Freie Universität Berlin

Claas Augner

A practical model of Software Telemetry

Addressing the need for knowledge about remotely deployed systems

Master thesis

Outline

Outline

1. Introduction

Motivation:
– Personal: frustration & scepticism as user + developer view
– Software ≈ result of customer-supplier agreement → requirements
– Problem 1: SW requirements/implementations often imperfect
– Problem 2: After deployment → only explicit customer feedback

Solution:
Software Telemetry (my definition) =
"*systematic collection, transmission and processing of selective information from a deployed software system, with the objective of acquiring actionable insights about the operation of the system*"

1. Introduction

Premise:
Knowledge about deployed software systems is important.

Research Problem:
Knowledge is difficult to obtain, if systems are remotely deployed.

1st Research Question: What are practices to obtain this knowledge?

2nd Research Question: How can these practices be modelled?

Procedure:

- Literature review → Background
- Review of software products → Practices
- Scientific modelling → Model
- Case study → Model instance
- Qualitative evaluation → Model evaluation

Outline

1. Introduction
2. Background
   - Context
   - Theory
   - Significance
3. Practices
4. Model
5. Case Study
6. Evaluation
7. Conclusion

2. Background

"A realtime software telemetry harness was designed …"

Historical context:
– "*telemetry*" ≈ "measuring at a distance"
– "*Software Telemetry*" – adopted by Microsoft/Mozilla
– Roots: Software Measurement, SW Monitoring (since 1960s)

Contemporary context:

Analytics	Instrumentation	Performance Evaluation
Error Reporting	Logging	Usage Data Collection

Theoretical underpinnings:
No SW Telemetry theory yet. Related theory:
– Software Maintenance (context)
– Software Operation Knowledge (artefact)
– Software Quality (goal)

2. Background

Research gap:
Similar research has been conducted.
But: No general research on practical applications yet.
And: No focus on remotely deployed applications yet.

Significance:
– Increasing complexity
– Increasing user expectations
– Paradigm shift from agile to Continuous Deployment
– Remote deployments
– SW maintenance longest phase

Outline

1. Introduction
2. Background
3. Practices
   - Procedure
   - Choice
   - Examples
4. Model
5. Case Study
6. Evaluation
7. Conclusion

3. Practices

Procedure:
Inclusion criteria → review dimensions
→ software selection → sources identification
→ review → integration of results

Inclusion criteria:
– SW must apply telemetry.
– SW must be open-source or telemetry well-documented.
– Selection should be diverse.

Review dimensions:

About	Goals	Collection	User involvement
Sources	Scopes	Transmission	Privacy / Security
Overview	Architecture	Processing	

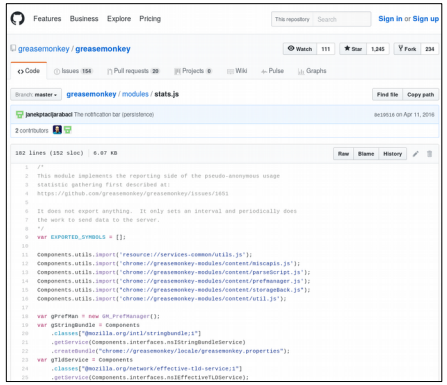
3. Practices

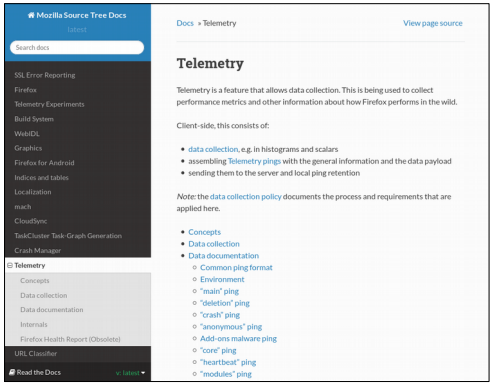
11 Software products:

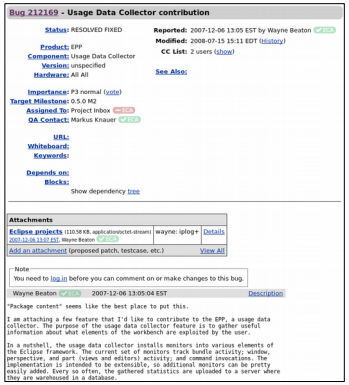
ATOM
eclipse
GIMP
chrome
Firefox
Nextcloud
SUBVERSION
FAIRPHONE
ubuntu
Windows

Primary sources: software, *source code*, documentation, etc.

Secondary sources: literature, bug tracker, etc.

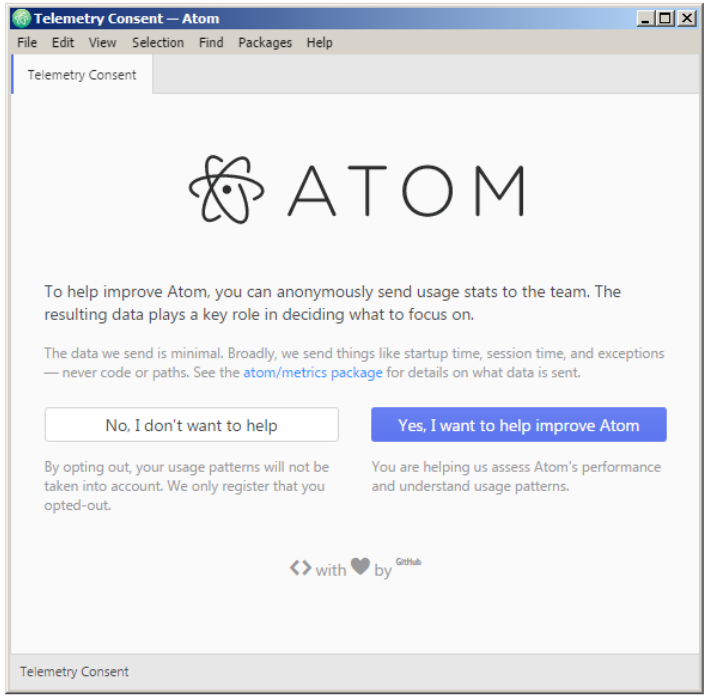


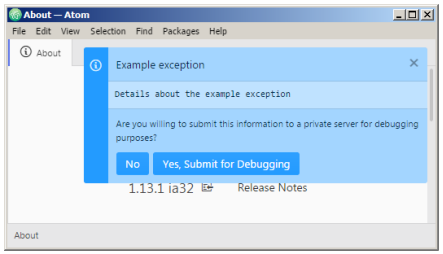


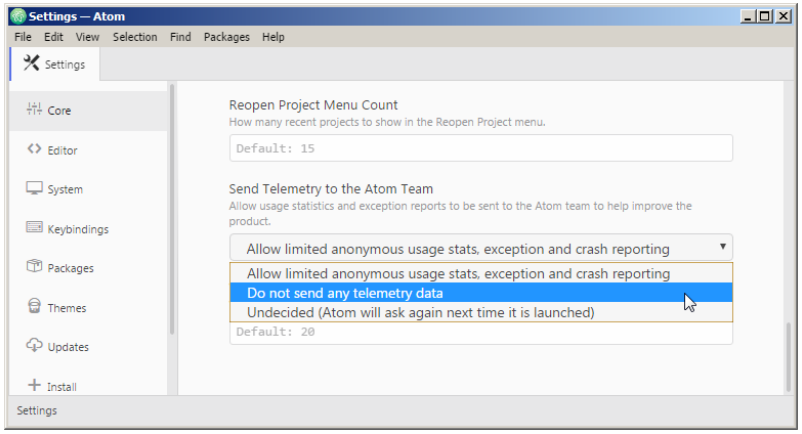


3. Practices

Example 1: ATOM







3. Practices

Example 2: Nextcloud

Administration

Claas

- Server settings
- Server info
- Sharing
- Theming
- Encryption
- Workflow
- Usage report
- Logging
- Additional settings
- Tips & tricks

Usage report

You can help us to improve Nextcloud be sending us some data about your current setup and usage.
We take your privacy seriously. The data are anonymized and you can enable/disable it at any time, by default it is always disabled. Below you can also adjust what kind of data are send and always see the last report send to us.
When the server receives a new report of your instance, all entries from previous reports are removed. In case you disable one of the settings below, you can send a new report, to delete the data, that is currently stored on the server.

Send new report now

- [x] Send "Usage report" monthly

Data to send

- [x] Server instance details (version, memcache used, locking/previews/avatars enabled?)
- [x] PHP environment (version, memory limit, max. execution time, max. file size)
- [x] Database environment (type, version, database size)
- [x] App list (for each app: name, version, is enabled?)
- [x] Statistic (number of: files, users, storages per type)
- [x] Number of shares (per type and permission setting)
- [x] Encryption information (is it enabled?, what is the default module)

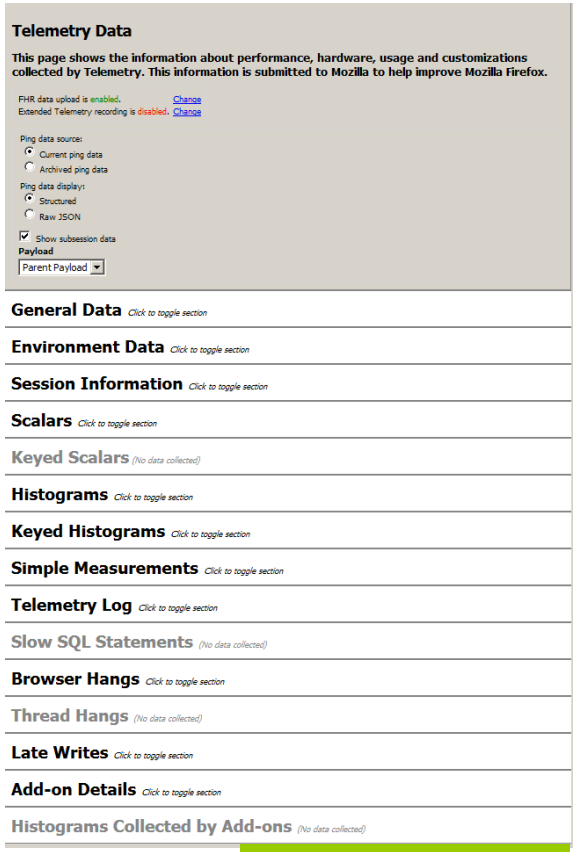
Last report


```
{
  "id": "526a8d1552887",
  "items": [
    [
      "server",
      "version",
      "9.1.1.5"
    ],
```

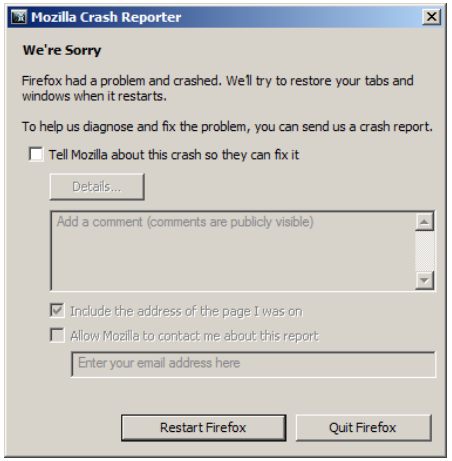

Sent on: 4. Oktober 2016

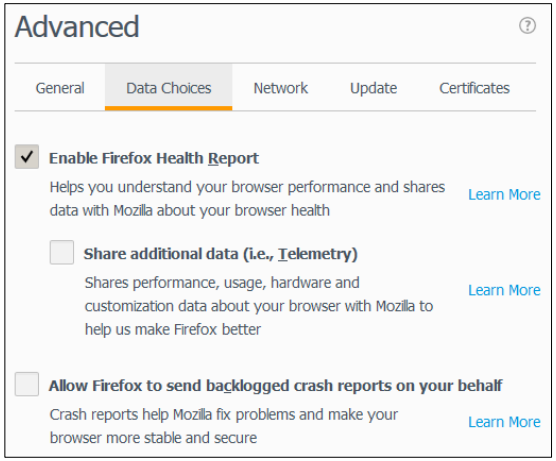
3. Practices

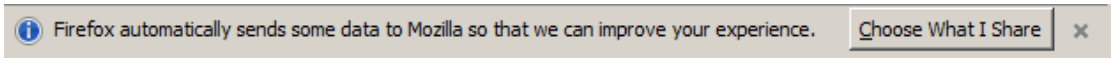
Example 3: Firefox

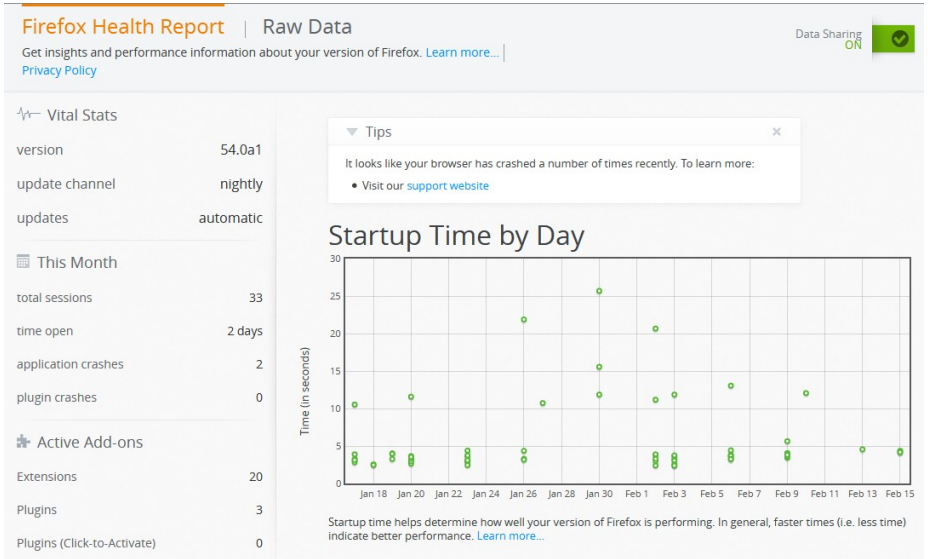


about:telemetry







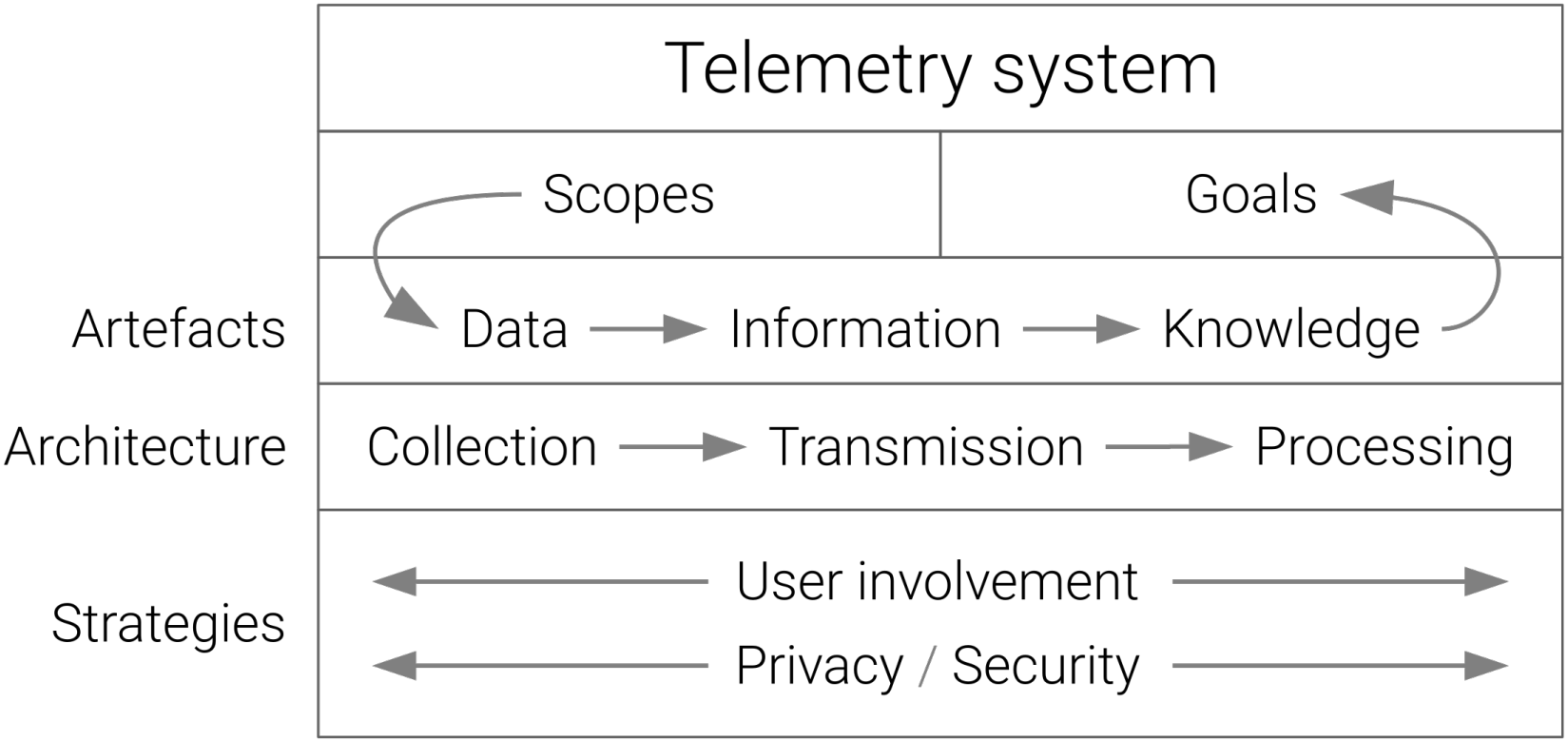


about:healthreport

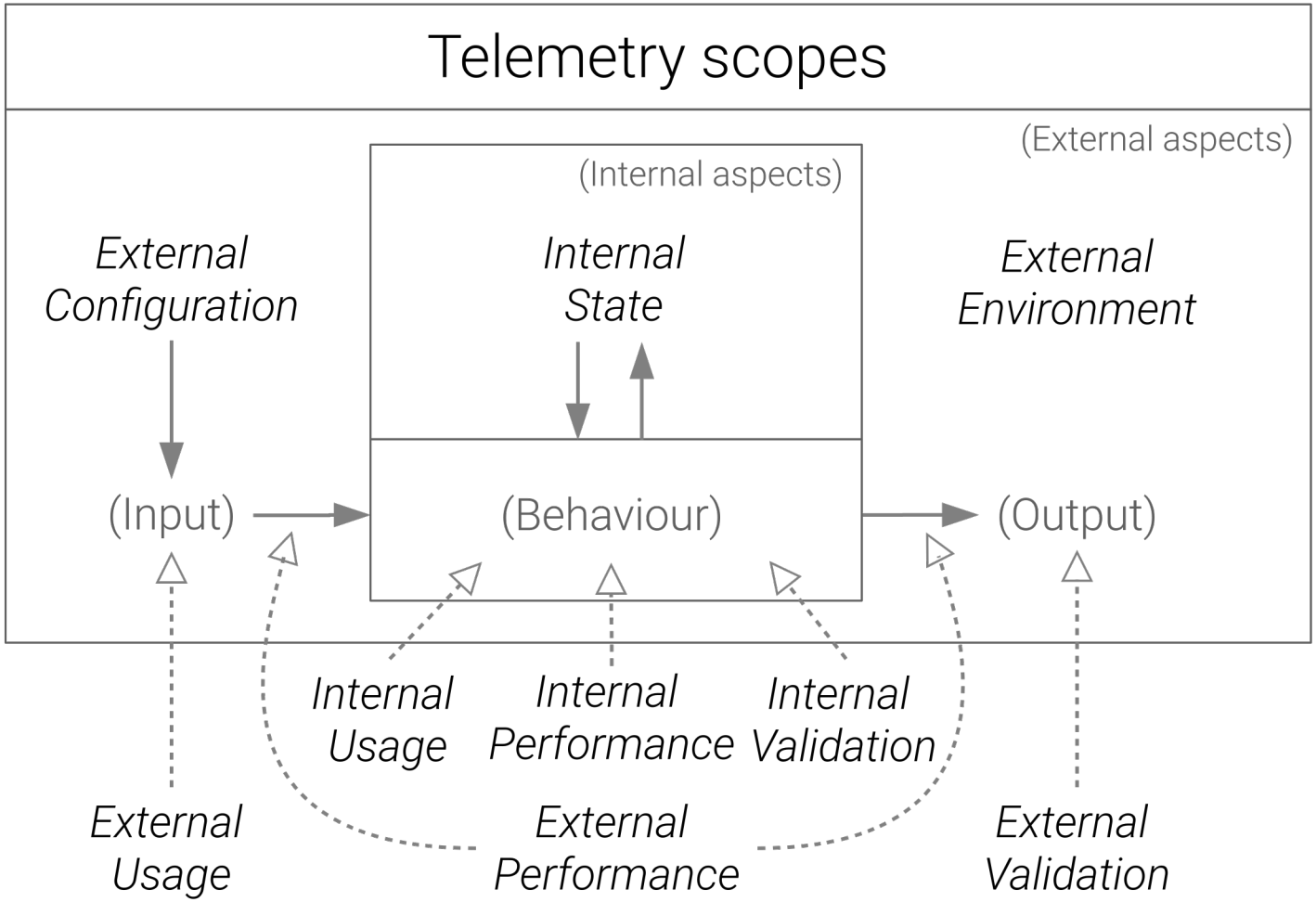
Outline

1. Introduction
2. Background
3. Practices
4. Model
   - Telemetry system
   - Telemetry scopes
   - Telemetry model
5. Case Study
6. Evaluation
7. Conclusion

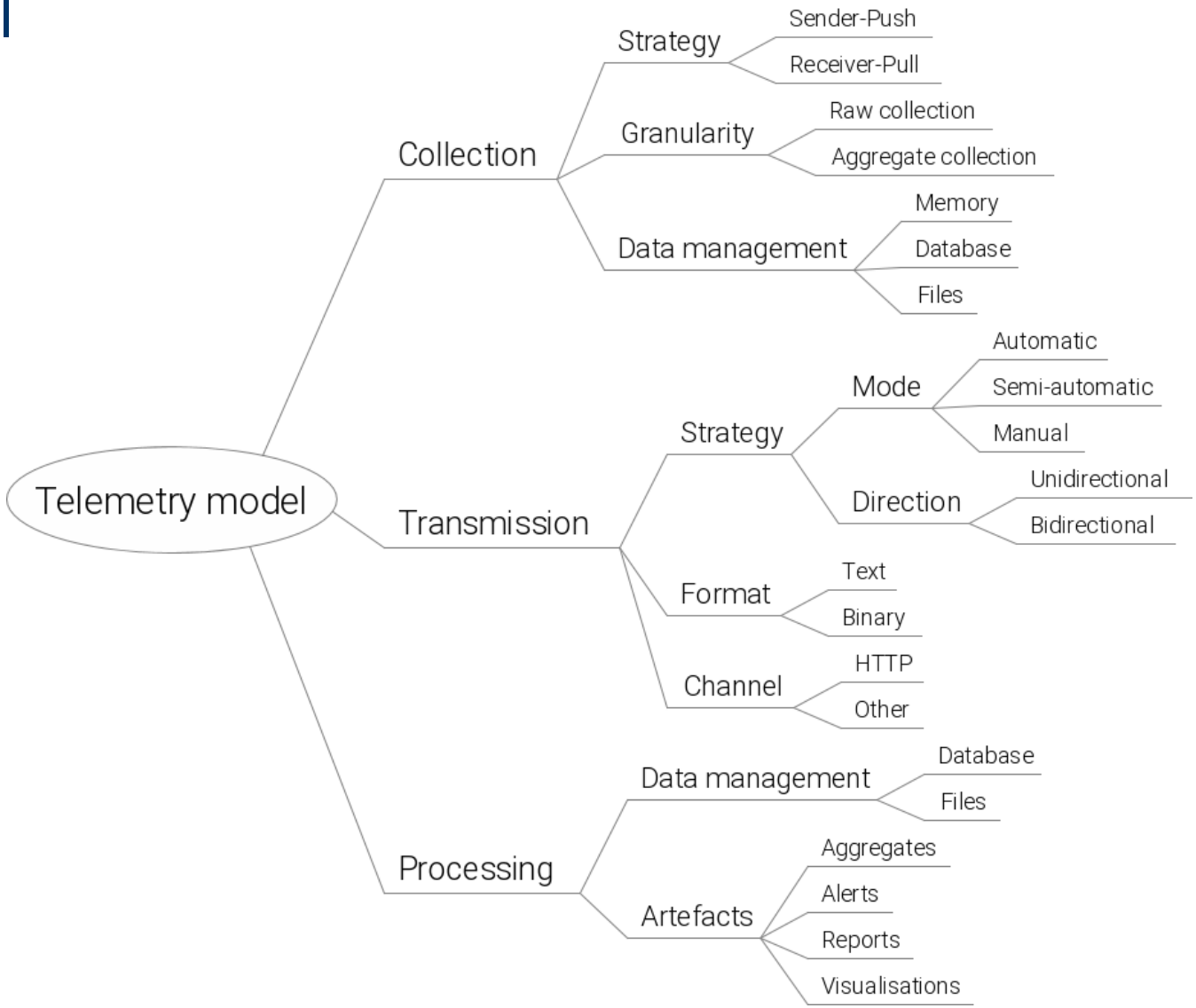
4. Model


Telemetry system
Scopes
Goals
Artefacts
Data
Information
Knowledge
Architecture
Collection
Transmission
Processing
Strategies
User involvement
Privacy / Security

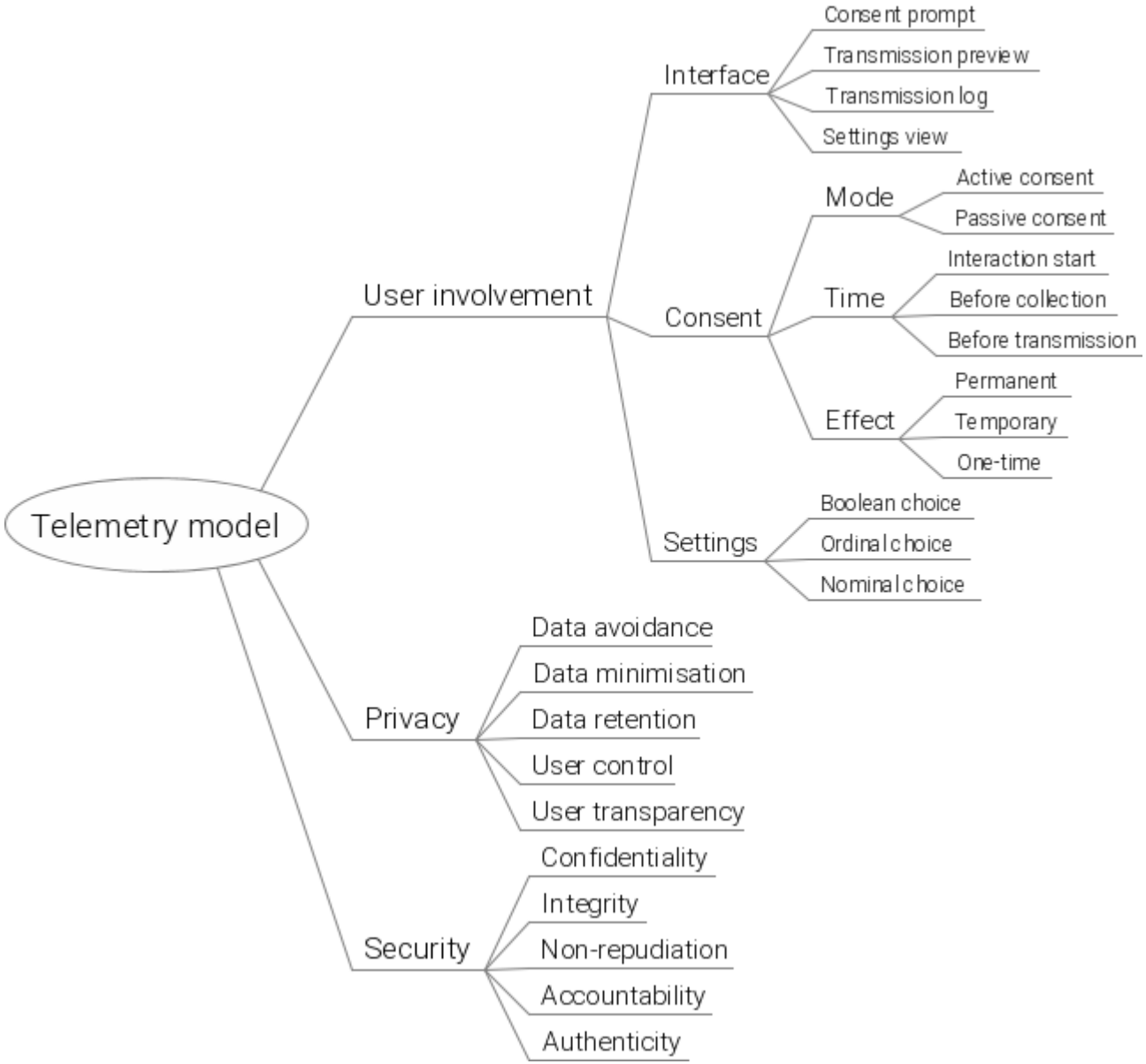
4. Model


Telemetry scopes
(External aspects)
(Internal aspects)
External Configuration
Internal State
External Environment
(Input)
(Behaviour)
(Output)
Internal Usage
Internal Performance
Internal Validation
External Usage
External Performance
External Validation

4. Model


Telemetry model
Collection
Strategy
Sender-Push
Receiver-Pull
Granularity
Raw collection
Aggregate collection
Data management
Memory
Database
Files
Transmission
Strategy
Mode
Automatic
Semi-automatic
Manual
Direction
Unidirectional
Bidirectional
Format
Text
Binary
Channel
HTTP
Other
Processing
Data management
Database
Files
Artefacts
Aggregates
Alerts
Reports
Visualisations

4. Model


Telemetry model
User involvement
Interface
Consent prompt
Transmission preview
Transmission log
Settings view
Consent
Mode
Active consent
Passive consent
Time
Interaction start
Before collection
Before transmission
Effect
Permanent
Temporary
One-time
Settings
Boolean choice
Ordinal choice
Nominal choice
Privacy
Data avoidance
Data minimisation
Data retention
User control
User transparency
Security
Confidentiality
Integrity
Non-repudiation
Accountability
Authenticity

Outline

1. Introduction
2. Background
3. Practices
4. Model
5. Case Study
   - Context / Objectives
   - Architecture
   - Implementation
6. Evaluation
7. Conclusion

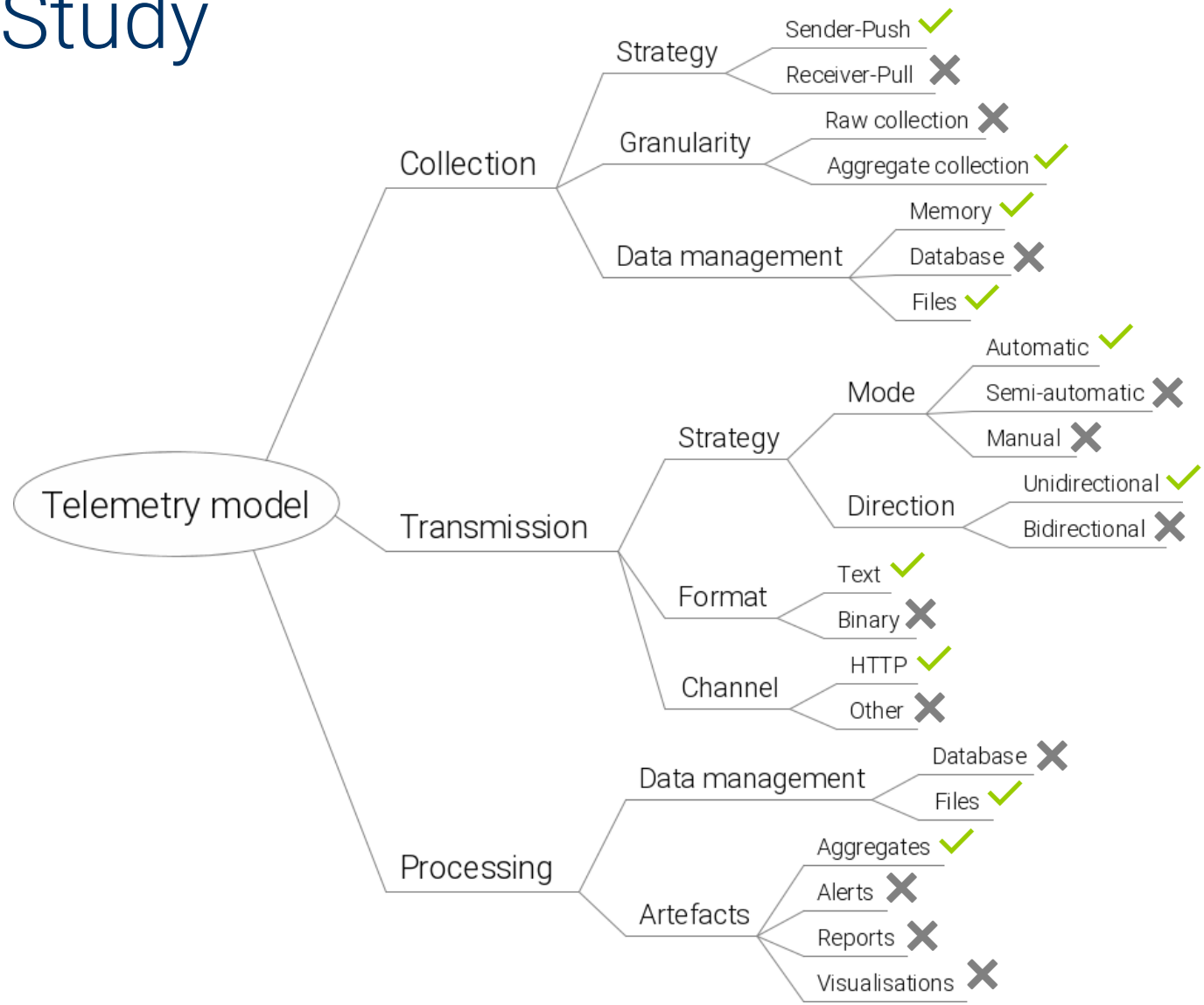
5. Case Study

Context:
– *Acme GmbH* = German medium-sized company with 1 SW product
– *ABC* = web application for enterprises, modularly structured, highly configurable and largely extensible

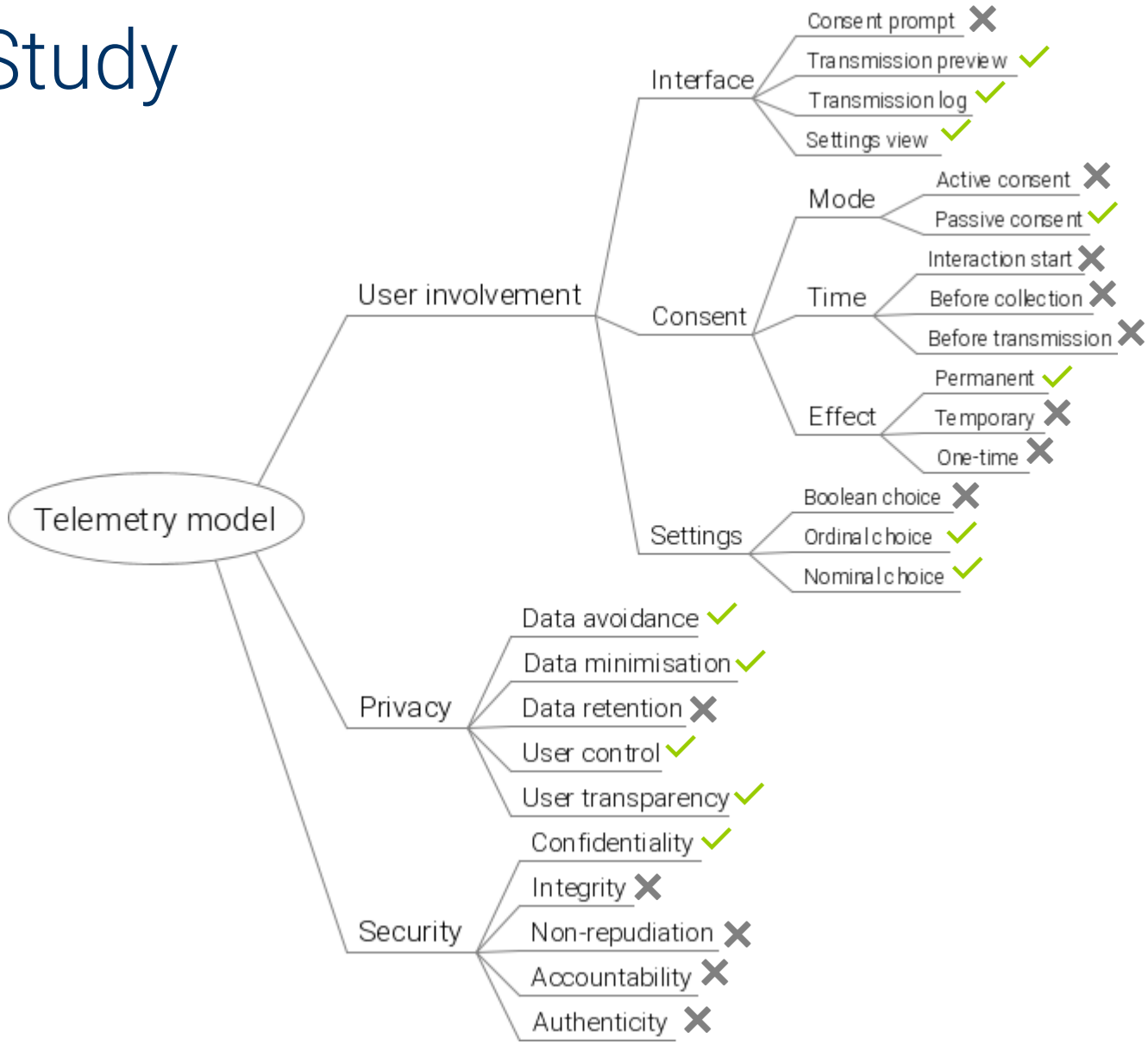
Objectives:
– General: Verify if model applicable to a real-world SW product.
– Acme: Insights regarding feature usage + performance.
– Author: Flexibility, user transparency, developer usability.

Architecture:
– 2 services (collection + transmission) + a `Telemetry` façade
– independent processing server
– Approach: New probes must be described in a Java `enum`

5. Case Study


Telemetry model
Collection
Strategy
Sender-Push
Receiver-Pull
Granularity
Raw collection
Aggregate collection
Data management
Memory
Database
Files
Transmission
Strategy
Mode
Automatic
Semi-automatic
Manual
Direction
Unidirectional
Bidirectional
Format
Text
Binary
Channel
HTTP
Other
Processing
Data management
Database
Files
Artefacts
Aggregates
Alerts
Reports
Visualisations

5. Case Study


Telemetry model
User involvement
Interface
Consent prompt ✗
Transmission preview ✓
Transmission log ✓
Settings view ✓
Consent
Mode
Active consent ✗
Passive consent ✓
Time
Interaction start ✗
Before collection ✗
Before transmission ✗
Effect
Permanent ✓
Temporary ✗
One-time ✗
Settings
Boolean choice ✗
Ordinal choice ✓
Nominal choice ✓
Privacy
Data avoidance ✓
Data minimisation ✓
Data retention ✗
User control ✓
User transparency ✓
Security
Confidentiality ✓
Integrity ✗
Non-repudiation ✗
Accountability ✗
Authenticity ✗

Outline

1. Introduction
2. Background
3. Practices
4. Model
5. Case Study
6. Evaluation
   - Data assessment
   - Experiment
   - Results
7. Conclusion

6. Evaluation

Data assessment:
– *Goal*: Functional / data evaluation
– *Procedure*: Insert 10 probes with hypothesis → run tests → verify

Experiment:
– *Goal*: Practical / developer evaluation
– *Procedure*: Test and survey with 3 developers

Results:
– Confirmed flexibility of the model.
– Confirmed functionality of all components.
– Provided evidence for practicability, effectiveness and usability.

Outline

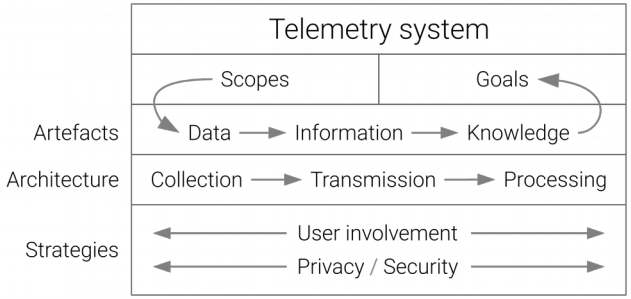
1. Introduction
2. Background
3. Practices
4. Model
5. Case Study
6. Evaluation
7. Conclusion
   - Limitations
   - Results
   - Future research

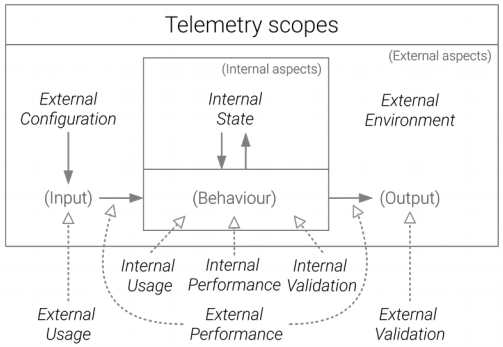
7. Conclusion

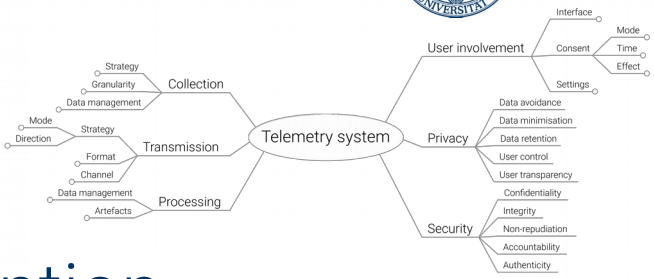
Limitations:
– Bias towards open source software.
– No Software Telemetry processes.
– No actual user involvement.
– No quantitative evaluation.

Results:
– Practices are diverse, but follow recurring patterns.
– Obtaining knowledge about remotely deployed systems is doable.
– Model provides basis for successful implementation.

Future research:
– Refine with regard to limitations.
– Study discontinuation examples of SW Telemetry.
– Study user perception of SW Telemetry.







Thank you for your attention.

A practical model of Software Telemetry

Addressing the need for knowledge about remotely deployed systems

Any questions?