0 Good tasks

Highlights

Closed task = meaningful user goal.
Check that you have identified all tasks.
Bundle small, related tasks.
Don't program the user dialog.

nat makes a good task?

The task concept sounds simple. In practice, however, analysts often define tasks poorly, choosing tasks that are too vague or too small. Here are some rules for selecting good user tasks:

1 **Closure rule.** Each user task must be "closed", i.e. finish with a meaningful goal that makes the user feel he has achieved something.

2 **Session rule.** Small closed tasks performed in the same work session should be grouped together under a single task description.

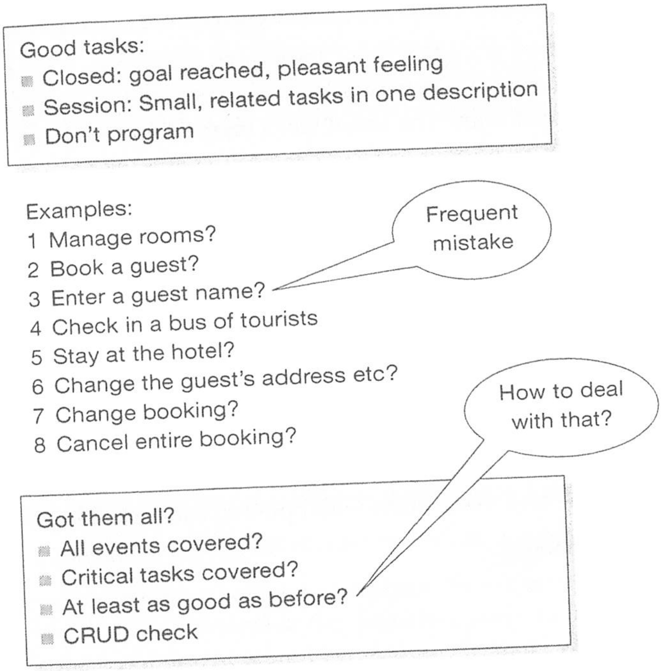
3 **Don't program.** Don't go into detail with how the task is performed, the exact sequence of steps involved, how to deal with special cases, etc.

The concept of "closure" originates from task analysis in human–computer interaction. A closed task has a goal that is meaningful to the user. Completing the task gives the user a pleasant feeling – he feels he has achieved something; maybe he deserves a cup of coffee. Psychologists say that most of us love closed tasks so much that we prefer to do the small things that we can complete now, rather than the large things that may take days.

While a task has to be closed, the sub-tasks in the description are usually so small that they are not closed, and consequently they are not real tasks.

Some tasks are closed when looked upon individually, but they belong together and are carried out in a single session, so they should be described as a single task. The reason is that we want efficient support for the group of tasks rather then the individual, small tasks.

Some developers specify too many details of the task, e.g. what to do if the booking number is not correct, which conditions trigger alternative paths among the sub-tasks, etc. These are programming details. At best they describe what is going on inside the computer, but that is premature design; also, the customer can rarely validate such details.



Good tasks:
- Closed: goal reached, pleasant feeling
- Session: Small, related tasks in one description
- Don't program

Examples:
1 Manage rooms?
2 Book a guest?
3 Enter a guest name?
4 Check in a bus of tourists
5 Stay at the hotel?
6 Change the guest's address etc?
7 Change booking?
8 Cancel entire booking?

Got them all?
- All events covered?
- Critical tasks covered?
- At least as good as before?
- CRUD check

The task description should explain what the user does or wants to do, and users are not programmed in the same way as computers. You should use variants to handle the complexity, not programming.

Examples

Let us try to apply the rules to the examples in Figure 3.10. Which of the examples are good user tasks?

Manage rooms is an important activity, but it is not closed. You cannot say that nov you have finished managing the rooms. It is an ongoing activity and thus not a good task.

Book a guest is a good task. It is closed and, when the receptionist has done it, something meaningful has been done. It may also be the time for a break – unless other guests need attention.

Enter guest name is not a good task. There is no closure. Receptionists would not feel that they have achieved something after entering the guest name. Entering the gues name is part of a larger, more meaningful activity, e.g. booking the guest. Surprisingly many tasks (use cases) seen in practice are defined on such a detailed, but meaningless level. This also means that the number of use cases is much too high.

Check in a bus of tourists is a good task. Although checking in one of them may be considered a closed task, the receptionist would feel that there is no time for a break until all are dealt with. The small tasks form a single session to be supported efficiently.

A stay at the hotel seems at first sight to be a strange task. It is not a task for the receptionist, but if we look at the guest as a type of user, staying at the hotel is a meaningful activity. Handling such activities is the whole purpose of the hotel. In section 3.11 we shall look at this from the guest's point of view and see how it gives rise to business process re-engineering.

Change the guest's name and address can be a closed task, for instance if a booked guest phones and says that he has moved house and want the confirmation to be sent to the new address. *Change the booking and cancel the booking* can also be separate closed tasks. However, the three tasks are often done in the same session and should be grouped into a single task description with optional sub-tasks like this:

Task: Change booking

Sub-tasks:

1 Find booking

2 Modify guest information (optional)

3 Cancel booking (optional)

ompleteness

How can we ensure that we identify all tasks? First of all, you need to interact with users to learn about the tasks. Section 8.2 explains about using interviews, observations, etc. to identify user tasks. Many developers try to guess the tasks by means of logic and common sense, sitting in their offices, but they fail to really understand what is going on.

However, even interviews and observations may not reveal all the tasks. How can we ensure that all tasks have been covered? It is impossible to guarantee such completeness, but here are some guidelines for getting close.

All domain events covered? An event is something that requests a service of our system. If our "system" is the reception, a domain event requests a service from the reception. Examples are that the guest calls to make a booking, that a guest arrives, etc. Each domain event gives rise to a user task. In this way we identify tasks for booking, check-in, etc. You should make a list of the domain events and ensure that each has a corresponding task.

difficult, or performed under stress. It is important to identify these because they need careful support. When observing users, you may see the time-consuming and frequent tasks, but rarely the difficult or stressful tasks; you have to ask about them. In the hotel reception scenario above, you might only identify the busload of tourists as a critical task if you asked about stressful and difficult situations.

At least as good as before? There is a difficult problem in this area: introducing a new system may turn a non-critical task into a critical one. As an example, one manufacturing company replaced their old IT system with a new COTS-based one. The old system had a very sophisticated screen picture that gave an excellent overview of pending repair jobs in the factory. However, the users were unaware that they used a sophisticated screen. The new system only showed traditional lists of data records that didn't give the same overview, and it would be very expensive to create new sophisticated screens. The old task using the repair list had not been stressful using the old system, but it was using the new one. Although the analysts had catered for all critical tasks, they had not realized that the new system would create a new critical task

This is one of the reasons the customer often wants a requirement like this:

R2 The product shall perform at least as well as our present product.

Suppliers strongly resist such requirements since it may be an immense task for them to study the old system in the detail necessary to meet this requirement. Verifying the requirement would be hard for the same reason. In this case, a skilled analyst might have spotted the sophisticated screen when studying the users, and identified it as something crucial.

Another technique that can help is to give the supplier a screen dump of all screen in the system, thus allowing him to spot unusual screens. Goal-domain tracing ma sometimes identify unnoticed, but critical, tasks. Section 8.7 explains the technique and shows how it detected such a task in the shipyard.

CRUD check. CRUD stands for Create-Read-Update-Delete. In order to make a CRUD check of the user tasks, you need a description of the data to be stored in th system. (Any of the descriptions mentioned in Chapter 2 will do.) You now look a each piece of data and ask yourself how that piece is created, read, updated, and deleted, and whether some task description deals with it.

If no task description deals with it, you will probably have to add such a task. Usually you identify some surprising new tasks in this way. In other cases there may be good reasons for not adding the task, for instance because the data is maintained in another system.

Some of the missing user tasks are often small tasks, such as updating the customer's address or deleting the customer. They may be grouped together according to the session principle into larger maintenance tasks. Section 9.2.3 explains more about CRUD checks.

Hard-to-catch activities Some activities may not be real work tasks or they may be hard to identify, but they may still require product functionality. Examples are ad hoc reports, games, surfing the Web without precise goals, supervisory functions (e.g. checking that everything in the plant is running smoothly).

An example from the hotel system is the need for an estimate of staff numbers needed in the next period (see section 3.4). Once this is suggested, the need is obvious, but would you have identified that need? There is no clear task or external event dealing with it, and there is probably no data in the database suggesting such a need.

We have no special remedy for identifying these activities. All we can suggest is that you have to study the domain better to ensure that all such activities are recognized; it is important to study all user groups. See Beyer and Holtzblatt (1998) for good techniques in that area. Identifying the business goals and tracing them to user activities and requirements may also help (see section 8.7).

New tasks may sometimes be created when the new product is introduced. Jan C. Clausen (personal communication) has for instance pointed out that when automatic workflow is introduced, a new task is created; the task of manually sorting out all the electronic documents that don't find a receiver in the automated way.