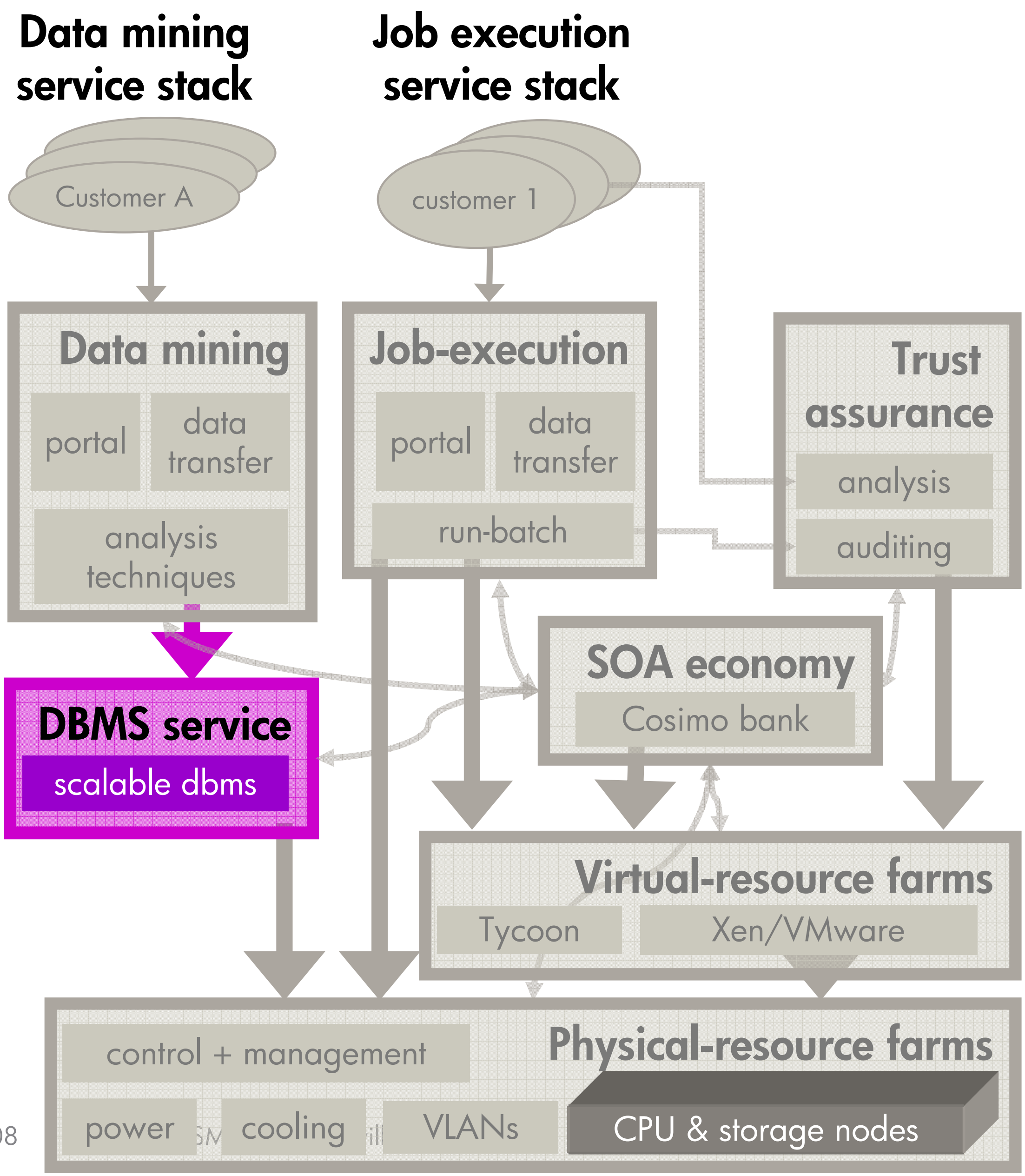
Utility functions, prices, and negotiation

john wilkes, hp labs

SMDB'08, Cancun, Mexico

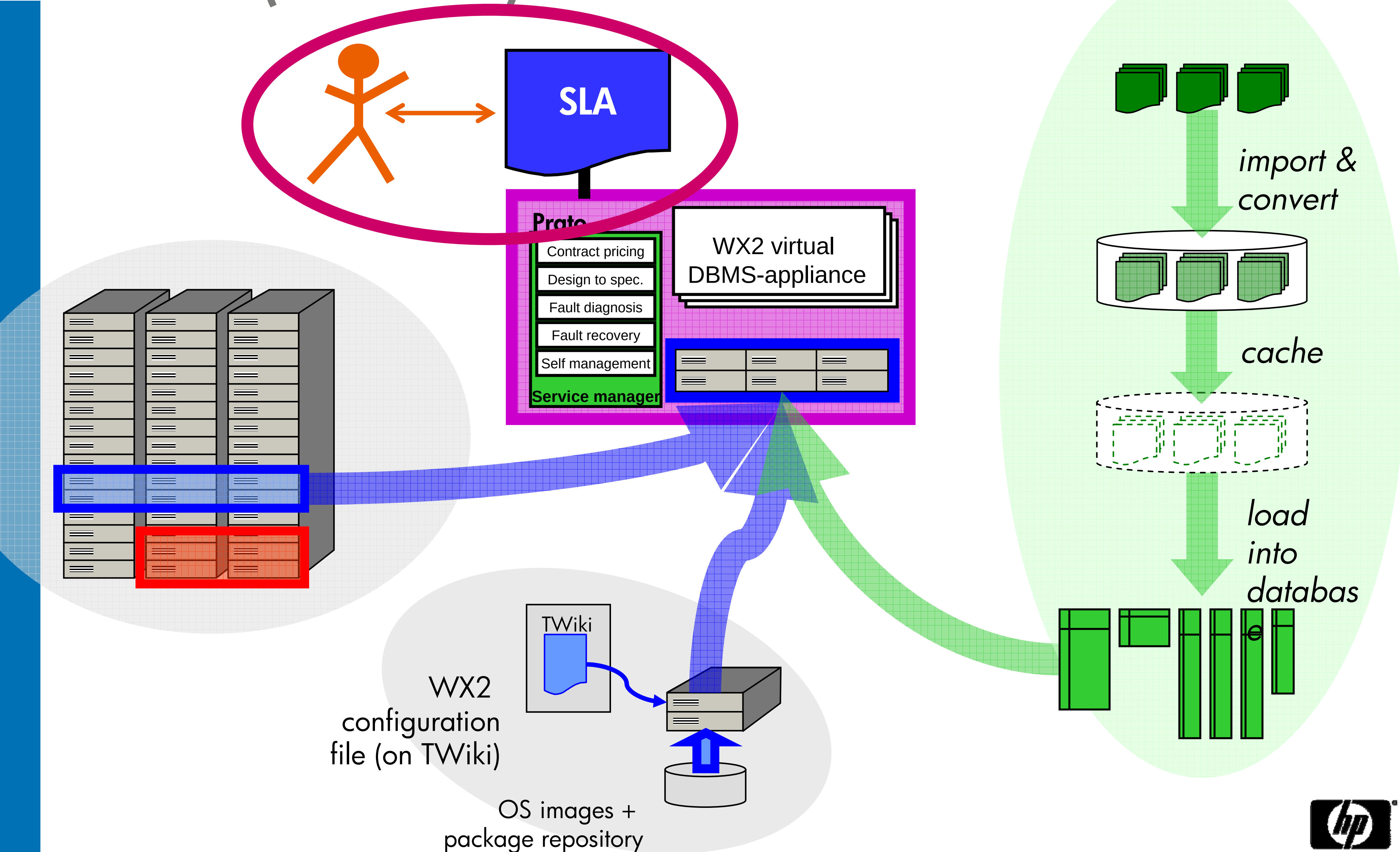
Context

a sample Tuscany ecosystem



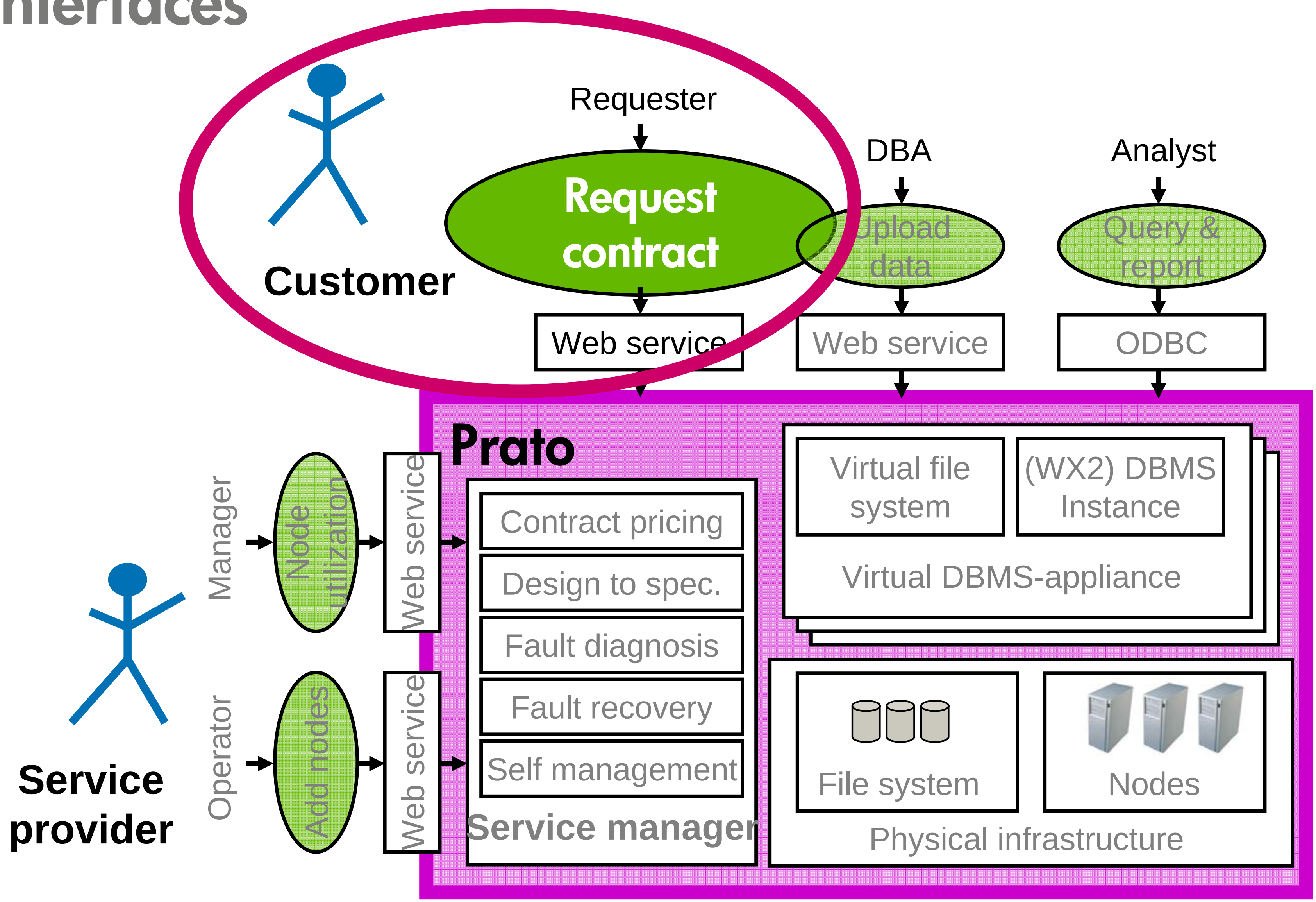
each is an example of an independent service provider

Prato: virtual-database-appliance-on-demand service (VDAaaS?)



Prato

interfaces



Utility functions, prices and negotiation

communicating business intent to (automated) IT

- What makes automation easier?
  - a single metric to optimize against
- What do business care about?
  - money!
- What is money a proxy for?
  - utility ➔ a measure of "goodness"

How to avoid unpleasant surprises?

- **Service Level Agreements (SLAs)**

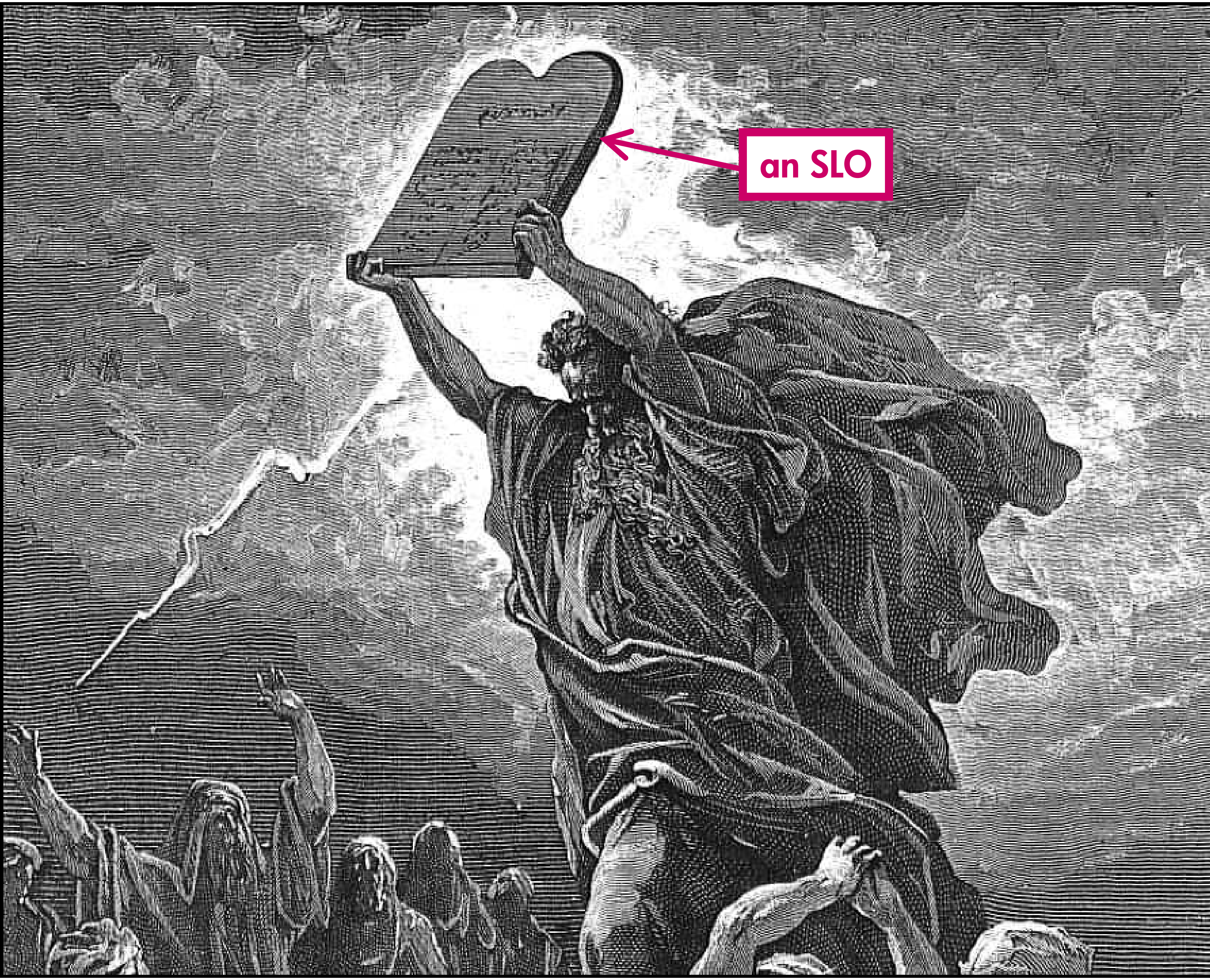
SLAs
as contracts

- a **Service Level Agreement (SLA) is a contract**
  - between mutually suspicious parties
    ➔ if you care about something, put it in the SLA!
  - agreement can be explicit or implicit

- assumptions
  - machine readable, can be reasoned about
  - two-party (other variations possible)

SLAs
WS-Agreement basics

1. Context
   – who, why, duration
2. Service terms
   – what service is offered, and how it is offered
3. Guarantee terms
   – scope + conditions (e.g., time of day)
   – Service Level Objectives (SLOs)
   – penalties and rewards


an SLO

Outcome-based pricing

a better way

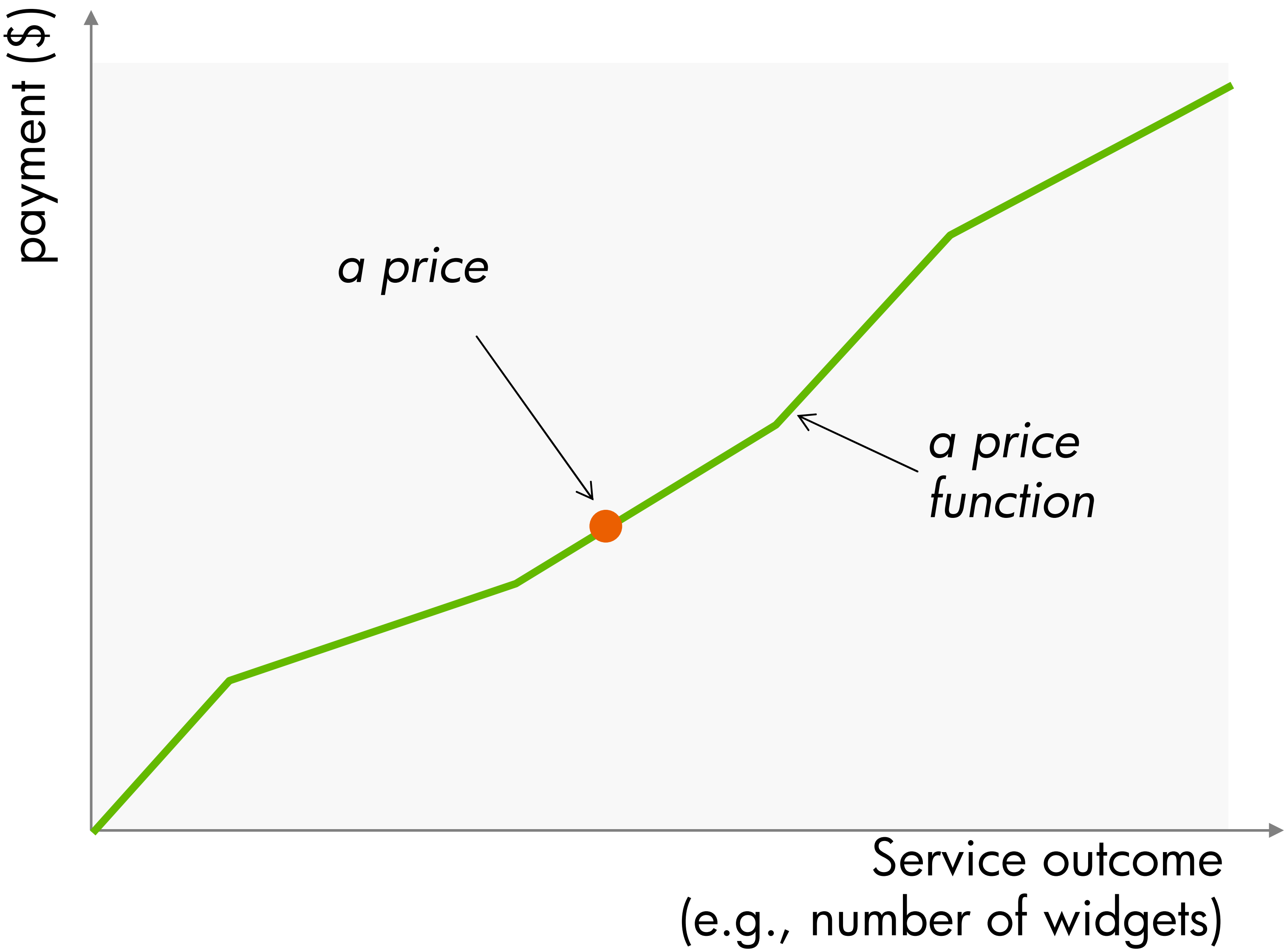
specify consequences, not behaviors!

- replace all the SLA guarantee terms
- by a single **price function**
- that specifies how much the service provider is paid for each possible **outcome**

- omitting all details of *how* the outcomes are achieved

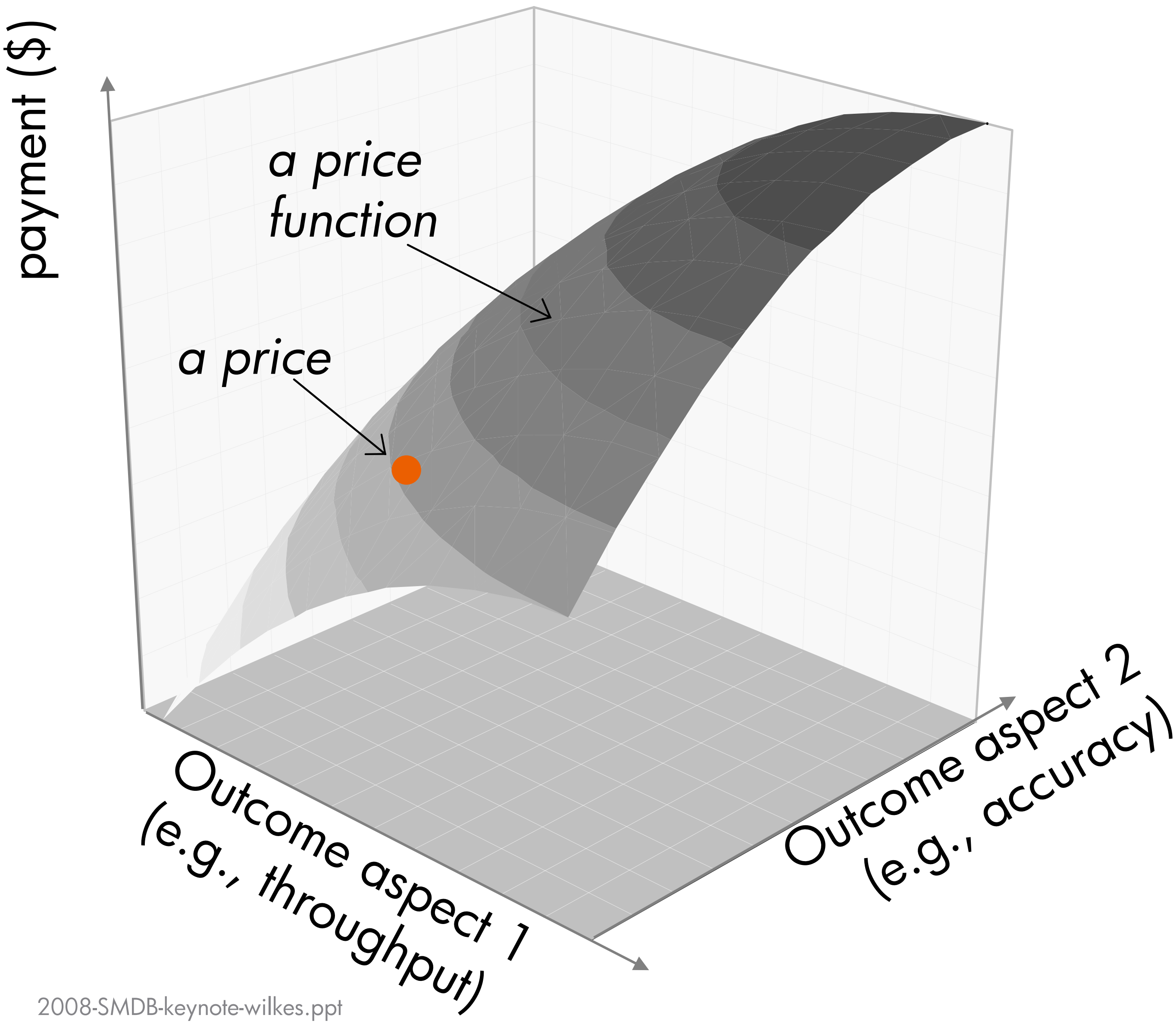
Outcome-based pricing
A price and a price function

specify consequences, not behaviors!



Outcome-based pricing
A price and a price function

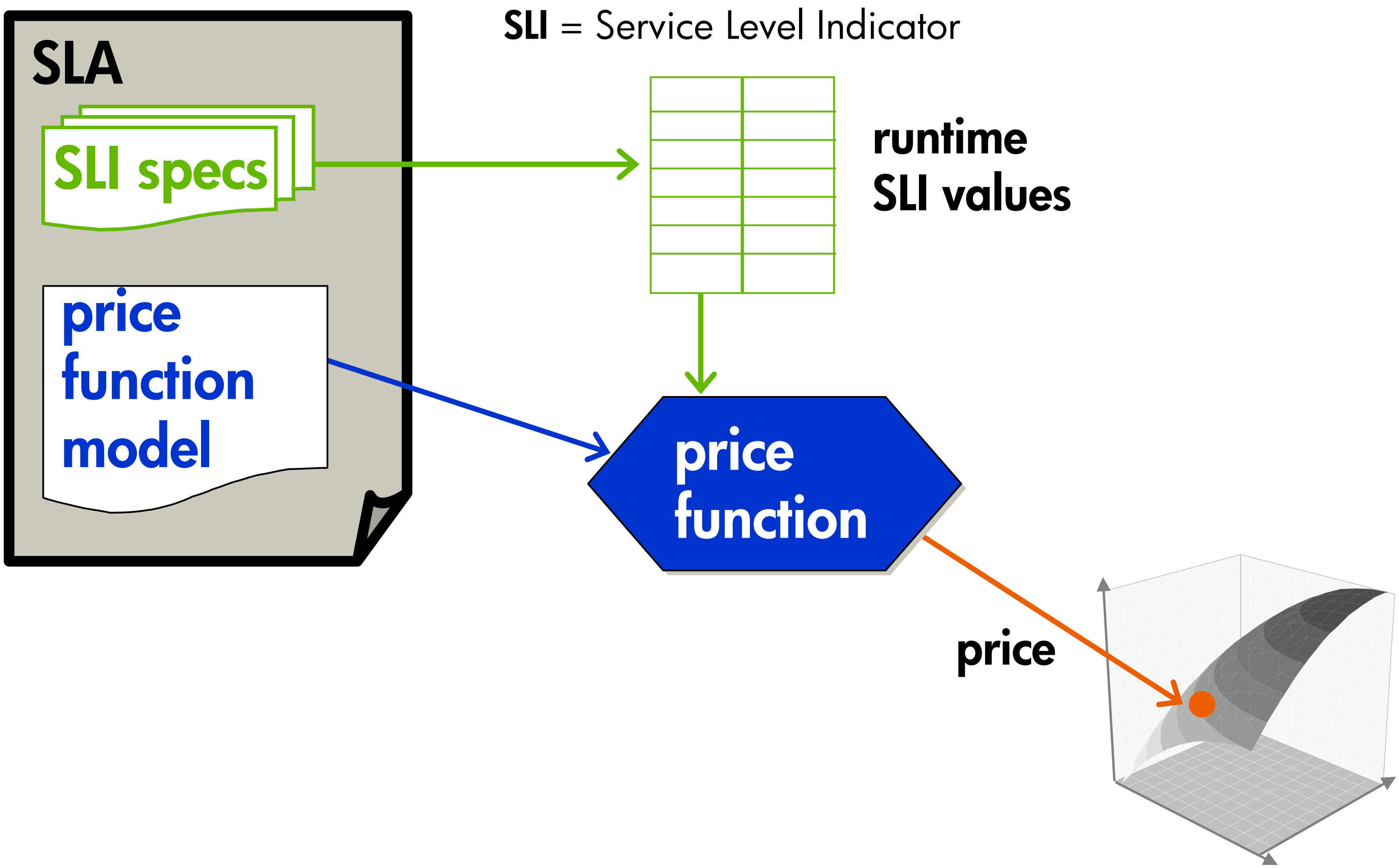
specify consequences, not behaviors!



Outcome-based pricing

what if … price functions

specify consequences, not behaviors!



Outcome-based pricing
what if … price functions

specify consequences, not behaviors!

- exactly one price function in each SLA
  - *function(set of metrics/parameters)* → a *price*

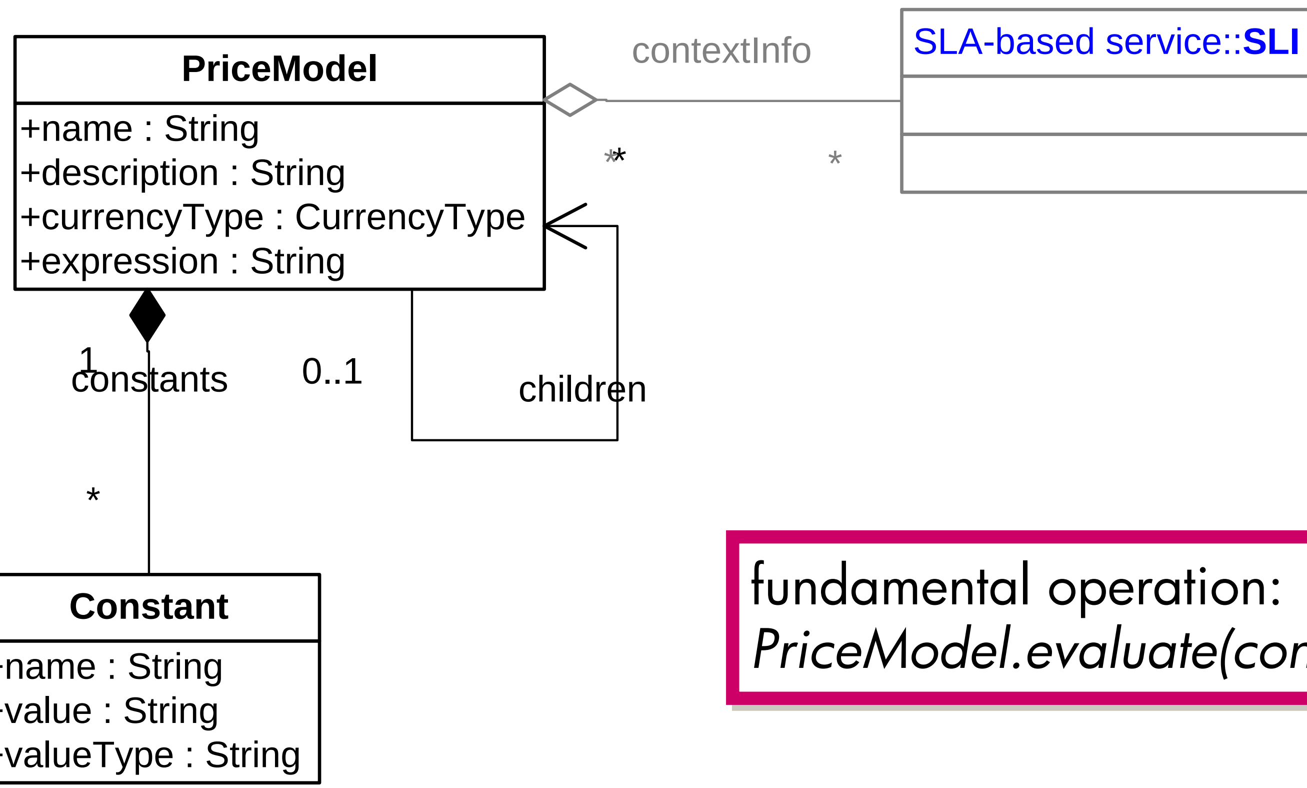
- evaluated by:
  - service provider to work out what to charge
  - client to predict what might happen
  - third party to audit

SLA structure

what if … price functions

specify consequences, not behaviors!

- typically structured as a tree
- summation, discounts, library-of-parts, etc



fundamental operation:
PriceModel.evaluate(context)

SLA structure
what if … price functions

specify consequences, not behaviors!

- benefits:
  - either side can predict price for given outcome
  - can be audited by 3rd party
  - consequences can be explored automatically
- requirements:
  - standalone, deterministic
  - flexible
  - well-defined, visible inputs: SLIs (Service Level Indicators)

SLA structure
what if … price functions

specify consequences, not behaviors!

Setting prices

- *Pricing* is the strategy used for setting prices
  - ***pricing strategy* → emits *price function***
  - e.g., loss leader; bundling, differential, …

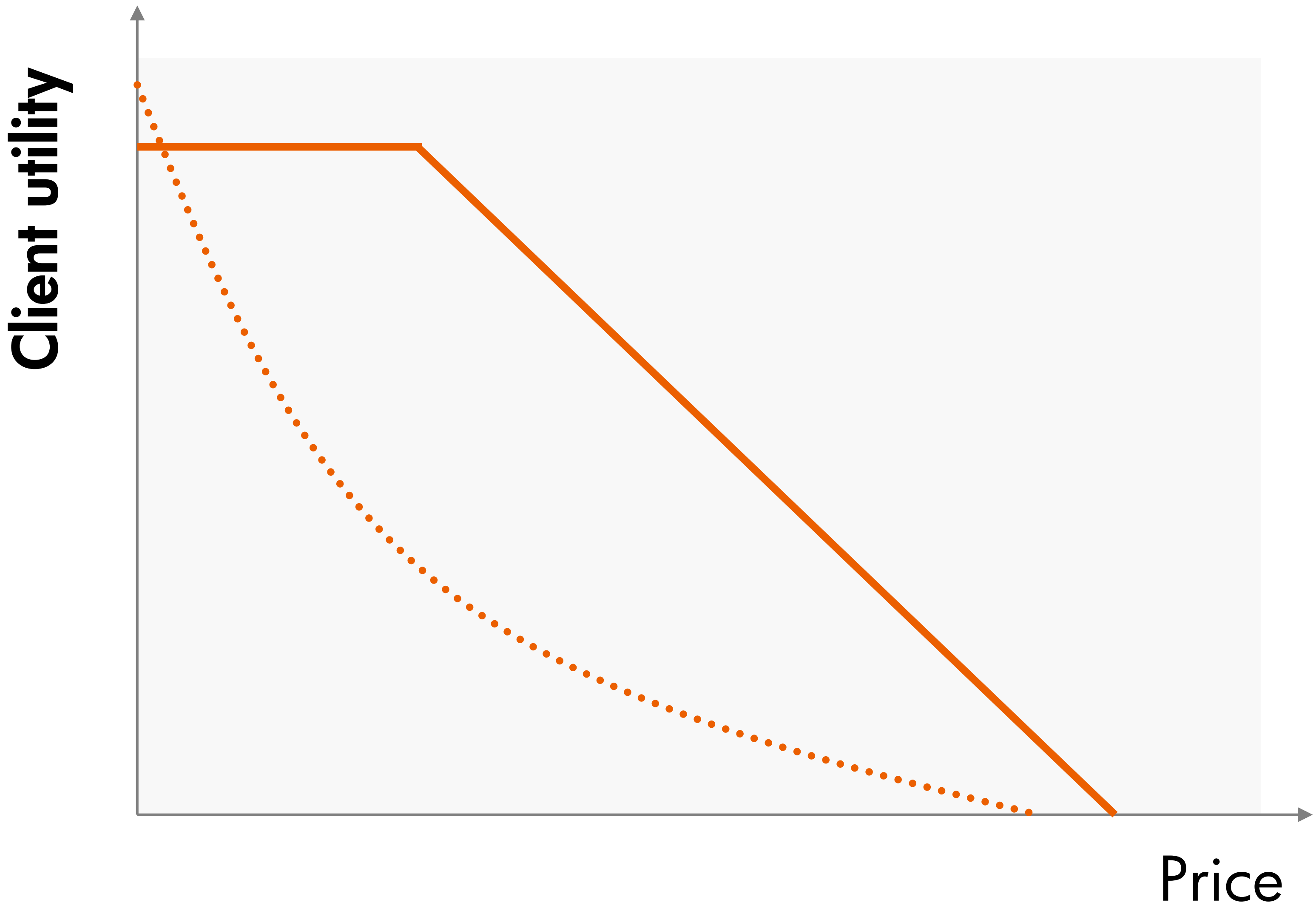
- Competition, price pressures → sets max prices
- Customer utility → limits what customers will pay
  - demand-elasticity curves

Utility

- Utility = local measure of goodness
  - more is better!
- Arbitrary, local units
  - cannot be:
    - compared across agents
    - normalized across agents
    - summed across agents
  - can be rescaled and re-normalized
    - e.g., > 0 ➔ win, <0 ➔ lose

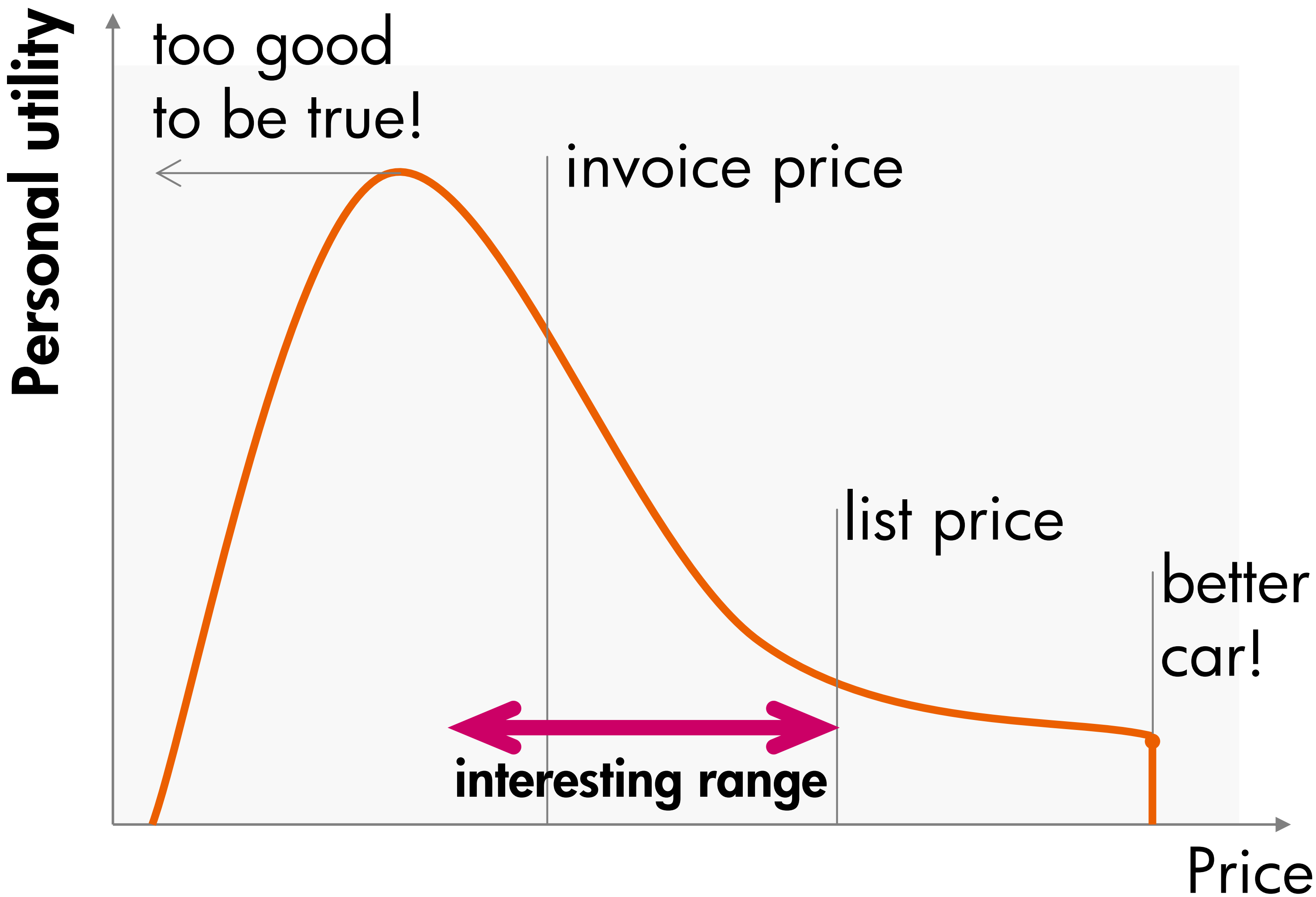
Utility for a fixed outcome

Some simple forms

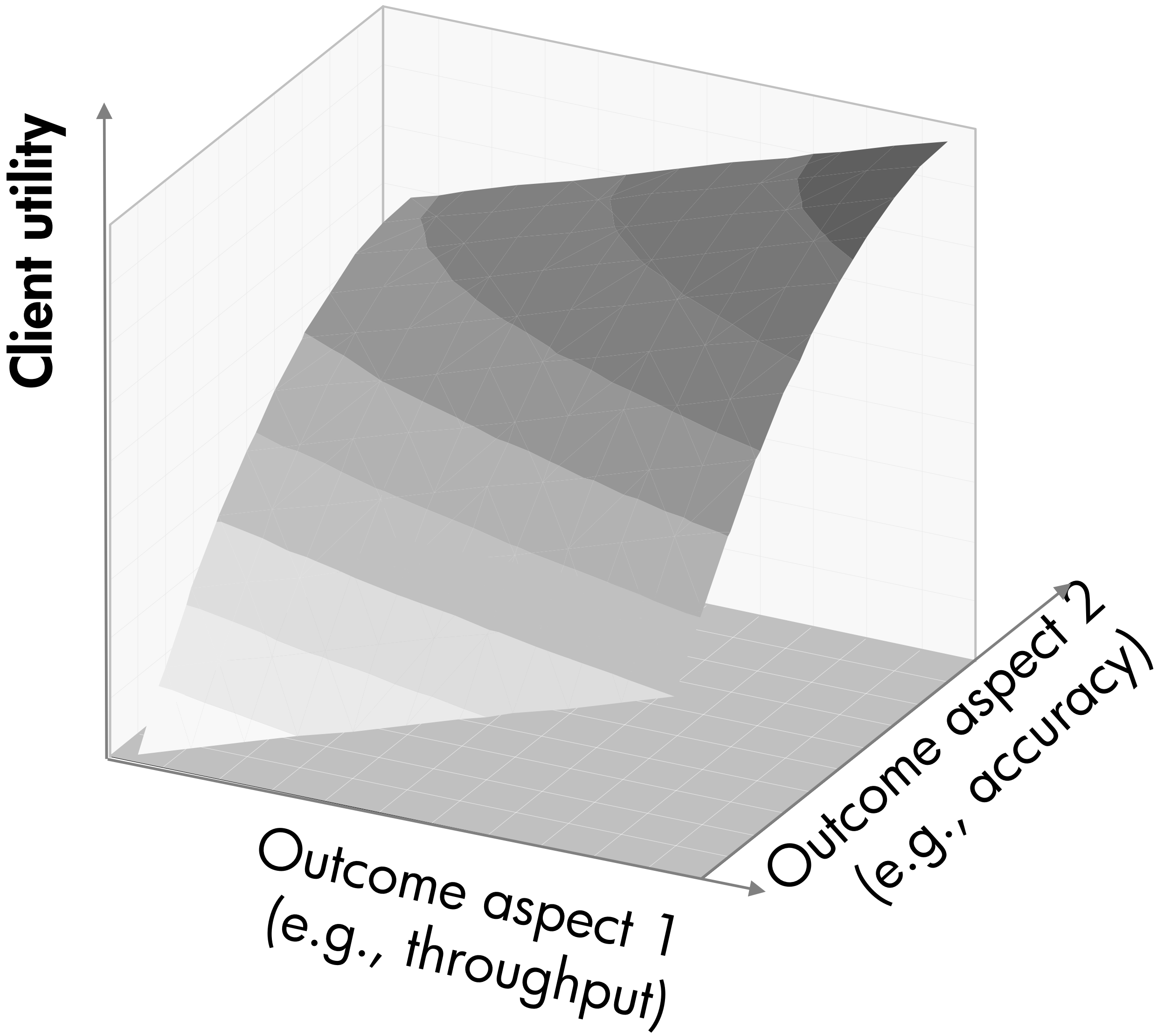


Utility for a fixed outcome

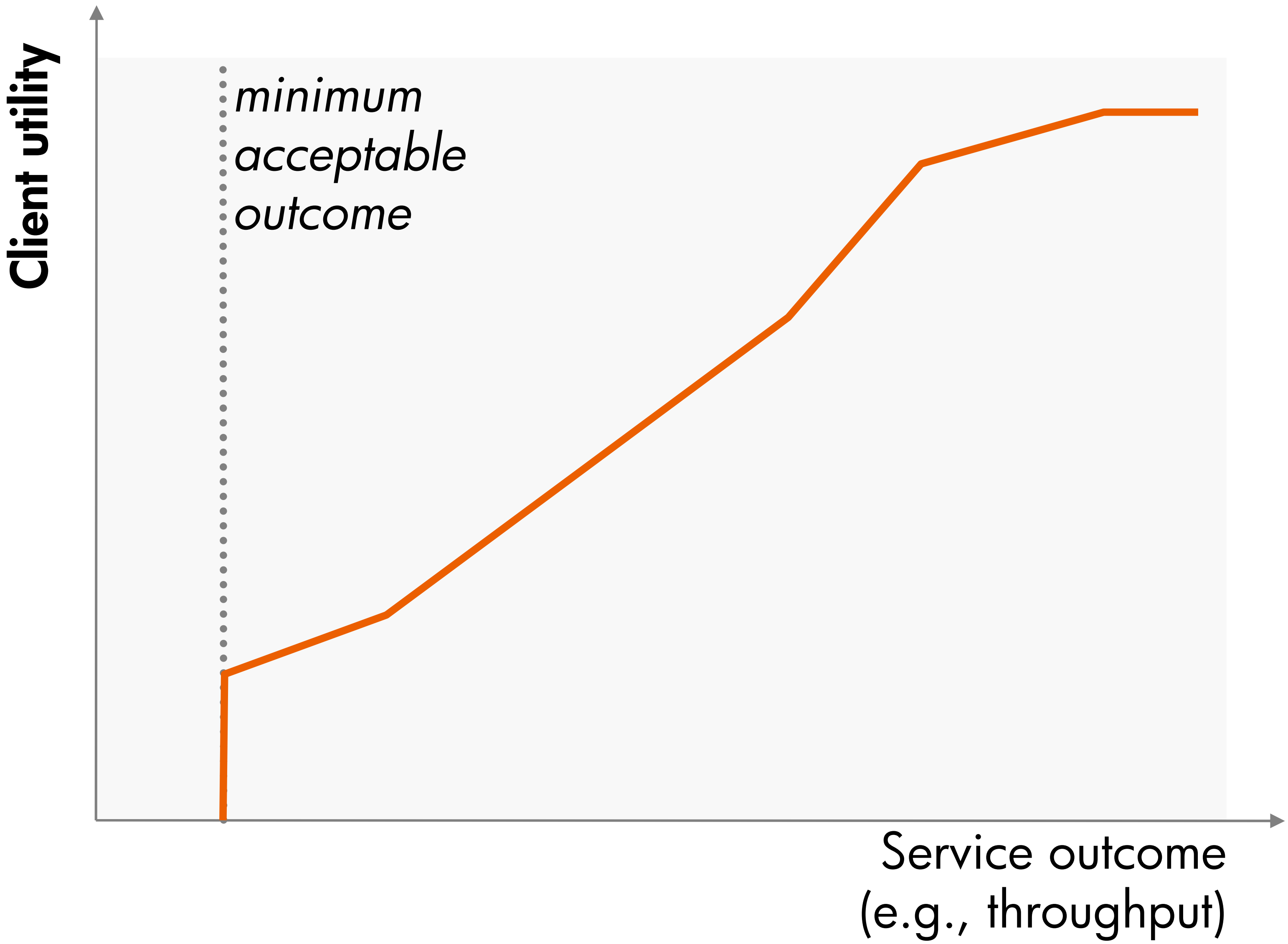
Example: buying a car



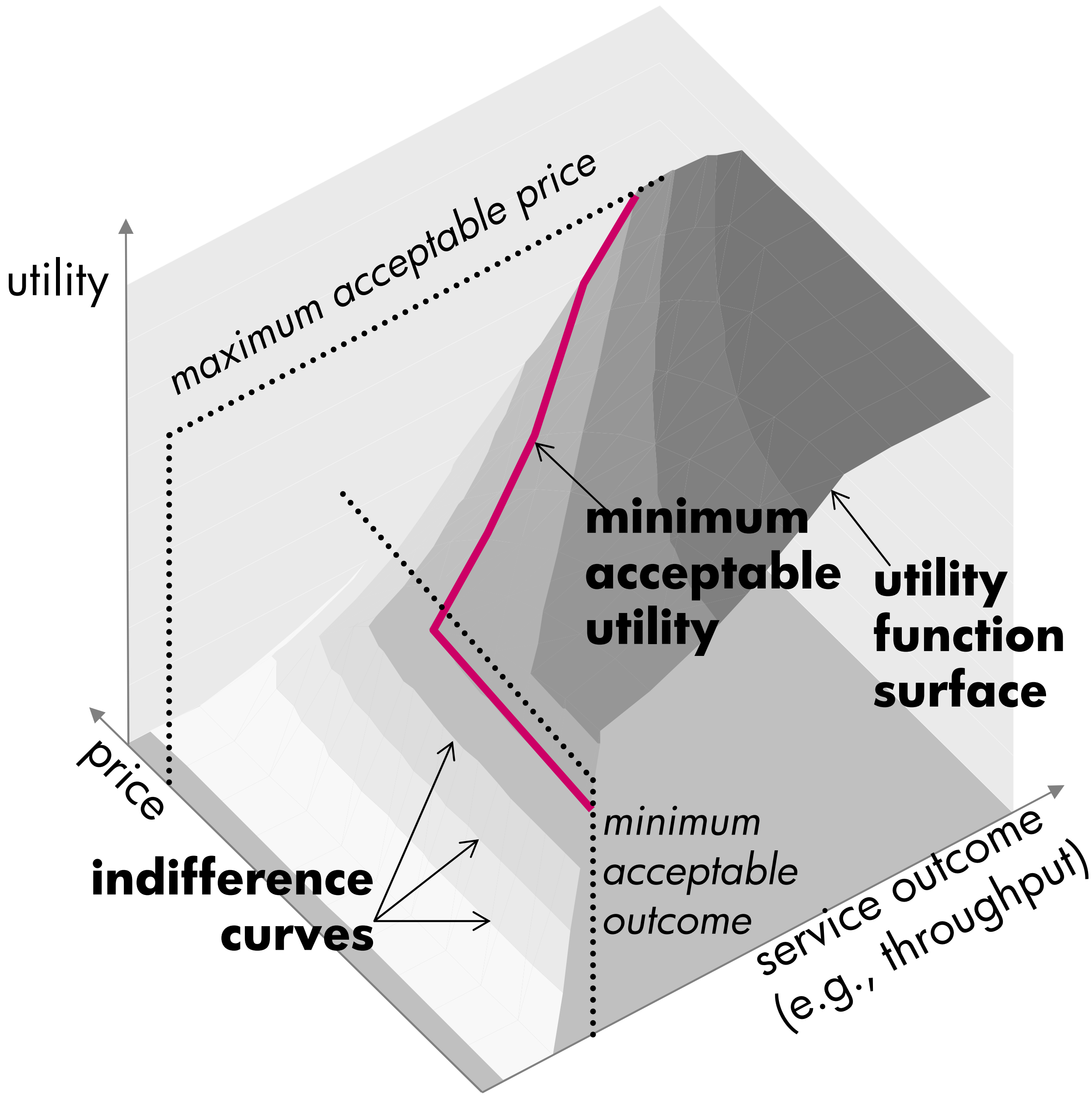
Client utility for 2 outcomes



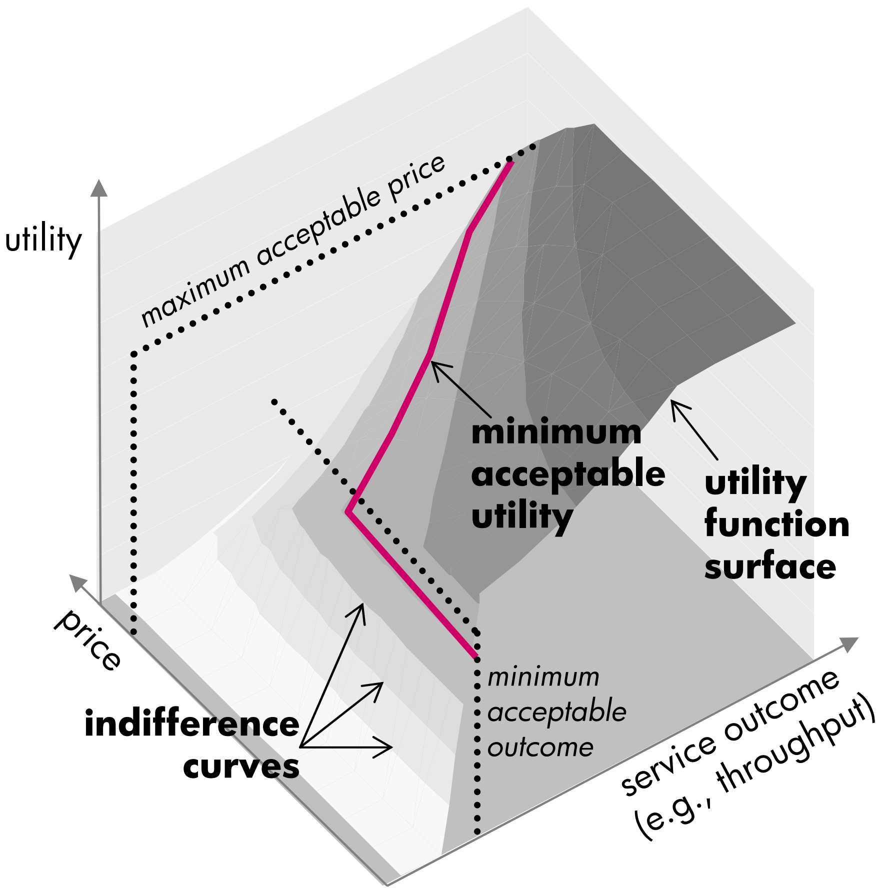
Client utility for 1 outcome

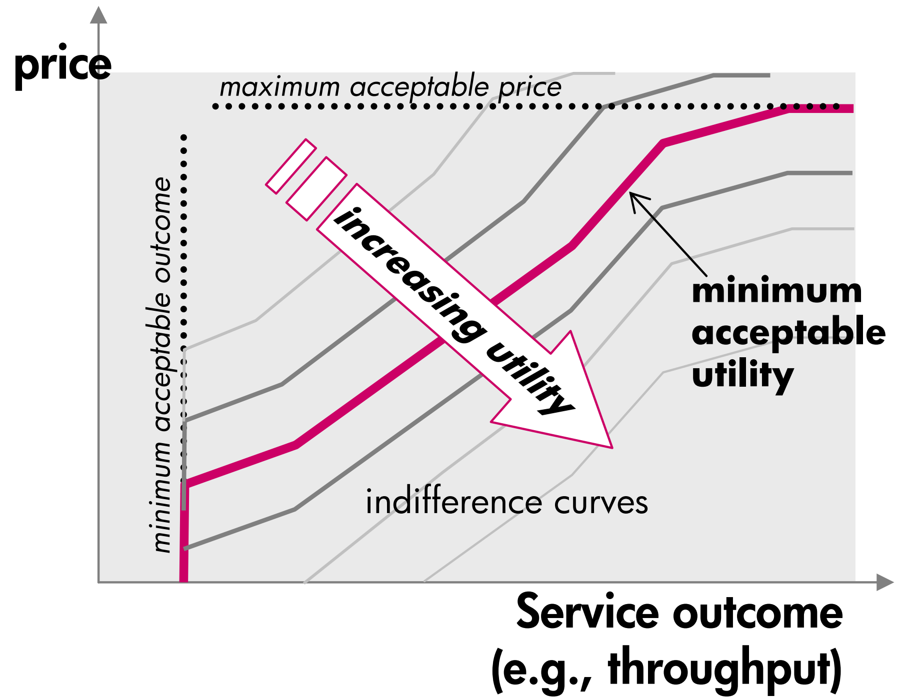


Client utility
1 outcome + price

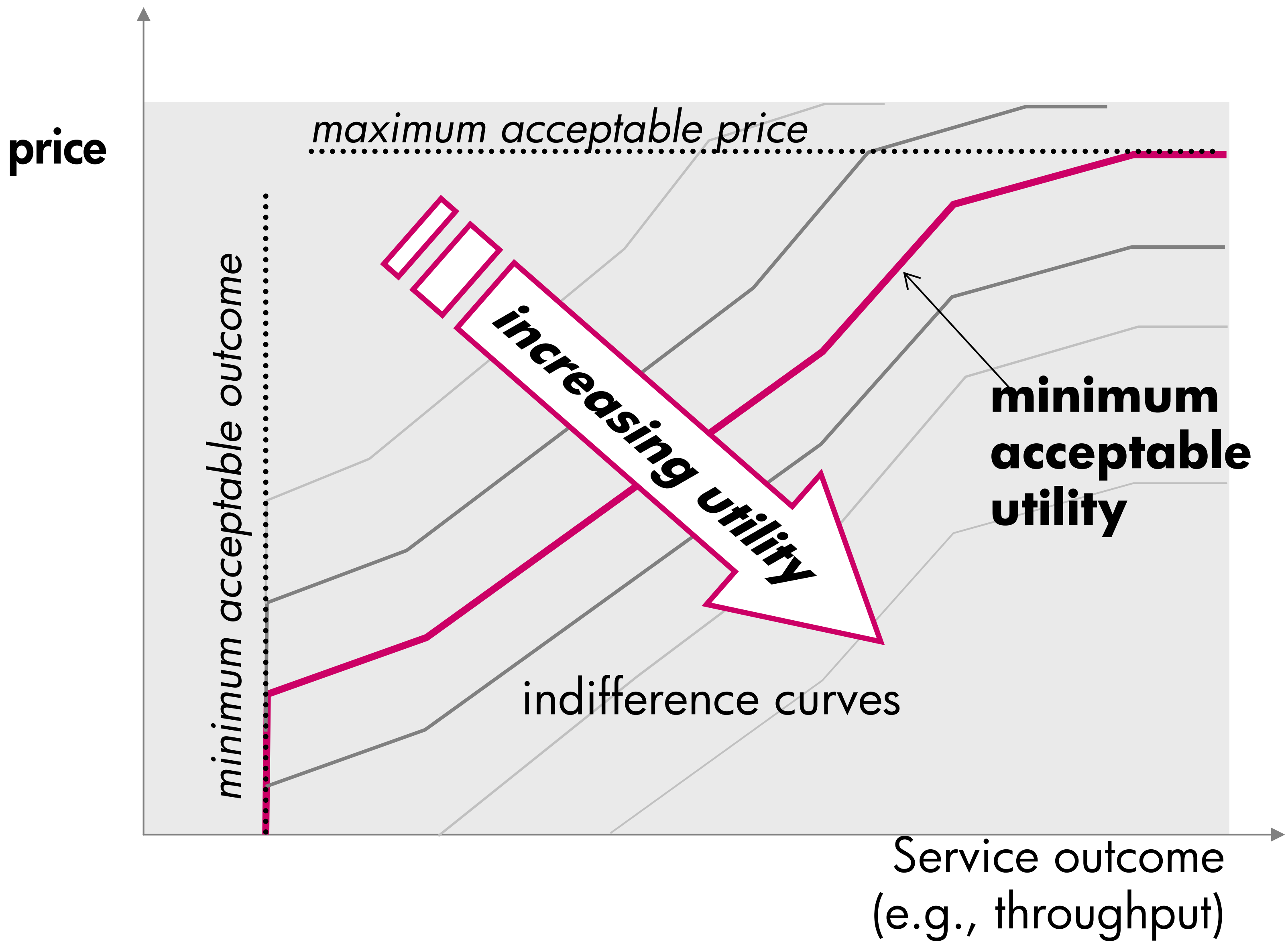


Client utility
1 outcome + price

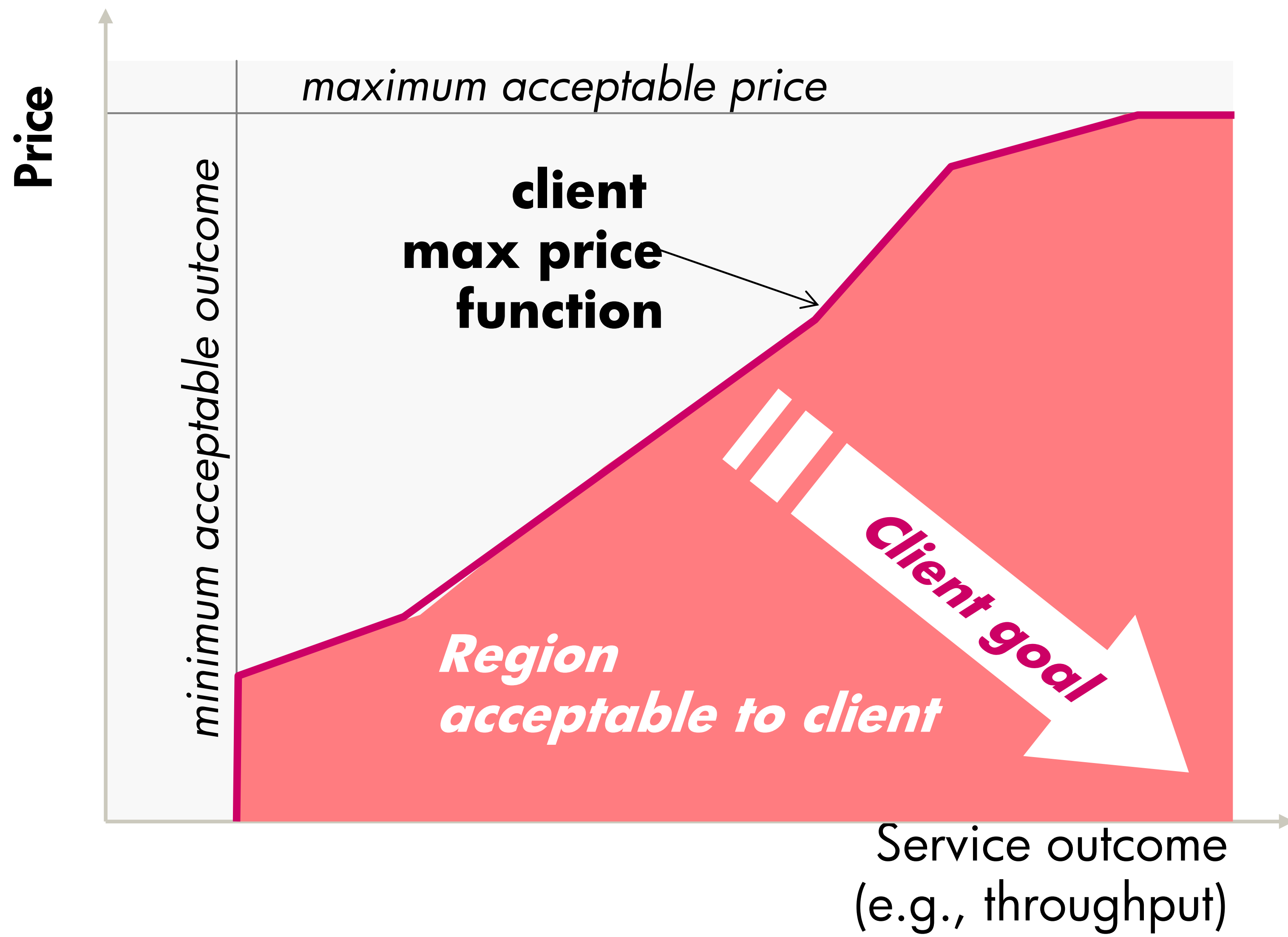




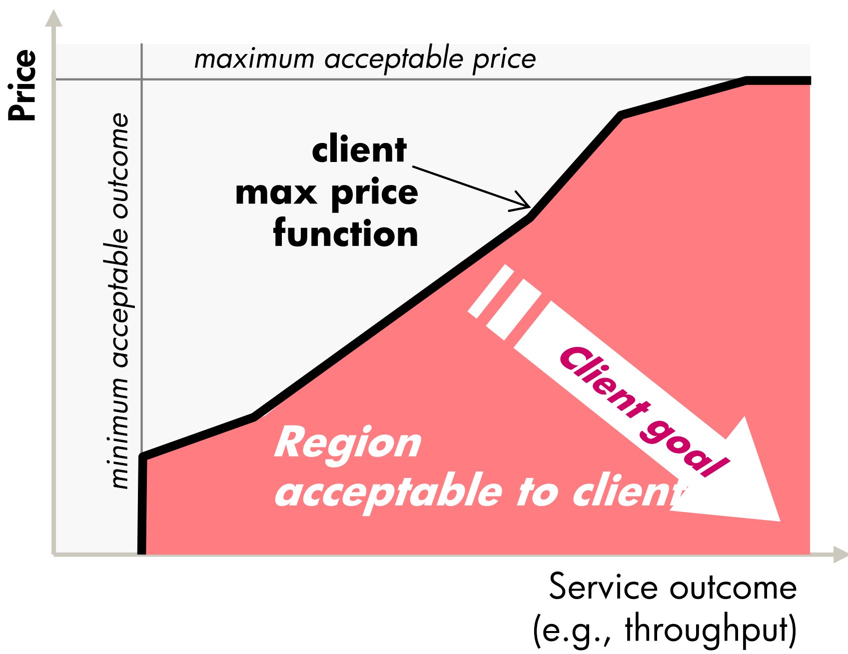
Client utility
1 outcome + price

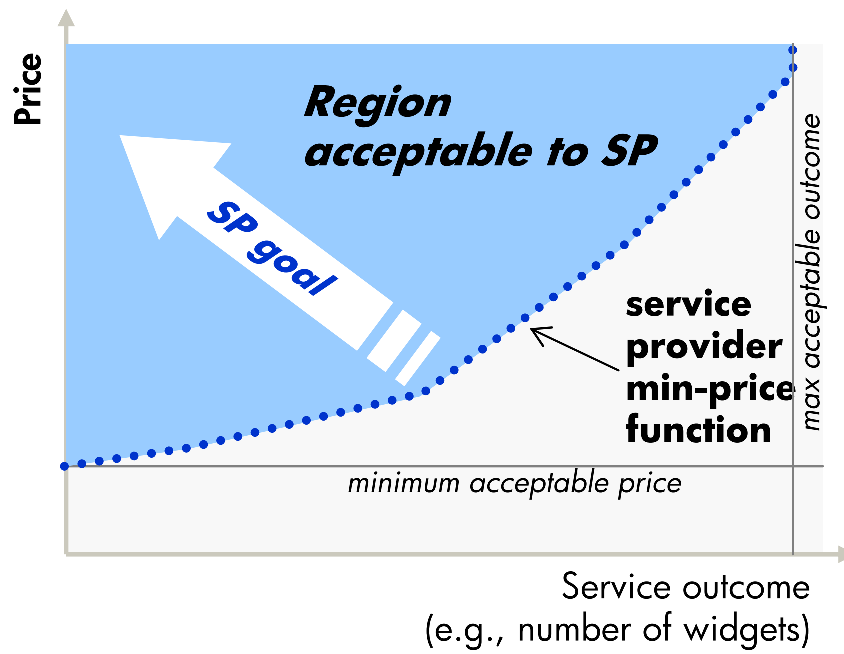


Client utility
1 outcome + price

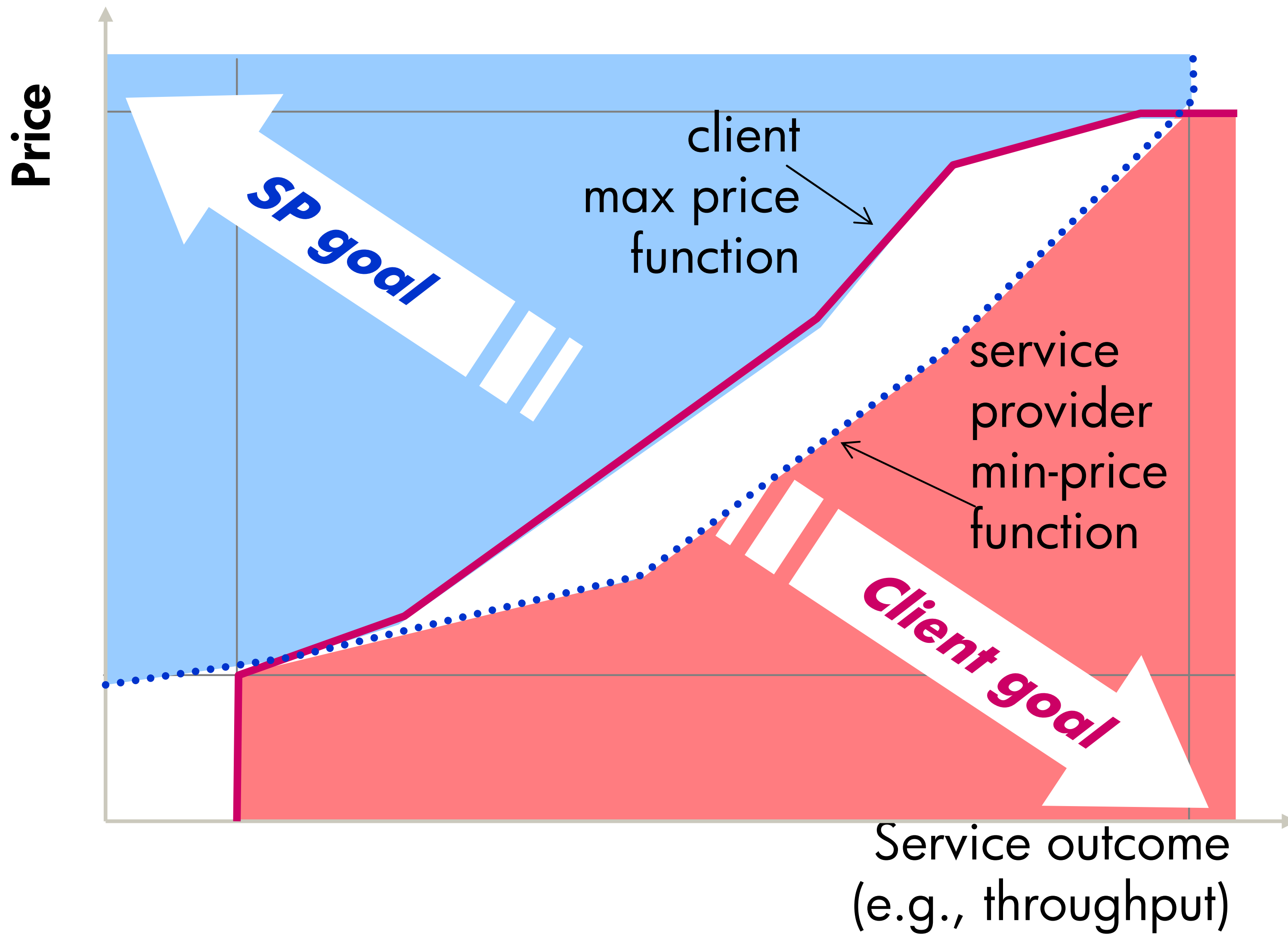


Service provider utility
1 outcome + price



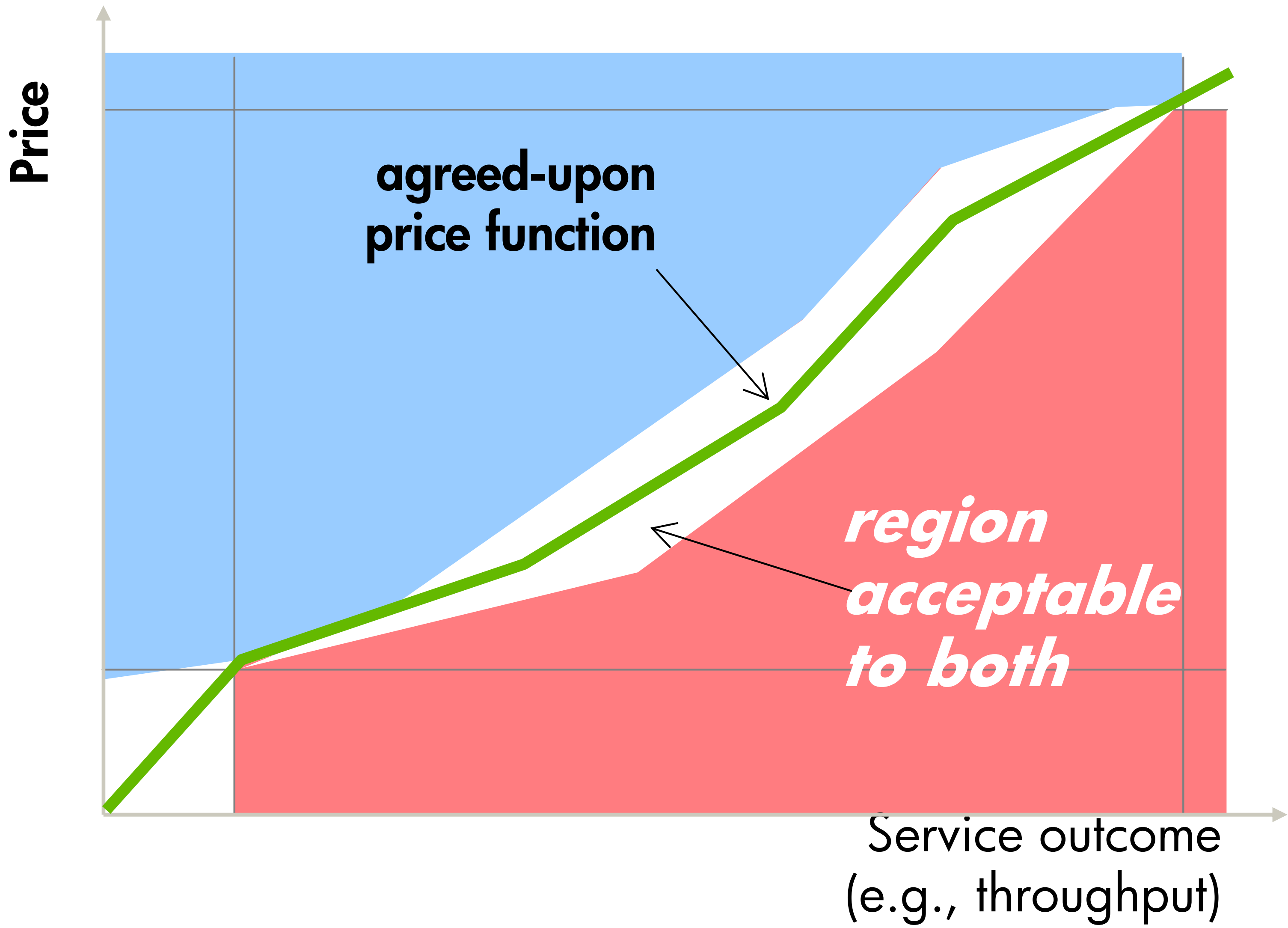


Negotiation



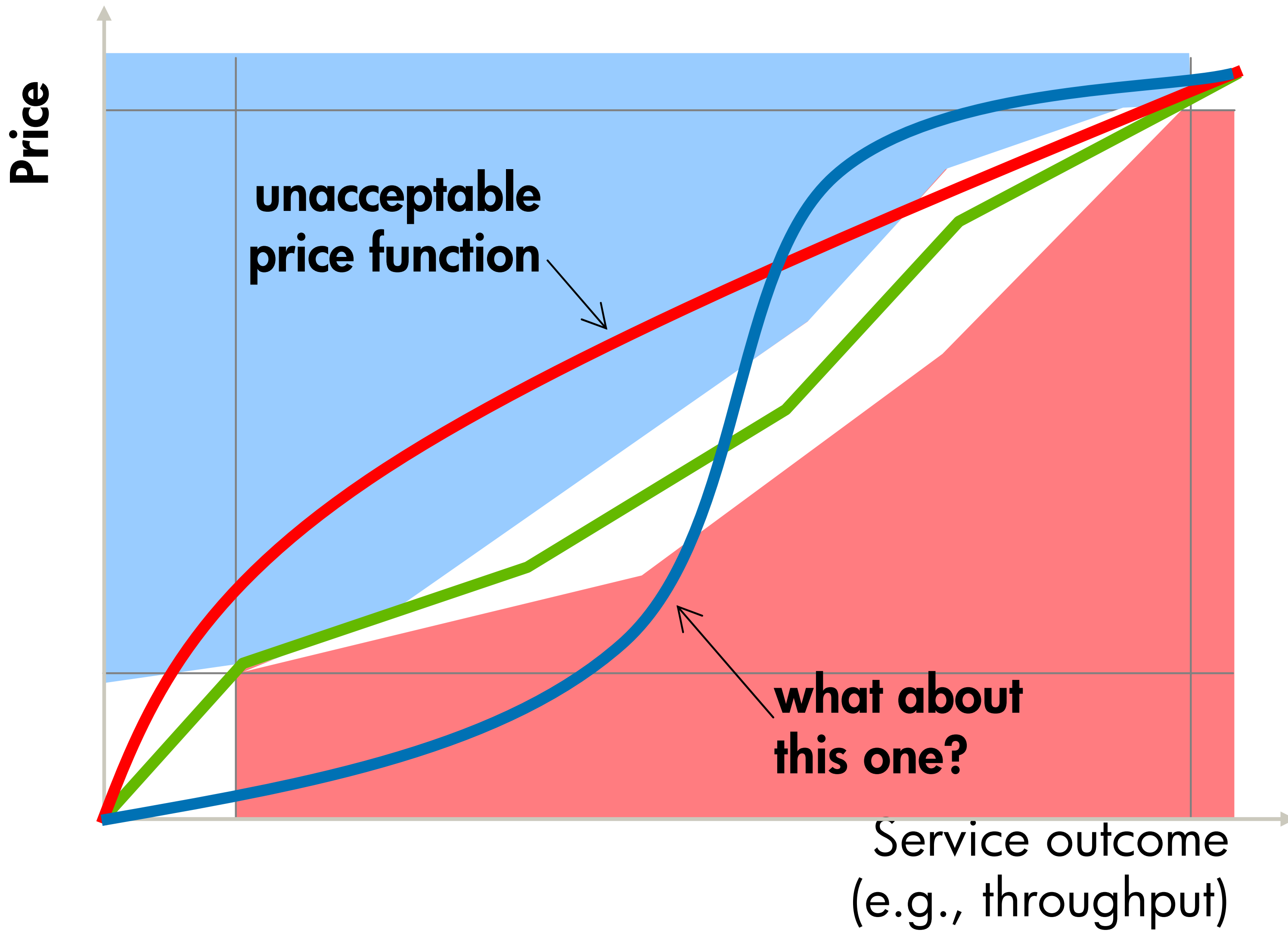
Negotiation goal

An agreed-upon price function



Negotiation goal

An agreed-upon price function



Negotiation
goals: basics

- success ➔ SLA; failure ➔ no SLA
  - there's a utility aspect to reaching an agreement, too
- support each party's interests
  - maximize achievable utility
  - caution: don't send max-price function to the other!
- purely rational agents
  - not people!

Negotiation
goals: fairness

- “Fairness” is entirely optional …
  - self-interested parties
  - but: people will walk away from a deal they consider unfair, even if they would benefit from it

- Approaches
  - note: cannot do “equal utility”
  - *k-pricing*: split the profit/loss difference
    ➔ requires trusted 3rd party

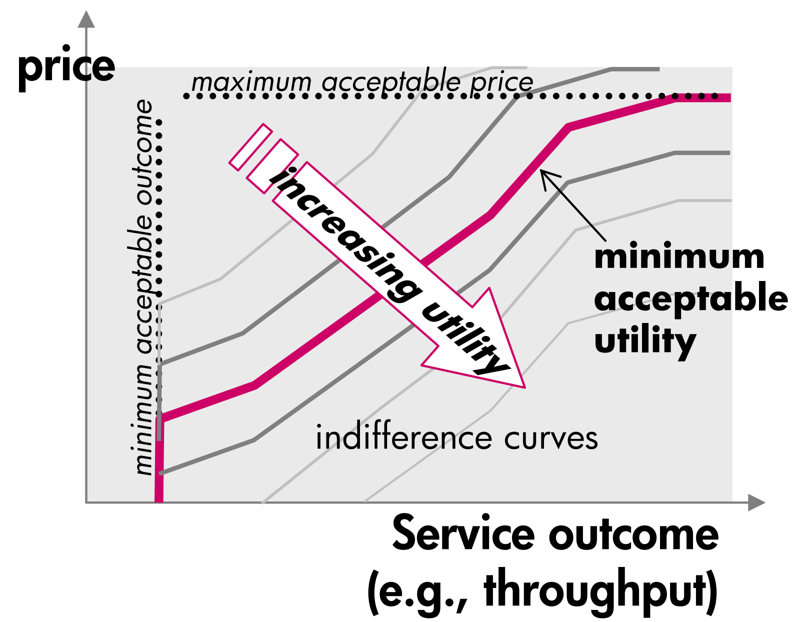
Negotiation
mechanisms/protocols

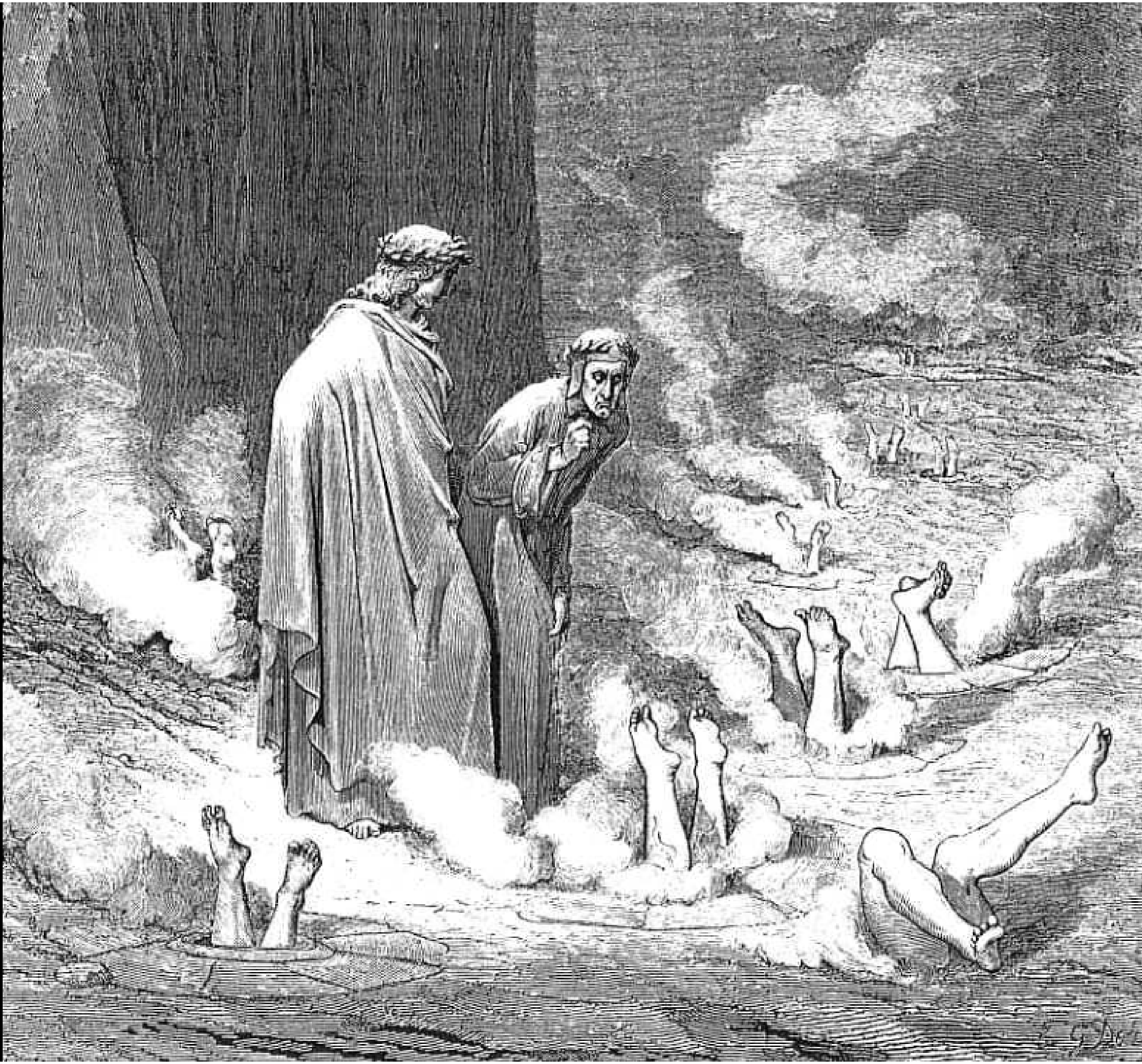
- here: two-party
  - can always add a third party (e.g., auctioneer)
- either party can set the price
  - e.g., price-setting service provider
  - price functions can be built jointly
    e.g., client → penalty, service provider → nominal cost
- many other aspects …
  - e.g., incentive compatibility

Negotiation
strategies

what to concede when?

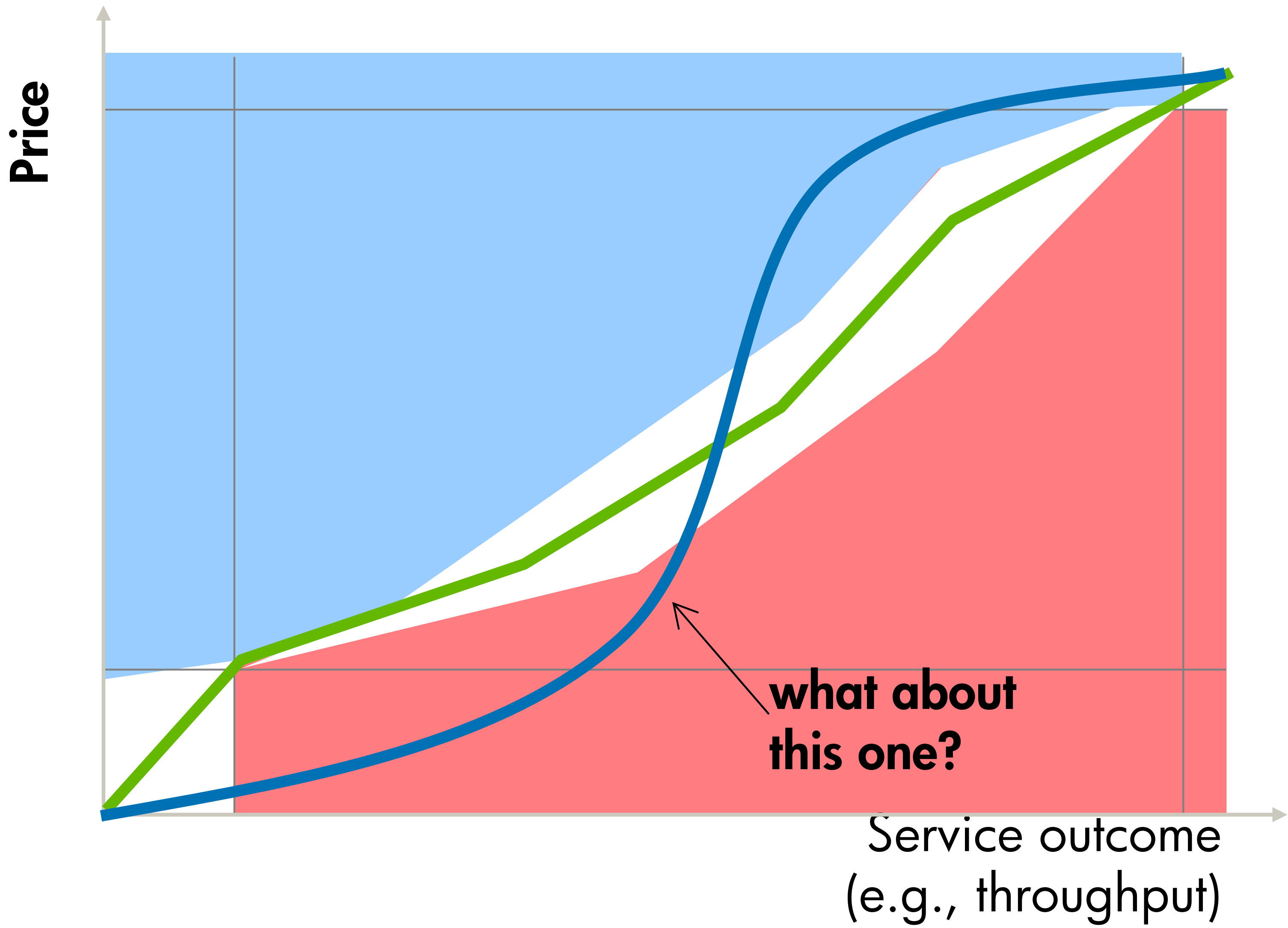
- e.g., push hardest where resistance is weakest
  - find where disparity in outcome/slope surface is greatest





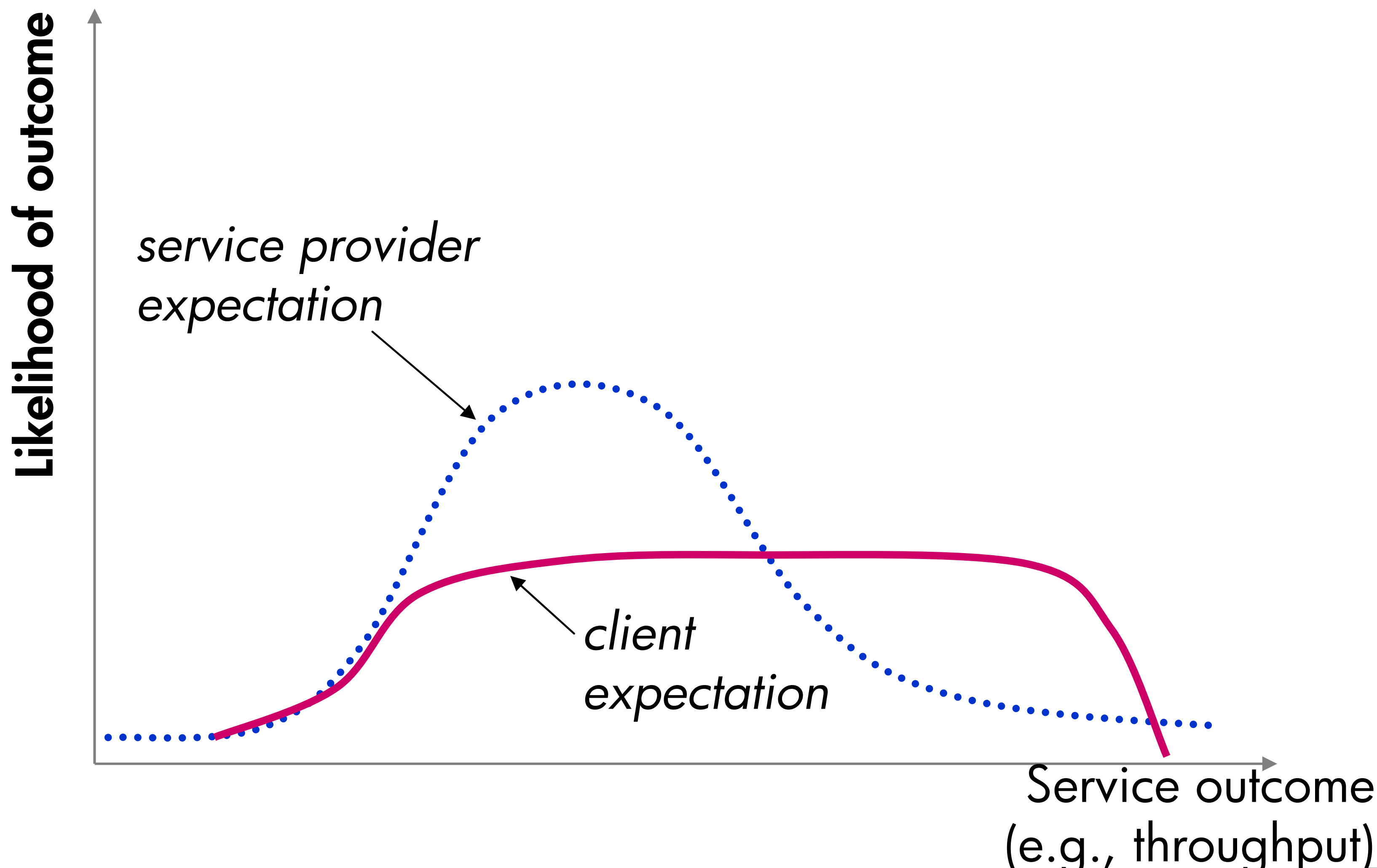
Negotiation goal

An agreed-upon price function



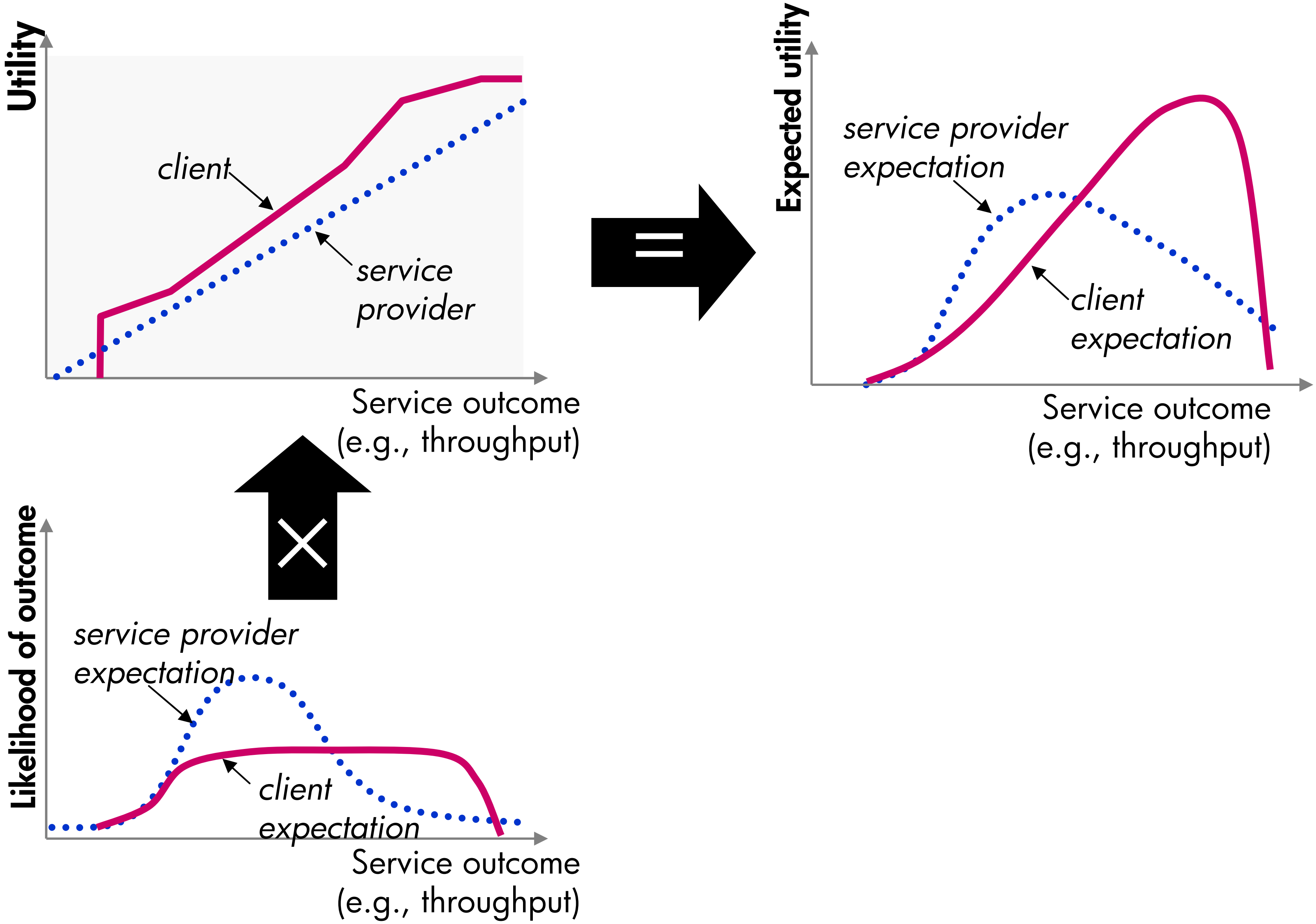
Variance in outcome

expected utility



Variance in outcome

expected utility



Variance in outcome
risk

- variance in outcome = **risk**

- examples:
  - offered load → poor performance, or more resources
  - component failure → poor availability
  - lack of resources → poor performance

- with what-if prices:
  - outcome variance → price variance

Variance in outcome

risk sharing

- who takes on the risk if effort required is unknown?
  - cost-plus prices: client
  - fixed prices: service provider
- example: how many resources?
  - model-based (e.g., response time + load → resources)
  - *systemic uncertainty* from model biases/inaccuracies
  - *stochastic uncertainty* from environment, workload, etc

Variance in outcome
risk lotteries, risk aversion

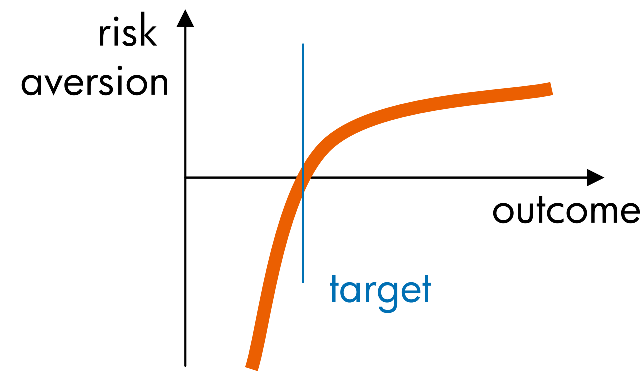
- a **risk lottery** is a game with multiple outcomes
  - pay $60, or a 50% chance of $100? (risk-averse $10)
  - pay $50, or a 50% chance of $100? (risk-neutral)
  - pay $45, or a 50% chance of $100? (risk-seeking $5)

- **risk aversion** is a measure of how much an agent dislikes the uncertainty/variance
  - strictly, risk is *any* variance; but people dislike downside risk more than they like upside risk

Variance in outcome
cumulative prospect theory

People are:

- *loss aversive*
  - a loss matters more than a gain of the same amount
- *target-relative*:
  - more receptive to risk below a target;
  - significantly averse to it above
- *long-shot biased:*
  - overweight rare, extreme events
  - discount “average” occurrences



Variance in outcome
penalties

- their *purpose* is to be bad for the victim
  - punitive vs. compensatory?
  - caution: *moral hazard*: a bad outcome that can be triggered by the *other* party
- pricing for penalties
  - estimate expected outcomes
  - add profit margin (or other pricing strategy)
  - add *risk aversion* (cf. insurance)

Future – what's next?

- Applying all these ideas to automated service providers
  - e.g., database management systems!

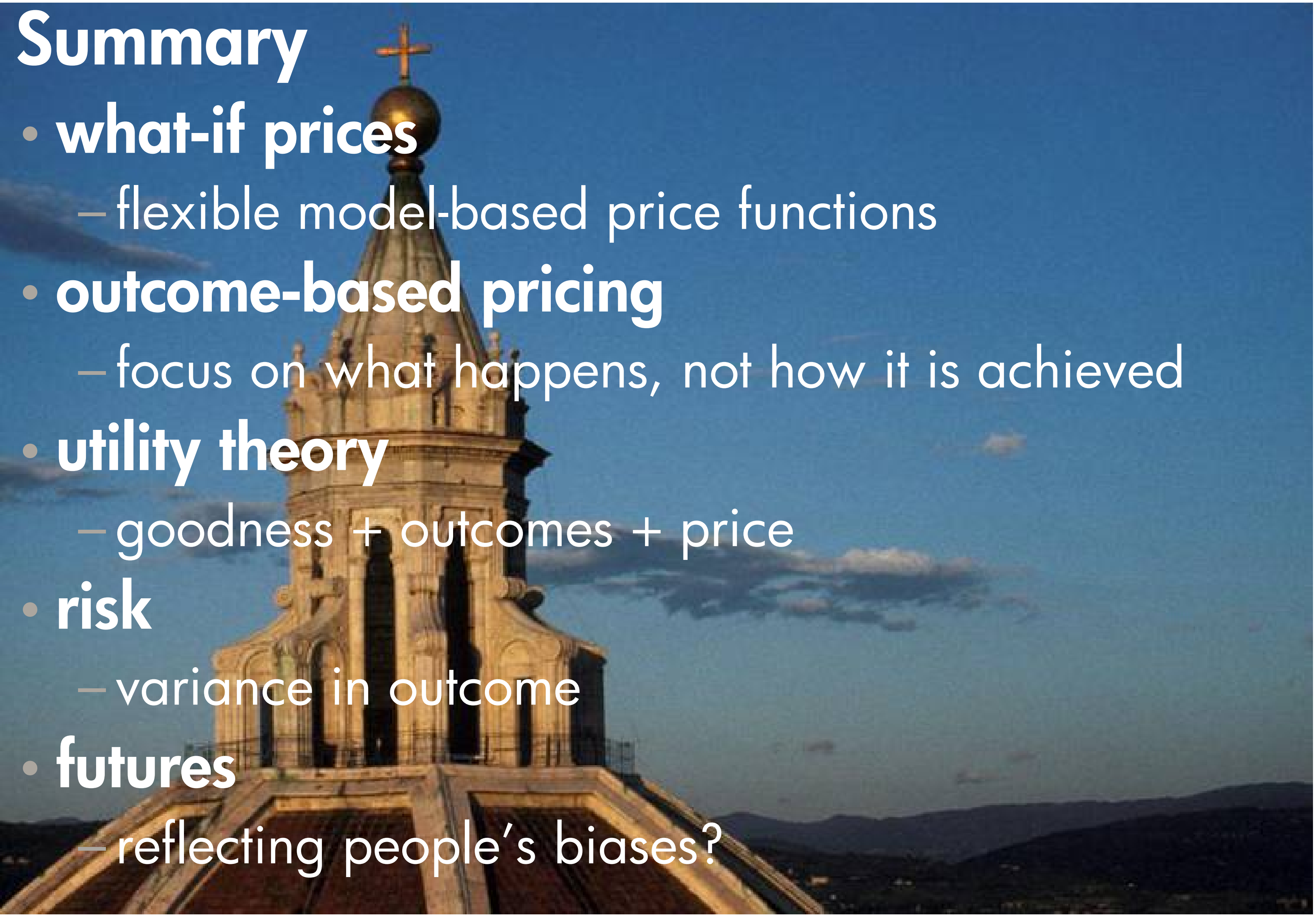
- Reflecting people's biases, not just being purely rational
  - e.g., cumulative prospect theory

special thanks to …

- HP Labs, Palo Alto, CA:
  - Sharad Singhal + Subu Iyer

- IHPC, Singapore:
  - Elaine Wong + Yang Yinping

- BDIM (Business-driven IT Management workshop):
  - Claudio Bartolini + Jacques Sauvé



Summary

- **what-if prices**
  - flexible model-based price functions
- **outcome-based pricing**
  - focus on what happens, not how it is achieved
- **utility theory**
  - goodness + outcomes + price
- **risk**
  - variance in outcome
- **futures**
  - reflecting people's biases?

http://www.hpl.hp.com/personal/john_wilkes/papers/#Tuscany

i n v e n t