business

In this exercise we look at patterns which are particularly relevant in the context of business applications. The first two tasks are on the analysis level and look at a few *analysis patterns*, while the later two tasks look at design patterns from a business application framework.

Task 1: Accountable Organizations

Organisations typically have a hierarchic structure, where sub-organisations report to bigger parts of the organisation above. This structure must be represented at analysis time in order to develop a correct model of the organisation.

1a)

How can we represent an organisation with a single hierarchy? How can we model the rules governing such a hierarchy?

1b)

Many organisations use different, overlapping hierarchies. For example, the Boston-based sub-group of IT-management reports to the IT-management group, which reports to Technical Infrastructure, which eventually reports to Executive Board. At the same time, the Boston-based sub-group of IT-management is responsible for managing the company's IT systems in Boston, so it reports to Boston branch's technical director, who reports to Boston branch's board of management, who report to Executive Board. Thus, the same organisational unit is involved in at least two organisational hierarchies in this company.

How can we model multiple hierarchies in an organisation? Where do the rules go?

1c)

Can the arguments and solutions given above be extended to persons dealing with organisations (or other persons)? If yes, how?

Task 2: Generating Complex Lists

In many business applications it is necessary to generate lists from multiple data sources for varying purposes. Often, for every item in the list it must be possible to determine the original data source, so that actions on the list can be reflected into actions on the original data. It is necessary to be able to vary the algorithm for list creation (including the number of lists to generate) as well as the algorithms to be applied on individual list items or complete lists.

For example, in a warehouse application, at some point, sets of current orders must be transformed into lists of items to be picked from the warehouse. We want to optimise these lists so that for each product only one line occurs in the list, with an amount corresponding to the accumulated amount for this product from all orders.

What design patterns can we use for this?

Task 3: Dynamic Life Cycle

Business objects often have a life cycle the individual elements of which stay the same, but whose arrangement may change depending on context. For example, in a warehouse, an order needs to be treated differently depending on whether it is an Internet order or a direct sales in a factory outlet. In the former case, the order is accepted, planned (determining which storage is to serve the order), prepared for picking, picked, shipped, and invoiced. In the latter case, it is accepted, prepared for picking, picked, and invoiced. Planning and shipping are not necessary, because the customer is right there in the outlet shop and picking can only sensibly happen in the warehouse associated to the outlet.

3a)

How can we realise the individual states in the life-cycle of the order? Remember that each state may require its own interface operations and data.

3b)

How can we realise the different life cycles while reusing as much of the order code as possible?