



SOCS

A Model for Service-Oriented Communication Systems

Bernd Reuther
Technische Universität Kaiserslautern
AG Integrated Communication Systems

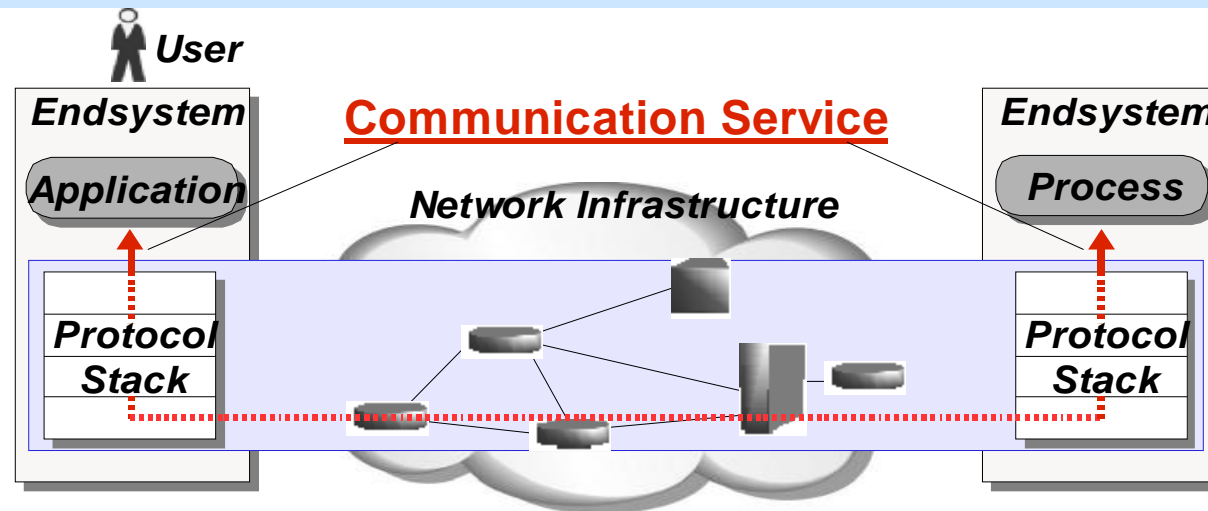
Email: reuther@informatik.uni-kl.de



Motivation

- Protocols are continuously enhanced and newly developed like any other software
- How to utilize changed/new protocols ?
 - If part of application or middleware !
 - If part of an application platform, e.g. transport protocols ?
- Problem of application developers:
which protocols should/can be used?
 - Common solution: use widely available standards
- ▶ Make protocols transparent for applications,
use communication services instead!

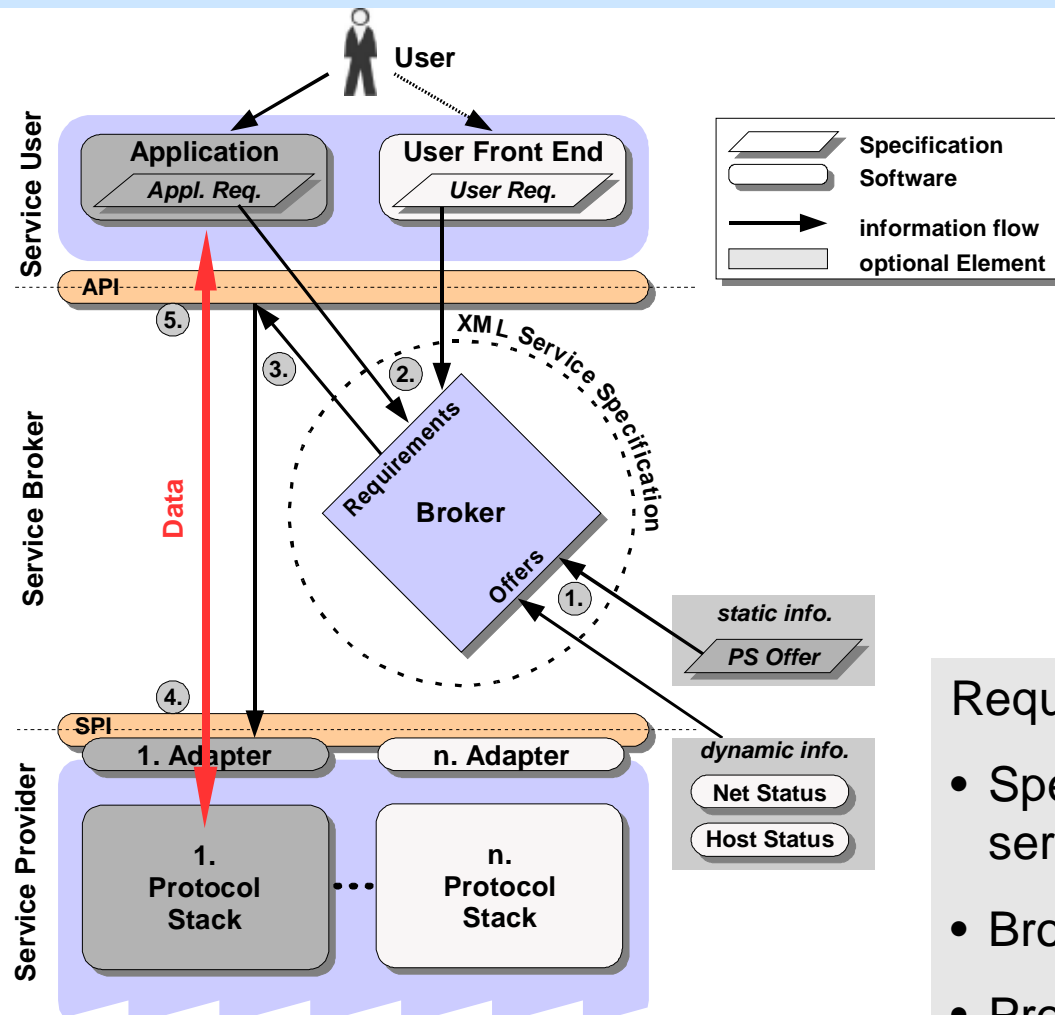
Communication Service



- Communication services are characterized by offering the exchange of data between two or more entities
 - Here communication services of the transport layer are considered only
- Communication services depend on
 - Protocols
 - Resources (Endsystem & Network-Infrastructure)
 - What is known about the application (behavior) and the user (authorization)



SOCS Model



1. Specification of offered services
2. Specification of requested services
3. Broker services
4. Access and
5. use the service

Required:

- Specification of communication services
- Broker
- Protocol independent API



Requirements on Service Specification

- Support brokering
 - Distinguish
 - what is necessary / guaranteed
 - what is desirable / not guaranteed
 - Optimize configuration of transport service provides (TSP)
 - User requirements
 - Information about users
 - Status of endsystem and network
- Should be flexible
 - Extendable according to what is specifies (semantic)
 - Adaptable to available information, i.e. support fine and coarse grained specifications



Specification of Services

- The interface is fixed: specify services by sets of properties only
 - Inherent properties (necessary / guaranteed)
 - Determine usable services
 - Do not rate a service
 - Qualitative properties (desirable / not guaranteed)
 - Determine optimal services
 - Explicitly rate a service
- Specification of a single property
 - Reference semantic by an URI
 - Values
 - Inherent properties: upper and lower bounds
 - Qualitative properties: rating
- Specification of all services of a TSP
 - Influence of configuration
 - Influence of dynamic conditions/constraints



Specification of Inherent Properties

- In Offers
 - URI specifying P_i
 - $lb, ub \in \mathbb{R}_\infty$ with $lb \leq ub$
 - Meaning: P_i is valid $\forall x \in [lb, ub]$
- In Requirements
 - URI specifying P_i
 - $lb, ub \in \mathbb{R}_\infty$ with $lb \leq ub$
 - Meaning: $\exists x \in [lb, ub]$ so that P_i is valid
- ▶ An offer for P_i matches a requirement for P_i , if the corresponding intervals overlap



Rating of Qualitative Service Properties

- Simple approach: quote expected absolute values
 - Example: end-to-end delay will/should be 30 ms
- Problem: in practice this is often impossible/inadequate
 - Properties depend on factors not known a priori
Example: available resources, status of devices or user behavior
 - There are no common units
Example: how to express quality of encryption?
 - Within requirement specifications:
“as good as possible” is more adequate than
“a value of x must be achieved”
- Common practice today: decision without knowing precise values
 - Rating occurs implicitly when a programmer “selects” a protocol
 - Protocols are compared to each other and not according to absolute values
Example: choose UDP instead of TCP when *low delay* and *low jitter* have a high priority, even though a *low loss* rate would be nice also

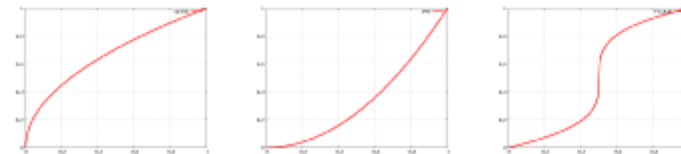
▶▶▶ Abstract description of Quality

- Specify quality of a property P_i by $qr_i \in [0, 1]$
 - $qr_i = 0 \rightarrow$ “worst considered quality”
 - $qr_i = 1 \rightarrow$ “best considered quality”
- Definition of qr may be based on
 - objective criteria (preferred), e.g. benchmarks
 - subjective criteria otherwise \leftarrow *Note: this is common practice today!*
 - at least an interpretation of “worst” and “best” is required
- In specifications of service offerings
 - $qr_{i,l}$ and $qr_{i,k}$ enables comparing TSP_l and TSP_k according to property P_i
- In specifications of service requirements
 - weights are used to balance qr_i against qr_j



Objective Criteria for Quality Description

1. Rate a TSP according to a property P_i
 - $Q_{P_i}: \text{TSP}_k \rightarrow \mathbb{R}$ an objective criteria
 - Each Q_{P_i} uses specific unit U
2. Define “best” and “worst” considered quality
 - $q_0, q_1 \in \mathbb{R}$ according to specific unit U
3. Linear mapping of rating to $qr' \in [0, 1]$
 - $R: [q_0, q_1] \rightarrow [0, 1]$ with
 $q_0 \leq \min_k (Q_{P_i}(\text{TSP}_k))$ and $q_1 \geq \max_k (Q_{P_i}(\text{TSP}_k))$
 - distinguish $q_0 < q_1$ and $q_0 > q_1$
4. Optional adaptation if qr' does not describe linear development of quality
 - $qr = f(qr')$ where $f(x)$ is non linear and invertible, examples:

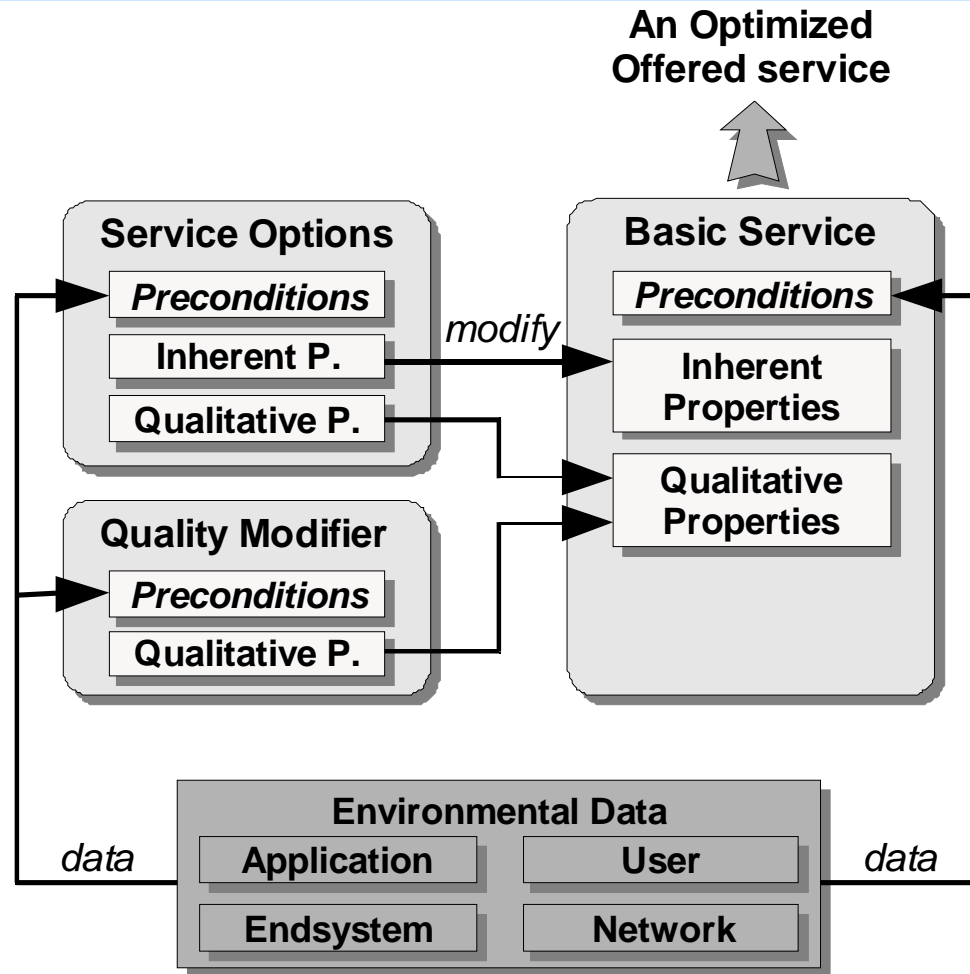


▶▶▶ Specification of Qualitative Properties

- In Offers
 - URI specifying P_i
 - $q_0, q_1 \in \mathbb{R}$ according to specific unit U, optional
 - $f(x)$ optional
 - qr the abstract quality description
- In Requirements
 - URI specifying P_i
 - $q_0, q_1 \in \mathbb{R}$ according to specific unit U, optional
 - $f(x)$ optional
 - qw weight of P_i in relation to other properties
- ▶ Using q_0, q_1 in offers indicates objective criteria, using q_0, q_1 in requirements enables adaptation of “worst” and “best” considered quality



Specification of TSP



Basic Service ~ service of a TSP in default configuration

Service Option ~ optional configuration, may be combined

Quality Modifier ~ influence of environment on quality

Preconditions ~ test for applicability (boolean expression)



Brokering

1. Join application and user requirements
2. Find all TSP that fulfill all necessary requirements
3. Optimize configuration according to quality
4. Use service(s)
 - a) Active initiation: use the best service
 - b) Passive initiation: use several of the best services (plus standard TSPs if appropriate)



Summary and Outlook

- Main challenge how to specify services
- Important to be highly flexible
 - Using sets of properties with fixed syntax only, are extensible according to semantic (i.e. what is specified)
 - An abstract quality description enables to compare services among each other and supports
 - subjective criteria
 - objective criteria
- Objective criteria are preferred to rate services.
But this demands (common) benchmarks for protocol properties
- Nevertheless dynamic selection of protocols at runtime, already enables to use alternative protocols transparently for applications



Thank you for your attention



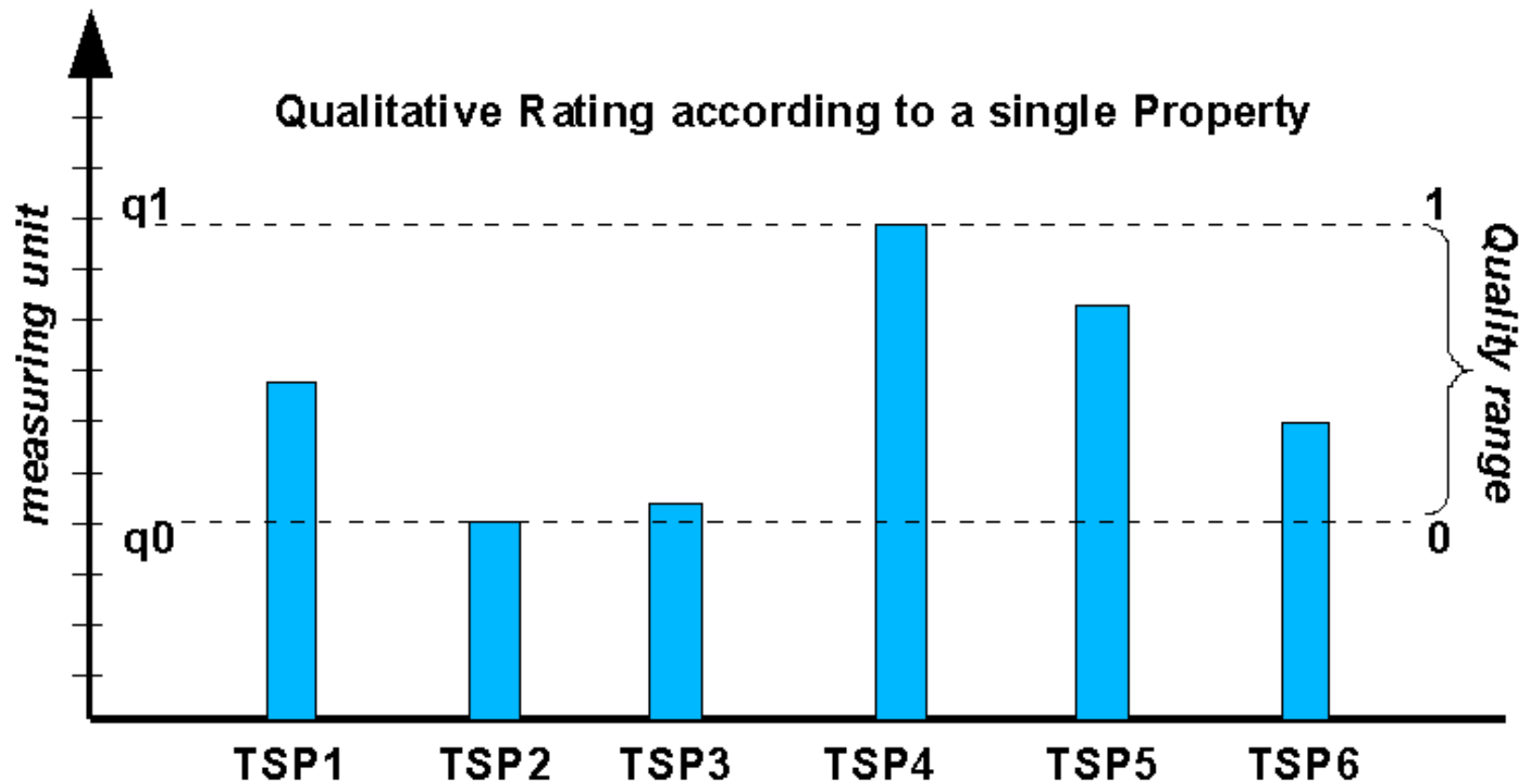
01. September 2006



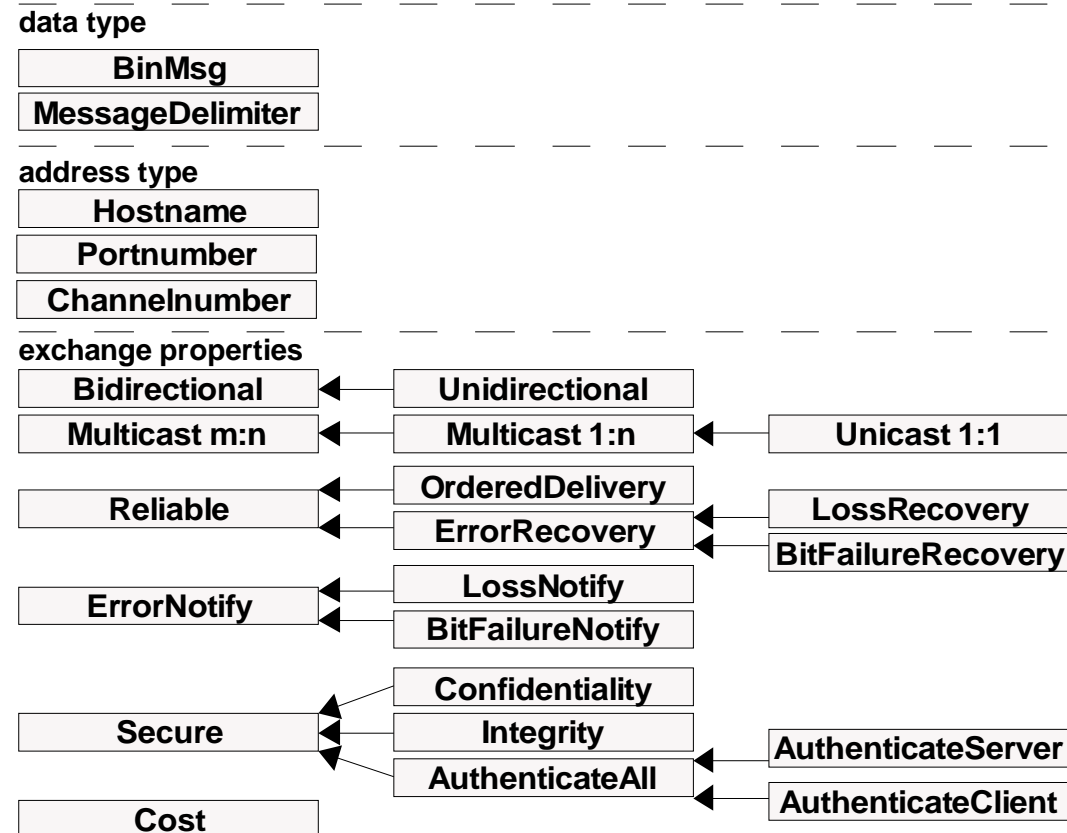
<http://www.icsy.de>



Example



Inherent Properties





Qualitative Properties

