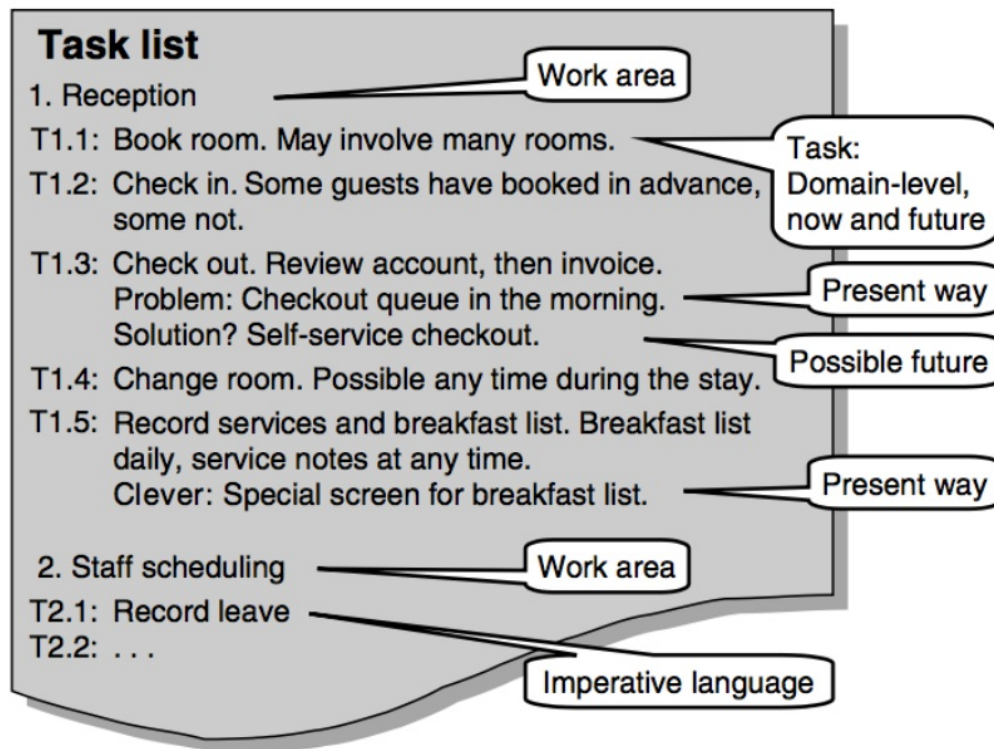


Templates

Annotated Task list



Task list

[num]. Text Here

T[num].[num] Text here Text Here

Task Description

T1.2: Check in	
Start: A guest arrives.	
End: The guest has got room(s) and key. Accounting started.	
Frequency: Total: Around 0.5 check-ins per room per night. Per user: 60 ...	
Difficult: A bus with 50 guests arrive	
Subtasks:	
1. Find room.	Domain-level, now and future. Imperative language
Problem: Guest wants neighbour rooms; price bargain.	
1a. Guest has booked in advance.	Variants Undo?
1b. No suitable room.	
2. Record guest data.	Missing subtask?
2a. Guest recorded at booking.	
2b. Regular guest.	
3. Record that guest is checked in.	Full reference to a subtask: T1.2-4
4. Deliver key.	
Problem: Guest forgets to return the key; guest wants two keys.	Past: Problems

Task descriptions

T[num]	Text here
Start:	Text here
End:	Text here
Frequency:	Text here
Difficult:	Text here
Subtasks:	
[num]. Text here (optional)	
Problem: Text Here	
Solution?: Text Here	
Smart: Text Here	
[num]a. Text Here	
[num]b. Text Here	

Work area and user profile

Work area: 1. Reception

Service guests – small and large issues.

Normally standing, for instance facing the guest. Frequent interrupts.
Often alone, e.g. during night.

User profile: Novice. Often a temporary job.

IT knowledge: Simple text processing. Younger persons have surfed a bit on the Web.

IT attitude: Part of the job, but not fascinating in itself.

Domain knowledge: Knows only the very basics, for instance what check-in is in the simplest case.

Domain attitude: OK but not the career of life. It is just a temporary job.

Discretionary use: Mandatory.

Physical abilities: Normal sight, hearing, size, etc.

User profile: Experienced. Often a lifetime job.

IT knowledge: Simple text processing. Some know more, of course.

IT attitude: Curious about how it works in the job.

Domain knowledge: Knows all the procedures and the special cases.

Domain attitude: Likes the job. Likes to be an expert.

[WORK AREA NAME]

User profile	IT Knowledge	IT Attitude	Domain knowledge	Domain attitude	Physical abilities
Text here	Text here	Text here	Text here	Text here	Text here

Design Defects

Design defects

D100. Understanding Breakfast window
vs. Stay window?

D101. Check credit card or get deposit.

D102. Understanding guest address across
stays?

D103. Long stays, or with many rooms.
(evergrowing list of defects)

Design Defects

D100. Text here

Subtasks vs Visible data vs Virtual Windows

T1.2: Check-in		
Start: A guest arrives . . .		
Subtasks:	Visible data:	Virtual windows:
1. Find room. Problem: neighbour rooms.	Free rooms of type x, price. Map.	Rooms. Crit: type, period. Map?
1a. Guest booked in advance. Problem: Fuzzy guest ID	Guest and stay details.	FindGuest, Stay. Crit: soundex, ...
1b. No suitable room.	All free rooms, price, discount.	Rooms. Crit: period.
2. Record guest data.	Guest detail.	Stay.
2a. Guest recorded at booking.	Guest detail.	FindGuest, Stay. Crit: soundex, ...
2b. Regular customer.	Guest detail.	FindGuest, ...
3. Record that guest is checked in.	Guest and stay details.	Stay.
4. Deliver key.	Room numbers.	Stay.

Relation between subtasks, visible data and virtual windows

T[num]	Text here
Start:	Text here
End:	Text here
Frequency:	Text here
Difficult:	Text here
Subtasks:	Visible data: Virtual windows:

1. Text here

Problem: Text Here

Solution?: Text Here

Smart: Text Here

1a. Text Here

1b. Text Here

Use cases

Task	User action	System-action & Feedback
1.1 Booking		
1. Find rooms	Enters search criteria in vwRooms. Clicks Find Rooms. Selects a room(ChooseRoom).	Shows free rooms. Shows selection.
1a. No suitable room	Enters new search criteria. Clicks FindRooms. Selects a room.	Shows free rooms. Shows selection.
2. Record guest	Clicks NewStay. Enters guest data in Stay window.	Shows new, empty Stay window.
2a. Regular guest	Enters search criteria in vwFindGuest. Clicks FindGuest. Looks down the list for the right person. Selects the guest if one matches. Clicks NewStay. Edits guest data in Stay window.	Shows guests that match. Shows more details for each guest. Shows new Stay window. Fills it with data about selected guest or data from the search criteria.
3. Record booking	Clicks Book.	Records guest data, stay data, room states. Updates vwRooms and vwStay. Visible feedback to user in vwStay?
3a. More rooms	Uses vwRooms as in subtask 1 or 1a. Clicks Book.	Records additional room states. Updates vwRooms.
4. Confirm (option)	Clicks PrintConfirm	Prints confirmation letter.

Use cases

Task: **User Action:** **System-action:**

[Num] Text here

1. Text here Text Here. Text Here

CREDO

CREDO = Create, Read, Edit, Delete, Overview

Data model versus virtual windows:

Entity Virt. window	Guest	Stay	Room	RoomState	ServiceRec.	ServiceType	Missing window data
Stay	CRE	CRED	r	re O	CREDO	R	
Rooms			CREDO	re O			
Breakfast			r		CREDO	R	roomID
Service charges						CREDO	
Missing fncts	DO	O		(C) D			

Notes: RoomState: personCount editable through Stay, all states through Rooms.
Breakfast: roomID . . .

Data model versus tasks:

Entity Task	Guest	Stay	Room	RoomState	ServiceRec.	ServiceType	Missing task data
Book	CRE O	C	O	Re O			neighbour
CheckinBooked	RE	E O	O	Re			
CheckinNonbkd	CRE O	C	O	Re O			neighbour
Checkout	RE	E O	R	e	R		
ChangeRoom		R	O	Re O			
RecordService			O		C O	R	
PriceChange			C EDO			CREDO	statistics
Missing tasks	D	D		(C) e D	ED		

Data model versus virtual windows

Entity: Entity1 Entity2 Entity3 Entity4

Virt. window:

Room1 CREDO CREDO CREDO CREDO

Data model versus tasks

Entity: Entity1 Entity2 Entity3 Entity4

Task:

Task1 CREDO CREDO CREDO CREDO

Measuring Task Time

ATM

Users: 20 bank customers, random selection.

Task 1: Withdraw \$100 from ATM. No instructions.

Measure: How many succeed in 2 min?

Task 2: Withdraw as much as possible (\$174)

Measure: How many succeed in 5 min?

Reqs: Task 1: 18 succeed.
Task 2: 12 succeed.

How to measure

What to measure

Requirement – target

Internal ordering system

Users: 5 secretaries in the company.
Have tried the internal ordering system.
Have not used it for a month.

Task 1: Order two boxes of letter paper + ...

Measure: Average time per user.

Reqs: Average time below 5 min.

What to measure
Risky!

Pros: Classic approach. Good when buying.

Cons: Not good for development.
Not possible early. Little feedback.

Task time

[SYSTEM
NAME]

Users: Text here

Task 1: Text here

Measure: Text here?

Task 1: [num] succeed

Requirements: Task [n]: [num] out of [num] must succeed within
___ min.

Measuring with Opinion Polls

Ask 20 novice users to complete the questionnaire.
Measure: Count number of entries per box.
Reqs: 80% find system easy to learn.
50% will recommend it to others.

How to measure
What to measure
Requirement

Questionnaire	agree	neutral	disagree
The system was easy to learn			
The system is easy to use			
The system helps me ...			
It is fun to use			
I will recommend it to others			

Pros: Widely used.
You may ask for any usability factor.

Cons: Doesn't match objective evidence.
Only indications during development.
Little feedback to developers.

Ask [num] users to complete the questionnaire

Measure: Text here

Requirements: [num]% Text here
[num]% Text here

Questionnaire

Agree Neutral Disagree

The system was easy to learn

The system is easy to use

The system helps me [TEXT HERE]

It is fun to use

I will recommend it to others

Measuring with Score for understanding

Ask 5 potential ATM users what these error messages mean:

Amount too large
PIN code invalid ...

Ask them also:

What would the system do if ...

Measure: Assess answers on scale A-D.

Reqs: 80% of answers marked A or B.

How to measure

What to measure

Requirement

Pros: Easy way to test understandability.
Best way to cover error messages.
Useful both early and late in development.
Cons: Only measures understandability.

Ask [num] [SYSTEM NAME] users what these error messages mean:

[ERROR NAME 1]

[ERROR NAME 2]

Ask them also:

[QUESTION]

Measure: Assess answers on scale A-D

Requirements: [num]% of answers marked A or B

Measuring with Guideline adherence

Ask an expert to review the user interface and identify deviations from guideline X. (Or ask two experts to come up with a joint list.)

How to measure

Measure: Number of deviations per screen.

What to measure

Reqs: At most one deviation per screen.

Requirement

Pros: Adherence helps users switch between systems.
Company-specific guidelines for internal systems can help even more.

Cons: Cannot guarantee high usability.
Developers find guidelines hard to follow – examples help best.

Ask an expert to review the UI and identify deviations from [GUIDELINE NAME HERE].

Measure: Number of deviations per screen.

Requirements: At most [NUM] deviations per screen.