Requirements gathering and specification

PSU CS 300 Lecture 4-1

Bart Massey
Assoc Prof Computer Science
Portland State University
<bart@cs.pdx.edu>
Notes

- Responsible for both course text and lectures
- Sadly, not enough time for everything
- We'll try to do more practice in class
User requirements

- “What”, not “how”
- Basis for
  - clean design
  - user validation
  - system test
- Connect domain to SW
Function vs hard stuff

- Function is easy
  - The input/output model
- State is hard
  - Breaks the model
- ILities are hard
  - Outside the model
Work product: Numbered pars

- Reqs / spec / arch split varies by dev org
- All produce SRS-type doc
  - Hierarchically numbered English pars with succinct, careful statements
  - Some formal language: “may”/“should”/“shall”
Modes: User-visible state

- Modes are bad, but often are unavoidable
- Much SRS complexity is tracking modal behavior
  - Magic notation helps
  - e.g. Leveson TCAS work
Prototypes

- To *gain knowledge*
  - of user reqs
  - of design properties
- Reusable *vs* discardable
- *vs* Increment/Iterate dev
  - Spiral model
  - Open source
The open source way

- No SRS or formal process
- Highly incremental / spiral
- Relies on
  - developer-customers
  - comms infrastructure
- Code as SRS
Good reqs checklist

- User-friendly
- “What” not “how”
- Valid
- Sound & complete
- Brief
- Precise
- Traceable
- Modifiable
- Testable
- Feasible
Top concerns

- **V&V**
  - Test oracle
  - Inspection target
  - Formal methods assertions

- **If you don't know what you're building, your process is doomed**
Requirements gathering and specification

PSU CS 300 Lecture 4-1

Bart Massey
Assoc Prof Computer Science
Portland State University
<bart@cs.pdx.edu>
Notes

- Responsible for both course text and lectures
- Sadly, not enough time for everything
- We'll try to do more practice in class
User requirements

- “What”, not “how”
- Basis for
  - clean design
  - user validation
  - system test
- Connect domain to SW
Function vs hard stuff

- **Function is easy**
  - The input/output model
- **State is hard**
  - Breaks the model
- **Ilities are hard**
  - Outside the model
Work product: Numbered pars

- Reqs / spec / arch split varies by dev org
- All produce *SRS*-type doc
  - Hierarchically numbered English pars with succinct, careful statements
  - Some formal language: “may”/“should”/“shall”
Modes: User-visible state

- **Modes are bad, but often are unavoidable**
- **Much SRS complexity is tracking modal behavior**
  - Magic notation helps
  - *e.g.* Leveson TCAS work
Prototypes

- **To *gain knowledge***
  - of user reqs
  - of design properties
- **Reusable vs discardable**
- **vs Increment/Iterate dev**
  - Spiral model
  - Open source
The open source way

- No SRS or formal process
- Highly incremental / spiral
- Relies on
  - developer-customers
  - comms infrastructure
- Code as SRS
Good reqs checklist

- User-friendly
- “What” not “how”
- Valid
- Sound & complete

- Brief
- Precise
- Traceable
- Modifiable
- Testable
- Feasible
Top concerns

• V&V
  - Test oracle
  - Inspection target
  - Formal methods assertions

• If you don't know what you're building, your process is doomed