Towards Effective and Efficient Behavior-based Trust Models

Klemens Böhm
Universität Karlsruhe (TH)
Motivation: Grid Computing in Particle Physics

- Physicists have designed and implemented services specific to particle physics (data analysis, computations, etc.).
- In general, physicists are willing to let others use their services.
- A service typically runs at the institution where it has been implemented.
- However, resources are limited etc. Typically, one cannot/does not want to process all service requests.
- Letting others pay for services is not acceptable
  - for cultural reasons, and
  - rigid specification of service characteristics would be necessary.
Motivation, continued

- Different users have different policies. Highly subjective issue.
- Wanted: Language to specify when a particular user may consume the service.
- Behavior-based:
  - Decision depends on previous behavior of the user.
  - Useful in settings that are anonymous.
  - Requires that information on previous behavior of the user is available.
- Aka. trust policies/trust policy languages.
Behavior-based Trust Policies (1)

- Example policies:
  - Alice: "I let someone use my resources (i.e., I trust him) if the average feedback about him is positive."
  - Bob: "I will provide service X for someone if there is no negative feedback about him within the last 24h."
  - Carol: "I will only interact with someone if the $k$ most reputable entities recommend him."
  - Dave: "I only perform the services for others if their performance regarding complex tasks has been satisfactorily."
Behavior-based Trust Policies (2)

- Not only service providers may use such policies, but also consumers of services. (E.g., "I only want to use services of providers without any negative feedback about them.")
What Can We Learn from the Examples? (1)

- Requirement 1:
  Trust policies may require complex operations, e.g., aggregation (‘average feedback’).

- Requirement 2:
  Adequate representation of knowledge that describes behavior of participants sought (behavior-specific knowledge).
  - Different types of behavior-specific knowledge: feedback, reputation, recommendation, trust.
  - Various aspects of behavior-specific knowledge considered (e.g., context, age of knowledge, etc.).
What Can We Learn from the Examples? (2)

• Representation of knowledge as directed graph $G(V,E)$
  - $V$…set of participants
  - $E$…set of edges based on behavior-specific knowledge

• Example:

![Diagram showing a directed graph with nodes A, B, C, D, E and arrows indicating relationships such as Feedback, Recommendation, and Trust.]

→ Application of graph algorithms to find trustworthy partners
  e.g., EigenTrust (Schlosser et al., 2003), PageRank (Brin and Page, 1996).
  *Centrality.*
Motivation, Continued

• Research issues:
  ▪ Define a language to formulate trust policies.
  ▪ Efficient evaluation of policies.
  ▪ Effectiveness issues.
Contributions

- Identify characteristics of/classify behavior-based trust models from literature. Definition of underlying concepts.

- Definition of query algebra for trust. Demonstrated its appropriateness.

- Experiments:
  - Efficiency and effectiveness of various trust policies (relying on centrality measures).
  - How does preprocessing of underlying data affect the effectiveness of centrality measures?
  - Which trust policies are used in reality? (ongoing work)
Agenda

• Introduction
• Terminology
• Centrality
• Query Algebra for Trust
  ▪ Idea
  ▪ Conventional Extensions
  ▪ Centrality Operator
• Experiments
  ▪ Setup
  ▪ Results
• Effectiveness
• Conclusions
Types of Behavior-specific Knowledge (1)

- **Feedback**
  - An entity's (*rater*) rating of an interaction performed by a partner (*ratee*).
  - Alice: "The last service execution by Bob was very satisfying."

- **Recommendation**
  - An entity's (*recommender*) opinion about the previous behavior of a partner (*recommende*).
  - Alice: "For services of type X, I can recommend Bob."
Types of Behavior-specific Knowledge (2)

- **Reputation**
  - General opinion of the whole network towards a single entity
    - Global characteristic of an entity.
  - Example: "With regard to services of type Y, Bob has an excellent reputation."

- **Trust**
  - An entity's (truster) degree of belief that a partner (trustee) will behave as expected.
  - Alice: "I trust Bob regarding services of type Z."
Aspects of Behavior-specific Knowledge (1)

• **Value** ∈ [-1, 1]
  - Continuous valuation allows for a finer granularity.
  - Alice: "Bob’s last service execution has been fairly good (~0.6)."

• **Context**
  - Allows to distinguish between different situations where entities can interact.
  - Alice: "Bob is good regarding computations of type X, but his performance wrt services of type Y has been poor."

• **Facets of a context**
  - Allows to distinguish between different perspectives of a context.
  - Alice: "The last service invocation has been very satisfying but also very slow."
Aspects of Behavior-specific Knowledge (2)

- **Timestamp**
  - Allows to emphasize the impact of current knowledge.
  - Alice: "Bob’s early service executions were satisfactory but recent ones were poor."

- **Certainty \( \in [0,1] \)**
  - Allows to quantify the certainty of an assessment.
  - Alice: "I am absolutely sure (e.g., \(~1.0\)) that Bob’s last performance was good."

- **Estimated Effort \( \in [0,1] \)**
  - Allows to quantify the perceived complexity of an interaction.
  - Alice: "Bob performed simple (e.g., \(~0.2\)) computations quite well but complex ones (e.g., \(~0.9\)) very poorly."
Agenda

- Introduction
- Terminology
- **Centrality**
  - Query Algebra for Trust
    - Idea
    - Conventional Extensions
    - Centrality Operator
- Experiments
  - Setup
  - Results
- Effectiveness
- Conclusions
Feedback Graph

- Derivation from feedback
  - Participants $\rightarrow$ vertices
  - Feedback from A about B $\rightarrow$ edge from Vertex A to Vertex B
  - Value of feedback $\rightarrow$ weight of corresponding edge

$\Rightarrow$ Characteristics of feedback graph
- Directed graph
- Not strongly connected
- Multiple edges
- Weighted edges

<table>
<thead>
<tr>
<th>Rater</th>
<th>Ratee</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>A</td>
<td>0.5</td>
</tr>
<tr>
<td>B</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>A</td>
<td>-0.2</td>
</tr>
<tr>
<td>B</td>
<td>D</td>
<td>0.6</td>
</tr>
<tr>
<td>C</td>
<td>D</td>
<td>-0.2</td>
</tr>
<tr>
<td>D</td>
<td>C</td>
<td>-0.5</td>
</tr>
</tbody>
</table>
Centrality Indices (1)

- Centrality index
  - Graph-based measure to quantify the importance of a vertex according to the graph structure.
  - Different existing measures: *Indegree*, *PageRank*, *Proximity Prestige*, *HITS*, *Integration & Radiality*, etc.
  - Different measures yield different rankings.

- According to various proposals, reputation of participant = his centrality value.
## Centrality Indices (2)

### Table 1: Centrality Indices for Nodes A, B, C, D, and E

<table>
<thead>
<tr>
<th></th>
<th>InDegree</th>
<th>PageRank</th>
<th>Proximity</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>0.2</td>
<td>1.12</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0.12</td>
<td>0.25</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0.06</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>0.27</td>
<td>0.67</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>0.35</td>
<td>0.5</td>
</tr>
</tbody>
</table>

### Table 2: Query Algebra for Trust

<table>
<thead>
<tr>
<th></th>
<th>InDegree</th>
<th>PageRank</th>
<th>Proximity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>D</td>
<td>E</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>3</td>
<td>E</td>
<td>A</td>
<td>E</td>
</tr>
<tr>
<td>4</td>
<td>B</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>5</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
</tbody>
</table>

---

"Towards Effective and Efficient Behavior-based Trust Models"
Centrality Measures Considered

- Local measures
  - consider only direct neighborhood of a vertex
  - \textit{InDegree}

- Eigenvector-based measures
  - recursively defined measures that consider direct and indirect neighborhoods of a vertex
  - \textit{PageRank}, Authority (HITS), Positional Weakness Function

- Distance-based measures
  - rely on shortest paths between vertices
  - \textit{Proximity Prestige}, Integration
# Requirements of CM on Input Graph

<table>
<thead>
<tr>
<th></th>
<th>Weighted Graphs</th>
<th>Unweighted Graphs (single edges)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Multiple edges (+/- weights)</td>
<td>Single edges (+ weights)</td>
</tr>
<tr>
<td>Local</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Eigenvector-based</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>Distance-based</td>
<td>-</td>
<td>✓</td>
</tr>
</tbody>
</table>

- **Result:**
  - Most measures require graph transformation techniques.
  - Identification of two subsequent transformation steps:
    - multiple weighted edges → single weighted edges,
    - single weighted edges → single unweighted edges.

- **Problem:** transformation incurs loss of information.
EigenTrust Transformation

- Example
  - Feedback graph:

- Input graph after transformation

\[
\begin{align*}
\max \{ 0 ; \ 0.5 + 1 - 0.2 \} &= 1.3 \\
\max \{ 0 ; \ 0.1 - 0.9 \} &= 0
\end{align*}
\]
**Beta Transformation**

- **Example**
  - Feedback graph:

- **Input graph after transformation**
  - $\beta = 0 \rightarrow \text{"no experience"} = \text{"max negative experience"}$
    
  $$\frac{0.5 + 1}{|0.5| + |1| + |-0.2|} = 0.88$$

  - $\beta = 0.5 \rightarrow \text{"no experience"} = \text{"neutral experience"}$
    
  $$\frac{0.1}{|0.1| + |-0.9|} = 0.1$$
Single weighted edges $\rightarrow$ Single Unweighted edges

- Example
  - Feedback graph:
    - Input graph after transformation
      - $\tau=0.5$
      - $\tau=0.9$
Impact on Trust Policy Language Envisioned

- Any centrality measure.
- Extensibility in this respect.
- Any transformation.
Agenda

• Introduction
• Terminology
• Centrality
• **Query Algebra for Trust**
  ▪ Idea
  ▪ Conventional Extensions
  ▪ Centrality Operator
• Experiments
  ▪ Setup
  ▪ Results
• Effectiveness
• Conclusions
Status Quo

- Existing behavior-based trust models
  - define representation of behavior-based knowledge,
  - define fixed evaluation scheme to derive the trust in a partner.

- Fixed evaluation scheme contradicts subjective nature of trust.

- Potential approach for making trust policies explicit: Logic-based trust policy languages. Definition of rules and clauses to derive the trustworthiness of a partner.

- Existing languages cannot satisfactorily cope with data-intensive computations required by behavior-based policies.
A Framework for behavior-based trust models

• Aspects of our framework:
  - *relational representation* of behavior-specific knowledge,
  - *algebra* for the formulation of behavior-based trust policies.

• Advantages:
  - Supports definition of arbitrary user-defined trust policies for behavior-based trust models, including all existing evaluation schemes from literature we currently are aware of.
  - Relational representation allows for a straightforward implementation.
Relational Representation of Knowledge

- Relations that represent behavior-specific knowledge: Feedback, Recommendation, Reputation, Trust

- Additional relation: Entity (ID)

- Alice: "I am quite sure that the execution of service S by Bob was good. It was a complex problem."

  ➔ New Feedback tuple.

<table>
<thead>
<tr>
<th>Rater</th>
<th>Ratee</th>
<th>Value</th>
<th>Context</th>
<th>Facet</th>
<th>Time</th>
<th>Certainty</th>
<th>Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Bob</td>
<td>0.95</td>
<td>S</td>
<td>Quality</td>
<td>12:09:45</td>
<td>0.75</td>
<td>0.8</td>
</tr>
</tbody>
</table>

➔ Goal: Trust policy language as mechanism to derive Trust, Recommendation and Reputation tuples.
Relational Representation of Knowledge

- In our scenario:
  - Only Feedback tuples reflect direct experiences.
  - Other knowledge must be derived from feedback (including Trust tuples).
Towards an Algebra-based Policy Language

- Source: Relational representation of knowledge.
- trust policy = query on the knowledge base. (Same with recommendation, reputation.)
- Common way to deal with relations: *Relational Algebra* (RA)
  - set of operators to be applied on relations,
  - closure property of operators allows for nesting of operators to complex algebra expressions.

⇒ Basic Idea: *Relational Algebra* (RA) as basis for our trust policy language.
Example Trust Policy

- Informal formulation:
  - "I trust individuals in context $c$ and facet $f_c$ if their average feedback value from the 10 most reputable entities exceeds a specific threshold."
  - Only feedback tuples with certainty>0.8 shall be considered."

⇒ Algebra expression of that policy:

```plaintext
PROJECTION[trusted](
  MAP[trusted, (avg_value>threshold)](
    GROUP[avg_value, AVG(Feedback.value), {ratee}](
      JOIN[Feedback.rater=Reputation.entity](
        TOP[10, Reputation.value](
          SELECTION[context=c, facet=f_c](Reputation)
          SELECTION[ratee=id_partner context=c, facet=f_c, certainty>0.8](Feedback)
          )
        )
      )
  )
)
```
Algebra-based Policy Language

- Observation:
  - Basic operators of the RA are not sufficient to formulate behavior-based trust policies.
  - Extension with additional operators are necessary.

→ Which further operators are essential to arrive at expressiveness desired?

- First step: Existing additional operators from literature
  - Top operator (e.g., Bertino et al., 2004)
  - Map operator (e.g., Aberer and Fischer, 1995)
Conventional Extensions to the RA (1)

- Top Operator: \( \text{TOP}[k, \text{attr}](\text{relation}) \)
  - returns the \( k \) tuples with the highest value of a attribute \( \text{attr} \)
  - Example:

<table>
<thead>
<tr>
<th>ID</th>
<th>...</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td></td>
<td>0.71</td>
</tr>
<tr>
<td>Carol</td>
<td></td>
<td>0.95</td>
</tr>
<tr>
<td>Alice</td>
<td></td>
<td>0.98</td>
</tr>
<tr>
<td>Eve</td>
<td></td>
<td>0.75</td>
</tr>
<tr>
<td>Dave</td>
<td></td>
<td>0.90</td>
</tr>
</tbody>
</table>

\( \text{TOP}[3, \text{Value}](\text{Reputation}) \)

<table>
<thead>
<tr>
<th>ID</th>
<th>...</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carol</td>
<td></td>
<td>0.95</td>
</tr>
<tr>
<td>Alice</td>
<td></td>
<td>0.98</td>
</tr>
<tr>
<td>Dave</td>
<td></td>
<td>0.90</td>
</tr>
</tbody>
</table>
Conventional Extensions to the RA (2)

- Map Operator: \( \text{MAP}[\text{attr}, \text{expression}(A_1, \ldots, A_n)](\text{relation}) \)
  - Allows the execution of user-defined functions over the attributes of a relation.
  - The functions are applied separately to each tuple; the results become a new attribute.
  - Example:

<table>
<thead>
<tr>
<th>Rater</th>
<th>Ratee</th>
<th>...</th>
<th>Value</th>
<th>Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Bob</td>
<td>...</td>
<td>1.0</td>
<td>0.2</td>
</tr>
<tr>
<td>Alice</td>
<td>Carol</td>
<td>...</td>
<td>0.8</td>
<td>0.9</td>
</tr>
</tbody>
</table>

\( \text{MAP[Weighted, (Value*Effort)](Feedback)} \)
An Operator for Centrality Computation

• Requirements for a centrality operator:
  - Flexible specification of the underlying graph
  - Support of various centrality measures within one operator

<table>
<thead>
<tr>
<th>Rater</th>
<th>Ratee</th>
<th>...</th>
<th>Value</th>
<th>Effort</th>
<th>Weighted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Bob</td>
<td></td>
<td>1.0</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>Alice</td>
<td>Carol</td>
<td></td>
<td>0.8</td>
<td>0.9</td>
<td>0.72</td>
</tr>
</tbody>
</table>

e.g., choice of the weight of an edge: "Value" vs. "Weighted"

Definition of centrality operator:

CENTRALITY[attr, A_v, A_s, A_t, A_w, Measure](R_vertices, R_edges)
**Centrality Operator - Example**

**INTRODUCTION**

**Terminology**

**Centrality**

**Query Algebra for Trust**

- Idea
- Conventional Extensions
- Centrality Operator

**Experiments**

**Effectiveness**

**Conclusions**

---

**CENTRALITY** [PageRank, ID, Recommender, Recommendee, Value, **PageRank**] (Entity, Recommendation)

**Recommendation**

<table>
<thead>
<tr>
<th>Recommender</th>
<th>Recommendee</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>C</td>
<td>0.9</td>
</tr>
<tr>
<td>A</td>
<td>E</td>
<td>0.2</td>
</tr>
<tr>
<td>B</td>
<td>A</td>
<td>1.0</td>
</tr>
<tr>
<td>B</td>
<td>D</td>
<td>0.5</td>
</tr>
<tr>
<td>C</td>
<td>B</td>
<td>0.6</td>
</tr>
</tbody>
</table>

**Entity**

<table>
<thead>
<tr>
<th>ID</th>
<th>Recommendee</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
</tr>
</tbody>
</table>

**PageRank**

<table>
<thead>
<tr>
<th>ID</th>
<th>PageRank</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.23</td>
</tr>
<tr>
<td>B</td>
<td>0.21</td>
</tr>
<tr>
<td>C</td>
<td>0.31</td>
</tr>
<tr>
<td>D</td>
<td>0.15</td>
</tr>
<tr>
<td>E</td>
<td>0.1</td>
</tr>
</tbody>
</table>

---

Klemens Böhm

*"Towards Effective and Efficient Behavior-based Trust Models"*
Agenda

- Introduction
- Terminology
- Centrality
- Query Algebra for Trust
  - Idea
  - Conventional Extensions
  - Centrality Operator
- Experiments
  - Setup
  - Results
- Effectiveness
- Conclusions
Centrality Operator

- Nature of centrality computation
  - Very time-consuming and resource-intensive.
  - Centrality computation is the most costly part of evaluation of a trust policy.

- Implemented centrality measures in PL/SQL (Oracle 10g)
  - PageRank, Positional Power Function (eigenvector centrality measures based on power iteration implementation)
  - Authorities, Proximity Prestige, Integration

- Experiments
  - Efficiency: Performance of our implementations
  - Quality of Centrality Measures: Comparison of ranking results
Creation of a Reference Ranking

- Core component: feedback generator
  - Simulates interactions between participants
  - Behavior of a participant specified by its cooperation value (\( = \) probability that participant cooperates)
  - Generation of feedback based on outcome of interactions (outcome depends on cooperation values)

→ Participants sorted by coop values: Reference ranking

→ Computation of CM on feedback: Result rankings

- Intuition: "Good" measures assign high ranks to participants with high cooperation value (and vice versa)
Quantifying the Distance between 2 Rankings

- Participants in reputation systems use cutoff strategies, i.e., a partner B is...
  - trustworthy, if rank of B is above a threshold $k$ (in Top-$k$ list),
  - untrustworthy, if rank of B is below $k$ (not in Top-$k$ list)

(exact rank of B is unimportant)

⇒ A participant can make 2 wrong decisions
  - Deeming an untrustworthy partner trustworthy
  - Deeming an trustworthy partner untrustworthy

Our measure: fraction of wrong decisions (FWD).
Amount of Feedback Data Needed

- **Example:** PageRank

- **Results**
  - A lot of feedback required to gain no more improvement
  - Quite good results with 25k feedback items (50 per peer)
  - Rather "slow" if participants change their behavior frequently
Dealing With "No Experience"

- Parameter $\beta$

- Example: PageRank

- Results:
  - "Neutral knowledge" ($\beta=0.5$) yields best results
  - $\beta>0.85$ (deeming unknown participants good) is inappropriate
  - $\beta=0$ yields quite good result + allows pruning of edges
    ➔ Trade-off: Quality vs. computation performance
Pruning Weighted Edges

- Parameter $\tau$
- Example: Proximity

- Results
  - $\tau > 0.5$: significant improvement for small Top-$k$ lists (explanation: more importance on "strong" edges)
  - The larger $\tau$ the more edges can be pruned

$\Rightarrow$ "Good" values of $\tau$ can improve quality + performance
Different degrees concerning loss of knowledge verifiable
(no distinction between "no experience" and "bad experience"
depends on ratio of uncooperative participants
Quality of EigenTrust-transformation
In general, Beta-transformation is superior

Results:

- Eigenvector-based measures

EigenTrust vs. Beta-Transformation
Comparison of All Centrality Measures

- $\beta=0.5$
- $\tau=0.95$

- Results
  - All measures yield similar results, esp. for smaller Top-$k$ lists
  - Distance-based measures profit from large value of $\tau$
  - With good $\beta, \tau$: All Measures quite suitable
  - Without graph:
    - Performances differ significantly
    - Distance-based measures perform poorest
  - Application of distance-based measures in larger networks questionable.
Insights from Experiments

- No centrality measure is clearly better.
- Both effectiveness and efficiency (in terms of computation effort) are important.
- Much feedback needed to have low FWD.
  - Policies need to be sophisticated.
  - Trust policy language must facilitate this.
Agenda

- Introduction
- Terminology
- Centrality
- Query Algebra for Trust
  - Idea
  - Conventional Extensions
  - Centrality Operator
- Experiments
  - Setup
  - Results
- Effectiveness
- Conclusions
Effectiveness

- Which combinations of policies are successful and lead to stable and efficient (in the economic sense) systems?
- Which policies do humans actually use in different situations?
Effectiveness – Experiments (1)

- Human participants playing a game:
  - System-controlled entities interact with each other.
  - An entity may issue one service request per round.
  - Entity processing a service request is chosen randomly.
  - Each player may specify trust policy of ‘his’ entity.

- Costs/benefits:
  - processing a service request incurs costs,
  - issuing a service request and issuing feedback are free,
  - having a service invocation processed yields benefit,
  - payment based on performance.
Effectiveness – Experiments (2)

- Humans enter 'their' trust policy in natural language.
- We translate these statements to algebra expressions.
- Different experiments, with different information available.
Formulierung einer neuen Policy

Es wird gerade gespielt. Daher können Sie aktuell die Policy für ihren Stellvertreter-Rechner nicht ändern. Sie können allerdings weiter neue Policies formulieren und alte, bis auf die aktuell aktive Policy, entfernen.

Im folgenden Textfeld können Sie eine neue Policy in natürlicher Sprache formulieren.

Hier Ihre bereits formulierten Policies

Ich vertraue allen, außer sie haben meine Aufträge abgelehnt. AKTIV
Punktestandsentwicklung

In untenstehender Graphik sehen Sie die Entwicklung Ihres eigenen Punktestandes (blau) sowie das theoretische Maximum (rot), was Sie hätten erreichen können (das theoretische Maximum wird erreicht, wenn Sie alle Ihre Aufträge bearbeitet bekommen, selbst aber keine Aufträge bearbeiten).
Insights So Far

- Same policies may result in different rankings in different runs of the game.
- Players have not yet resorted to centrality measures to formulate policies – so far.
- The more information on others is available, the more 'hectic' players become.
Agenda

• Introduction
• Terminology
• Centrality
• Query Algebra for Trust
  ▪ Idea
  ▪ Conventional Extensions
  ▪ Centrality Operator
• Experiments
  ▪ Setup
  ▪ Results
• Effectiveness
• Conclusions

Klemens Böhm
"Towards Effective and Efficient Behavior-based Trust Models"
Summary

• The following questions are fundamental:
  - How to establish trust in distributed systems?
  - How do individuals (humans) interact in such settings?
  - How should a language look like that allows to formalize these issues?
  - How to evaluate expressions in the language efficiently?
Summary

- What have we done so far?
  - Collected various meaningful behavior-based trust policies from literature and our own attempts.
  - Motivation of an algebra-based approach for formulation of behavior-based trust policies.
  - Definition of a relational representation of behavior-specific knowledge.
  - Definition of a query algebra for trust
    - Listing of necessary operators from literature (basic operators from the RA incl. existing extensions),
    - Definition of a centrality operator to compute various centrality measures.
  - Presentation of experimental results.
  - Design of setting for effectiveness experiments.
Outlook

- Declarative language instead of algebra.
- Efficiency issues, in particular query optimization (equivalences of algebra expressions, selectivity estimation, view materialization, multi-query optimization).
- Distribution.
- Privacy. Relationship between privacy and economic success.