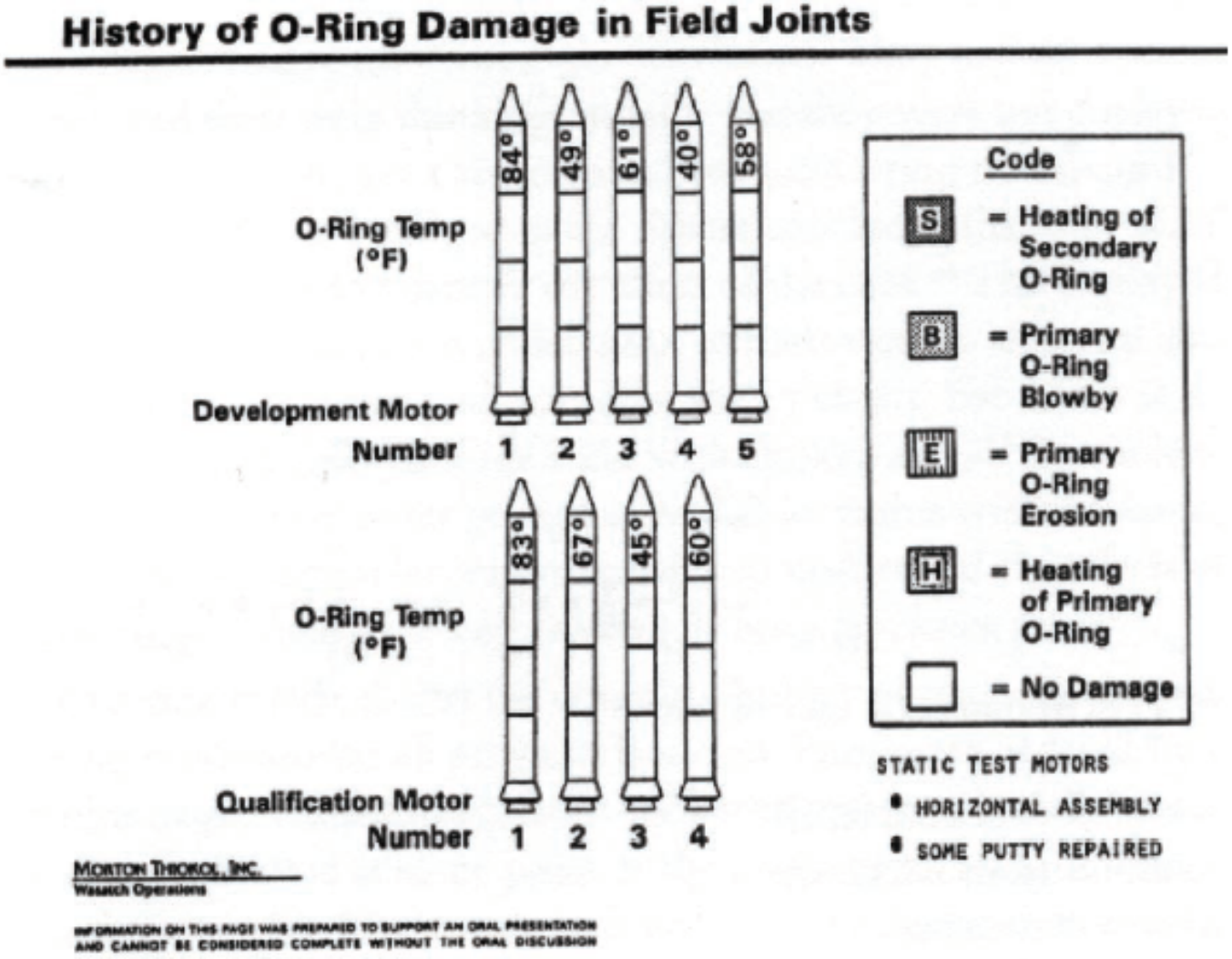
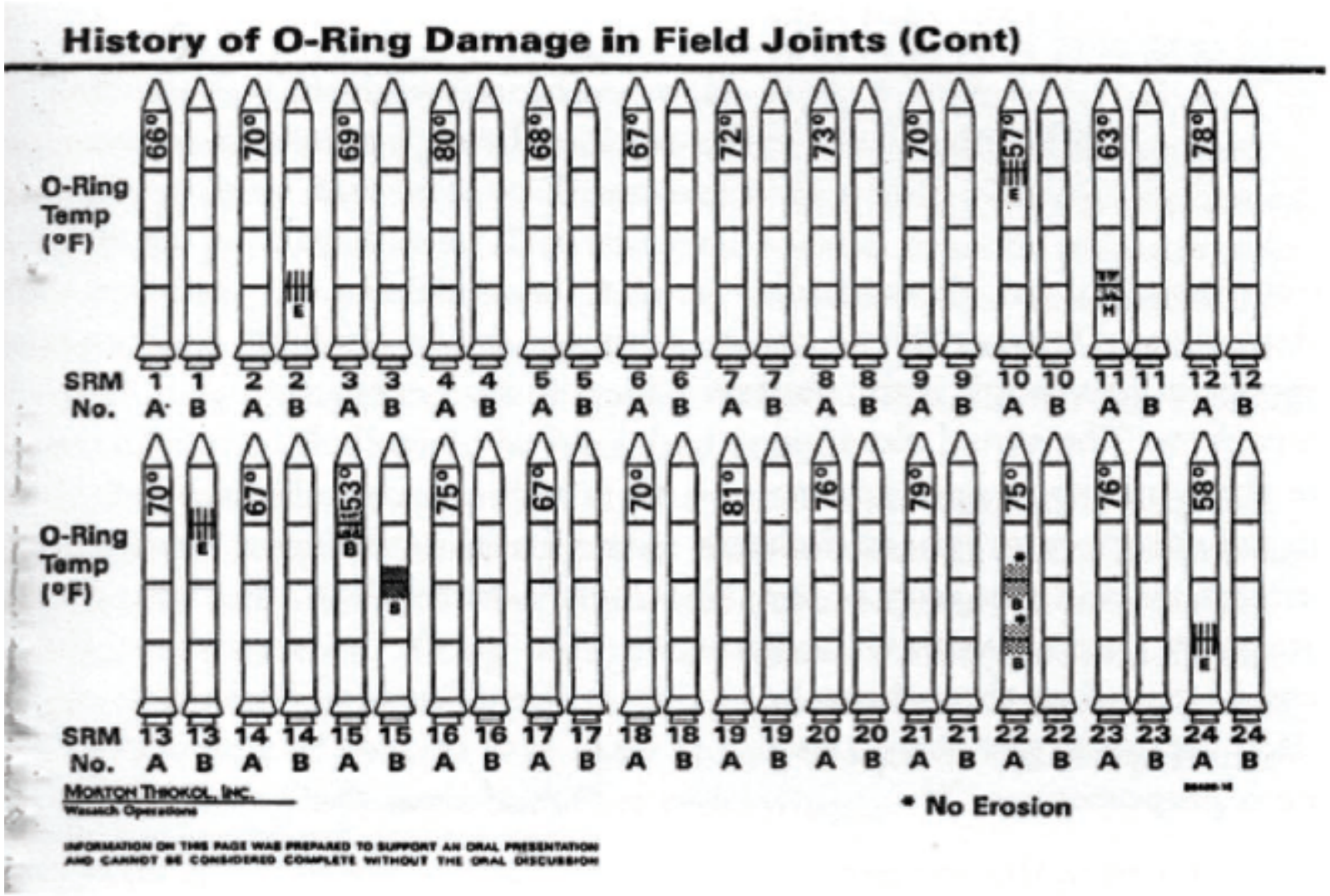
Requirements, Scenarios and Task Analysis

Human Factors

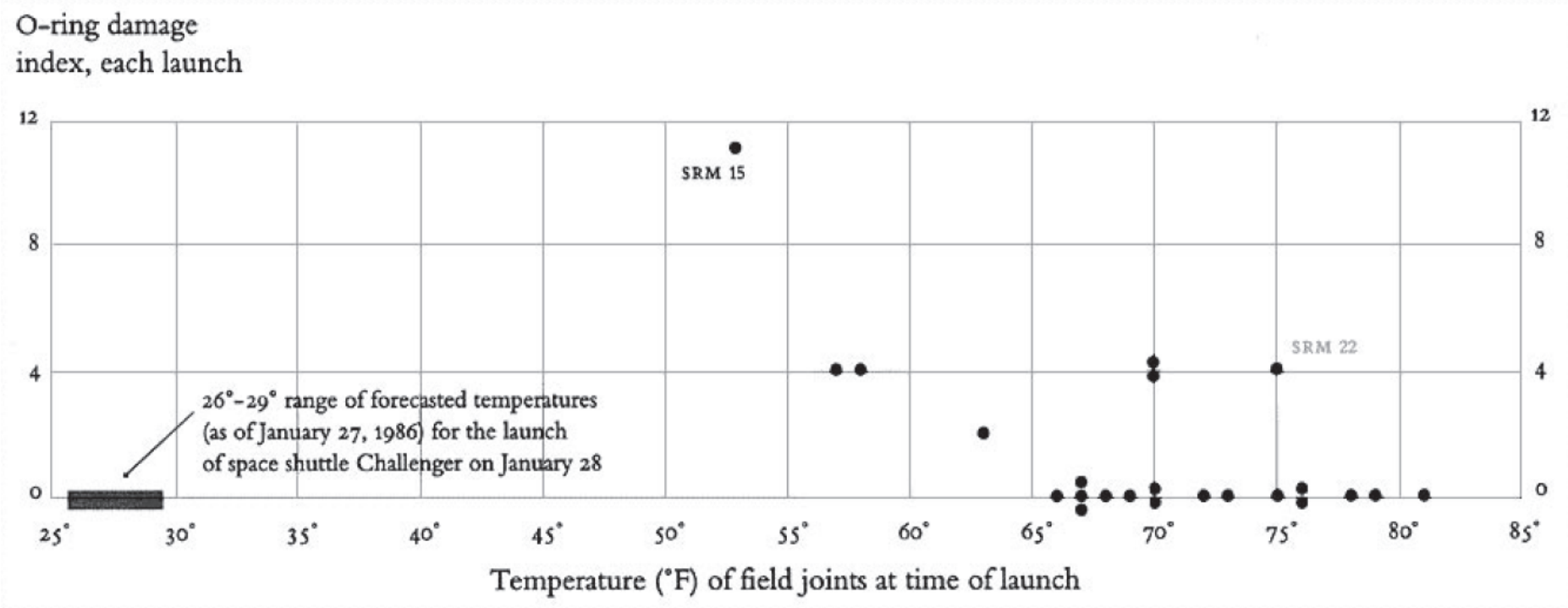
A data graphics example



A data graphics example



An alternative visualisation



(From Tufte)

Requirements

- Understand as much as possible about users, task, context
- Produce a stable set of requirements
- How:
    - Data gathering activities
    - Data analysis activities
    - Expression as ‘requirements’

All of this is iterative.
Allows us to build the right system.

Establishing requirements

- What do users want? What do users ‘need’?

  Requirements need clarification, refinement, completion, re-scoping

  Input: requirements document (maybe)

  Output: stable requirements

- Why ‘establish’?

  Requirements arise from understanding users’ needs

  Requirements can be justified & related to data

Different kinds of requirements

- Functional:
  - What the system should do
  - Historically the main focus of requirements activities
  - Data:
    - What kinds of data need to be stored?
    - How will they be stored (e.g. database)?
- Non-functional:
  - How well the system will do it.
  - Memory size, response time, usability.

Different kinds of requirements

Environment or context of use:

— physical: dusty? noisy? vibration? light? heat? humidity? ….

—social: sharing of files, of displays, in paper, across great distances, work individually, privacy for clients

— organisational: hierarchy, IT department's attitude and remit, user support, communications structure and infrastructure, availability of training

Different kinds of requirements

- Users: Who are they?
  - Characteristics: ability, background, attitude to computers
  - System use: novice, expert, casual, frequent
  - Novice: step-by-step (prompted), constrained, clear information
  - Expert: flexibility, access/power
  - Frequent: short cuts
  - Casual/infrequent: clear instructions, e.g. menu paths

Kinds of requirements

What factors (environmental, user, usability) would affect the following systems?

- Self-service filling and payment system for a petrol station.
- On-board ship data analysis system for geologists searching for oil.
- Social networking website.

Data gathering for requirements

Interviews:

- Props, e.g. sample scenarios of use, prototypes, can be used in interviews
- Good for exploring issues
- But are time consuming and may be infeasible to visit everyone

Focus groups:

- Group interviews
- Good at gaining a consensus view and/or highlighting areas of conflict
- But can be dominated by individuals

Data gathering for requirements

Questionnaires:

- Often used in conjunction with other techniques
- Can give quantitative or qualitative data
- Good for answering specific questions from a large, dispersed group of people

Researching similar products:

- Good for prompting requirements

Data gathering for requirements

Direct observation:

- Gain insights into stakeholders' tasks
- Good for understanding the nature and context of the tasks
- But, it requires time and commitment from a member of the design team, and it can result in a huge amount of data

Indirect observation:

- Not often used in requirements activity
- Good for logging current tasks

Data gathering for requirements

Studying documentation:

- Procedures and rules are often written down in manuals
- Good source of data about the steps involved in an activity, and any regulations governing a task
- Not to be used in isolation but good for key words and prompting interviews
- Good for understanding legislation, and getting background information
- No stakeholder time, which is a limiting factor on the other techniques

Data gathering for requirements

- Stakeholders:
  - Identifying and involving stakeholders: users, managers, developers, customer reps?, union reps?, shareholders?
  - Dominance of certain stakeholders.
  - ‘Real’ users, not managers:
    traditionally a problem in software engineering, but better now
  - Availability of key people
- Process:
  - Workshops, interviews, workplace studies, co-opt stakeholders onto the development team
  - Communication between parties:
    - within development team
    - with customer/user
    - between users… different parts of an organisation use different terminology

More problems with data gathering

- Domain knowledge distributed and implicit.
  - difficult to dig up and understand
  - knowledge articulation: how do you walk?
- Requirements management: version control, ownership
- Political problems within the organisation
- Balancing functional and usability demands
- Economic and business environment changes during development process.

Some basic guidelines

- Focus on identifying the stakeholders' needs
- Involve all the stakeholder groups
- Involve more than one representative from each stakeholder group
- Use a combination of data gathering techniques

Some basic guidelines

- Support the process with props such as prototypes and task descriptions
- Run a pilot session
- You will need to compromise on the data you collect and the analysis to be done, but before you can make sensible compromises, you need to know what you'd *really* like
- Consider carefully how to record the data
- Start soon after data gathering session

Task descriptions

- Scenarios
  - an informal narrative story, simple, ‘natural’, personal, not generalisable
- Use cases
  - assume interaction with a system
  - assume detailed understanding of the interaction
- Essential use cases
  - abstract away from the details
  - does not have the same assumptions as use cases
- Task models
  - Detailed breakdowns of steps involved
  - Should avoid committing to implementation details

Scenarios

- Scenarios are stories for design: rich stories of interaction.
  - communicate with others
  - validate other models
  - understand dynamics
- Can be short “user intends to press *save*, but accidentally presses *quit*, and loses his work”.
- Can focus on describing a situation, or detailed context.

Scenarios

- Include:
  - Actors
  - Background information about actors
  - Assumptions about their environment
  - Goals and objectives
  - Sequences of actions and events.
- Shared among stakeholders in system design.
- Useful if circumstances are novel.

Linearity

Scenarios – one linear path through system

Pros:

- life and time are linear
- easy to understand (stories and narrative are natural)
- concrete (errors less likely)

Cons:

- no choice,no branches,no special conditions
- miss the unintended

So:

- use several scenarios
- use several methods

Explore the scenario

- What will users want to do?
- Step-by-step walkthrough
  - what can they see (sketches, screen shots)
  - what do they do (keyboard, mouse etc.)
  - what are they thinking?
- Explore interaction
  - what happens when
- Explore cognition
  - what are the users thinking
- Explore architecture
  - what is happening inside

Scenario

- Andy needs a doctor’s appointment for his young daughter Kirsty in the next week or so. The appointment needs to be outside school-time and Andy’s core working hours, and ideally with Dr Fox, who is the children's specialist. Andy uses a PC and the internet at work, so has no difficulty in accessing the appointments booking system. He logs in and from a series of drop down boxes, chooses to have free times for Dr Fox displayed for the next two weeks….[continued scenario would describe how Andy books the appointment and receives confirmation].

Scenario – from usability.gov

- Mr. and Mrs. Macomb are retired schoolteachers who are now in their 70s. Their Social Security checks are an important part of their income. They've just sold their big house and moved to a small apartment. They know that one of the many chores they need to do now is tell the Social Security Administration that they have moved. They don't know where the nearest Social Security office is and it's getting harder for them to do a lot of walking or driving. If it is easy and safe enough, they would like to use the computer to notify the Social Security Administration of their move. However, they are somewhat nervous about doing a task like this by computer. They never used computers in their jobs. However, their son, Steve, gave them a computer last year, set it up for them, and showed them how to use email and go to websites. They have never been to the Social Security Administration's website, so they don't know how it is organized. Also, they are reluctant to give out personal information online, so they want to know how safe it is to tell the agency about their new address this way.

Exercise

- Consider a collaborative calendar system deployed over handheld devices.
- The system should support arranging appointments and meetings, and dealing with conflicts.
- Some meetings will be urgent. Some will be regular.
- Develop a scenario for a commercial organisation.

Coursework

- Form groups.
- Design a system to be used for job seeking/recruitment. Consider both job seekers and recruiters. The system needs to be appropriately designed for both. You might wish to specialise in one type of employment or industry, or to have support for multiple types in your design.
- Initial steps:
  - Form your team.
  - Choose two educational applications to compare yourself to.
  - Write a number of scenarios covering the main functionality and any of the different roles that might be involved (at least one per team member).

Task analysis

- What are people trying to achieve?
- Why are they trying to achieve it?
- How are they going about it?
  - What people do
  - What things they work with
  - What they must know
- Task descriptions are often used to envision new systems or devices
- Used mainly to investigate an existing situation
- It is important not to focus on superficial activities
- Many techniques, the most popular is Hierarchical Task Analysis (HTA)

Approaches to task analysis

- Task decomposition
  - splitting task into (ordered) subtasks
- Knowledge based techniques
  - what the user knows about the task and how it is organised
- General method:
  - observe
  - collect unstructured lists of words and actions
  - organise using notation or diagrams

An Example

- To prepare plane for landing
  - Lower the landing gear
  - Check the gear is fully deployed
  - Arm the spoilers
  - Set flaps to 25 degrees
  - Set flaps to 40 degrees

- Must know about:
  - landing gear, flaps, spoilers.

Hierarchical Task Analysis

- Involves breaking a task down into subtasks, then sub-sub-tasks and so on. These are grouped as plans which specify how the tasks might be performed in practice
- HTA focuses on physical and observable actions, and includes looking at actions not related to software or an interaction device
- Start with a user goal which is examined and the main tasks for achieving it are identified

Task Decomposition

- Aims:
  - describe the actions people do
  - structure them within task subtask hierarchy
  - describe order of subtasks

- Focus on Hierarchical Task Analysis (HTA)
  - text and diagrams to show hierarchy
  - plans to describe order

Textual HTA description

- 0. in order to prepare for landing (pilot non-flying)
    1. deploy the landing gear
    2. arm the spoilers
        - 2.1. check the door open light
        - 2.2. pull spoiler lever
    3. set the flaps
        - 3.1 set flaps to 25 degrees
        - 3.2 check flaps set to 25 and confirm
        - 3.3 set flaps to 40 degrees
        - 3.4 check flaps set to 40 and confirm
    4. confirm checklist complete

Plan 0: do 1 – 2 and 3 when required - 4

Plan 2: do 2.1 first and then 2.2 when door open light goes out.

Plan 3: do 3.1 when asked to by pilot flying, do 3.1 then 3.2, do 3.3 when asked to by pilot flying, then 3.4.

Tasks and Goals

- Users have goal directed units of activity:
  - have specific problems to solve;
  - have tasks to accomplish;
  - feedback from interface must address this.
- Tasks are complex:
  - few relate solely to the computer;
  - decomposed into sub-tasks.

Supporting user activities

Example: buying a train ticket

- study list of train departures
- mentally note time and platform number of next train
- *stand in line at ticket counter*
- *on reaching counter, state destination and journey type*
- *receive quote for price of ticket*
- *pay money*
- *receive ticket and change*
- walk over to drinks machine

Describing tasks

1. Find time of next train
   a) study list of train departures
   b) mentally note time and platform number of next train
2. Purchase ticket
   a) stand in line at ticket counter
   b) on reaching counter, state destination and journey type
   c) receive quote for price of ticket
   d) pay money
   e) receive ticket and change
3. Obtain cup of coffee
   a) walk over to drinks machine
   b) insert money
   c) press button for black coffee
   d) wait for cup to drop and contents to be poured
   e) remove cup from machine.

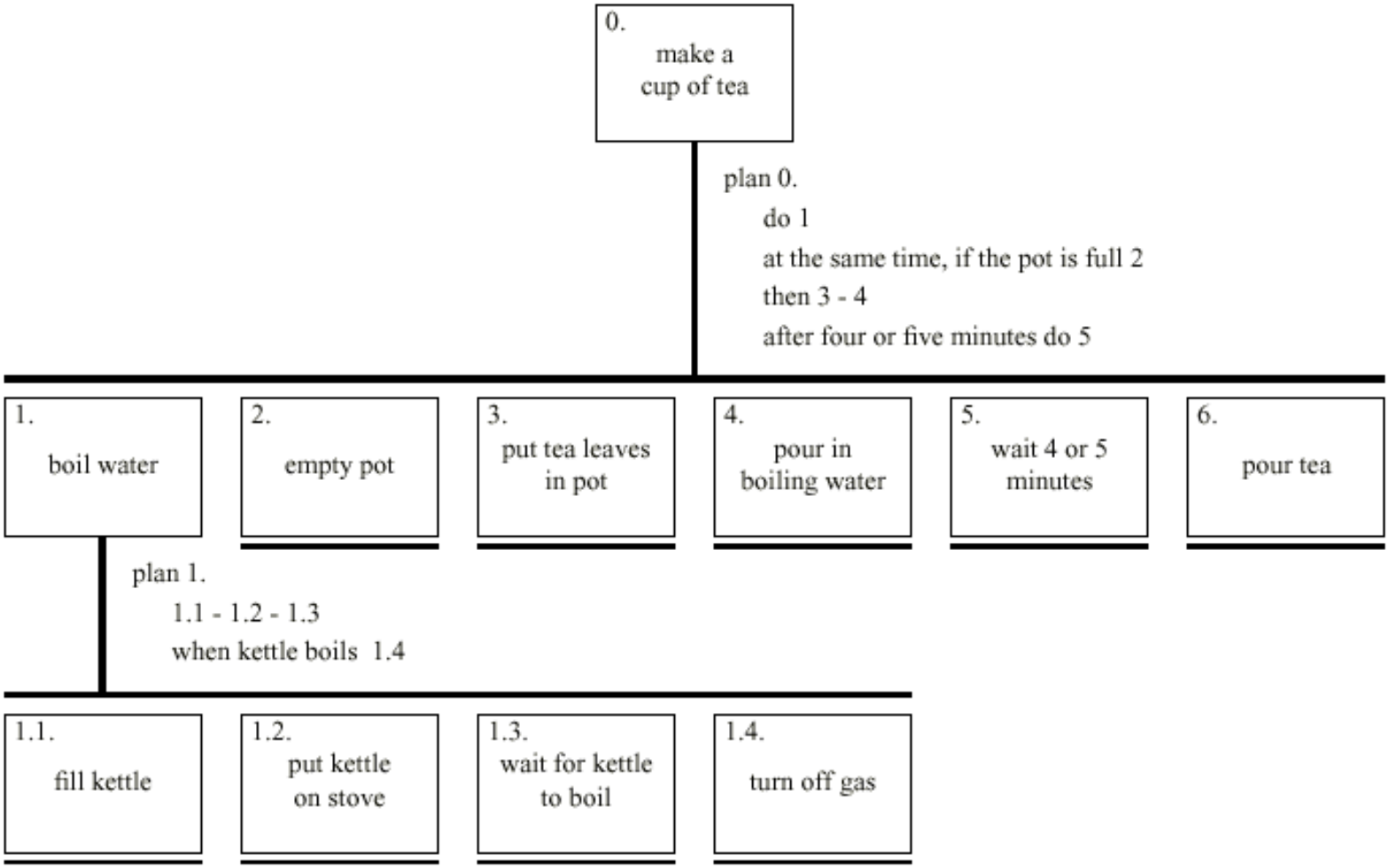
Generating the hierarchy

1. Get list of tasks
2. Group tasks into higher level tasks
3. Decompose lowest level tasks further

Stopping rules - How do we know when to stop?

- Is “empty the dust bag” simple enough?
- Purpose: expand only relevant tasks
- Motor actions: lowest sensible level

Diagrammatic HTA



Refining the description

- Given initial HTA (textual or diagram)
  - How to check/improve it?
- Some heuristics:
  - paired actions

    e.g., where is ‘turn on gas’
  - restructure

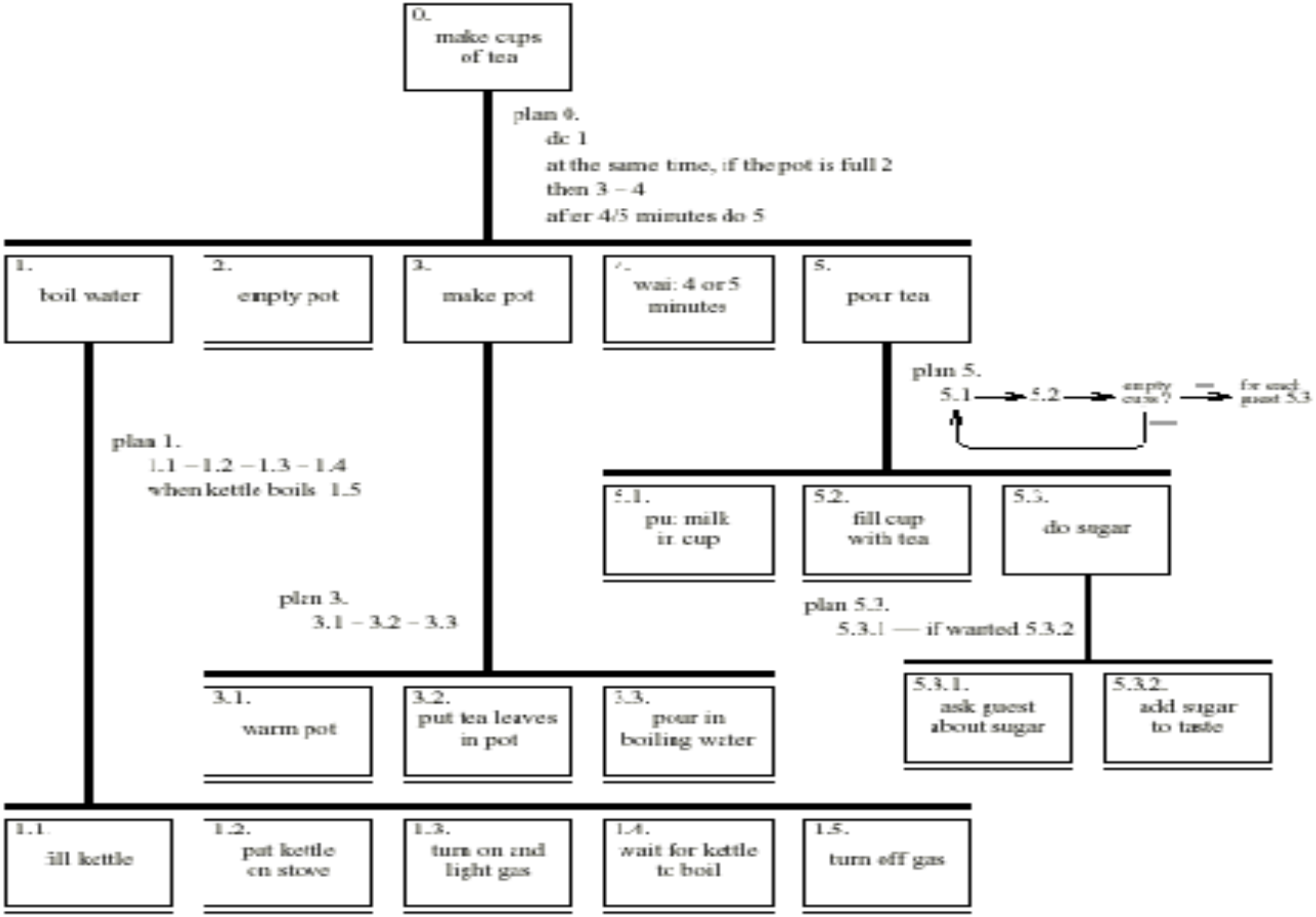
    e.g., generate task ‘make pot’
  - balance

    e.g., is ‘pour tea’ simpler than making pot?
  - generalise

    e.g., make one cup or two

    ….. or more

Refined HTA for making tea



Types of plan

fixed sequence - 1.1 then 1.2 then 1.3

optional tasks - if the pot is full 2

waiting for events - when kettle boils 1.4

cycles - do 5.1 5.2 while there are still empty cups

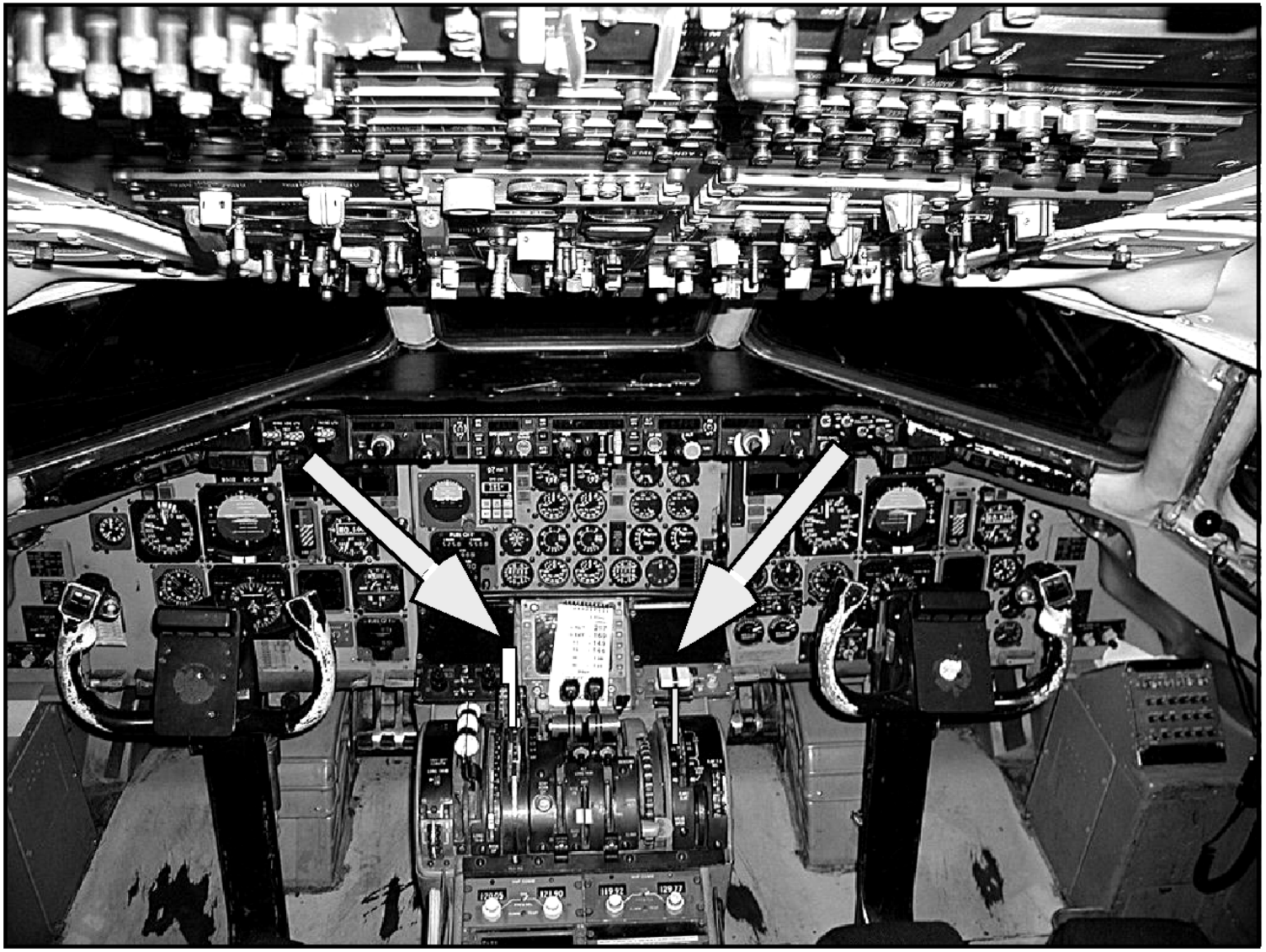
time-sharing - do 1; at the same time ...

discretionary - do any of 3.1, 3.2 or 3.3 in any order

mixtures - most plans involve several of the above

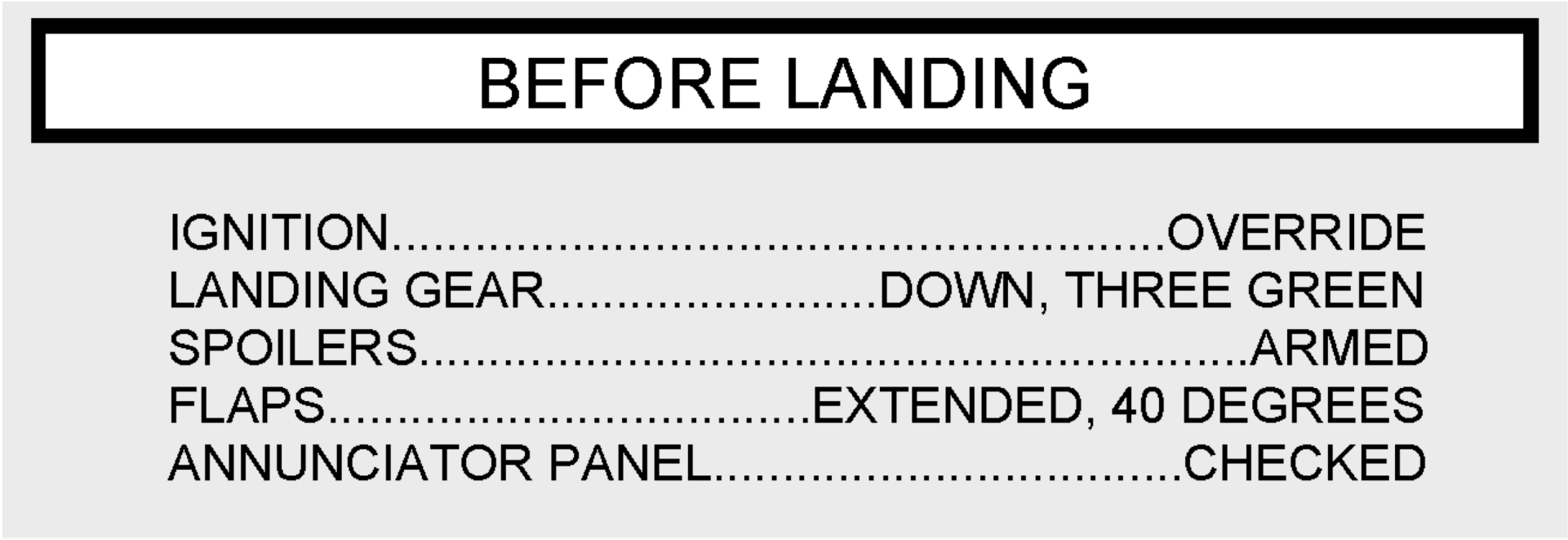
Exercise 2 – Task Analysis

- Now consider the explicit (sub)tasks involved in your scenarios.
- After specifying the tasks, try to produce a more unified Task Model.
- What is the most complex task from the user perspective?
- What feedback and information should be given to the user to support this task?



- The white arrow on the right points to the flap handle and the arrow on the left points to the spoiler level

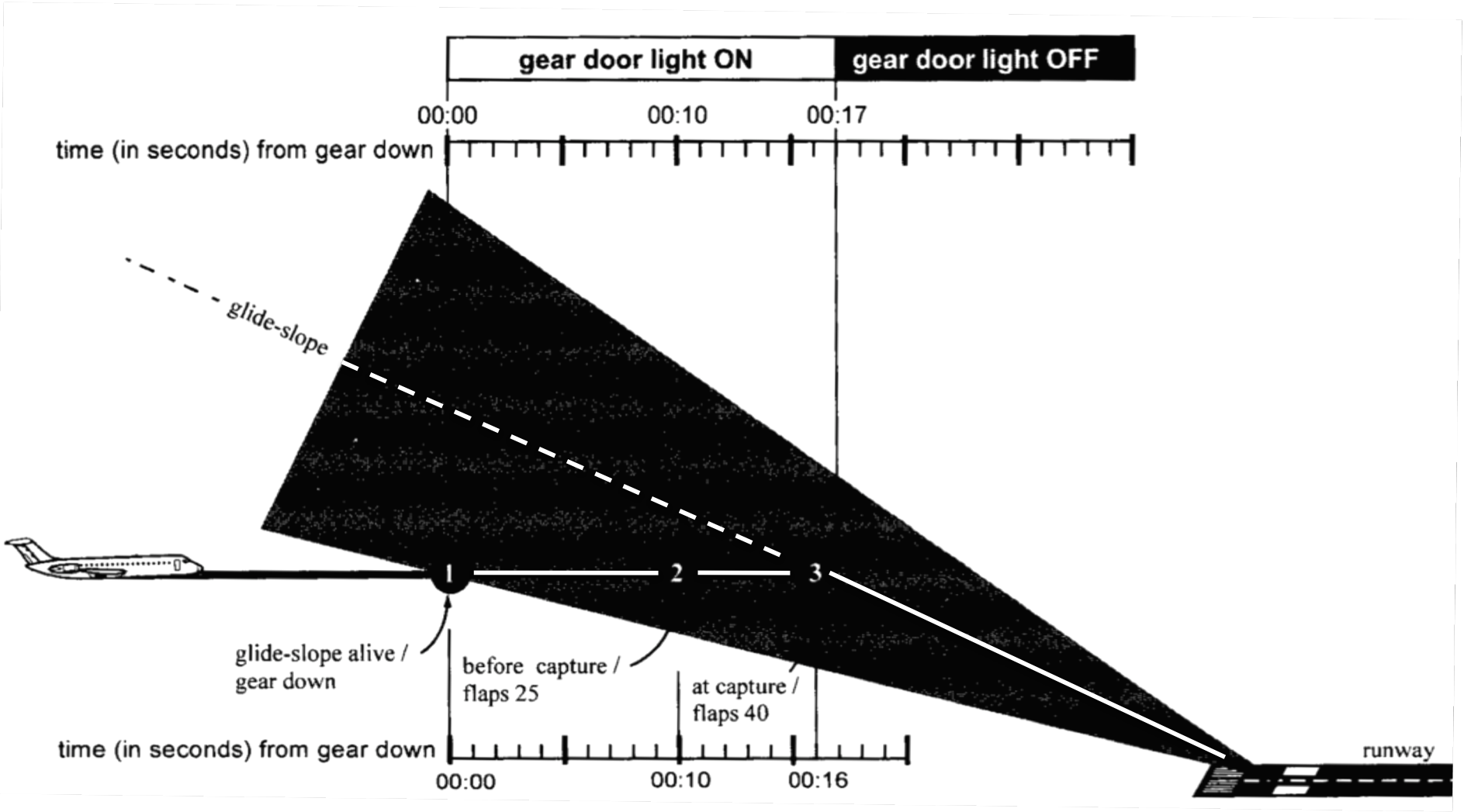
Checklist



BEFORE LANDING

IGNITION.....OVERRIDE
LANDING GEAR.....DOWN, THREE GREEN
SPOILERS.....ARMED
FLAPS.....EXTENDED, 40 DEGREES
ANNUNCIATOR PANEL.....CHECKED

Forms of analysis



From Asaf Degani, Taming HAL, Chapter 13.

Knowledge Based Analyses

- Aim is to capture knowledge needed to perform a task.
- Focus on:
  - Objects - used in task
  - Actions - performed
- Taxonomies represent levels of abstraction
- Assess the amount of common knowledge between different tasks.
- Can also help in production of teaching materials.
- A lot of technology based on ontologies...

Example: parallel programming

- Started with qualitative interview study, interviews transcribed and coded.
- Many codes concerned categories of problem which programmers faced.
- Build model of the problem space, combining qualitative data with domain expertise, existing tools, and literature.
- Put forward 7 categories of problem.
- Identify information which helps in diagnosing these problems.

Taxonomy

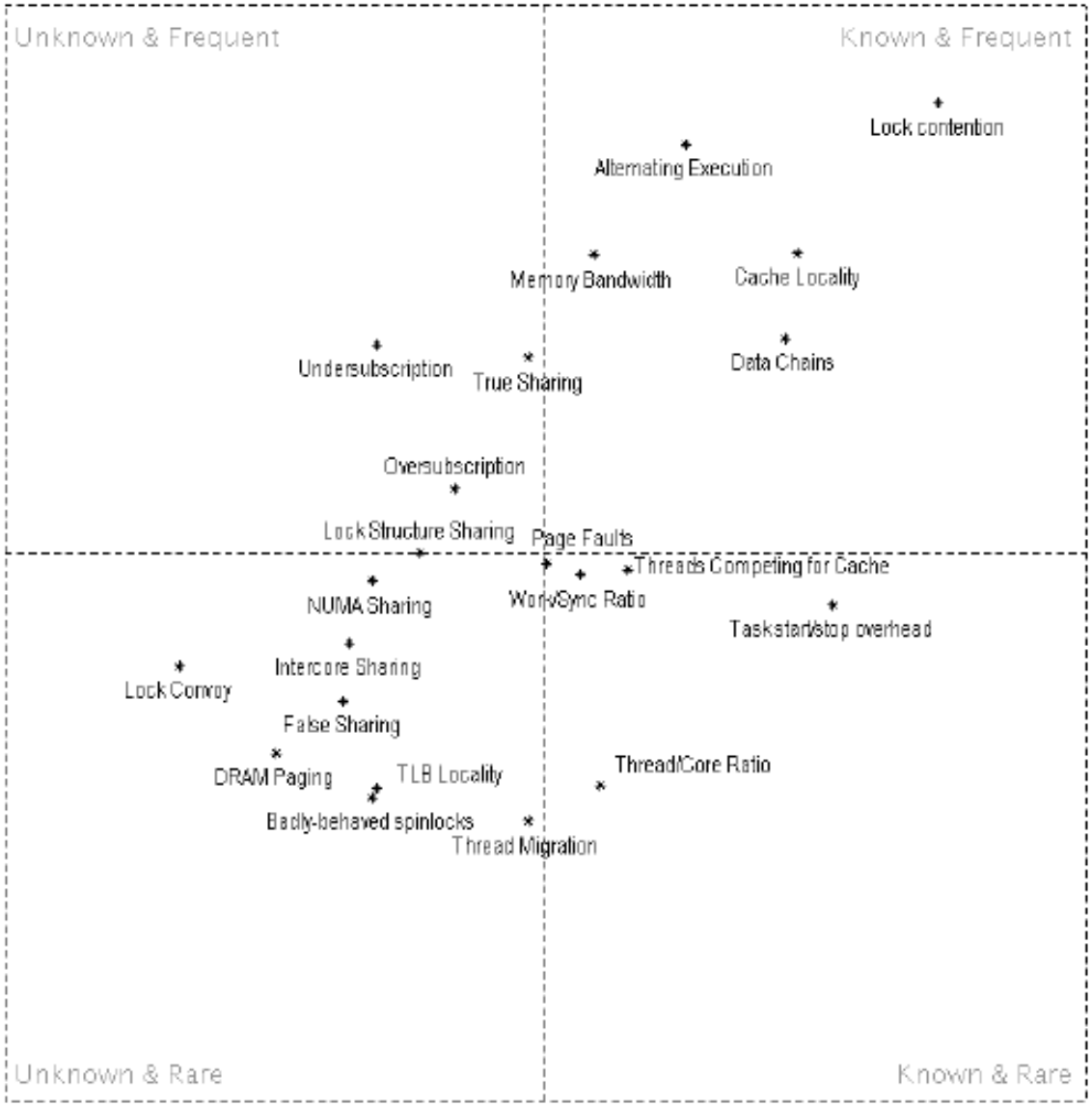
Represents expert knowledge of the domain

Detail of problem space

Starting point for design of tools

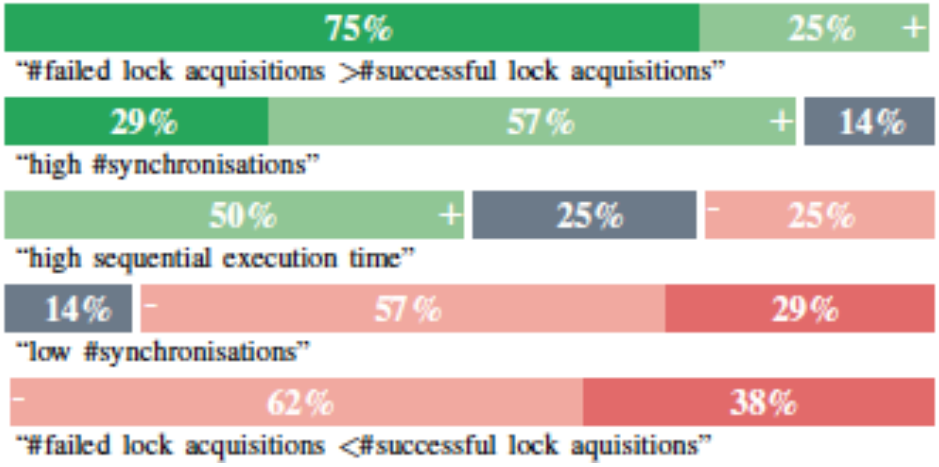
Category	Problem
Task granularity	Oversubscription Task start/stop overhead Thread migration
Synchronisation	Low work to synchronisation ratio Lock contention Lock convoys Badly-behaved spinlocks
Data sharing	True sharing of updated data Data sharing b/w CPUs on NUMA Sharing of lock data structures Sharing data between distant cores
Load balancing	Undersubscription Alternating sequential/parallel exec. Chains of data dependencies Bad threads to cores ratio
Data locality	Poor cache locality Poor TLB locality NUMA memory shared b/w CPUs DRAM memory pages Page faults
Resource sharing	Exceeding memory bandwidth Threads competing for cache False data sharing
Input/output	Shared files Shared disk Shared network connection

How important?



Validity of model

Experts validate individual components of the model



Expert agreement on indicators of lock contention problems

A Richer View of Expert Knowledge

- Expertise for diagnosing & predicting.
- Situation awareness
- Perceptual skills
- Developing and knowing when to apply tricks of the trade
- Improvising
- Recognizing anomalies
- Compensating for equipment limitations

Affordances

- Psychological term
- For physical objects
  - shape and size suggest actions
    - pick up, twist, throw
  - also cultural – buttons ‘afford’ pushing
- For screen objects
  - button-like object ‘affords’ mouse click
  - physical-like objects suggest use
- Culture of computer use
  - icons ‘afford’ clicking
  - or even double clicking … not like real buttons!

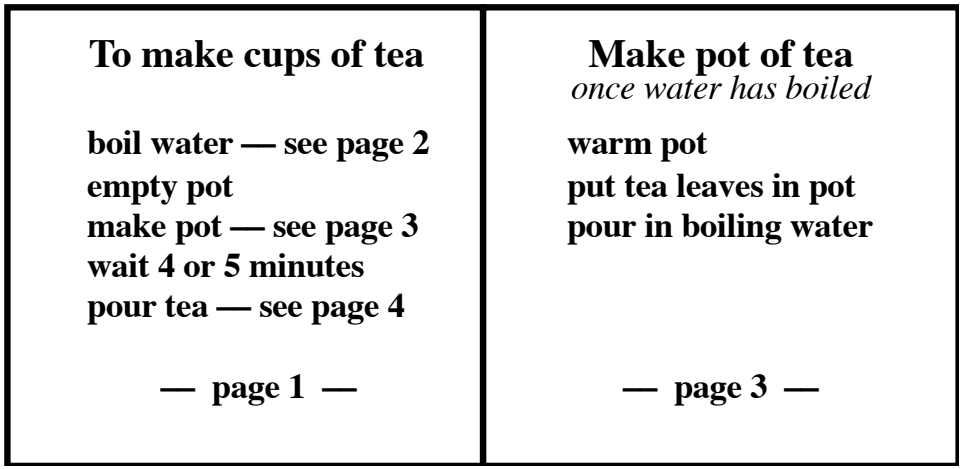
‘affords’ grasping

Affordances

- "A technical term that refers to the properties of objects - what sorts of operations and manipulations can be done to a particular object."
  - In "Psychology of Everyday Things" (POET)
    Originally from Psychologist JJ Gibson
- Important for GUI: Perceived Affordance
  - "what a person thinks can be done with an object."

Manuals and Documentation

- Conceptual Manual
  - from knowledge based analysis
  - good for open ended tasks
- Procedural ‘How to do it’ Manual
  - from HTA description
  - good for novices
  - assumes all tasks known



To make cups of tea

boil water — see page 2
empty pot
make pot — see page 3
wait 4 or 5 minutes
pour tea — see page 4

— page 1 —

Make pot of tea
once water has boiled

warm pot
put tea leaves in pot
pour in boiling water

— page 3 —

Uses of Task Analysis II

Requirements capture and systems design

- lifts focus from system to use
- suggests candidates for automation
- uncovers user's conceptual model

- Detailed interface design
  - taxonomies suggest menu layout
  - object/action lists suggest interface objects
  - task frequency guides default choices
  - existing task sequences guide dialogue design
- Task analysis is never complete/comprehensive
  - rigid task based design $\Rightarrow$ inflexible system

Summary

- Getting requirements right is crucial
- There are different kinds of requirement, each is significant for interaction design
- The most commonly-used techniques for data gathering are: questionnaires, interviews, focus groups, direct observation, studying documentation and researching similar products
- Scenarios, use cases and essential use cases can be used to articulate existing and envisioned work practices.
- Task analysis techniques such as HTA help to investigate existing systems and practices