



Agent-Based Trust Model Involving Multiple Qualities

E. Michael Maximilien maxim@us.ibm.com
Munindar P. Singh singh@ncsu.edu



Outline

- **Background**
- **Problem**
- **Contributions**
- **Approach**
- **Service trust model**
 - Simple preferences
 - Complex preferences
- **Evaluation**
 - Setup
 - Results
- **Related Work**
- **Directions for Future Work**

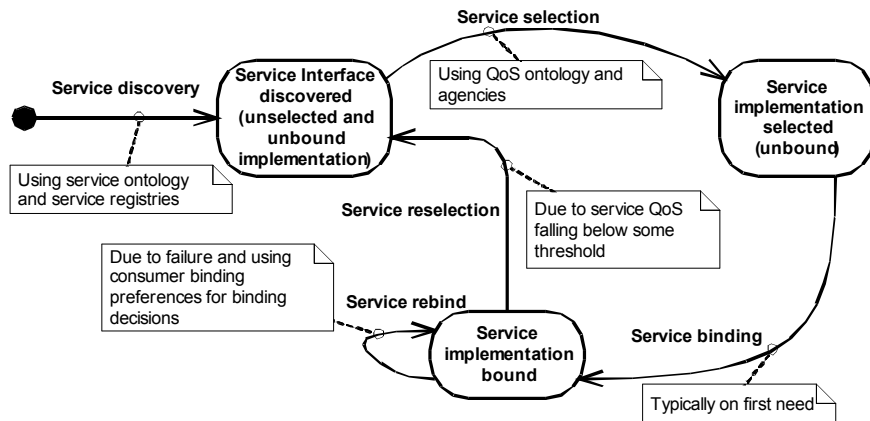
Background

- **Web services as agents and agents as Web services**
- **Web services**
 - Expose business functions
 - Embody relationships of interacting agents
- **SOA supports socio-economic systems of interacting businesses and evokes such systems as a metaphor for software development**
- **Key advantage of SOA is dynamism of resulting applications**
- **Interacting parties need to be able to rationally select each other**

Problem

- **SOA environment**
 - Various idiosyncratic service consumer QoS preferences
 - Multiple service providers with various trust profiles
 - Multiple service implementations with varying QoS profiles
- **Problem is how to select *best* service?**
 - Wide variety of service providers and implementations
 - Selection has to match service consumer's preferences
 - How to enable dynamic (i.e., runtime) selection of services using nonfunctional (QoS) attributes?

Problem – Service lifecycle overview



Approach

- **Assumptions**
 - Service agent environment is open, dynamic, and no central authority
 - Agents are free to share opinions on service interactions and information available to all
 - Agents report true assessments on QoS attributes that they gathered while interacting with service instances
- **Conceptual notion for trust**
 - Predictor of quality values for future usage of service instance
 - Aggregate value (reputation) corresponding to historical levels of QoS
 - Match between QoS needs and advertisements
- **Thesis is that selection has to be from on an empirical basis**
 - History of service providers and implementations
 - Sharing of quality values by service consumers

Contributions

- **Solution to service selection based on a QoS-based trust model**
 - Model of consumer QoS preferences
 - Model of provider QoS advertisements
 - New trust model with calculation that matches consumers to providers
 - **Improved trust model that takes into account service consumer's subtle QoS preferences**
- **Multiagent system framework to attach agents to service consumers and help them automate service interactions tasks**
- **Flexible ontology of QoS***
- **Emergent *self-adjusting trust* (Autonomic Computing)**
 - As services start behaving incorrectly they are purged from selection considerations
 - When services start behaving correctly again they are gradually reconsidered for selection

* E. M. Maximilien and M. P. Singh "A Framework and Ontology for Dynamic Web Services Selection" IEEE Internet Computing, 8(5):84-93, Sept. 2004.

Trust Model – Basic objects

- **Services**
 - Domain, interface, and set of implementations
- **Qualities**
 - Domain-specific and applicable to different domains
 - Values collected by service agents
- **Service consumers**
 - Service interface and quality preferences
- **Service providers**
 - Service interface, set of service implementations, and quality advertisements for each service implementation
- **Service selection problem**
 - Select service implementation with highest level of trust while taking into account consumer's quality preferences
 - Trust function allows us to rank service implementations

$$i = \arg \max_{i \in I_s} \{trust(i, \Phi_a)\}$$

Where $trust() : I_s \times \Phi \mapsto \mathcal{R}$ is a service trust function.

Trust Model – Simple preferences

- **Consumer quality preferences**
 - Quality value range and preferred value
- **Provider quality advertisements**
 - Quality value range and typical value
- **Compatibility of quality preference and advertisement**
 - Quality preference range should be within advertisement range
- **Quality reputation and trust**
 - Aggregation of normalized quality values collected with time discounting

▪ **Service selection**

$$R_{Qi} = \frac{1}{n} \sum_{k=0}^n q_k \delta^{-t(q_k)}$$

Let $Q_{min} = \min(\alpha_{min}, \pi_{min})$ and $Q_{max} = \max(\alpha_{max}, \pi_{max})$.
 Let $\vec{Q}_i = \langle Q_{min}, \alpha_{typical}, \pi_{pref}, Q_{max}, R_Q^{(i)} \rangle$.

$$qTrust(\vec{Q}_i, \pi_{pref}) = \text{moment}(\vec{Q}_i, \pi_{pref})^{-\frac{1}{2}}$$

where $\text{moment}(\vec{Q}_i, \pi_{pref}) \neq 0$

$$\text{serviceTrust}(i) = \sum_{\substack{Q \in \Phi_d \\ \pi \in \vec{Q}_d}} qTrust(Q_i, \pi_{pref})$$

$$\text{trust}(i_p, \Phi_d) = \text{serviceTrust}(i_p)$$

Service Trust Model – Complex Preferences

- **Quality tradeoffs**
 - Ordering of preferred qualities
- **Quality relationships**
 - Ontological: $Q_{Rel} = \{ \text{Opposite (O), Parallel (P)} \} \times \{ \text{Weak (W), Mild (M), Strong (S)} \}$
 - Statistical: Correlation between normalized quality values
- **Average quality relationships**

$$\varrho(Q_j) = \frac{1}{n-j} \sum_{m=j+1}^n \rho(Q_j, Q_m) \times (Q_j \bowtie Q_m) \text{ with } j \neq n$$

Q_j is average relationship between quality j and remaining qualities in ordering

- **New service trust function**

$$\text{trust}(i_p) = \frac{1}{n} \sum_{\substack{Q \in \Phi_c \wedge (\pi \in \vec{Q} \vec{\alpha}) \\ j=0}}^n w_j \times qTrust(Q_j, \pi_{pref}) \times [1 + \varrho(Q_j)]$$

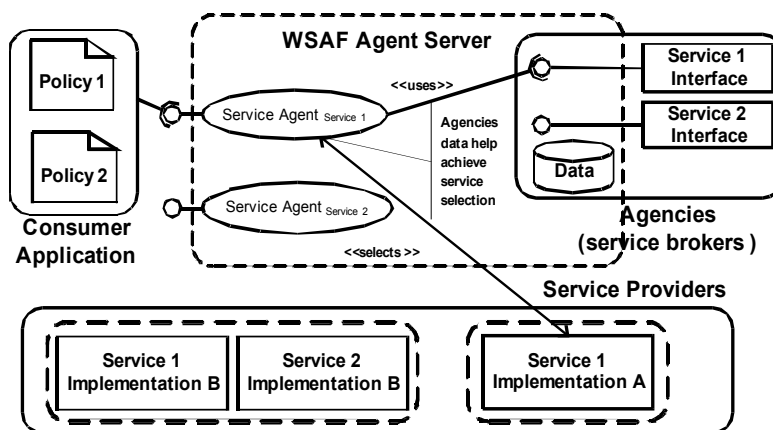
Weights are selected to match the consumer's quality preferences

Framework – Architecture Overview

- **Service Consumers and Providers**
 - Consumers express their QoS preferences
 - Providers express their QoS advertisements

- **Service Agents**
 - Bootstrapped with QoS ontology and proxy service for consumer (expose augmented interface)
 - Share data in agencies
 - Match QoS policies, calculate trust values, and select services for consumers
 - Implement binding policies

Framework – Architecture Overview (cont.)



Evaluation – Previous experiments and results

Experiment Design

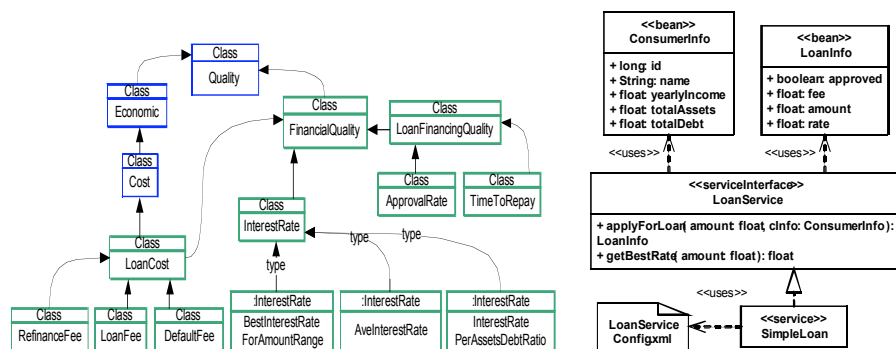
- Service interface is *SortService*
- Three types of providers: *Bubble*, *Merge*, and *Quick*
- Three qualities: *Availability*, *Reliability*, and *ResponseTime*
- Three types of consumers: *Mellow*, *Careful*, and *Rushed*
- Three pools of five consumers each
 - Consumers of each pool biased to one provider type
- Three pools of five providers
 - All providers except last member of each pool is doped
- Simulations (various iterations per simulation) with adjusted doping and agency data parameters
- Doping of a service implementation quality allow us to adjust service's quality response

Results*

- Without agency data, agents randomly select services
- With doping, emergence of autonomic characteristic
- Delay in doping also delay convergence

* E. M. Maximilien and M. P. Singh "Toward Autonomic Web Services Trust and Selection" In Proc. of ICSSOC 2004, Dec. 2004.

LoanFinancing Lower Ontology and LoanService Design



Evaluation – Setup

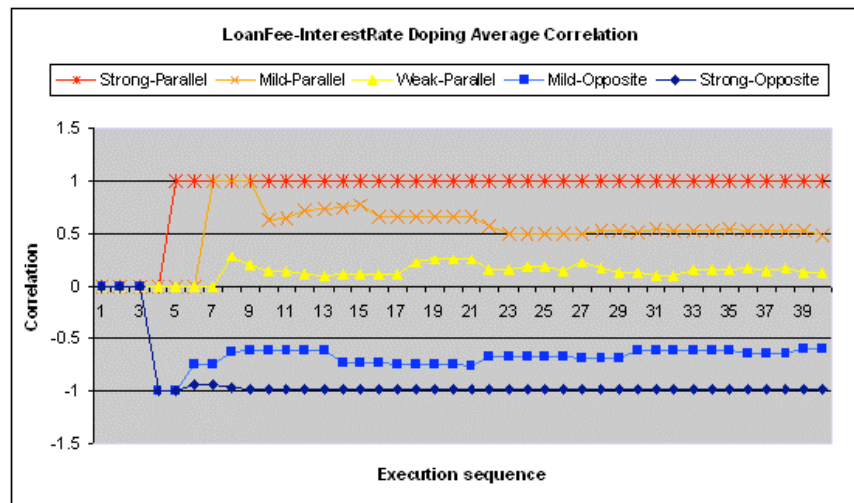
Design

- Domain is *LoanFinancing*
- Service is a simplified *LoanService*
- Three qualities are average: *InterestRate (IR)*, *LoanFee (LF)*, and *ApprovalRate (AR)*
- Five service implementations (same advertisements but varying Q_{Rel}) and group of consumers (same preferences)
- Service implementations have LF and IR doped according to table
- Add explorer agents (EAgents) to agents community
 - EAgent script to select all service instances in round-robin fashion
 - EAgent script task similar to other service agents

Quality relationships

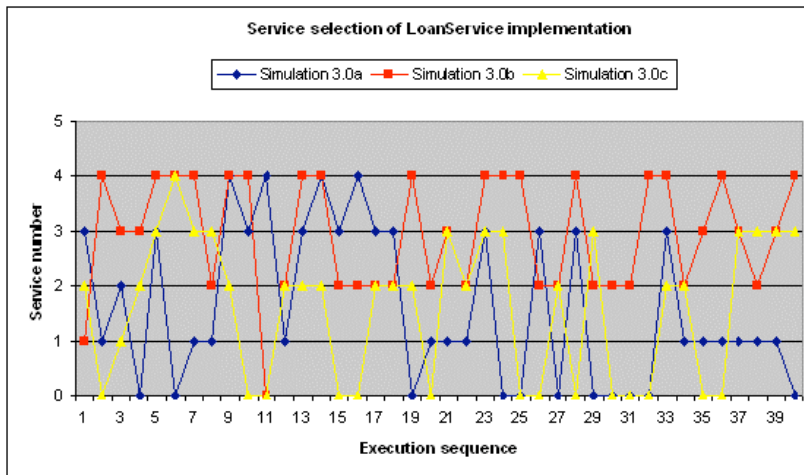
Sim #	Weights	LF, IR service doping
0	{1.00, 0.80, 0.70}	NA
1	{1.00, 0.80, 0.70}	{ SO, SO, SO, SO, SP }
2	{1.00, 0.80, 0.70}	{ MO, MO, MO, MO, SP }
3	{1.00, 0.80, 0.70}	{ SO, SO, MO, MO, SP }
4, 5	{1, $\frac{1}{2}$, $\frac{1}{3}$ }	{ WP, WP, MP, MP, SP }

Results – *LoanService* LF and IR average correlations



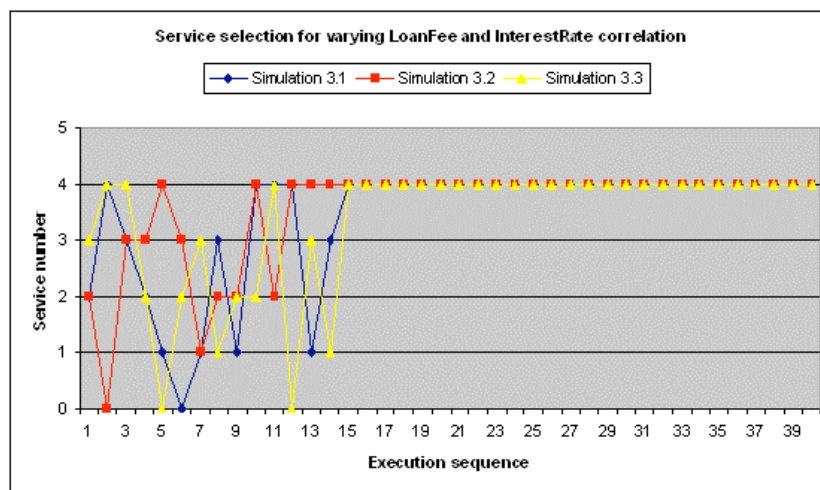
Results – Selection with simple preference

LF and IR service doping: { SO, SO, SO, SO, SP } (SP is best)



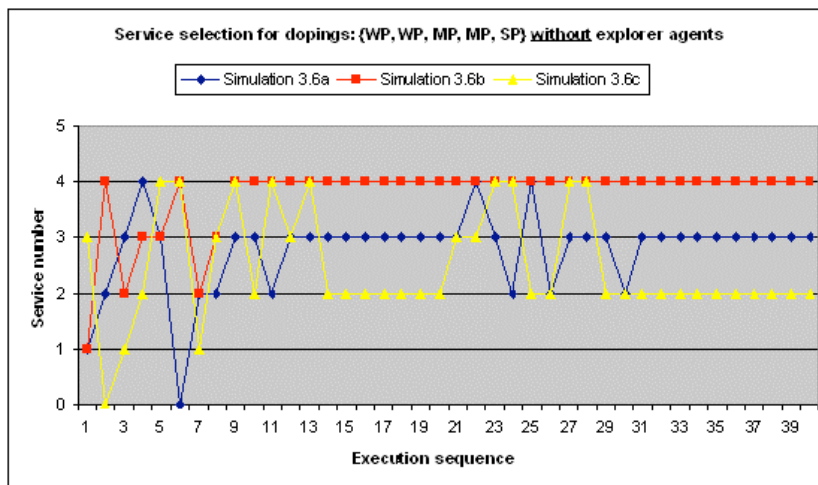
Results – Selection, new trust model, and varying Q_{Rel}

LF and IR service doping: { MO, MO, MO, MO, SP } (SP is best)



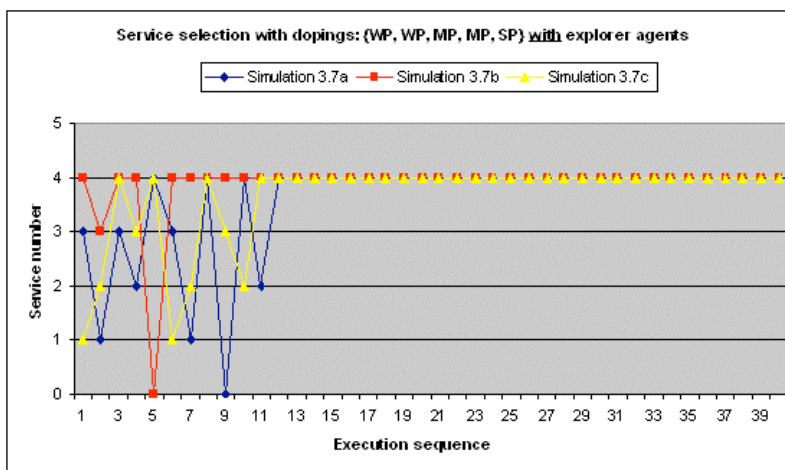
Results – Selection, varying Q_{Rel} w/o EAgents

LF and IR service doping: { WP, WP, MP, MP, SP } (SP is best)

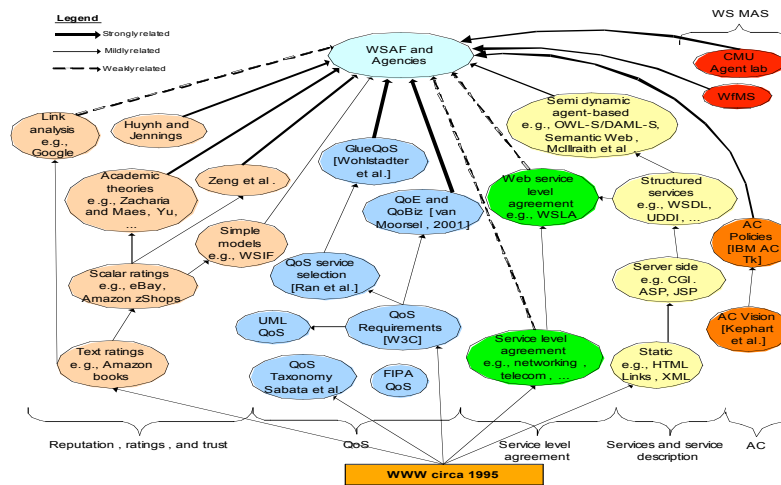


Results – Selection, varying Q_{Rel} with EAgents

LF and IR service doping: { WP, WP, MP, MP, SP } (SP is best)



Related Work



AAMAS 2005, Utrecht, The Netherlands

 July 28th, 2005

Directions

- **Trust and Trustworthiness**
 - Multiple (more than two) quality relationships
 - Include notion of *honesty* in experiments (difference between advertisements and reputation)
 - Incentive mechanism for sharing quality values
 - Incentive mechanism for preventing *free rider* service consumers
 - Expand policy language to allow for trust negotiation
 - Complement global trust (reputation) model with:
 - Local trust model
 - Capability to establish long running trust contracts, à la service level agreement (SLA)
- **Software Engineering**
 - Provider agents
 - Standardization of QoS ontologies
- **Autonomic Computing**
 - Analytical model for explorer agents

AAMAS 2005, Utrecht, The Netherlands

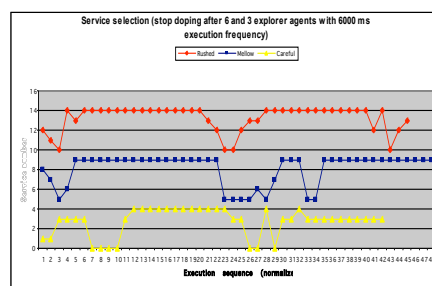
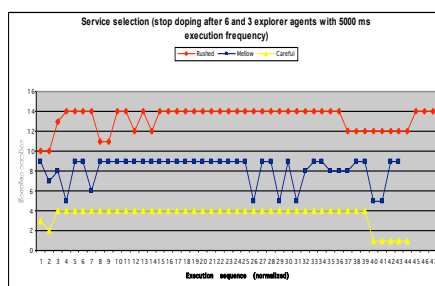
 July 28th, 2005

<http://maximilien.org>
(or google maximilien)

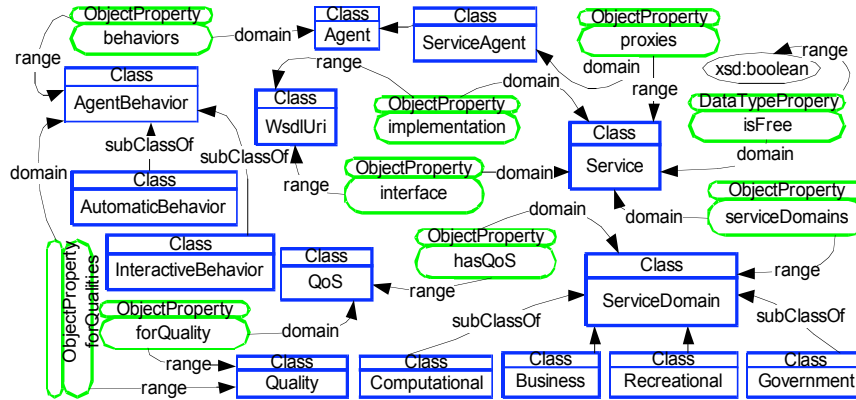
<http://www.csc.ncsu.edu/faculty/mpsingh/>
(or google singh)

Effect of Execution Frequency

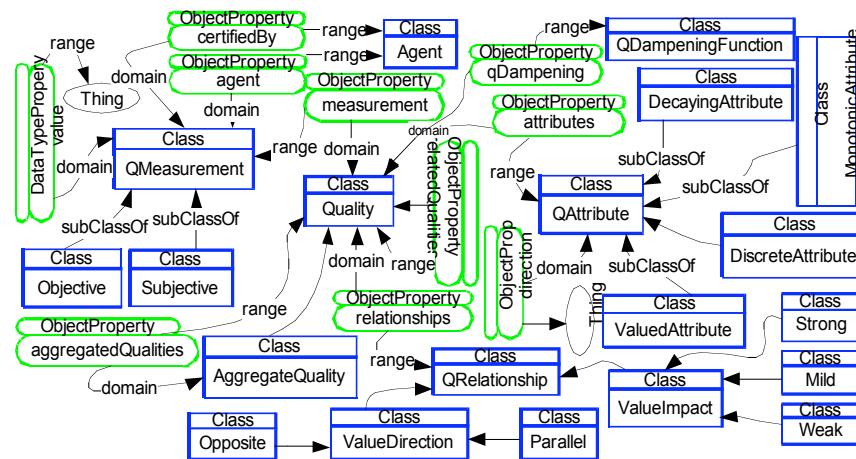
- **Decrease execution frequency**
 - Faster reselection of newly well behaved instances



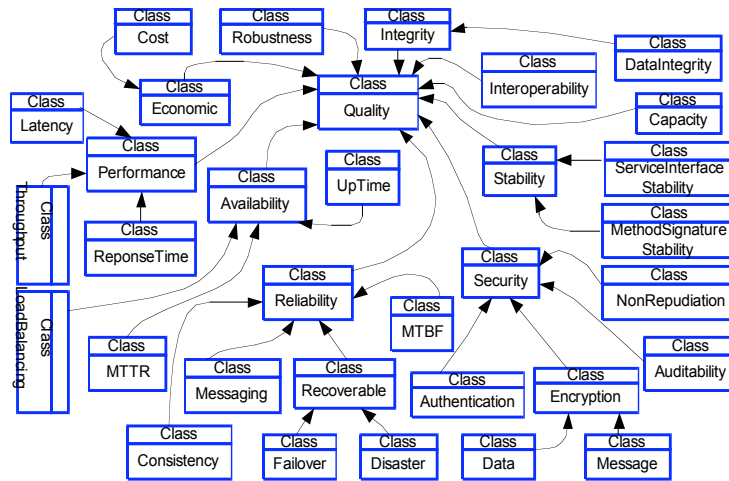
Backup: Framework – Service Ontology



Backup: Framework – QoS Upper Ontology



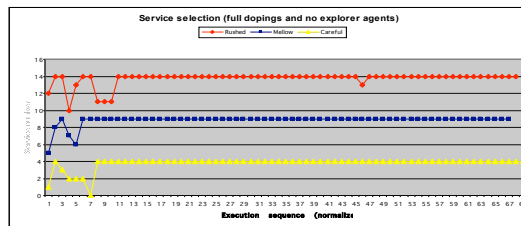
Backup: Framework – QoS Middle Ontology



Backup: Baseline Selection Without and With Explorer Agents (EAgents)

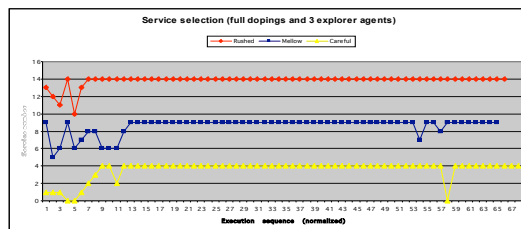
Service doping without EAgents

- Convergence to sole clean instance



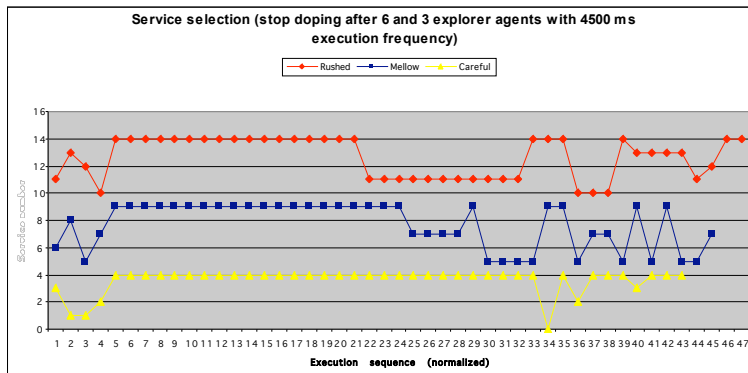
Service doping with EAgents

- Convergence to sole clean instance



Backup: Full Self-Adjusting Trust With Explorer Agents

- **Stop doping after six selections and three EAgents**
 - Convergence then random selection within pool



Backup: Doping Start, Stop, and Restart

- **Start doping, stop after six, and restart after 15**
 - Convergence, random selection, and convergence again

