Requirements Specification

Lectures 4&5a, DAT230, Requirements Engineering
Robert Feldt, 2010-09-08 & 2010-09-14

Notes about course

- Individual assignment 1:
    - Yes, it has personality tests in there
    - Yes, I should have explained why/how more clearly, purpose is:
        - Research: Can personality factors explain (group) performance?
        - Pedagogical: SE is more than technology
        - Practical: Personality testing is reality in modern workplaces
        - Real-world: Compare your to industrial SW Engineers
    - You will get your own results and can compare with norms
    - Never used for grading! Never connected to your name!

Notes about course

- Mandatory guest lecture on Friday 10th of February
    - Room Babord, House Äran, Chalmers Lindholmen
    - Map: http://www.chalmers.se/HyperText/karta_lindholmen.pdf
- Individual assignment 2:
    - Available on course home page later today
    - Questions on SEMAT based on guest lecture
    - MAX 3 pages PDF in IEEE format, Submit through Fire
    - Deadline: 16/9 09:00

Notes about course

- This weeks exercise, IEEE 830 SRS example
  - Either go today (15:15, EB) or tomorrow (15:15, EE)
  - Not BOTH, they are the same since you are many

Recap from last lecture

Recap

- Elicitation to find/gather/create/refine/specify reqs & understand stakeholder needs
- Many different elicitation techniques
  - Interviews, Group sessions, Observation are key
  - Always: care, be human, listen, focus on them, glossary
- Other sources: Docs, Strategies, Problem domain, History, Competitors, Environment
- Different abstraction levels
- Structured interview more powerful than open-ended

Elicitation methods

Elicitation methods

Interviews

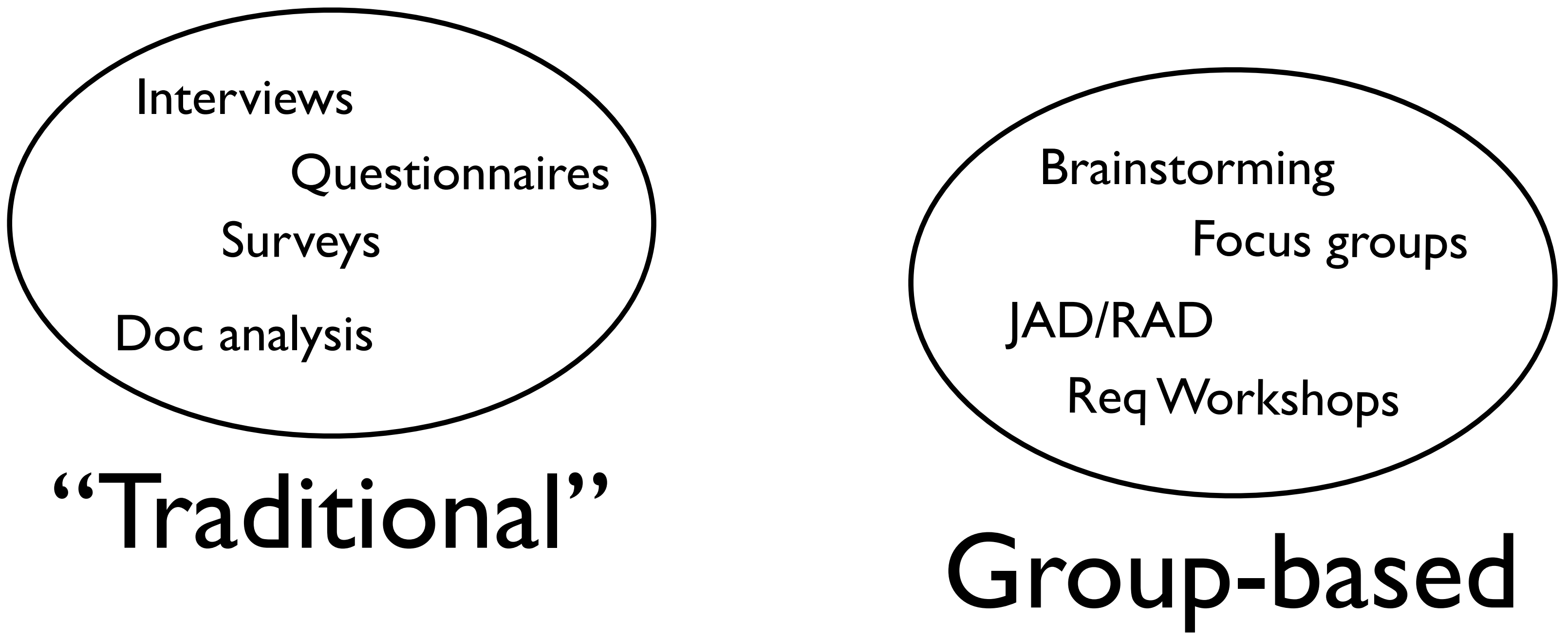
Questionnaires

Surveys

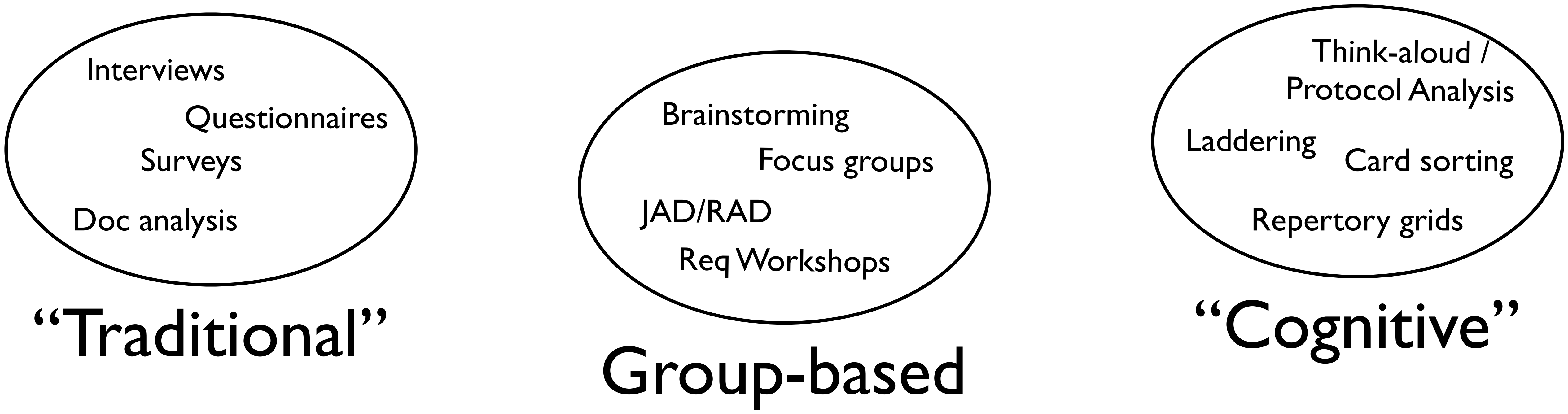
Doc analysis

“Traditional”

Elicitation methods



Elicitation methods



Elicitation methods

Interviews
Questionnaires
Surveys
Doc analysis

“Traditional”

Brainstorming
Focus groups
JAD/RAD
Req Workshops

Group-based

Think-aloud /
Protocol Analysis
Laddering
Card sorting
Repertory grids

“Cognitive”

Ethnography
Observation
Conversation analysis

Contextual

Elicitation methods

Interviews
Questionnaires
Surveys
Doc analysis

“Traditional”

Brainstorming
Focus groups
JAD/RAD
Req Workshops

Group-based

Think-aloud / Protocol Analysis
Laddering
Card sorting
Repertory grids

“Cognitive”

Ethnography
Observation
Conversation analysis

Contextual

KAOS
I*
CREWS

Model-driven

Elicitation methods

“Traditional”
- Interviews
- Questionnaires
- Surveys
- Doc analysis

Group-based
- Brainstorming
- Focus groups
- JAD/RAD
- Req Workshops

“Cognitive”
- Think-aloud / Protocol Analysis
- Laddering
- Card sorting
- Repertory grids

Contextual
- Ethnography
- Observation
- Conversation analysis

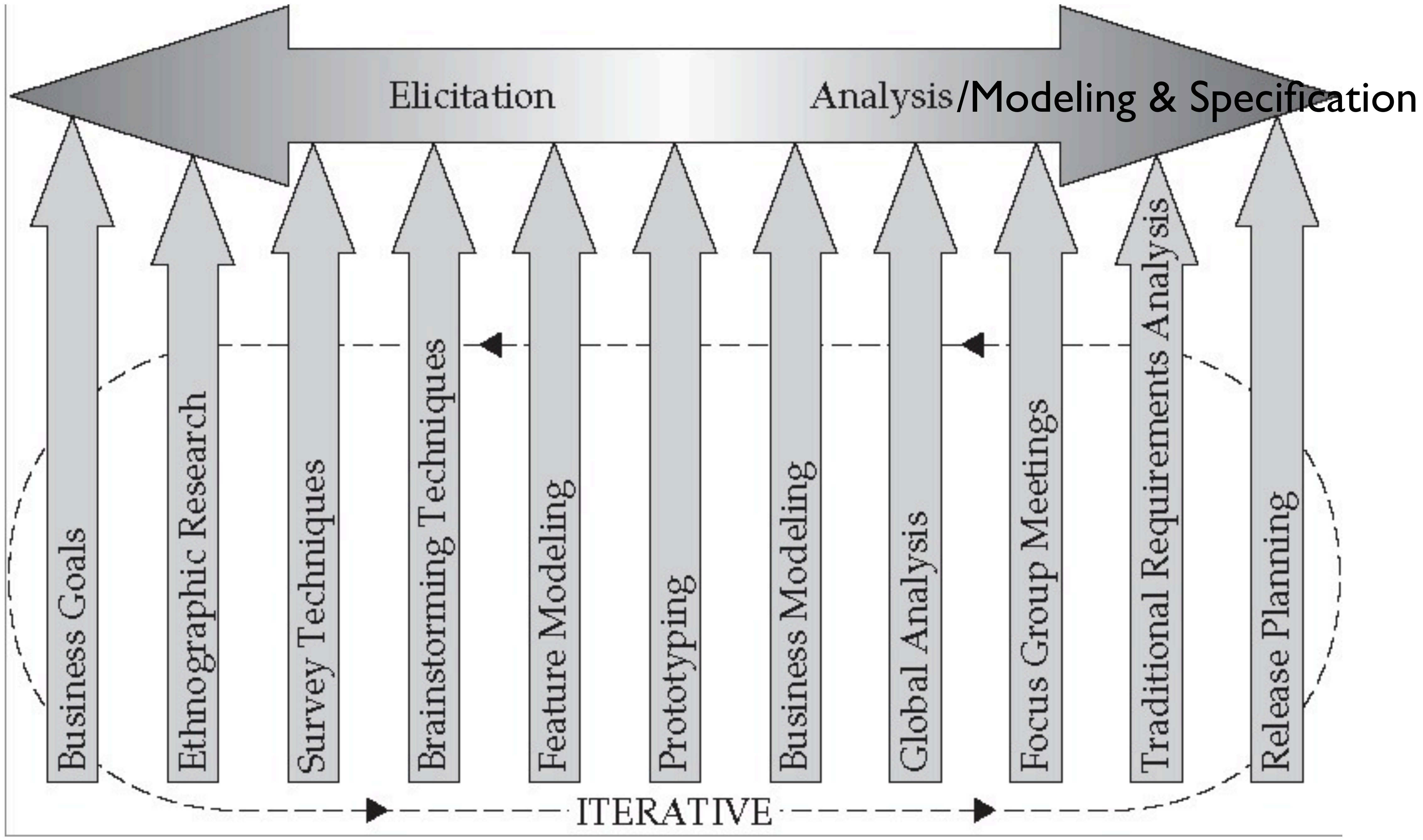
Model-driven
- KAOS
- I*
- CREWS

Prototyping
- Working prototypes
- Mashups
- Drawings

A question to ponder:

Can you think of an elicitation situation where you would choose to start elicitation with hand-drawn UI sketches or is that never good early?

A continuum


Elicitation
Analysis/Modeling & Specification
Business Goals
Ethnographic Research
Survey Techniques
Brainstorming Techniques
Feature Modeling
Prototyping
Business Modeling
Global Analysis
Focus Group Meetings
Traditional Requirements Analysis
Release Planning
ITERATIVE

What is Req Modeling?

What is Req Modeling?

“The construction of
abstract descriptions of
reqs/goals/systems/behavior”

What is Req Modeling?

"The construction of
abstract descriptions of
reqs/goals/systems/behavior"

Used in several RE activities:
elicitation, analysis, specification

What is Req Specification?

What is Req Specification?

"The deliberate documentation of requirements to a degree that makes the associated risks tolerable"

What is Req Specification?

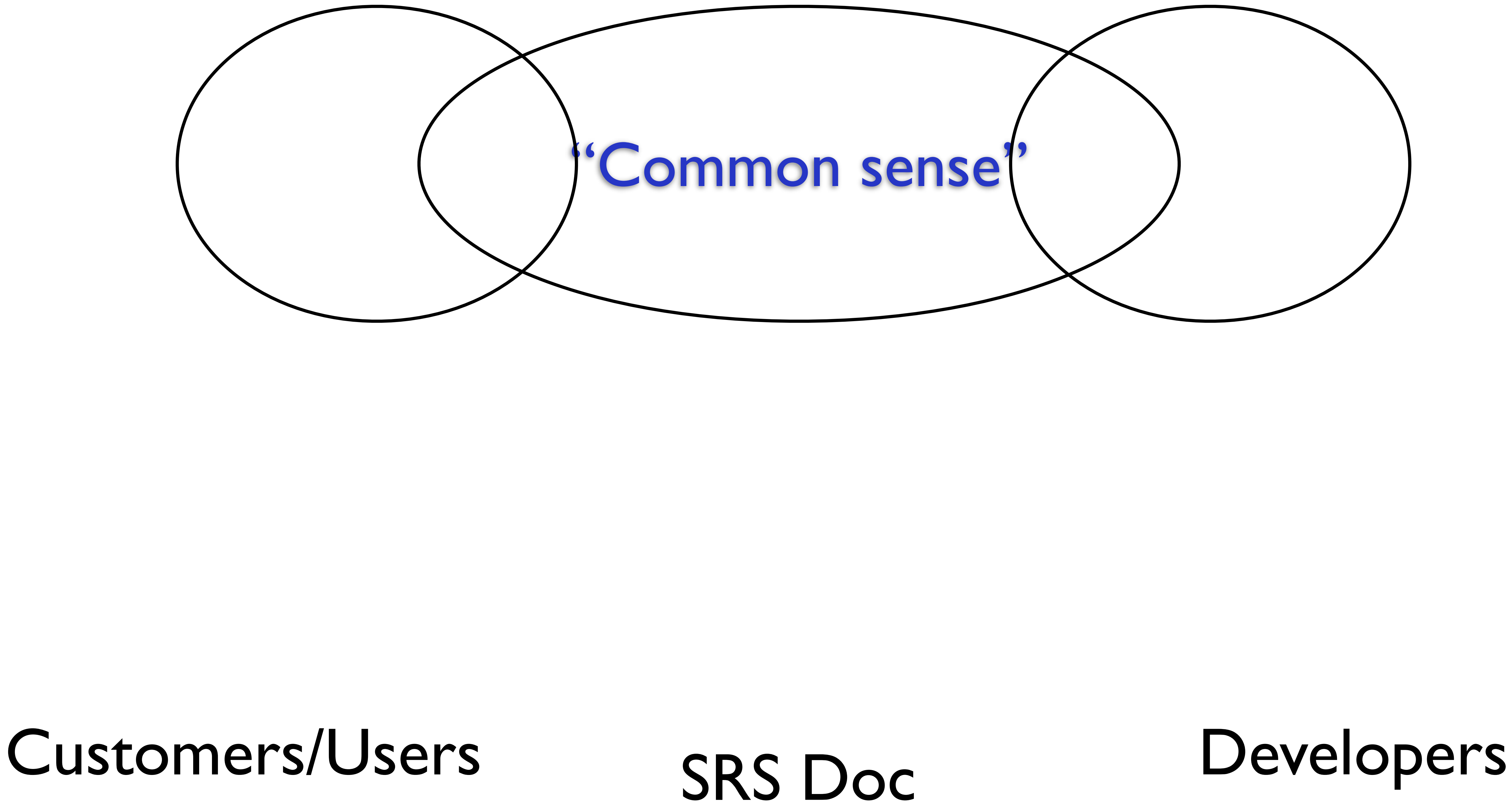
“The deliberate documentation of requirements to a degree that makes the associated risks tolerable”

i.e. writing requirements down in a form so that we avoid later problems

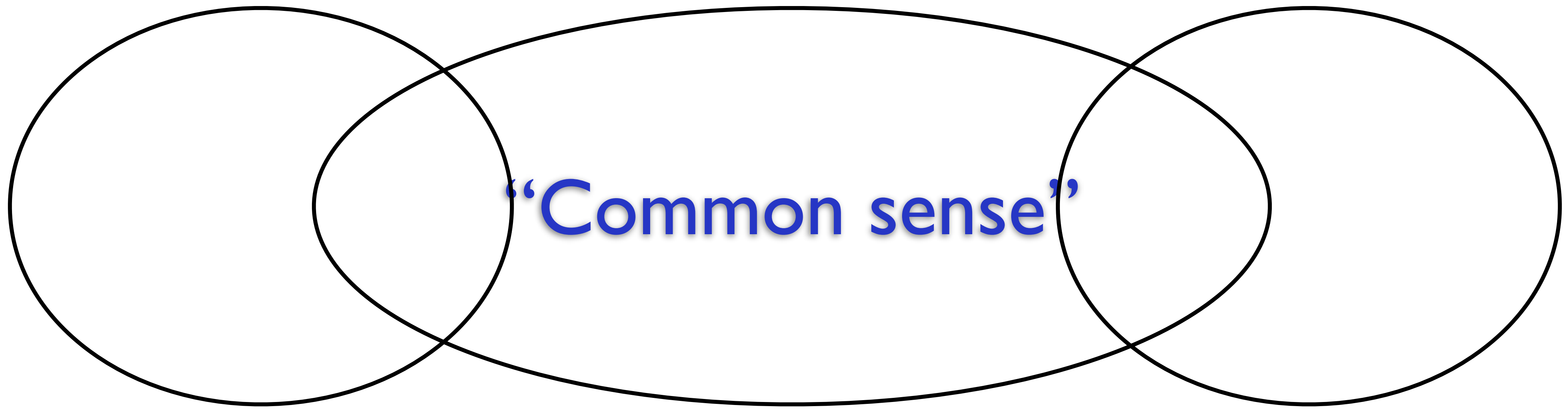
What are risks without doc?

- Reqs still ambiguous & open-ended after elicitation =>
- Developers make decisions/assumptions later =>
- User <-> Dev difference: User not satisfied
- Dev <-> Dev difference: Inconsistent system
- Overall: Costs high!
- BUT:
  - Goal is ideal PRODUCT not ideal Req Doc!
  - Thus: Just enough Req Spec to reduce Risks!

Cost-effectiveness



Cost-effectiveness

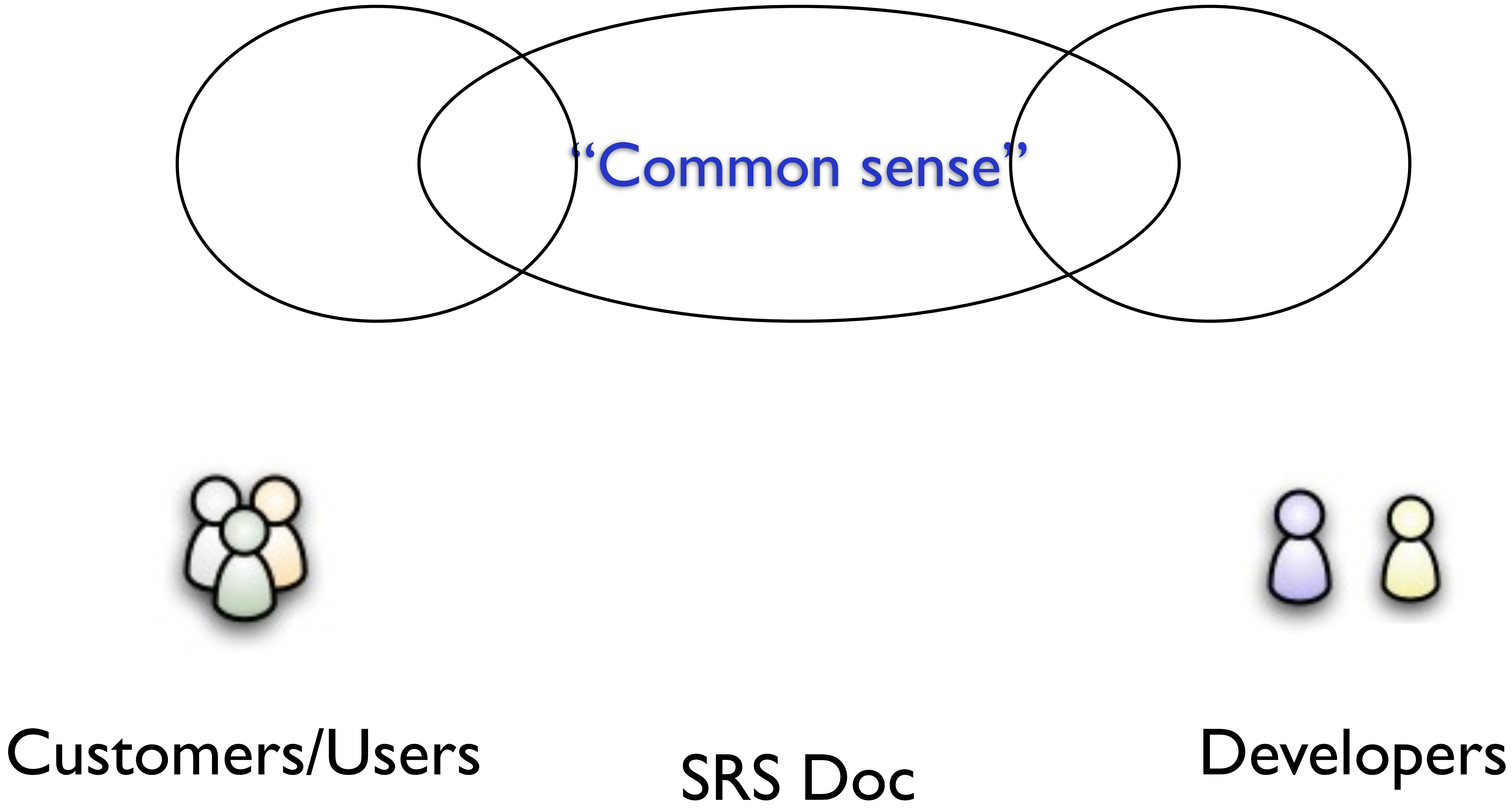


Customers/Users

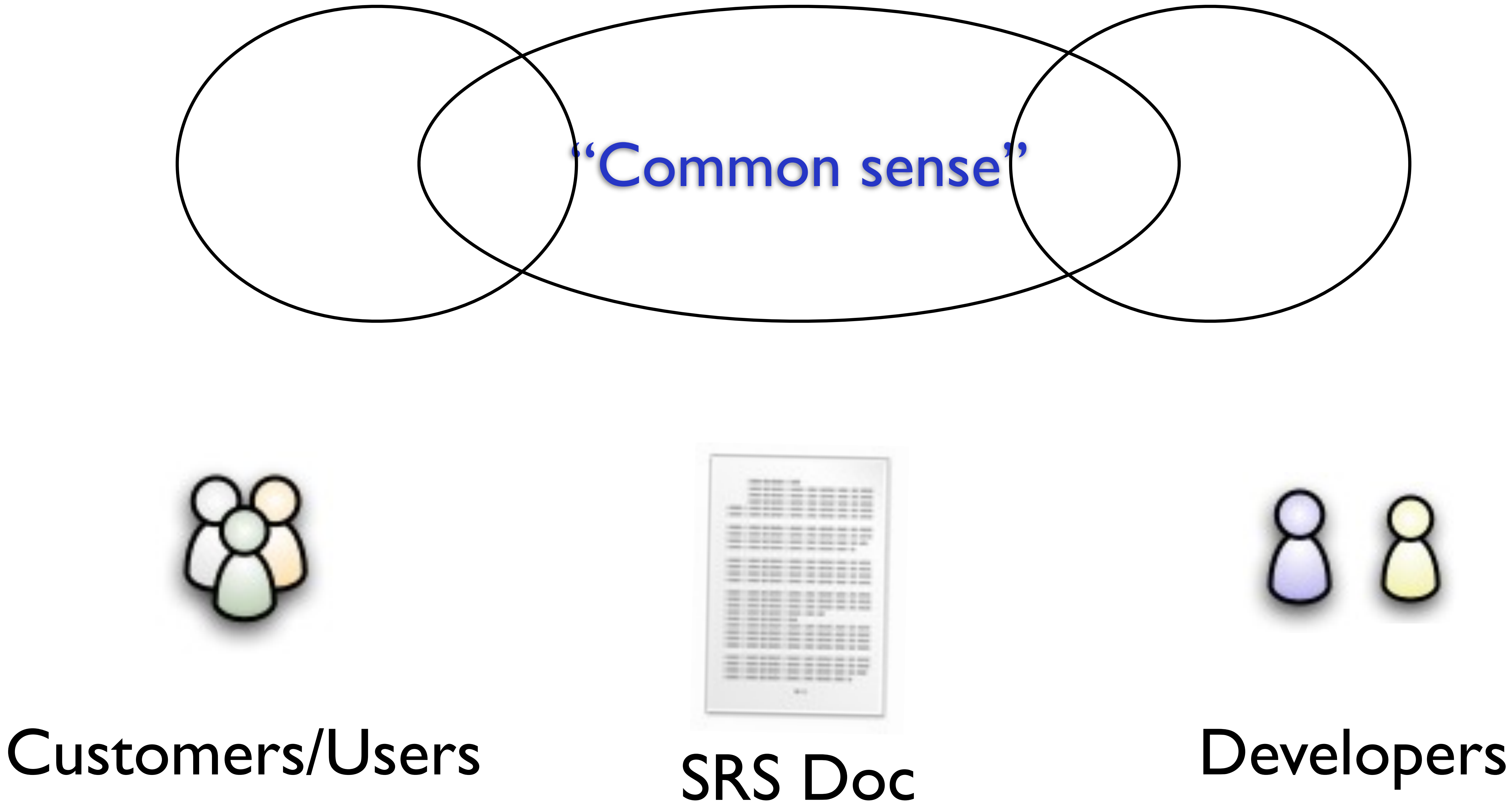
SRS Doc

Developers

Cost-effectiveness



Cost-effectiveness



Roles of Req Doc

- Communication device between all parties
  - Customers, Marketing, Sales, Finance, Management, Devs, Testers
- Drives design and choices
- Drives testing
- Drives project management
- Basis for evolution / releases

Specification Techniques

Word doc
Excel doc
DB / Req tool

Text

Scenario
Use case
Storyboard
Stimulus-response sequence

Interaction- / Sequence-based

State transition diagram
UML state diagram

State-based

Decision tables
Decision trees

Decision-based

CSP
Z
VDM
Property-based

Formal

PLanguage
Volere
Probabilistic Quality Patterns

Quality Requirements

UI standards
Text
Prototype
Sketches
Look'n'feel samples

User Interfaces

Selecting techniques

- Stakeholders must understand => Natural Language
- Models where NatLang has risks:
    - Complex interactions/sequences/states/decisions
    - Interfaces
    - BUT not "One model to rule them all!"
- Quality requirements:
    - Quantify
    - Capture in structured english or PLanguage

Industrial survey: Methods for ReqEng?

Uses...	**“Yes”**
Reviews of requirements	63.8%
Model-based development	25.0%
Prototype-based development	24.3%
Prioritization of reqs	23.7%
Personas for req elicitation	20.4%
UML	17.8%
Modeling/formalisms for reqs	11.8%
Software Product Lines	5.9%

152 answers from Swedish industry, Spring 2009

Tool for Req Eng work?

Svarade	**Andel**
Office (Word, Excel, Visio)	23.8%
None	15.3%
Requisite Pro	10.2%
Quality Center	9.6%
Don’t know	5.1%
Focal Point / DOORS	4.0%
Caliber	3.4%
Customer-specific	3.4%
RSA	3.4%
Clear Case	3.4%
Req Test	3.4%
Rest / Other (max 2 mentions per tool)	18.6%

177 tools mentioned in total

IEEE 830

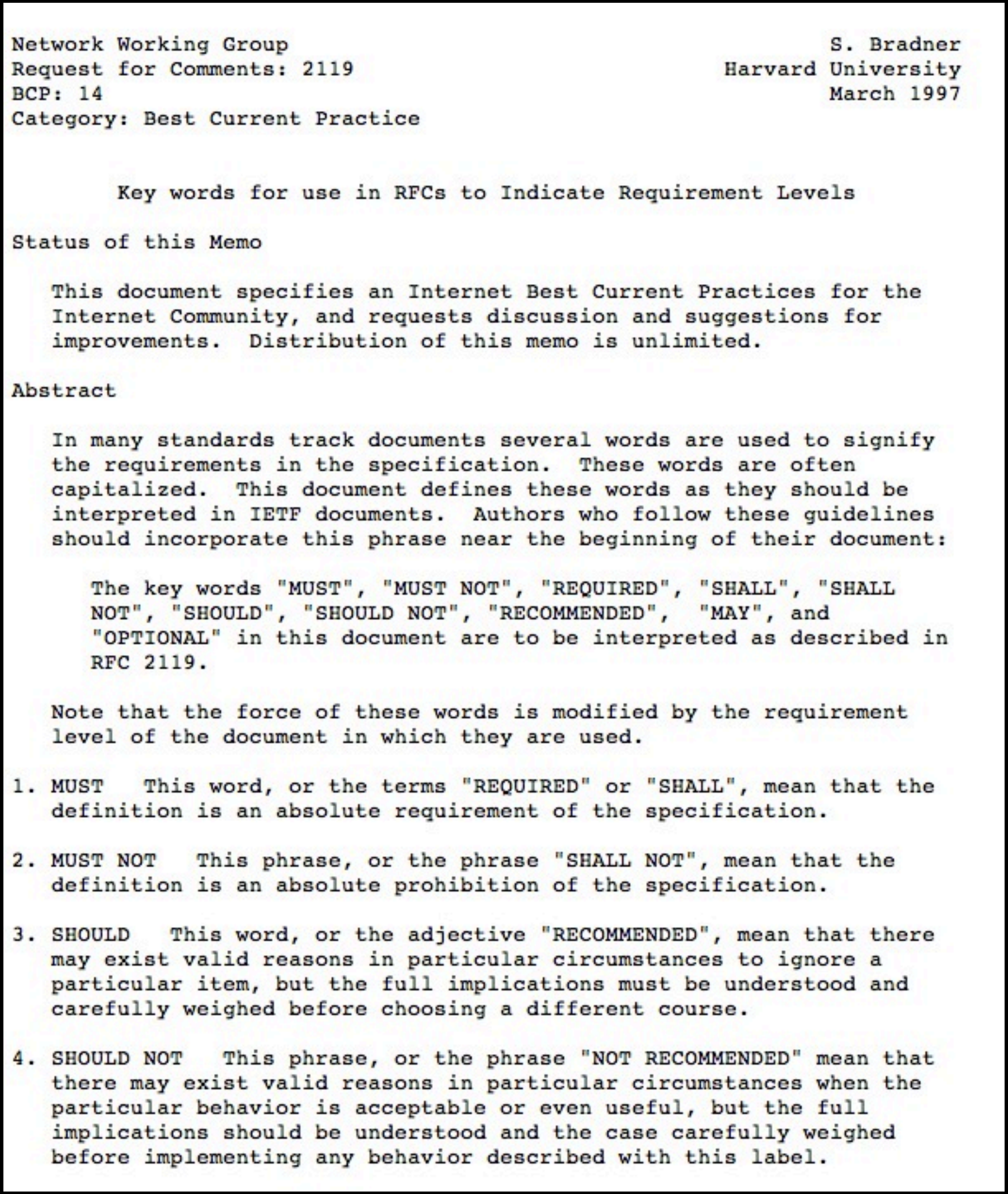
- Recommended practice for SRS
- “Avoid design and project reqs in SRS” (often not followed)
- “No design or implementation details”
- “No nomenclature specified”
  - “NatLang ambiguous => always independent review”
  - “Req specification languages are time-consuming & customers seldomly understand them”
  - In practice: always NatLang + some models/diagrams + maybe use cases / scenarios

SRS has high quality if

- Complete
- Correct
- Feasible
- Necessary
- Prioritized
- Unambiguous
- Verifiable
- In-line with business goals
- Traceable
- Not Design! Not Combined Reqs! Not Redundant!

Language

- Use complete sentences! Use correct grammar & spelling!
- Keep sentences short
- Use Active Voice
- User Terms Consistently
- State requirements in a consistent fashion
  - ex: "The [actor] shall [action verb] [observable result]"
  - "The door management system shall display all users that have exited the building in the past 48 hours"
- Avoid Vague Terms. Avoid Comparative Words.
- RFC 2119



Network Working Group
Request for Comments: 2119
BCP: 14
Category: Best Current Practice

S. Bradner
Harvard University
March 1997

Key words for use in RFCs to Indicate Requirement Levels

Status of this Memo

This document specifies an Internet Best Current Practices for the Internet Community, and requests discussion and suggestions for improvements. Distribution of this memo is unlimited.

Abstract

In many standards track documents several words are used to signify the requirements in the specification. These words are often capitalized. This document defines these words as they should be interpreted in IETF documents. Authors who follow these guidelines should incorporate this phrase near the beginning of their document:

> The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Note that the force of these words is modified by the requirement level of the document in which they are used.

1. MUST This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.

2. MUST NOT This phrase, or the phrase "SHALL NOT", mean that the definition is an absolute prohibition of the specification.

3. SHOULD This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.

4. SHOULD NOT This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

RFC 2119

- MUST = REQUIRED = SHALL
  - Absolute requirement of a specification
  - MUST NOT / SHALL NOT: Absolute prohibition
- SHOULD = RECOMMENDED
  - May exist valid reasons to ignore in particular circumstances, but the full implications must be understood
  - SHOULD NOT / NOT RECOMMENDED
- MAY = OPTIONAL
  - item is truly optional

IEEE 830 SRS Outline

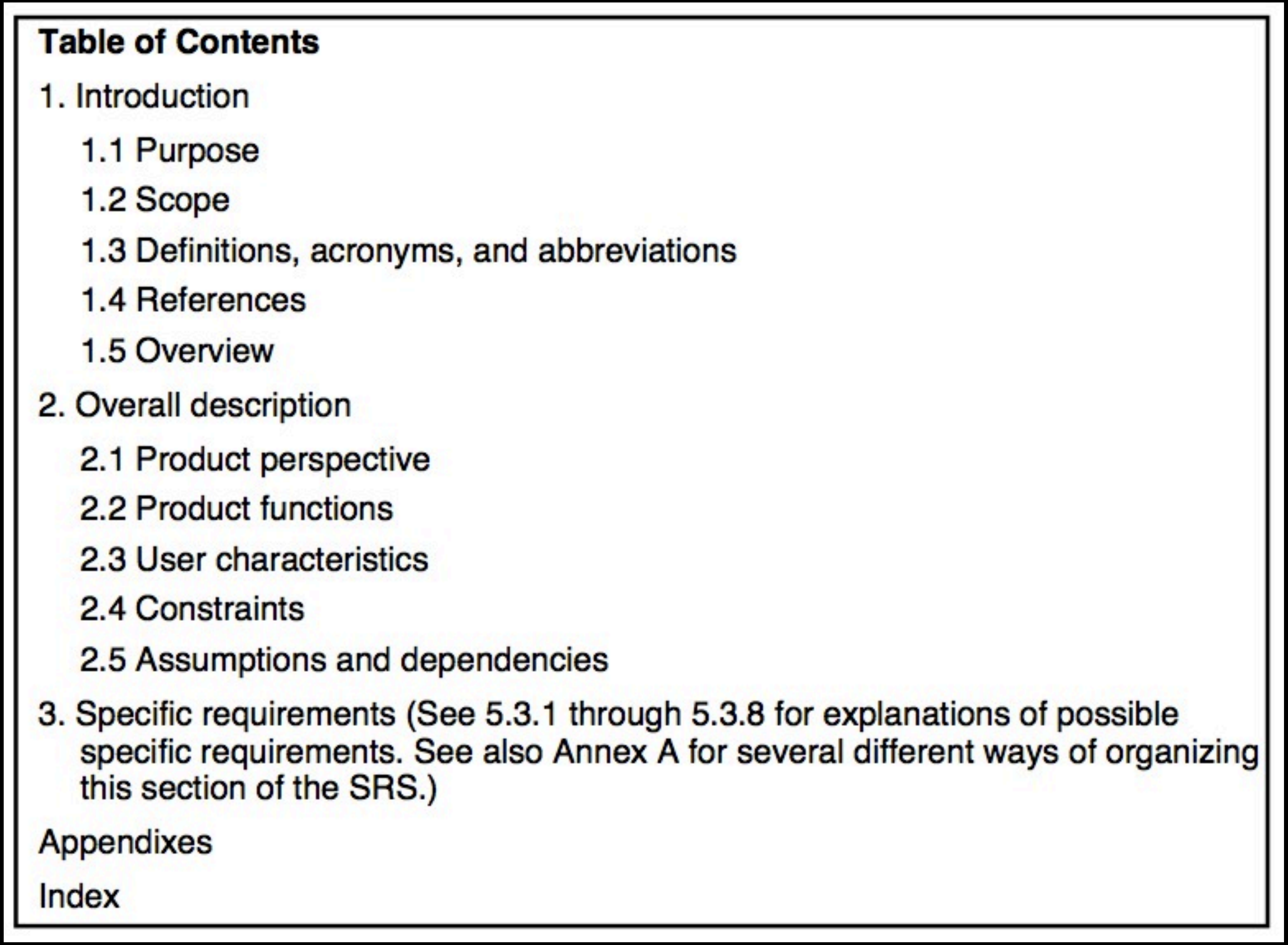


Table of Contents

IEEE 830: 3. Specific Reqs

- 3.1 External interfaces
- 3.2 Functions
- 3.3 Software Systems Attributes
  - Reliability, Availability, Security, Maintainability, Portability

- Performance requirements
- Logical database requirements (schemas etc)
- Design constraints
- Standards compliance

IEEE 830: Supporting info

- 4. Index
- Appendices
  - A.1 Glossary
  - ...

IEEE 830: Example of section 3

- 3.2.1 System feature X
  - 1. Id, Description, Priority
  - 2. Stimulus/response sequence
  - 3. Functional requirements
    - Functional req 1
    - Functional req 2

- Organise by mode, user class, object, stimulus, functional hierarchy, or your own relevant combination

IEEE 830: Alternatives for section 3

- Natural language (as above)
- Use Cases, Sequence-based, ...
- I* (I-star, goal-driven methodology)
- Formal languages
- Decision tables, State diagram, Graphical languages, ...
- i.e. any technique from map above (or combination of techniques)
- if combination is used: give overview and explain why this choice

Standard UML Use cases

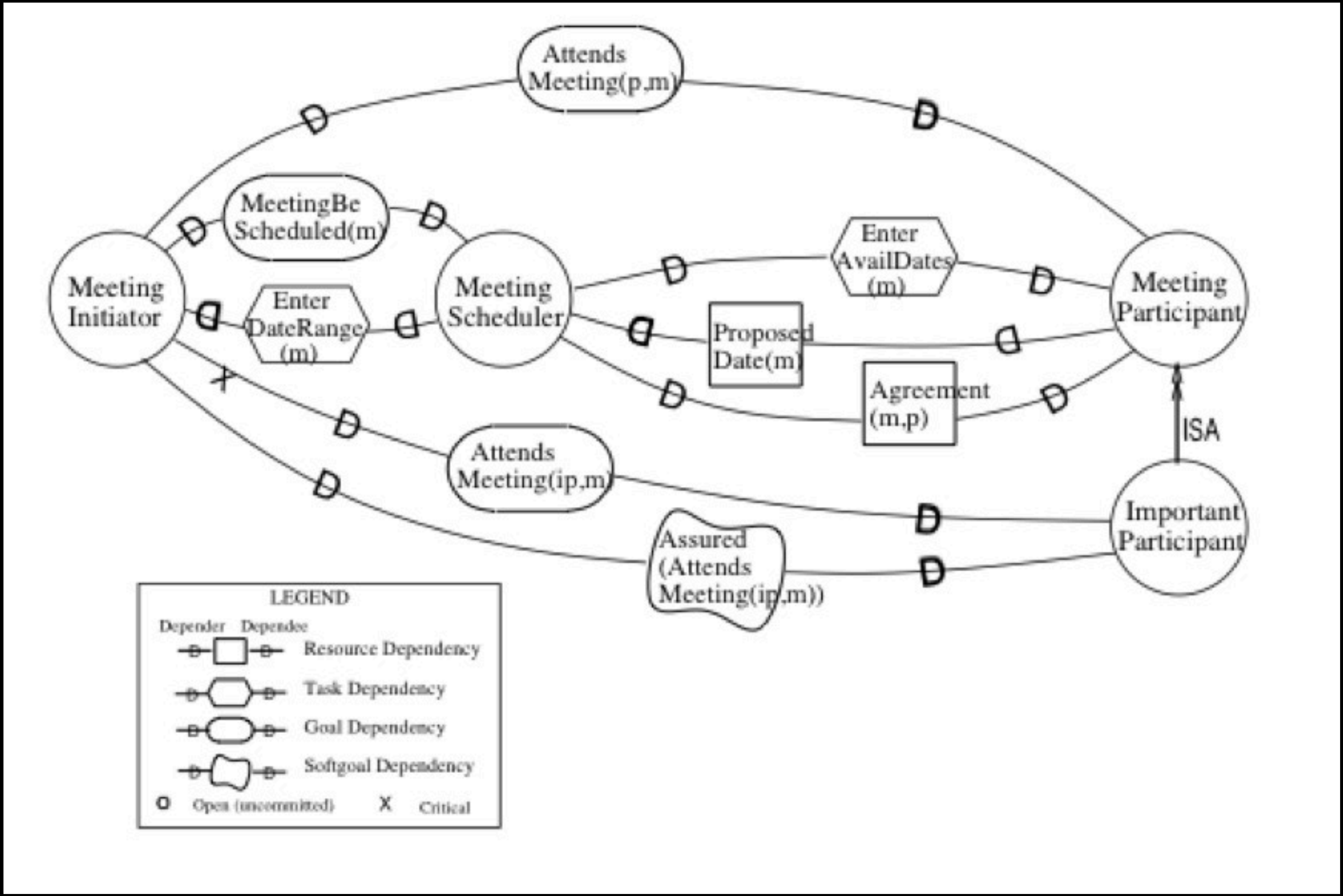
Enter Building (ID: UC3)
Description: A user enters the building
Pre: The person is a user in the system
Post: Person has entered the building

User Intention	**System Response**
1. User swipes magnetic card	2. Verifies that card is valid 3. Asks for user code
4. User enters the code	5. Verifies that code is valid for swiped card 6. Opens door 7. Sound buzzer
8. User opens door & enters building	9. Logs entry of user

I*

- http://www.cs.toronto.edu/km/istar/
- Models Agents and their Intentions
- Early Req Specification together with Customers
- 1. Strategic Dependency Model
    - Actors and Dependencies
    - Certain Actions performed by certain Actors
    - Ex: User depends on system to open door to meet goal to enter building
- 2. Strategic Rationale Model
  - Looks inside actors, what drives them

I* example


Attends Meeting(p,m)
MeetingBe Scheduled(m)
Meeting Initiator
Enter DateRange (m)
Meeting Scheduler
Enter AvailDates (m)
Proposed Date(m)
Agreement (m,p)
Meeting Participant
ISA
Attends Meeting(ip,m)
Assured (Attends Meeting(ip,m))
Important Participant
D
X
LEGEND
Depender
Dependee
Resource Dependency
Task Dependency
Goal Dependency
Softgoal Dependency
Open (uncommitted)
Critical

Formal languages: Z

- Mathematical language for describing computing system
- Model-based, models abstract data type (ADT)
- ADT = system state and operations on it
    - State = state variables and their values
    - Operation = can change state
- Good match to imperative programming languages
- Also extension for OO languages; form of inheritance
- Very mature, used since 1970's

State Transition Diagram (Z example)

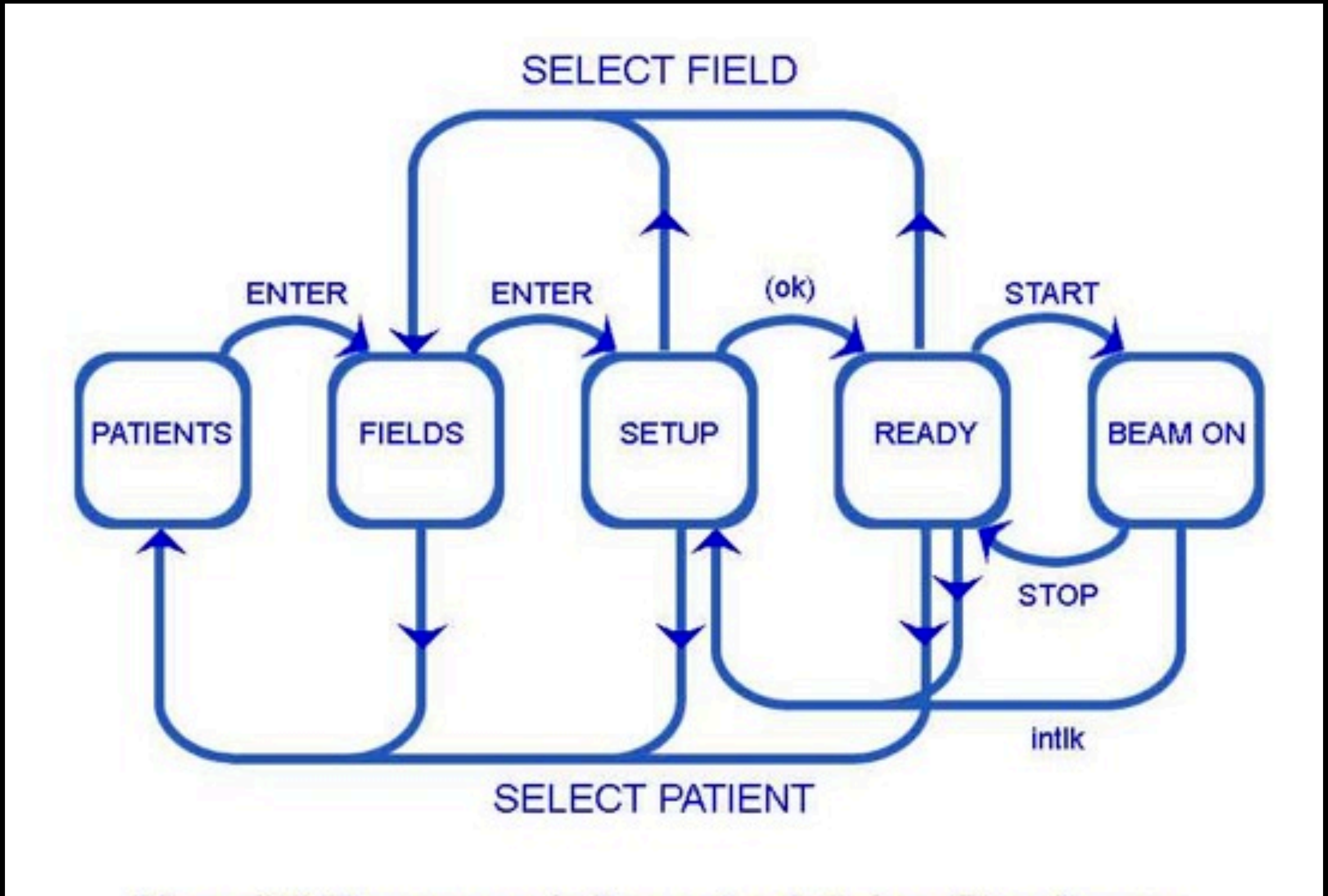
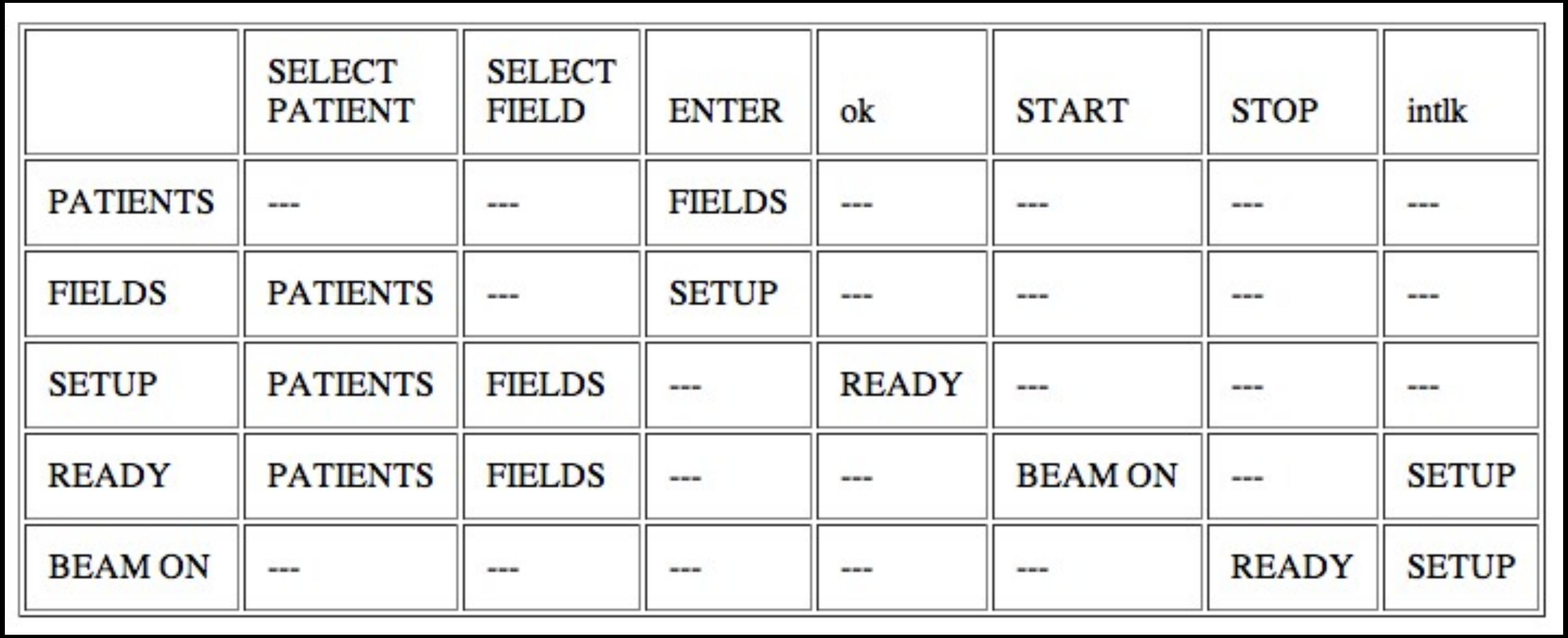


Figure 6.6: Therapy control cascade: state transition diagram

Graphic courtesy of kimberlybatteau.com

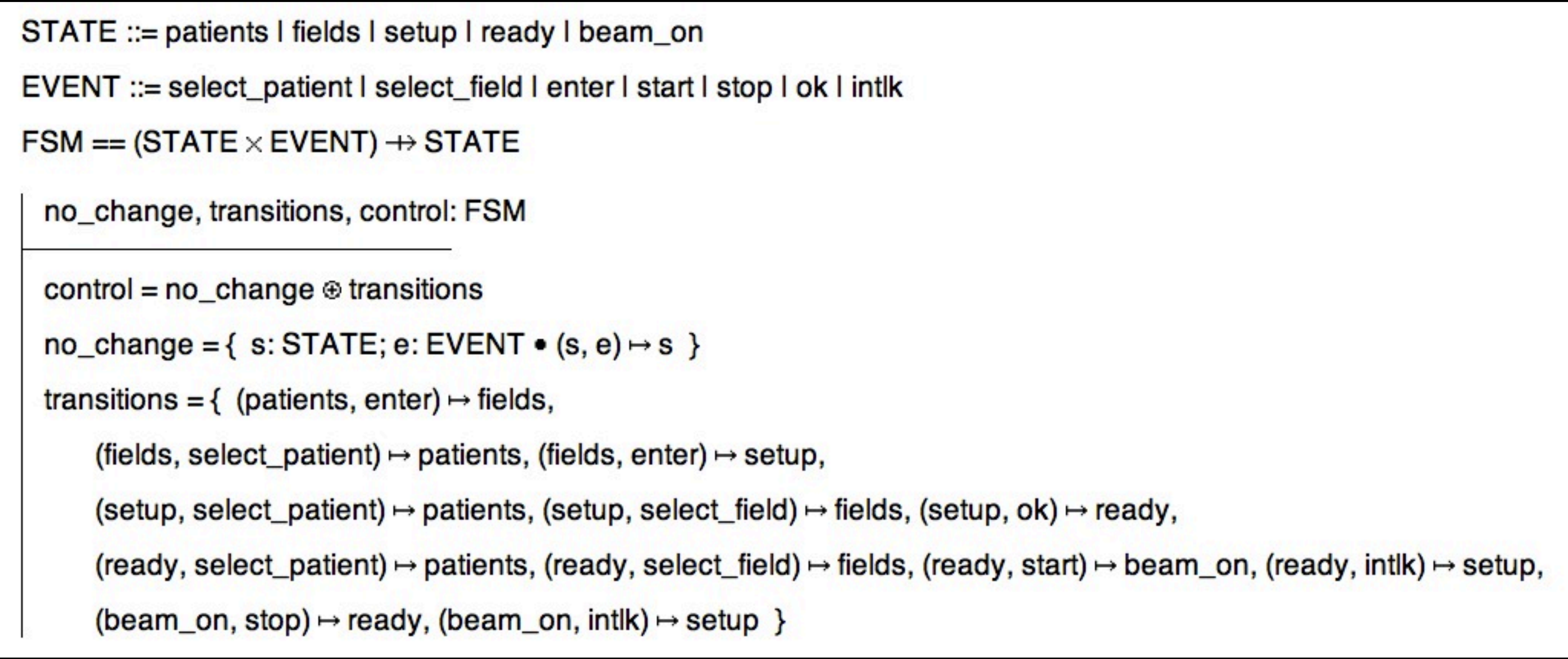
From J. Jacky, "The way of Z", chapter 6

State Transition Table (Z example)



	SELECT PATIENT	SELECT FIELD	ENTER	ok	START	STOP	intlk
PATIENTS	---	---	FIELDS	---	---	---	---
FIELDS	PATIENTS	---	SETUP	---	---	---	---
SETUP	PATIENTS	FIELDS	---	READY	---	---	---
READY	PATIENTS	FIELDS	---	---	BEAM ON	---	SETUP
BEAM ON	---	---	---	---	---	READY	SETUP

And now in Z



STATE ::= patients | fields | setup | ready | beam_on

EVENT ::= select_patient | select_field | enter | start | stop | ok | intlk

FSM == (STATE $\times$ EVENT) ⇸ STATE

no_change, transitions, control: FSM

---

control = no_change $\oplus$ transitions

no_change = { s: STATE; e: EVENT $\bullet$ (s, e) $\mapsto$ s }

transitions = { (patients, enter) $\mapsto$ fields,

(fields, select_patient) $\mapsto$ patients, (fields, enter) $\mapsto$ setup,

(setup, select_patient) $\mapsto$ patients, (setup, select_field) $\mapsto$ fields, (setup, ok) $\mapsto$ ready,

(ready, select_patient) $\mapsto$ patients, (ready, select_field) $\mapsto$ fields, (ready, start) $\mapsto$ beam_on, (ready, intlk) $\mapsto$ setup,

(beam_on, stop) $\mapsto$ ready, (beam_on, intlk) $\mapsto$ setup }