Key performance indicators in data warehouses

Manfred Jeusfeld
University of Skövde, Sweden

About myself

- studied computer science at RWTH Aachen, Germany (1980-86)
- doctoral dissertation from University of Passau, Germany (topic deductive object bases)
- senior researcher at RWTH Aachen (1992 - 1997)
- assistant professor at Tilburg University, Netherlands (1997 - 2013)
- senior lecturer at University of Skövde, Sweden (2013 - now)

Co-developed the ConceptBase.cc system

Worked in EU DWQ (data warehouse quality) project, and others

Started CEUR-WS.org (online workshop proceedings)

The problem statement

How can key performance indicators be realized by a data warehouse system?

Can a data warehouse design be derived from KPI specifications?

How can a query implementing the KPI be derived from its specification?

Why at all are KPIs useful and what do they express?

Frankly, I have no satisfactory answer to these questions but I want to understand with you the problem and develop a strategy how to come to satisfactory answers.

Def.:

A **key performance indicator** (KPI) evaluates the success of an organization or o particular activity in which it engages.

(source: Wikipedia)

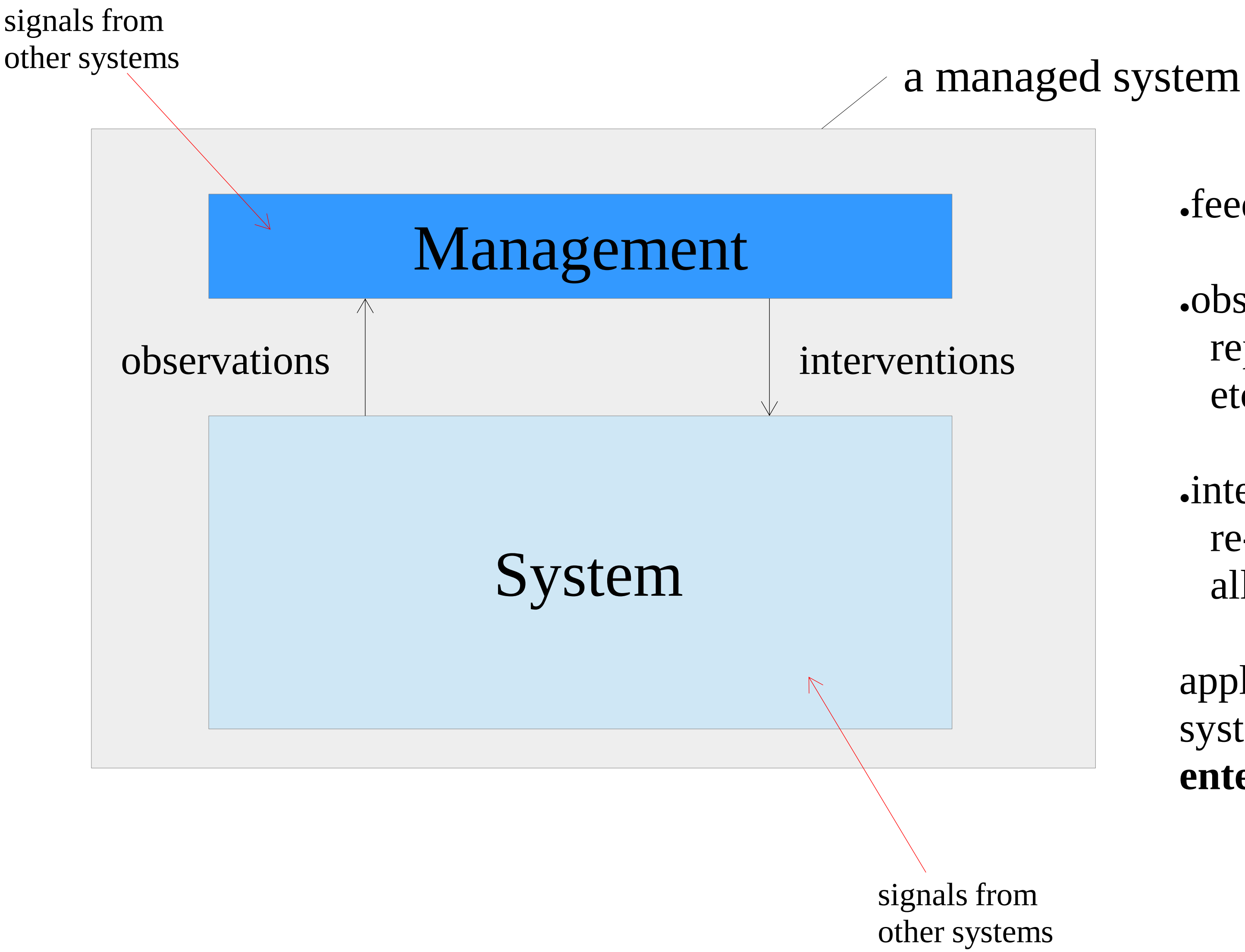
Examples:

- number of defects (of products/services)
- customer satisfaction
- profit margin
- services delivered before the promised delivery time
- machine utilization

Each enterprise may have its own set of KPIs depending on its business sector and (current) business goals.

Example (oil industry): number of days between two accidents where employees are hurt

The underlying mechanism: managed systems

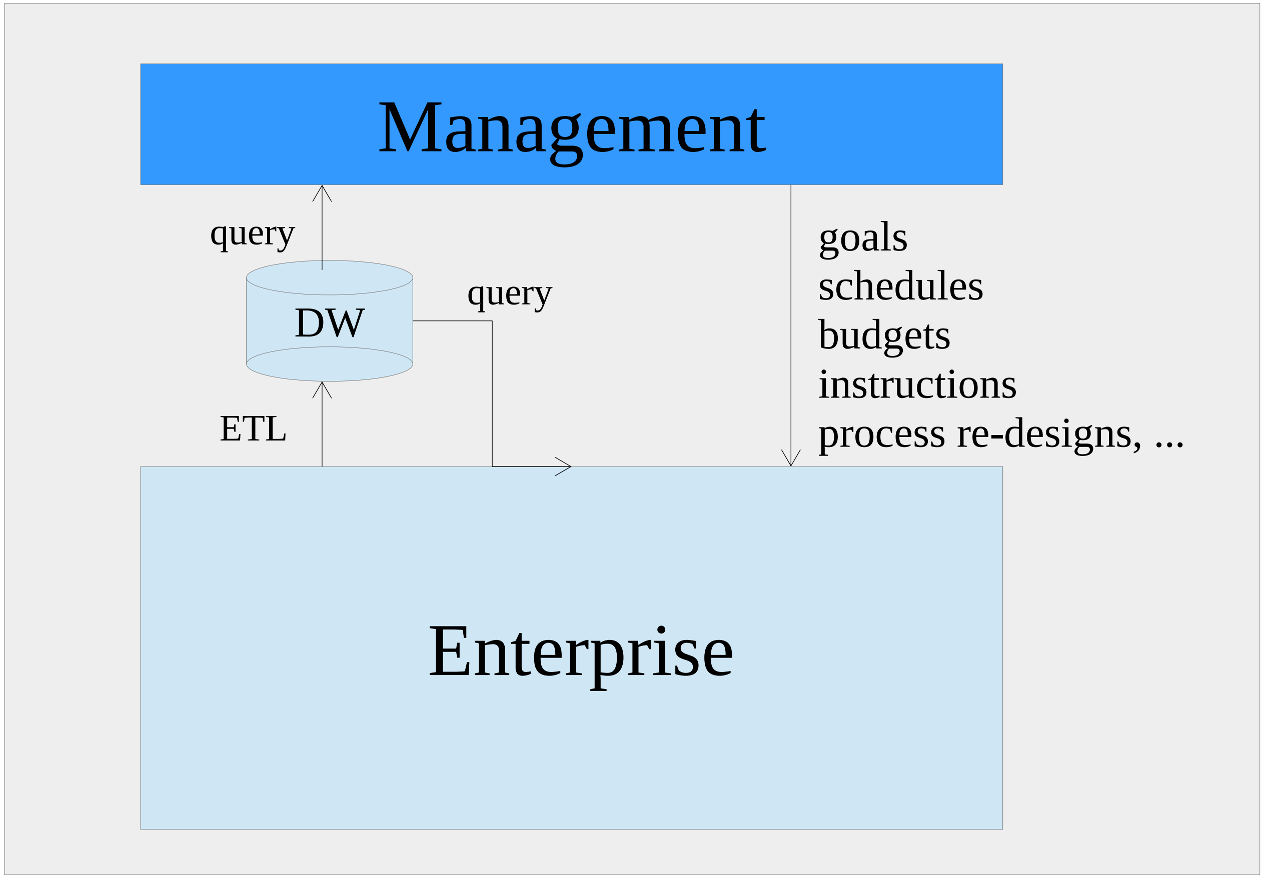


- feedback cycle
- observations can be reports, measurements, etc.
- interventions can be re-configurations, resource allocations, etc.

applies to many types of systems, in particular **enterprises**

- systems are part of larger systems
- systems have sub-systems
- the management is a sub-system of the managed system

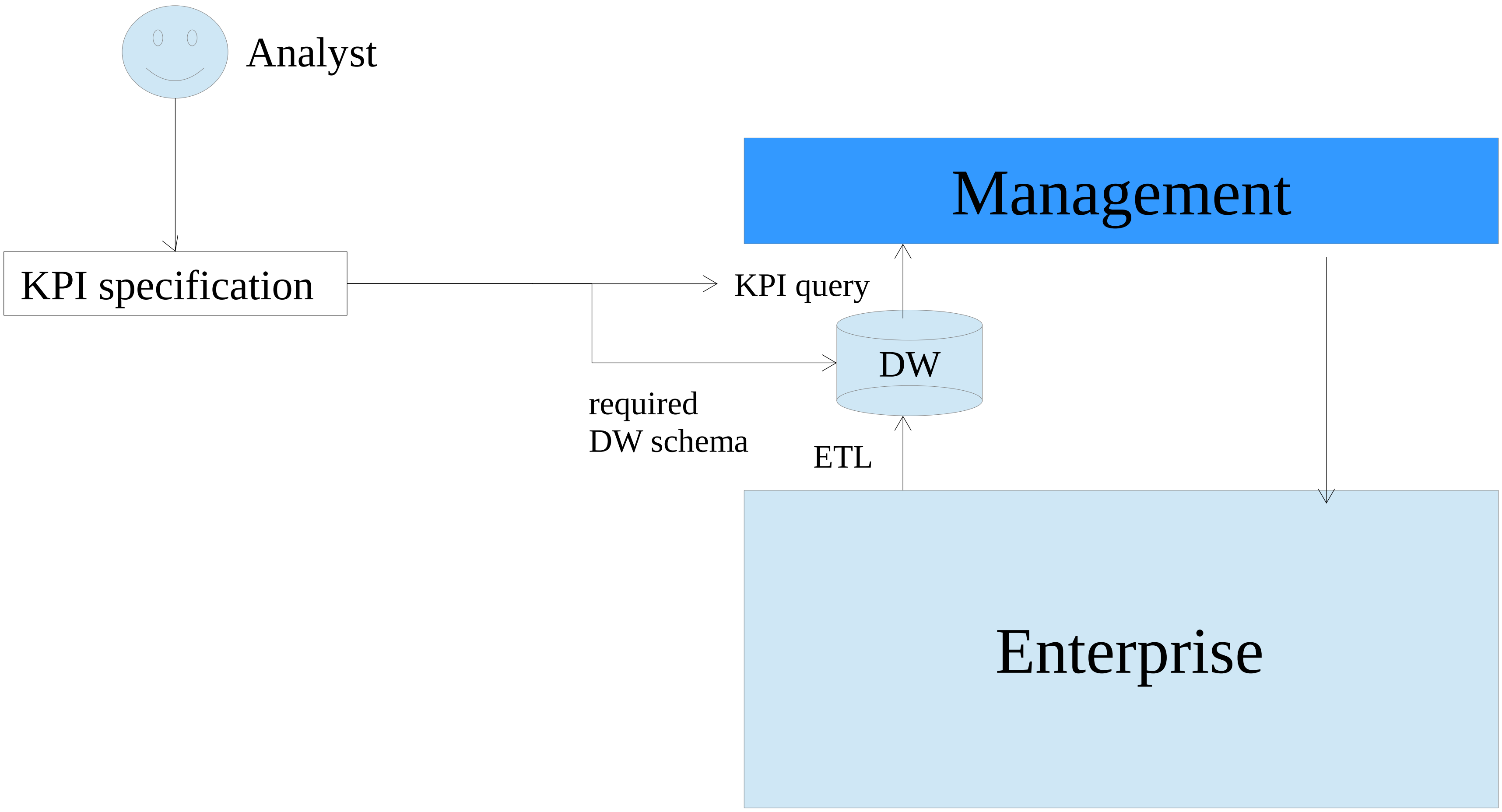
A data warehouse structures observations



- ETL processes collect observations from the enterprise (and its departments) into multi-dimensional, subject-oriented data structures (data cubes)

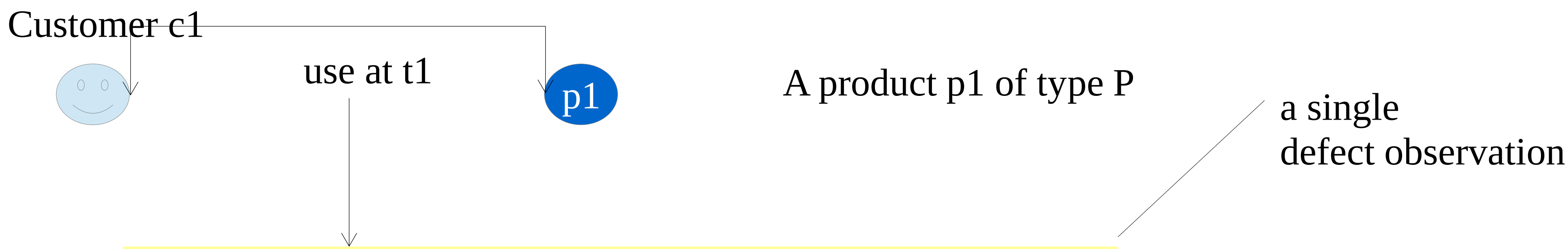
- the actors in the enterprise may also use the DW directly, e.g. for real-time process management

The problem in terms of the architecture

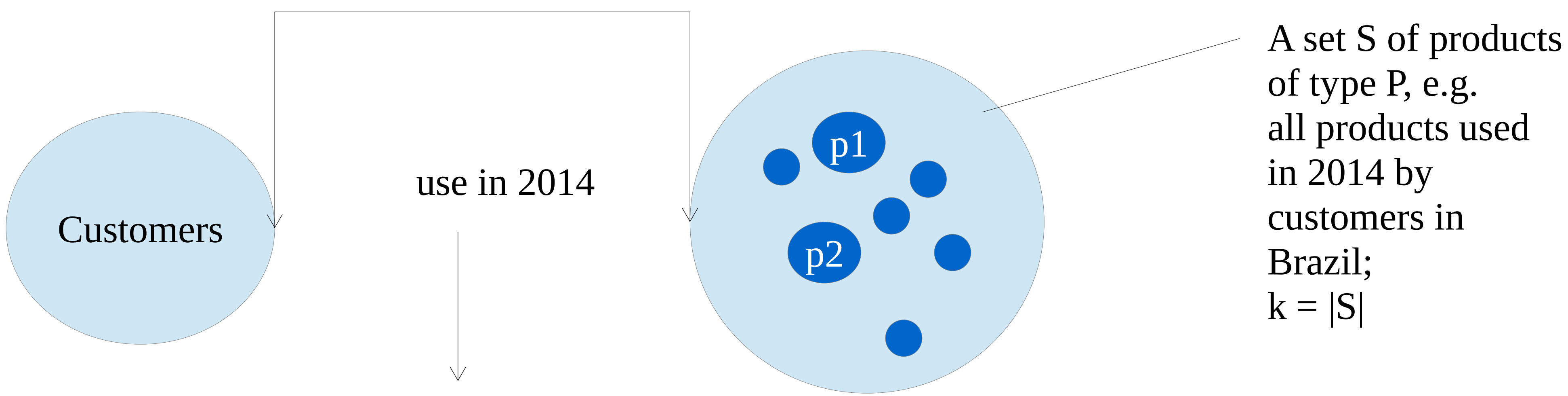


1) Specify the KPI
2) Generate the required DW schema (or schema pattern)
3)Generate the queries on top of the query that evaluate to the KPI

Example KPI: Number of reported defects of a product



Customer c1 observes a defect of product p1 at time t1.



Customer c1 observes a defect of product p1 at time 2014-01-12, 12:31
Customer c2 observes a defect of product p2 at time 2014-02-01, 17:14
...

n defect observations

$D_{2014,Brazil} = n / k$ (defect density of product P in Brazil in 2014)

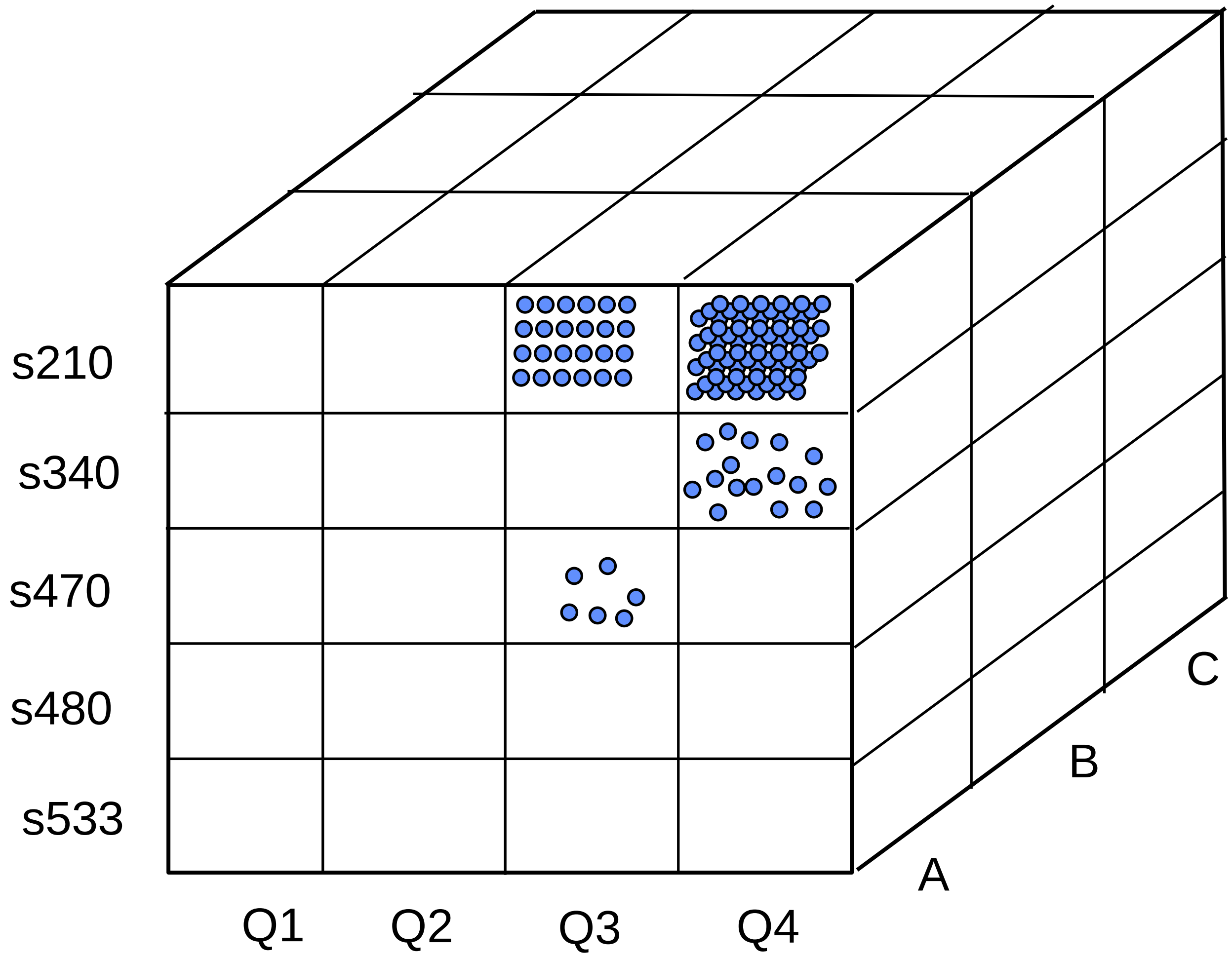
- KPIs typically have implicit **dimensions**

- KPIs are based on **observations** of some processes, e.g. the “use” process of a customer

- KPIs are **aggregated** from many observations about similar participating subjects / objects

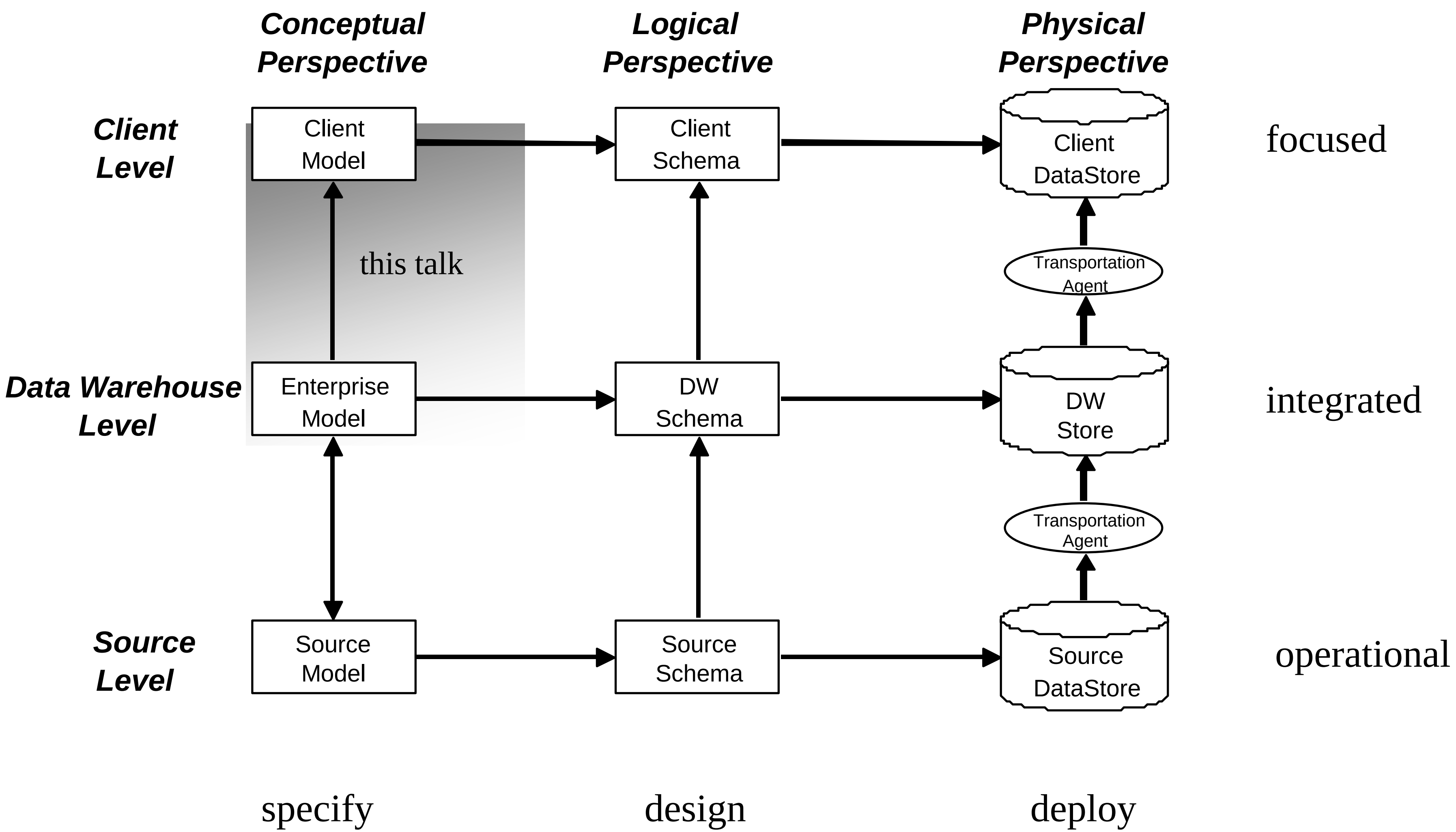
Thus, a data warehouse is a natural implementation platform for KPIs!

Data cubes: a way of looking at facts (=observations)



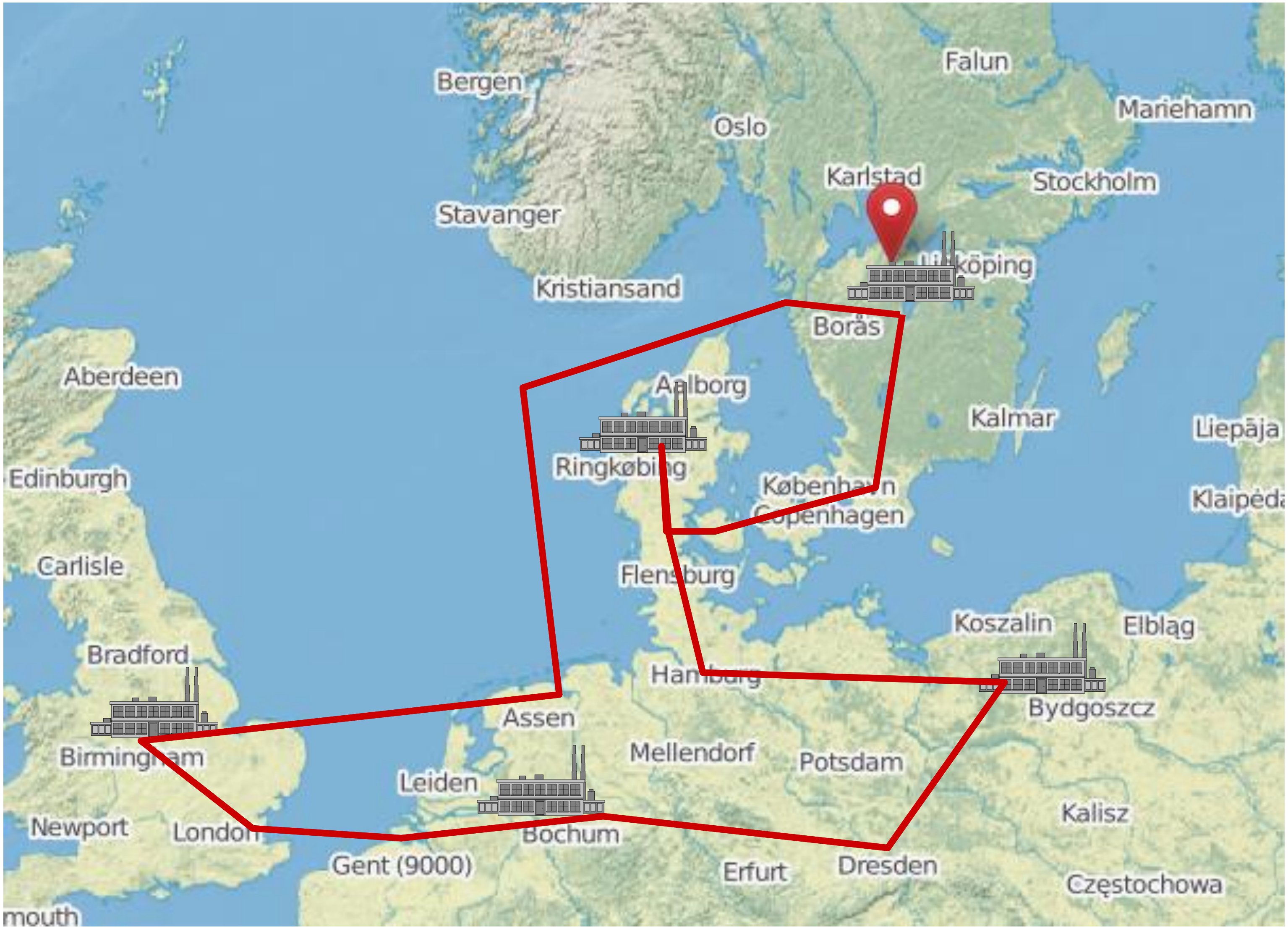
Each point • stands for a fact (here: a sale). In each cell of the data cube, a set of facts is contained. The measurement is then an aggregation operation on the set, e.g. count, or the sales value. The finer the intervals on the dimensions, the less facts are in the cells. At the finest grain, there is at most one fact in a cell.

Levels and perspectives in data warehousing



[Jarke et al 1999]

All systems are part of even larger systems!



... that are even more difficult to understand or control

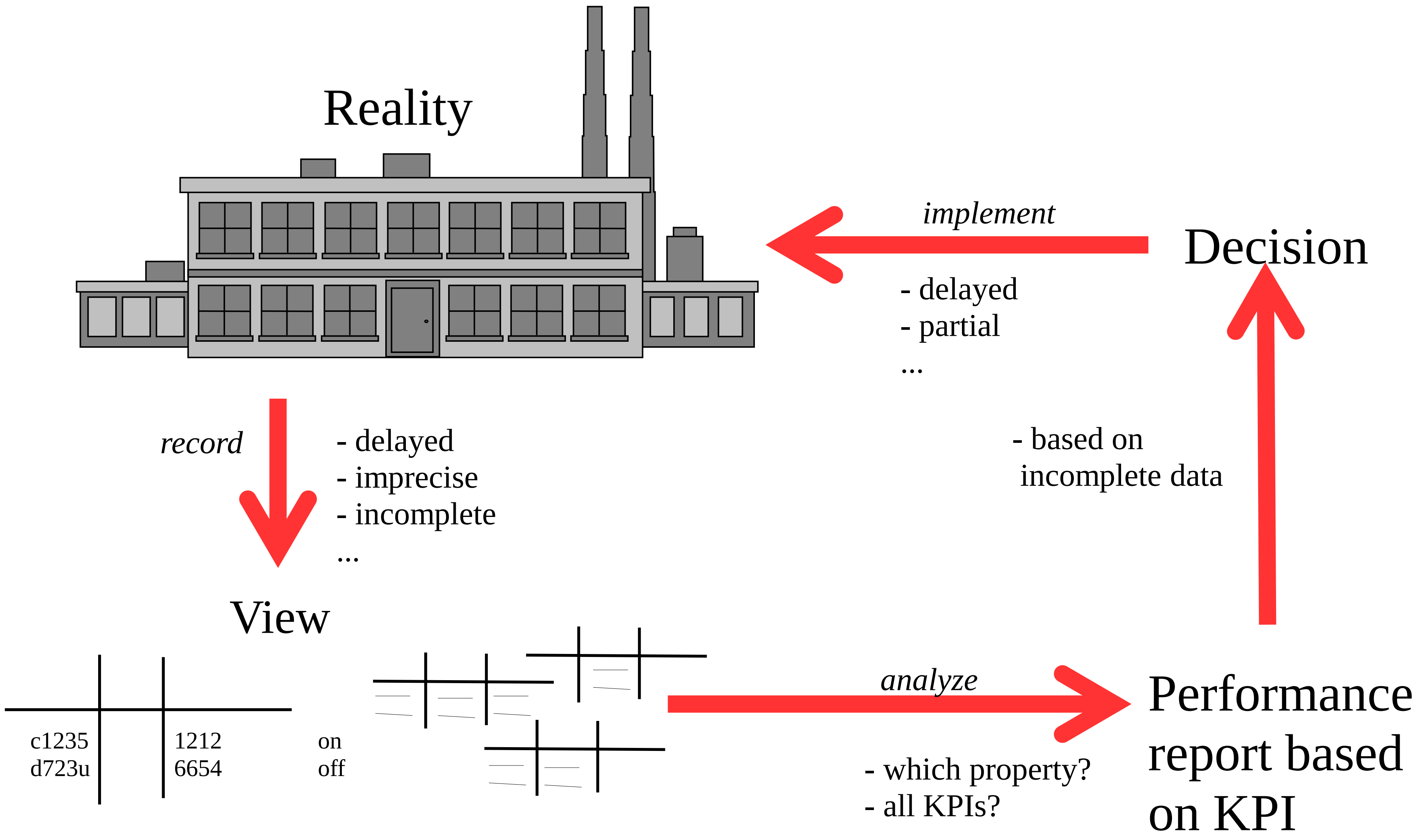
Some statements on performance measurement

“You cannot control what you cannot measure.” (attributed to W.E. Deming)

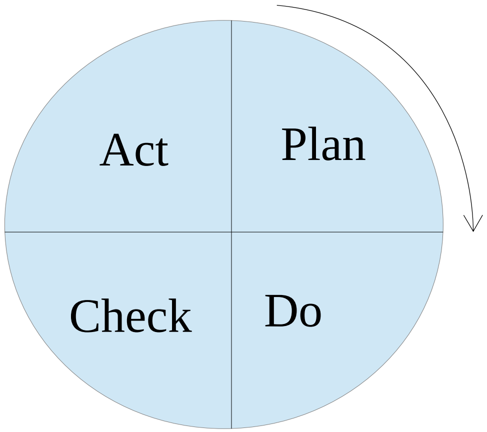
“Projects without clear goals will not achieve their goals clearly.” (Gilb)

“Measure what is measurable, and what is not measurable make measurable.”

Information systems are incomplete views of the reality



The Deming Cycle (Plan-Do-Act-Check)



Plan: define process, set measurable goals / targets

Do: Collect measurements from the current process

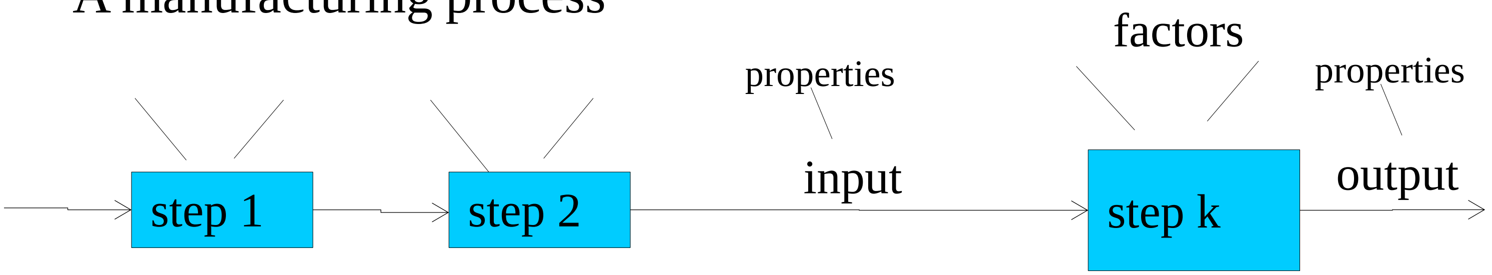
Check: Establish the difference between actual and expected results

Act: If the process fulfills the goals, it becomes the new standard, otherwise create a new plan

→ continuous improvement

Statistical process control (SPC)

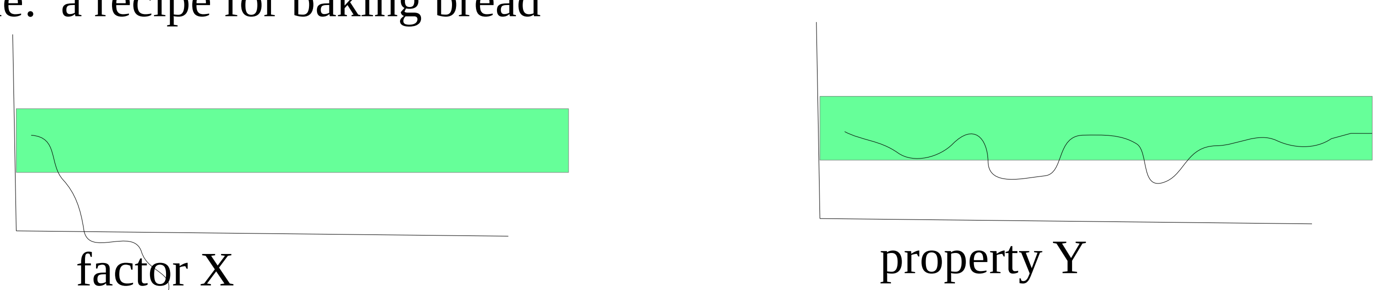
A manufacturing process



The quality (properties) of the output statistically depend on the properties of the input(s) and the factors (circumstances) of the production steps.

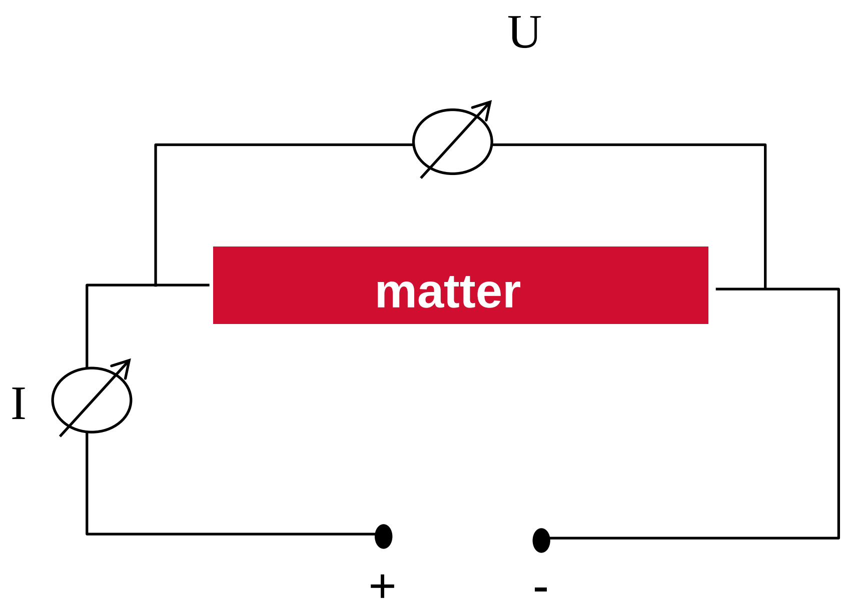
Hence, rather than checking the quality at the very end, one should keep the factors and inputs of the steps in “acceptable” intervals to maximize the probability the the product has the desirable properties

Example: a recipe for baking bread



The property Y statistically depends on factor X.

Use of Measurements in Science



Set U and measure I in a **repeatable** experiment. Observe results:

U	I	U/I
10	5.1	1.96
15	7.4	2.03
20	10.2	1.96
25	12.2	1,97
...	...	...
1000	170	5.88

a scientists <u>observes</u> experiments, forms a <u>model</u> (here Ohm's law ***R=U/I=const***), and <u>verifies</u> the model;

at the start, the design of the experiment and the model are not fixed

the model is not always globally true; for example, if the parameter U exceeds a certain level, then the matter will heat up and the resistance R will yield other values

certain parameters are neglected (e.g. the noise level in the room)

Hence, we ultimately are interested in such laws that help us predict the future.

Q: What entities could be measured?

Processes: collections of activities (like invoice handling)
Products: any artifact resulting from a process activity
Resources: entities required by a process activity

Q: Can we measure an entity just by referring to its state?

internal attribute: can be measured purely in terms of the entity itself
example: weight of a product
external attribute: can only be measured by taking the context of the entity into account (which activity produced it, which resources were spent, how does the entity behave in a certain situation, etc.)
example: number of failures experienced by the user
response time of a database query

Problem: People tend to restrict themselves on internal attributes since they can be measured easier. An internal attribute cannot always replace an external attribute.

Purpose: Provide guidelines to select and implement metrics

GOAL

Overall goals of your organization

QUESTION

List of questions whose answers are needed to determine whether a goal has been met

METRIC

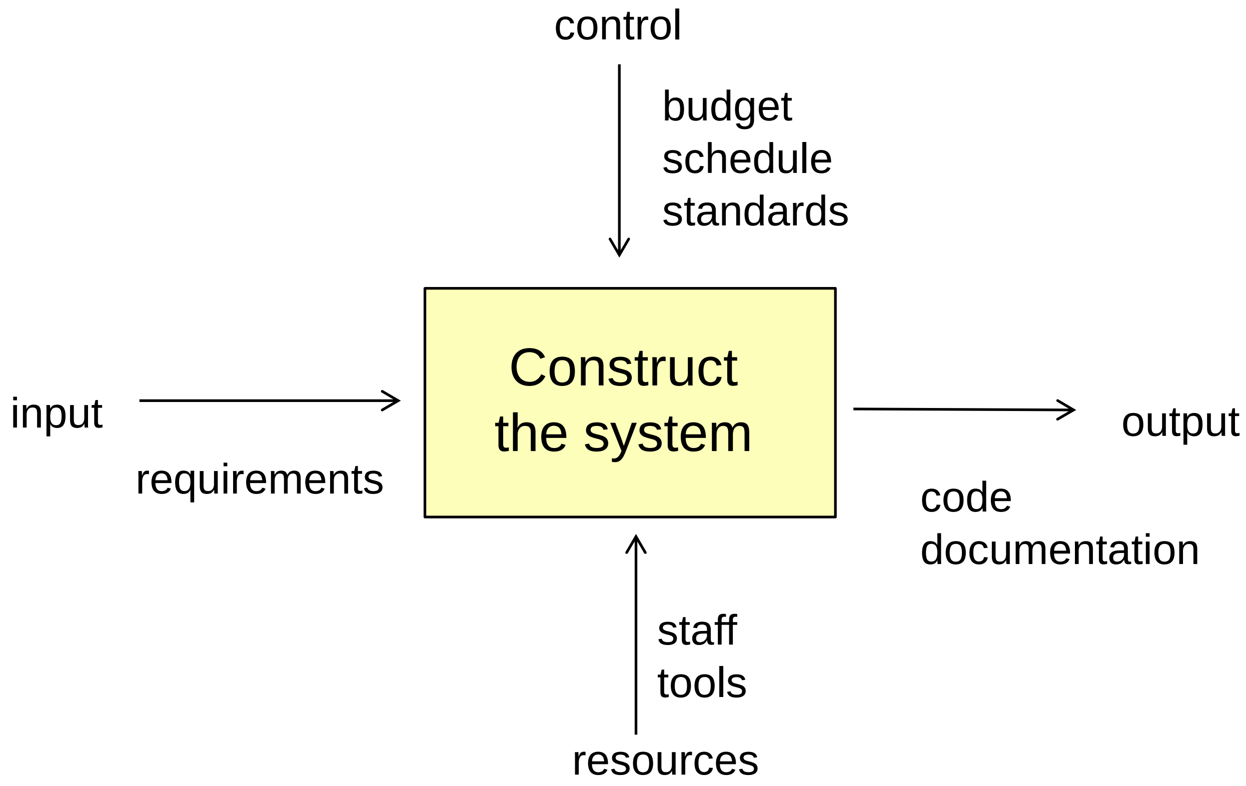
Selection of attributes to be measured, and metric to be used for obtaining the answers

Notes:

GQM prevents you to do measurements unrelated to goals to answer a question, more than one measurement may be required a single measurement can be used to answer multiple questions

Ref:Victor R. Basili, "Software Modeling and Measurement: The Goal/Question/Metric Paradigm," University of Maryland, CS-TR-2956, UMIACS-TR-92-96, September 1992

Purpose: The software development process is classified in five levels from ad hoc (the least predictable and controllable) to optimizing (the most predictable and controllable) .



*) SADT style representation of the systems development process

Ref.: Norman E. Fenton, Shari Lawrence Pfleeger:Software Metrics - A Rigorous & Practical Approach. 2nd Edition, PWS Publishing, Boston, USA, ISBN 0-534-95425-1

Level 1: "Ad hoc" (Initial)

Inputs are ill-defined

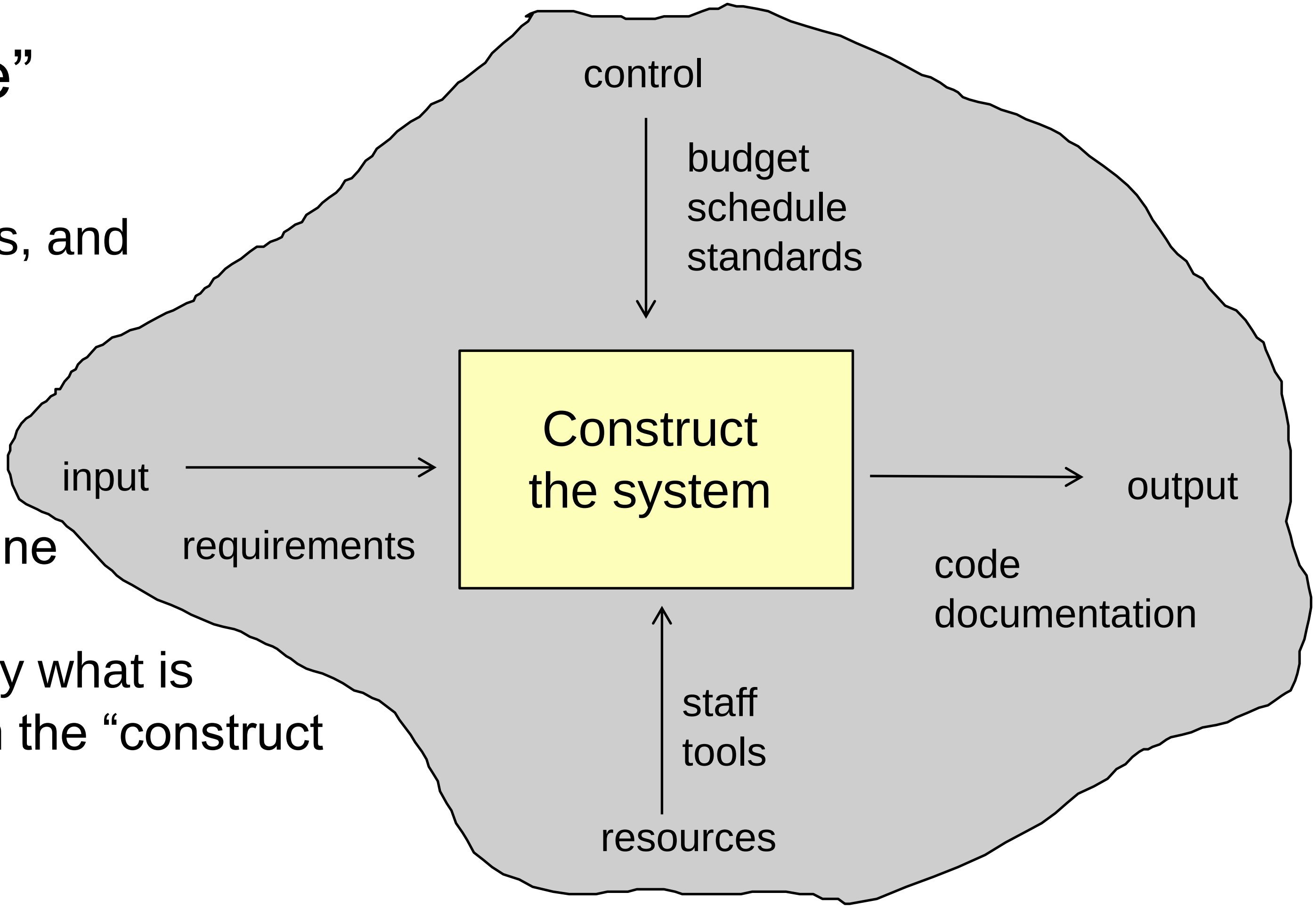
Outputs expected (programs, documentation)
"We don't exactly know the type of requirements when we start a software development project but we do know that we want to have executable programs at the end."

Transition from input to output is undefined and not controlled.
"An analyst/designer/programmer may use the method that suits her/him best. We are not interested in prescribing a standard on how to develop a system."

→ Productivity and quality measures vary largely because there is no adequate structure or control. These measure depend on the ad hoc decision made by the development team. Hard to define measures that can be used to compare a project/entity with other projects/entities.

→ Only simple measurements on output products that can be used to understand the process and that may indicate to switch to a higher maturity level.

Level 2: “Repeatable”



Inputs, outputs, constraints, and resources are identified

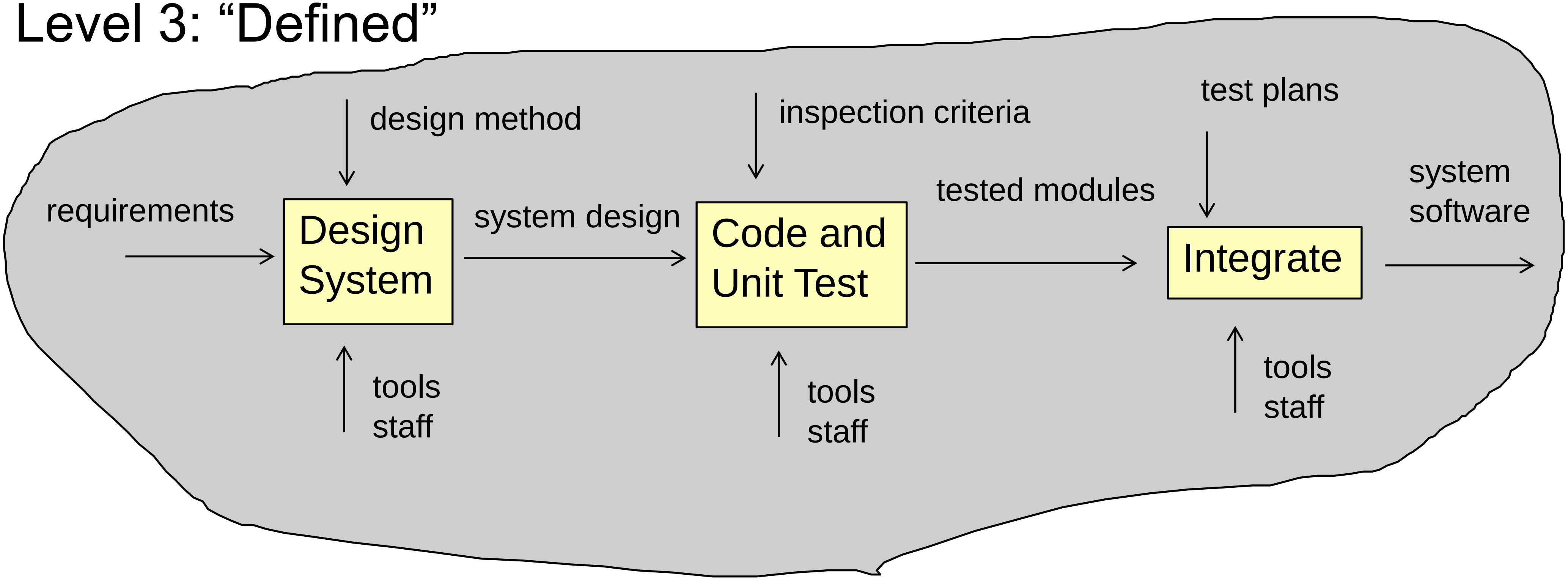
Process “Construct the system” is repeatable like a “black box” subroutine

One can measure basically what is on the arrows to and from the “construct the system” activity

Measurements associated to project management are suitable for this level.
For example: cost per KLOC, KLOC per budget.

It may well be that certain activities of systems development in your team are ad hoc while others are repeatable. For example, the “coding” activity may be repeatable while the testing is ad hoc.

Level 3: “Defined”

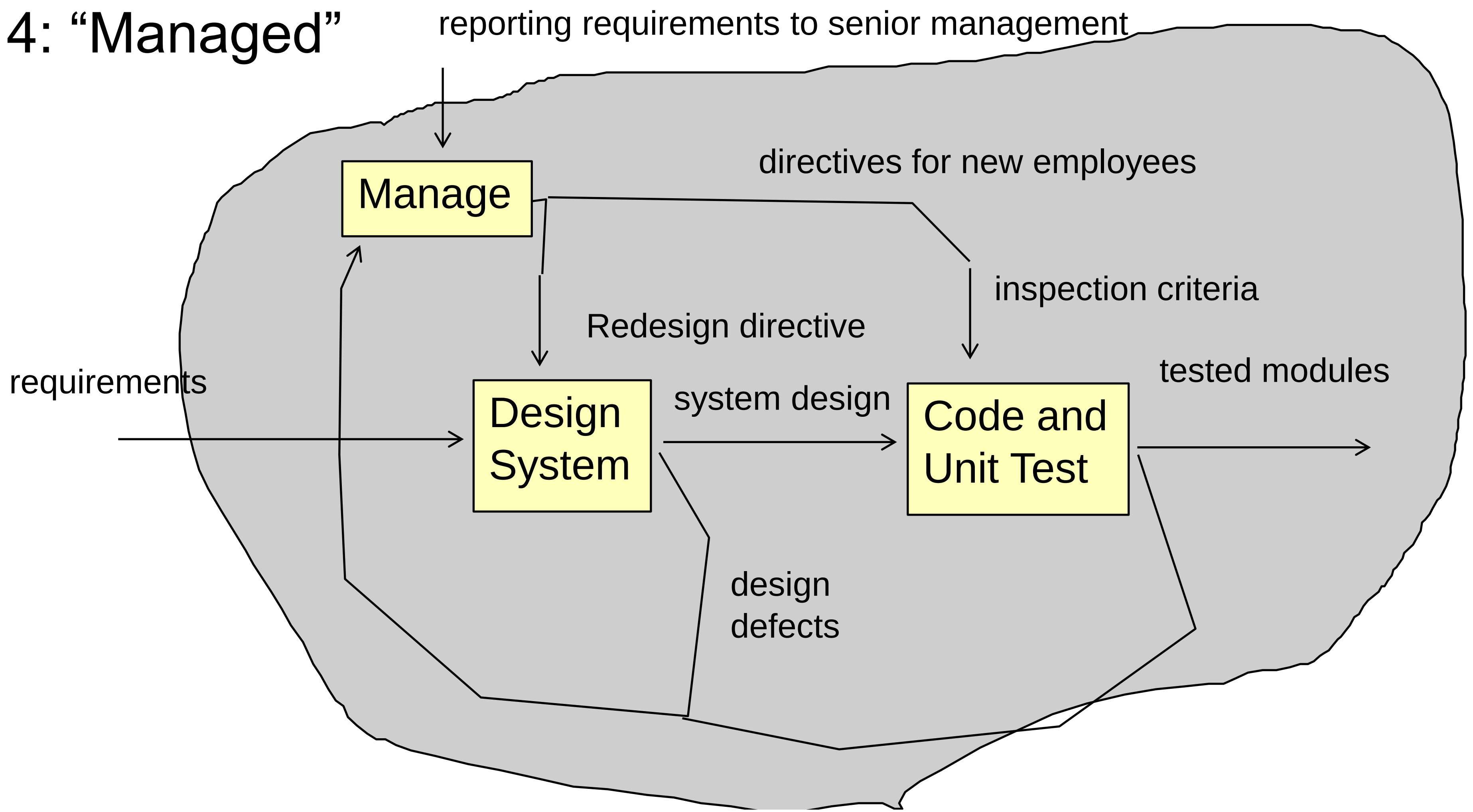


The is a “visibility” of sub-activities of the systems development process
Intermediate activities and their inputs/outputs are known and understood

→ Measure intermediate products
Predict measures for subsequent processes from known measures of earlier processes
Make measurements for the various types of input/output, e.g. defect density in code, defect density in system design, etc.

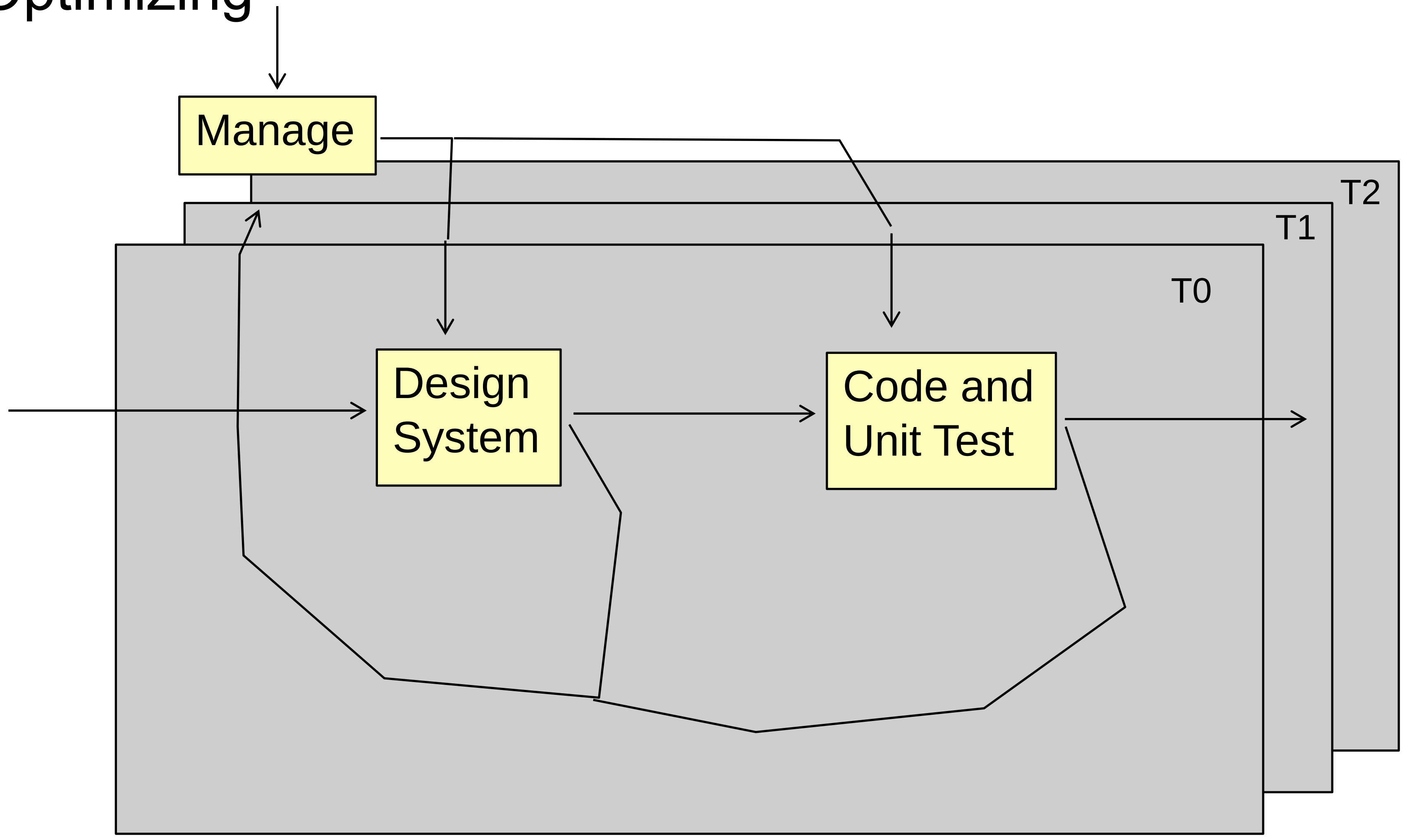
→ Control the processes based on their measurements, e.g. when a measure on the system design predicts low system quality then revise “Design System”.

Level 4: "Managed"



"Manage" process oversees the system development, collects feedback
Systematically create directives (like "redesign") based on measurements
Feedback control how resources are allocated to processes, e.g. more efforts in testing when some measures on system design indicate that the number of expected faults is high
Measure products, processes and feedback to <u>control</u>

Level 5: “Optimizing”

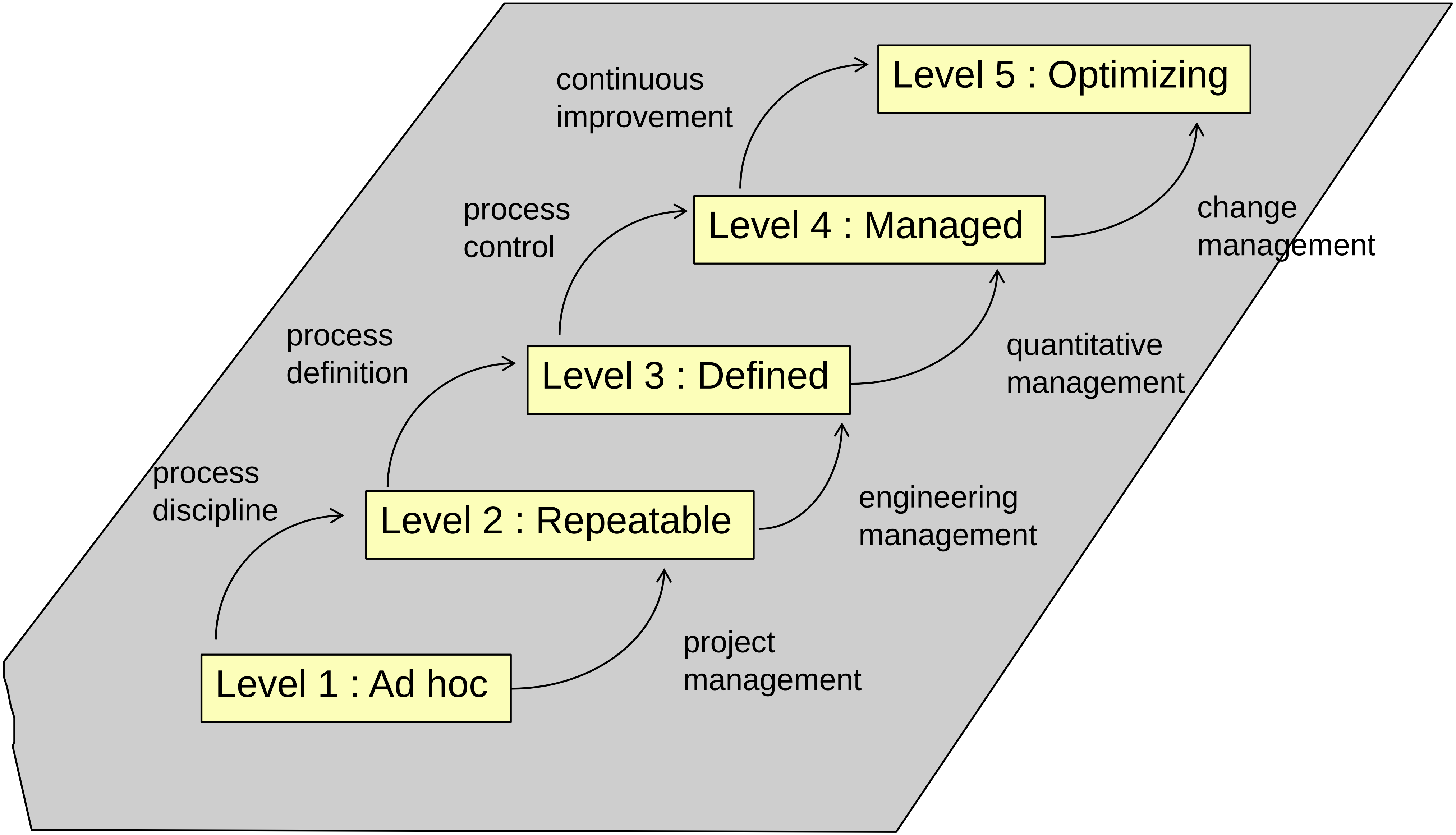


Allow changes to the system development process

For example: Allow to include a prototyping activity when certain measures indicate that requirements collected from the user are fuzzy

Measure products, processes and feedback to <u>control</u> and <u>change</u> the process

From “Ad hoc” to “Optimizing”



At what level is an enterprise?

Question: How can Key Performance Indicators (KPI) be implemented?

How do they relate to the database/DW schema of an enterprise?

What type of thing is a KPI? Are there common patterns?

KPI' are measurements

The function M is usually partial, i.e. undefined for certain input entities.

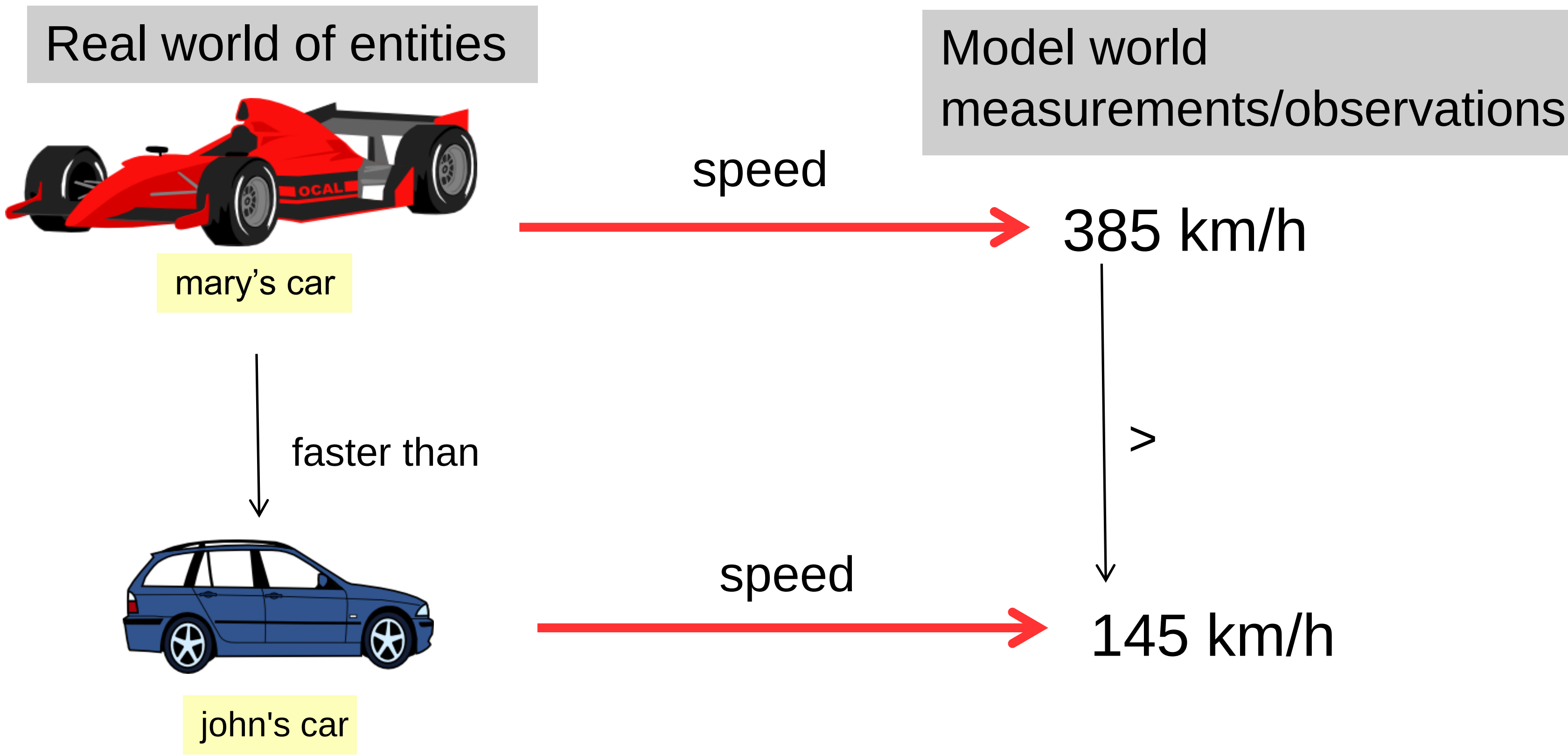
$$M: D \to R$$

- M: measurement function for an attribute
- D: Domain of **entities** to be measured
- R: Range of possible values for a measurement

Ranges can be number sets (integers, reals, intervals) or sets of symbolic names (e.g., “bad”, “average”, “good”, “very good”)

The measurement rules (esp. the units like centimeters vs. inches) must be fixed. Even when the unit is the same, results may be incomparable when the measurement method is not fixed. Example: measure weight with or without clothes

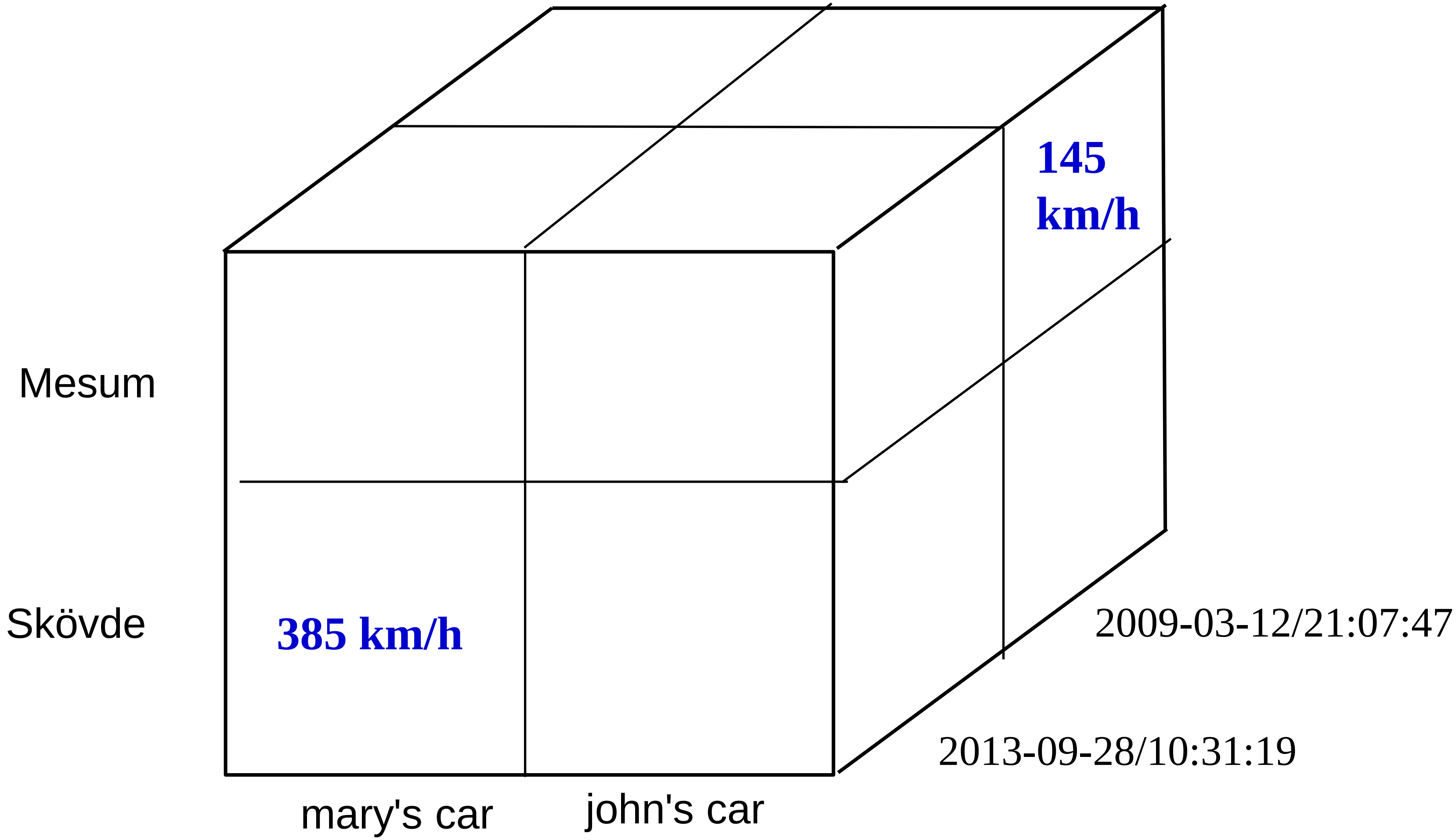
Representation condition for binary relations (see Fenton&Pfleeger)



Observation 1: Mary's car has a speed of 385 km/h at Skövde on 2013-09-28/10:31:19.
Observation 2: John's car has a speed of 145 km/h at Mesum on 2009-03-12/21:07:47.

Car A is faster than B if and only if speed(A) > speed(B)

Observations ("facts") in a data cube



The labels at the dimensions denote circumstances of the observations (time, location, participating entities, ...).

Observations = measurements functionally dependent on participating entities

$$d_1, d_2, ..., d_k \rightarrow m$$

dimensions
(participating entities)

measurement attribute

For the example observations:

Mary's car, Skövde,2013-09-28/10:31:19 → 385 km/h
John's car, Mesum,2009-03-12/21:07:47 → 145 km/h

As function expression:

speed(Mary's car, Skövde,2013-09-28/10:31:19)= 385 km/h

Values vs. entities

An entity has a referent (identifier) and describing properties (attributes). The identifier itself carries no meaning.

A value's label completely defines its meaning, e.g. a number.

Mary's car, Skövde, 2013-09-28/10:31:19 → 385 km/h

entities

reified value

value

The observation is about Mary' s car but also about Skövde. Both entities <u>participate</u> in the observation. The time is not a real entity but we “reify” it, i.e. we treat it as if it were an entity.

Observations as tables

car	location	timepoint	speed
Mary's car	*Skövde*	*2013-09-28/10:31:19*	*385 km/h*
John's car	*Mesum*	*2009-03-12/21:07:47*	*145 km/h*

participating entities

measured value

- there could be several measurement attributes in the same table if they are observed at the same circumstances

- we however focus on just a single measurement attribute per table or closer to star schema:

fact table: speeds

carid	locid	timid	speed
1001	*21*	*5001*	*385.0*
1002	*22*	*5002*	*145.0*

dimension table: car

carid	carname	...
1001	*Mary' s car*	
1002	*John's car*	

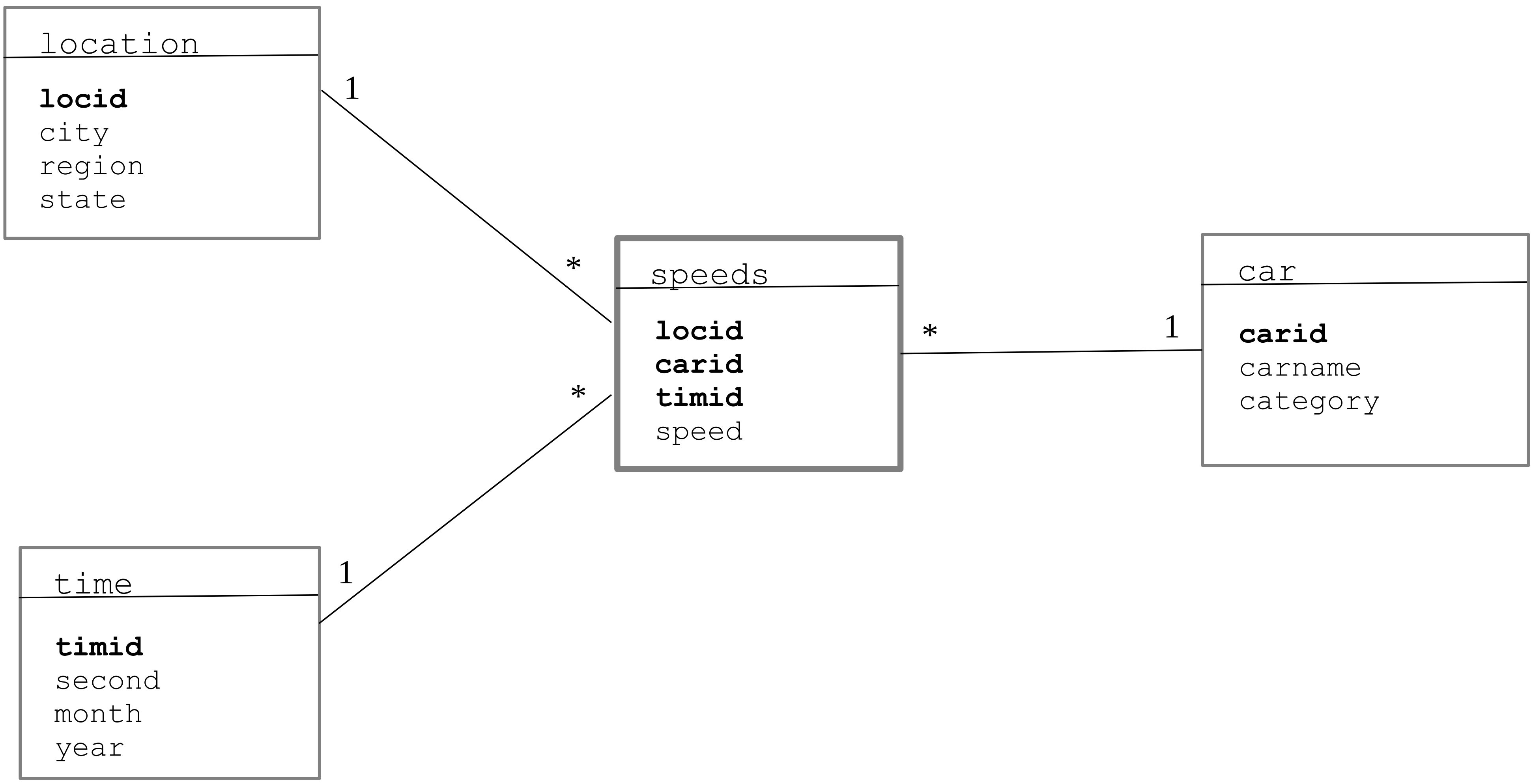
dimension table: time

timid	timevalue	...
5001	*2013-09-28/10:31:19*	
5002	*2009-03-12/21:07:47*	

dimension table: location

locid	locname	...
21	*Skövde*	
22	*Mesum*	

Star schema of the speeds example



Reminder: The primary key of the fact table consists of foreign keys referencing the dimension tables.

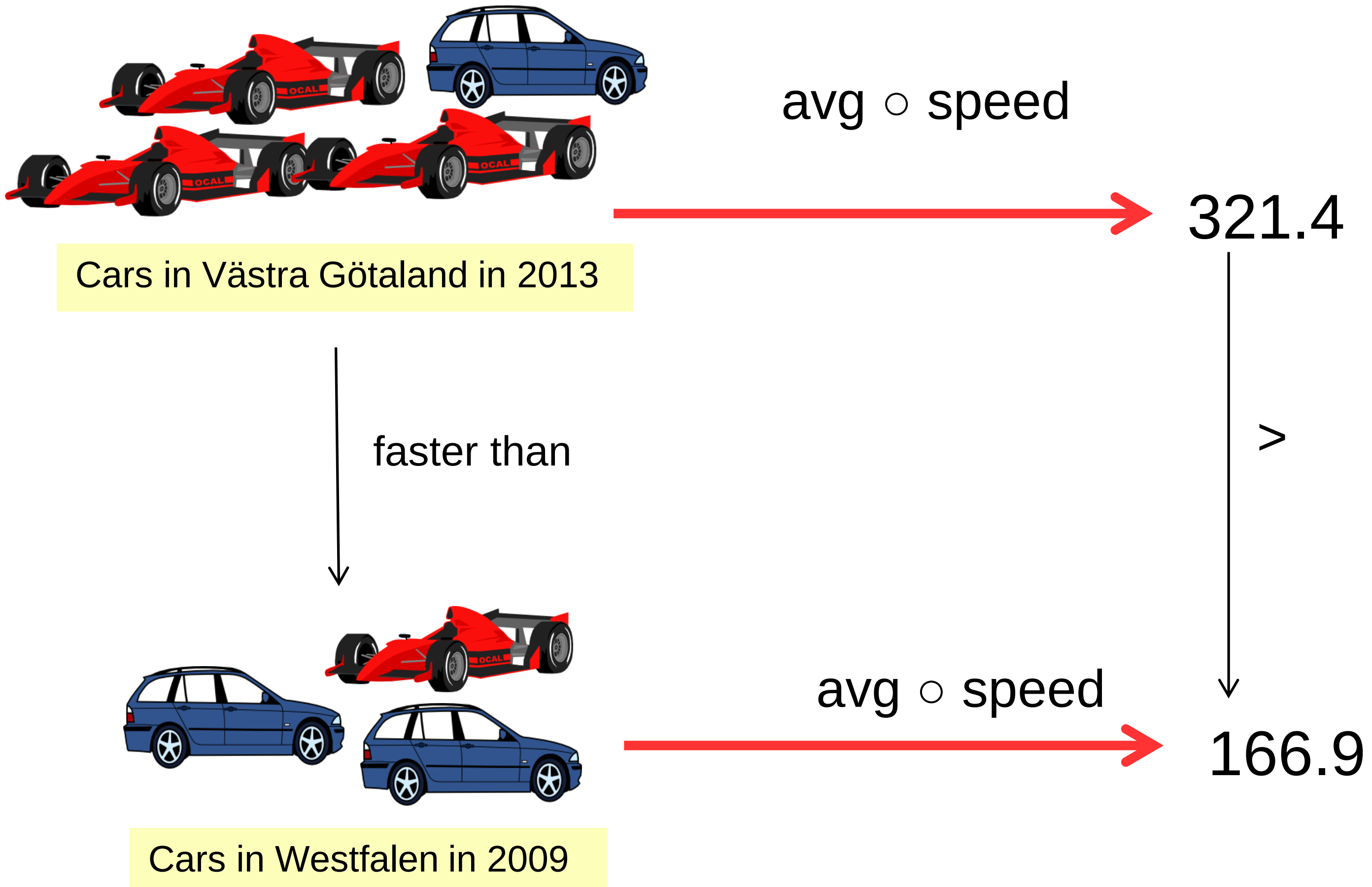
Table definitions in SQL

```
CREATE TABLE SPEEDS (
CARID INT,
LOCID INT,
TIMID INT,
SPEED FLOAT,
PRIMARY KEY (CARID,LOCID,TIMID)
) ;
```

fact table for
the observations

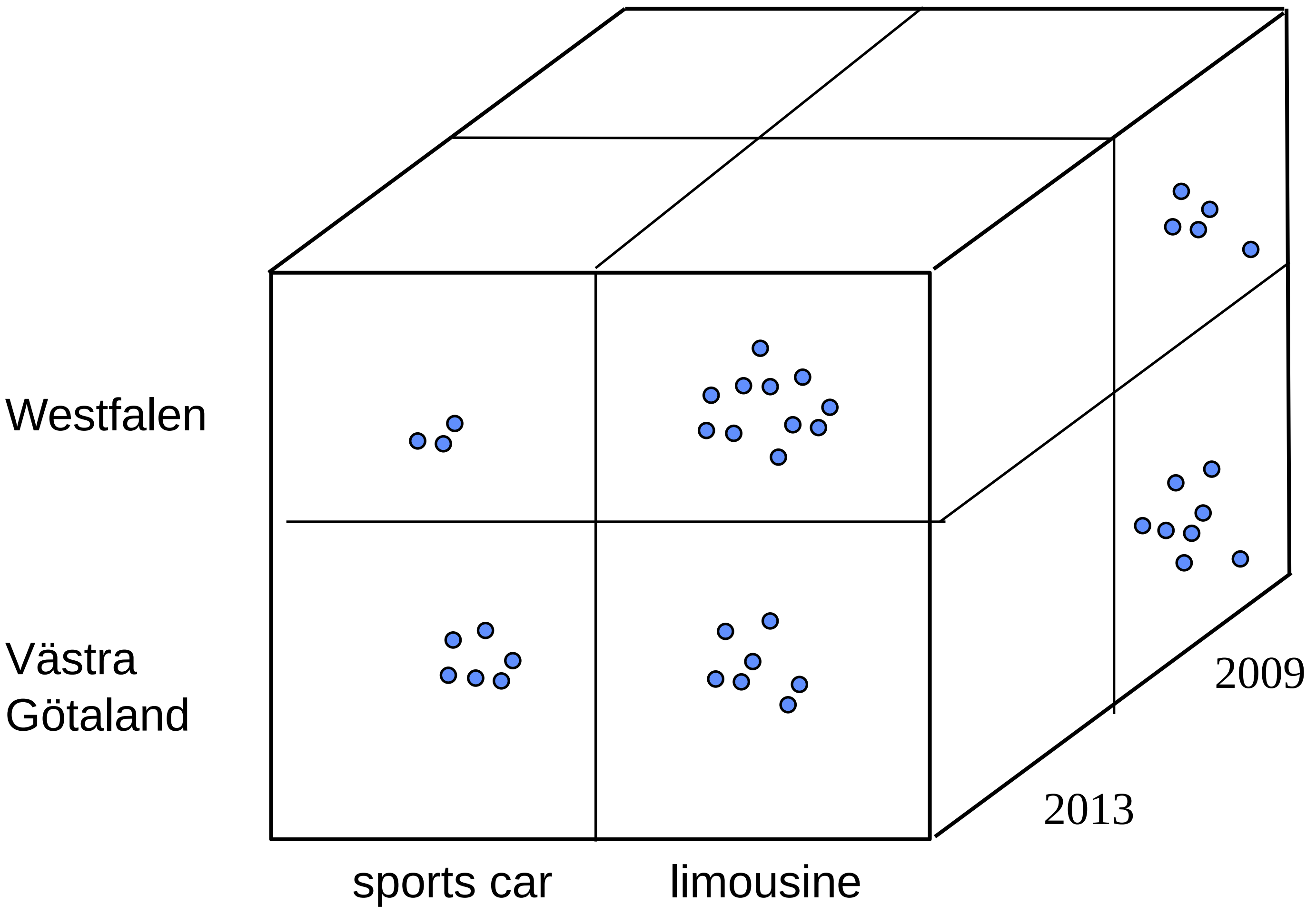
The foreign key references are left out here.
See full definition after the table definitions for the dimensions.

Sets of entities can be measured and compared as well!



Other aggregation operators than avg possible as well.

Aggregated observations in a data cube



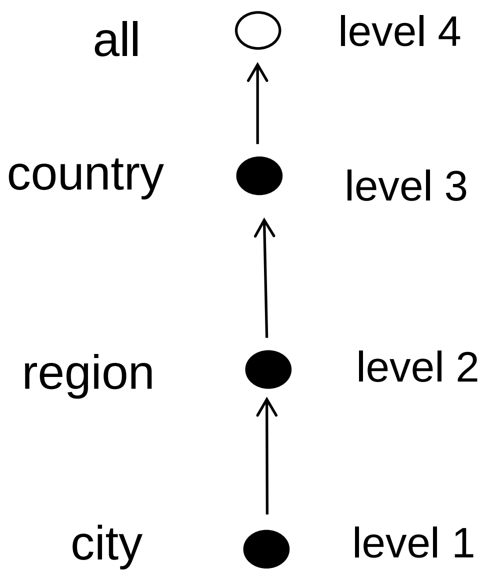
Each blue dot stands for a speed observation. The dimensions are now rolled up to some degree.

Hierarchy levels for location dimension

location

locid	city	region	country	level
1	Skövde	Västra Götaland	SWE	1
2	Mesum	Westfalen	GER	1
3	null	Västra Götaland	SWE	2
4	null	Westfalen	GER	2
5	null	null	SWE	3
6	null	null	GER	3
0	null	null	null	4

```
CREATE TABLE LOCATION (
LOCID INT ,
CITY VARCHAR(20),
REGION VARCHAR(30),
COUNTRY CHAR(3),
LEVEL INT NOT NULL,
PRIMARY KEY (LOCID)
);
```



When querying a fact table, one may not mix fact entries with dimension keys at different levels!

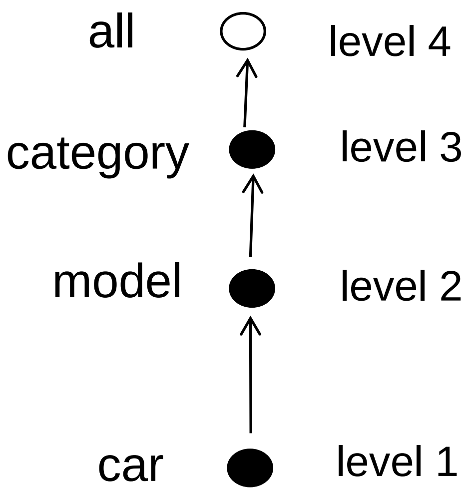
Note that the key of the location table now allows to refer to different levels of the location dimension! Hence, the fact table can contain entries at various aggregation levels.

Hierarchy levels for car dimension

car

carid	nplate	carmodel	category	level
1	DH53637	GOLFV	compact	1
2	SJ73637	Ferrari5	sportscar	1
3	null	GOLFV	compact	2
4	null	Ferrari5	sportscar	2
5	null	null	compact	3
6	null	null	sportscar	3
0	null	null	null	4

```
CREATE TABLE CAR (
CARID INT,
NPLATE VARCHAR(12),
MODEL VARCHAR(25),
CATEGORY VARCHAR(20),
LEVEL INT NOT NULL,
PRIMARY KEY (CARID)
);
```

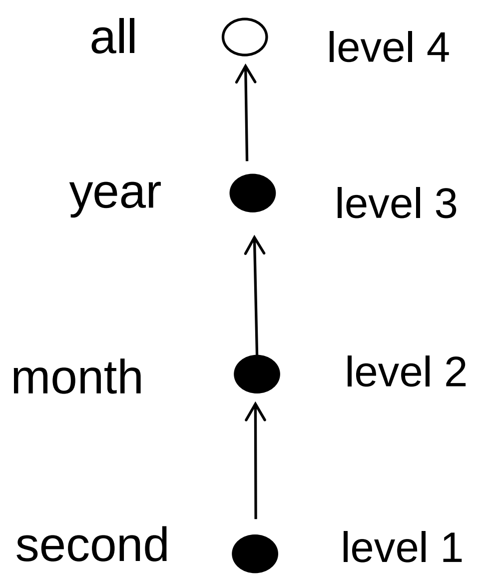


Hierarchy levels for time dimension

timetbl

timid	second	month	year	level
1	2013/09/28--10:31:19	201309	2013	1
2	2009/03/12--21:07:47	200903	2009	1
3	null	201309	2013	2
4	null	200903	2009	2
5	null	null	2013	3
6	null	null	2009	3
0	null	null	null	4

```
CREATE TABLE TIMETBL (
TIMID INT,
SECOND DATETIME,
MONTH CHAR(6),
YEAR CHAR(4),
LEVEL INT NOT NULL,
PRIMARY KEY (TIMID)
);
```



Full fact table SPEEDS

```
CREATE TABLE SPEEDS (
CARID INT,
LOCID INT,
      TIMID INT,
SPEED FLOAT,
PRIMARY KEY (CARID,LOCID,TIMID),
      FOREIGN KEY (CARID) REFERENCES CAR (CARID),
FOREIGN KEY (LOCID) REFERENCES LOCATION (LOCID),
FOREIGN KEY (TIMID) REFERENCES TIMETBL (TIMID),
) ;
```

The type FLOAT is in SQL-Server for 8byte binary floating point numbers.
Other DBMS like MySQL use the label 'DOUBLE' for this type.

Subsequently, we assume that the fact table only contains tuples at the lowest level of granularity (level 1).

Otherwise, one would have to use the level attribute of the dimension tables to restrict the query to level 1 facts.

This assumption is only for keeping the subsequent considerations simple. Of course, a real KPI implementation with DW's shall utilize the materialization of the higher aggregation levels!

KPI 1: "Average speed of sports cars in 2013 in Västra Götaland"

avg{speed(Sportscar,2013,Västra Götaland)}

measure

observations

participating entities

```
SELECT AVG(SPEED) FROM SPEEDS,CAR,TIMETBL,LOCATION WHERE
SPEEDS.CARID = CAR.CARID AND
SPEEDS.TIMID = TIMETBL.TIMID AND
SPEEDS.LOCID = LOCATION.LOCID AND
CAR.CATEGORY = "Sportscar" AND
TIMETBL.YEAR = "2013" AND
LOCATION.REGION = "Västra Götaland" ;
```

join to dimensions

selection on dimensions

So, this is a classical datawarehouse-like query on the data cube.

Measures on entity sets deliver (multi-)sets of values

level=1: individual entities as arguments

speed(mary's car,2013-09-28/10:31:19,Skövde) → 385

entities

level>1: entity sets as arguments

speed(mary's car,2013-09,Västra Götaland) → {75,102,102,385}

entity   entity sets

Västra Götaland = { Ale,Alingsås,Bengtsfors,Bollebygd,Borås,Dals-Ed,Essunga,Falköping, Färgelanda,Grästorp,Gullspång,Götene,Göteborg,Herrljunga,Hjo,Härryd, Karlsbor,Kungälv,Lerum,Lidköping,Lilla Edet,Lysekil,Mariestad,Mark, Mellerud,Munkedal,Mölndal,Orust,Partille,Skara,Skövde,Sotenäs, Stenungsund,Strömstad,Svenljunga,Tanum,Tibro,Tidaholm,Tjörn, Tranemo,Trollhättan,Töreboda,Uddevalla,Ulricehamn,Vara,Vårgårda, Vänersborg,Åmål,Öckerö }

multi-set: set where elements can occur more than once.

Explicit entity sets

speed(*{mary's car,john's car},*
{2013-09-28...2013-10-05},
Västra Götaland+Malmö) → {63,75,87,102,102,121,147,385}

So, in general the measures have entity sets as arguments
and then deliver multi-sets of values.

The multi-sets are subject to aggregation such as avg,sum,min,...

KPI 2: “Number of cars in Västra Götaland”

count{c:Car| c.registration.region=Västra Götaland}

participating entities   observations

```
SELECT COUNT(CAR.CARID) FROM CAR,REGISTRATION WHERE
       CAR.CARID = REGISTRATION.CARID AND
       REGISTRATION.REGION = “Västra Götaland” ;
```

We need another table such as REGISTRATION to link a car to the required region.

The registration of a car at a certain location in a region is an observation. The observation has no measurement attribute here but could have further dimensions like time, car owner, etc.

So, this is a KPI where we only can count to map observations to numbers.

Q: Could a car be registered twice and then counted double? Check yourself!

KPI 3: “Number of speed observations in Västra Götaland where speed was > 100.”

count{s:speeds| s.location.region=Västra Götaland, s.speed> 100}

observations

participating entities

measure

```
SELECT COUNT(*) FROM SPEEDS,LOCATION WHERE
       SPEED > 100.0 AND
       SPEEDS.LOCID = LOCATION.LOCID AND
       LOCATION.REGION = “Västra Götaland” ;
```

Time and car dimensions are not used here, so any car & time matches.

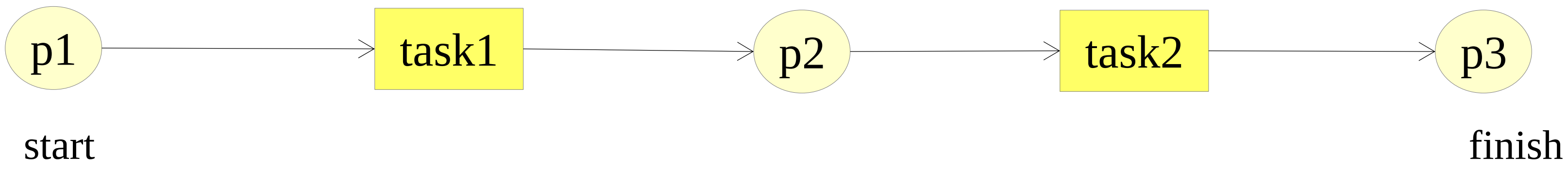
KPI 4: “Number of speed observations in Västra Götaland where speed > 100 / #cars in Västra Götaland.”

This is just KPI3/KPI2, so we once we have the basic KPIs, then we can combine them to form derived KPIs.

Note that the two KPIs are using a common parameter that must have the same meaning for both KPIs, i.e. the region Västra Götaland referred to in the SPEEDS table is the same as referred to in the REGISTRATION table.

This may not always be the case for dimension, e.g. the time is location-dependent. We might have to convert local times thus to universal times.

KPIs in processes



KPI 5: “How long do objects (case, product, shipment,...) need from start to end?”

KPI 6: “How many objects of type T arrived in the start place on a given date?”

KPI 7: “How long does task1 need to process an object?”

KPI 8: “How long does an object wait on place p?”

N.B.: The above diagram can be read as petri net.

Step 1: Identify observation types, entities, and measures

entity — measure

"object X arrives at place p at time t"

"object X departs from place p at time t"

entity — measure

KPI 7: "How long does task1 need to process an object?"

$$processtime(task1) = avg\{departuretime(o,p1) - arrivaltime(o,p2) \mid o \in OBJECT\}$$

Assumption: We only consider objects o for which both departure time and arrival time have values.

Step 2: Define tables for the observation types

arrival

objid	placeid	arrtime
o1	p1	10:31
o2	p1	10:37
o1	p2	11:03
o1	p3	12:23

```
CREATE TABLE ARRIVAL (
        OBJID VCHAR,
        PLACEID VCHAR,
        ARRTIME DOUBLE,
        PRIMARY KEY (OBJID,PLACEID)
)
```

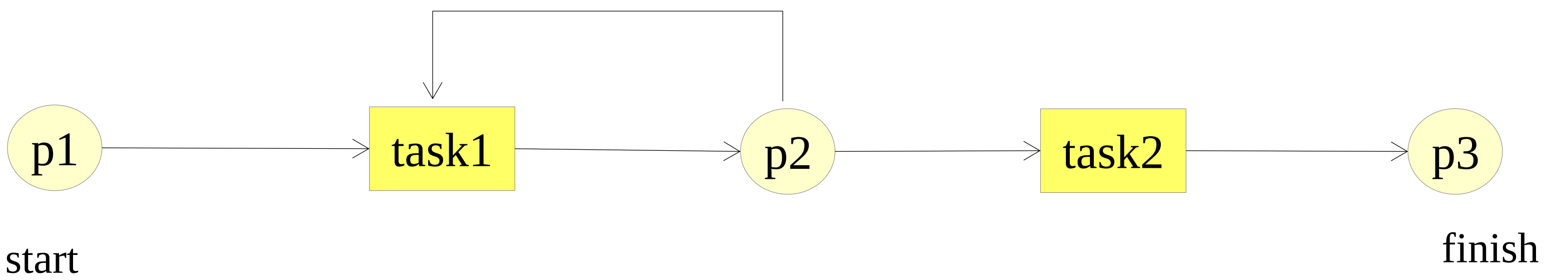
departure

objid	placeid	deptime
o1	p1	10:45
o2	p1	11:12
o1	p2	11:27

```
CREATE TABLE DEPARTURE (
        OBJID VCHAR,
        PLACEID VCHAR,
        DEPTIME DOUBLE,
        PRIMARY KEY (OBJID,PLACEID)
)
```

Assumption: objects are not re-entering places, i.e. we do not consider processes with loops here
NB: We could also have a single fact table for observing arrival and departure time, but then NULL values would occur

Question: How would the schema look like when objects can re-enter places?



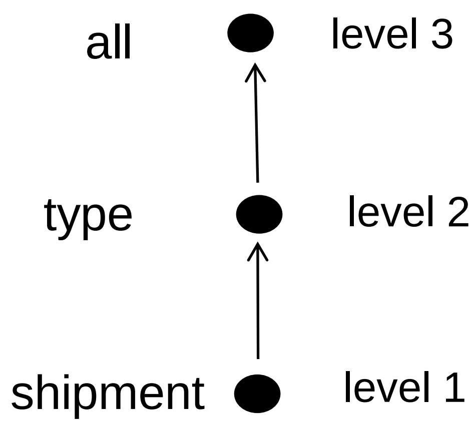
Analyze yourself!

Hierarchy levels for object dimension

object

objid	shipment	type	level
o1	238754623	letter	1
o2	854767732	parcel	1
o3	null	letter	2
o4	null	parcel	2
0	null	null	3

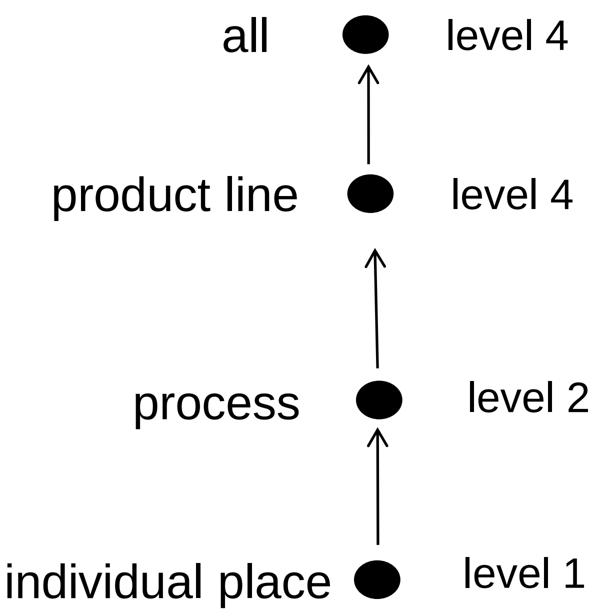
```
CREATE TABLE OBJECT (
        OBJID INT UNSIGNED ,
        SHIPMENT CHAR(9),
        LEVEL INT NOT NULL,
        PRIMARY KEY (OBJID)
)
```



Hierarchy levels for place dimension

place

placeid	placelabel	process	productline	level
1	p1	post delivery	postal service	1
2	p2	post delivery	postal service	1
3	p3	post delivery	postal service	1
4	null	post delivery	postal service	2
5	null	sell stamps	postal service	2
6	null	null	postal service	3
0	null	null	null	4




```
CREATE TABLE PLACE (
        PLACEID INT UNSIGNED,
        PLACELABEL VCHAR,
        PROCESS VCHAR,
        PRODUCTLINE VCHAR,
        LEVEL INT NOT NULL,
        PRIMARY KEY (PLACEID)
)
```

“Arrival time of an object at a place.”

arrivaltime(o,p)

measure

observations

participating entities

```
SELECT ARRTIME FROM ARRIVAL WHERE
       OBJID = o AND
       PLACEID = p  ;
```

As fact table for all possible objects and places:

```
SELECT OBJID,PLACEID,ARRTIME FROM ARRIVAL;
```

“Departure time of an object at a place.”

departuretime(o,p)

measure

observations

participating entities

```
SELECT DEPTIME FROM DEPARTURE WHERE
       OBJID = o AND
       PLACEID = p  ;
```

KPI 5: “How long do objects (case, product, shipment,...) need from start to end?”

$$leadtime(o) = arrivaltime(o,pe) - arrivaltime(o,ps)$$

end place of the process

start place of the process

Note: We disallowed loops in our (too simple) process schema.

KPI 6: “How many objects of type T arrived in the start place on a given date?”

count({a:arrival| a.objid.type=T, a.arrtime IN D,a.placeid=p1})

defined measure — observations — participating entities — used measure

```
SELECT COUNT(*) FROM ARRIVAL,OBJECT WHERE
        ARRIVAL.OBJID = OBJECT.OBJID AND
        OBJECT.TYPE = T AND
        ARRIVAL.PLACEID = p1 AND
        "ARRTIME IN Day D" ;
```

mapping to SQL not complete; need to link times to dates

KPI 7: “How long does task1 need to process an object?”

processtime(task1) = {departuretime(o,p1) - arrivaltime(o,p2)| o in OBJECT}

- result is a multi-set of numbers

avgprocesstime(task1) = avg(processtime(task1))

- we can aggregate the multi-set

mapping to SQL omitted

KPI 8: “How long does an object wait on place p?”

avgwaittime(p) = avg{arrivaltime(o,p) - departuretime(o,p)| o in OBJECT}

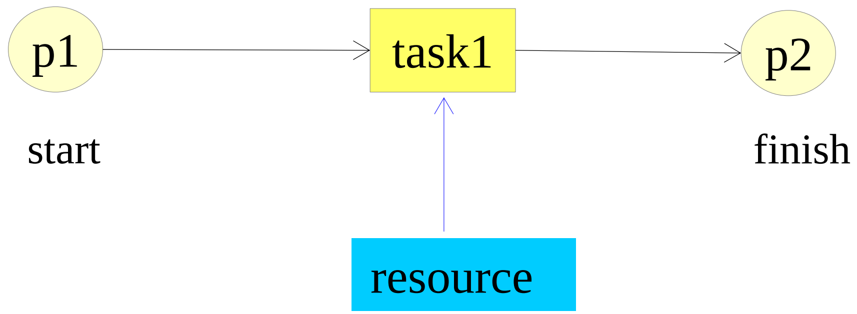
KPI 9: “What is the aggregated waittime of an object in a given process?”

procwaittime(proc) = sum{arrivaltime(o,p) - departuretime(o,p)|
o in OBJECT, p.process=proc}

only correct when the process
has no loops!

mapping to SQL omitted

KPI's on resource consumption



Examples:

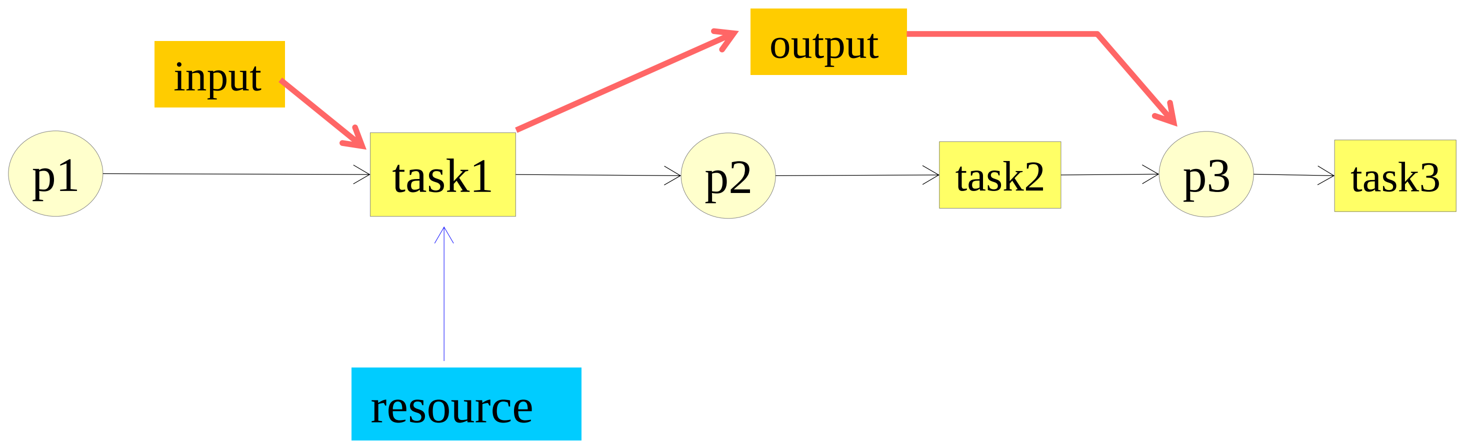
How many person hours are spent on task1 for an object?

What percentage of the shipment time of task1 is requiring the activity of the truck's cooling device?

What is the average power consumption of machine X performing task1?

KPI definition and mapping to tables/queries left to your exercise!

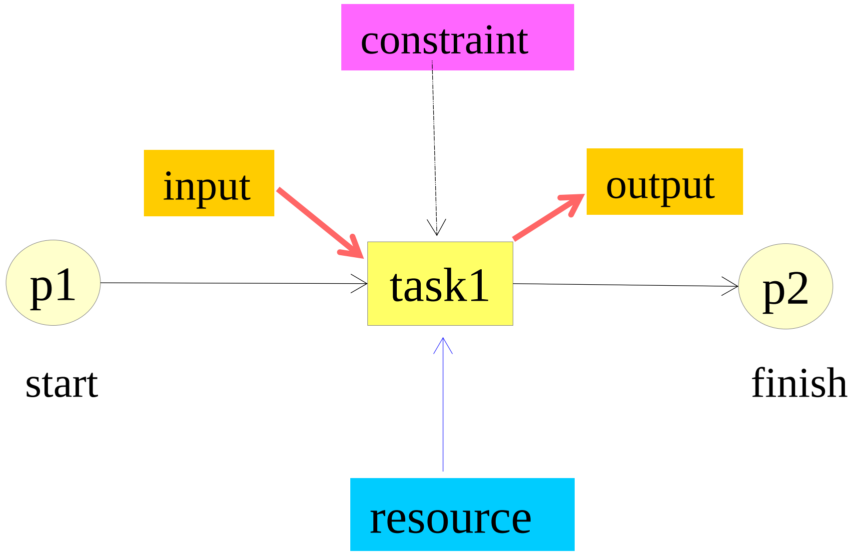
KPI's on input/output products



“How much aluminum is needed to build the engine of type T for a car?”

“How is the defect density of part X related to the amount of time spent on producing part X?”

KPI's on scheduling and budget constraints



"How many projects overspend their budget (or deadline)"?

"Is a tight project deadline affecting the quality of the result?"

inspired by SADT

Example KPIs (inspired by kpilibrary.com)

inventory turn time : average time in months that it takes to sell the whole inventory for a given product and a given warehouse

schedule adherence : difference of the actual production scheduling from the planned scheduling

truck turnaround time : time between the arrival of a truck at a station and its departure

first time correct deliveries : percentage of product shipments that correctly arrive at the customer at the first delivery attempt

Exercise (~ 30 min):

1) What entities are involved?

2) What is the underlying process model?

3) Which DW schema can cater for the KPI?

4) Define the query to evaluate the KPI

Research questions

- Which patterns of KPI's occur in the industry? How to describe the patterns?

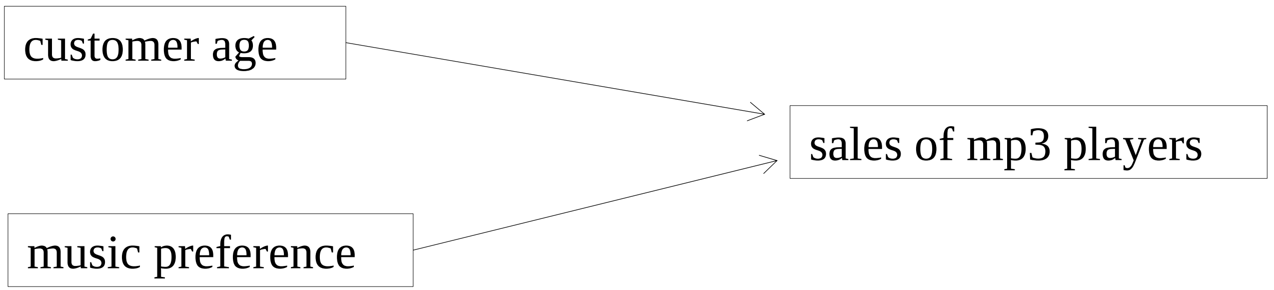
  http://kpilibrary.com

- Is there an algebraic/textual notation for KPIs that is both readable by domain experts and formal enough to be mapped to table structures and SQL queries?

- What parts of the PKI implementation can be automated? What additional knowledge has to be included to automate the implementation?

A theory based on KPIs?

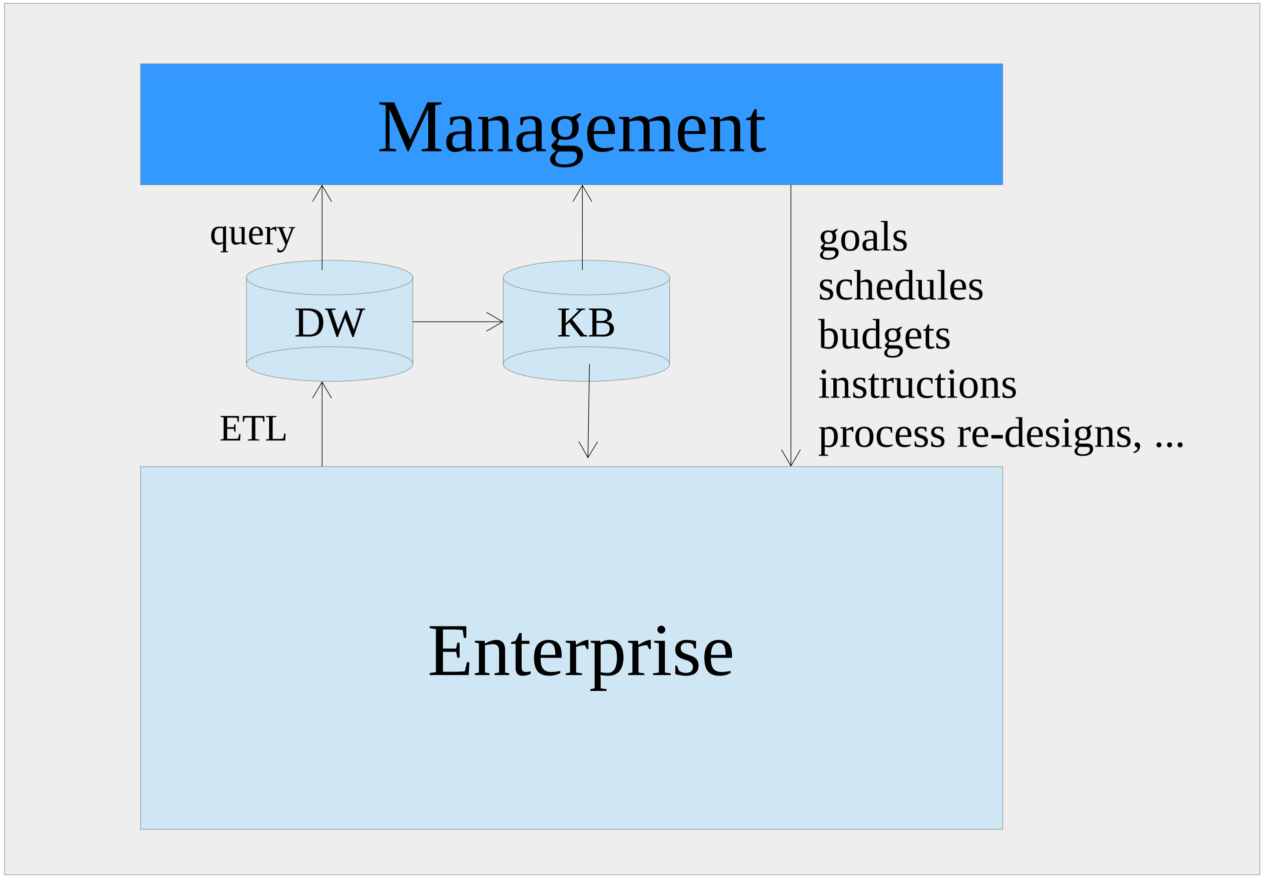
In statistics, dependent and independent variables are used to validate whether a certain theory in terms of these variables is valid?



eS = c * (18 - age) * mpref

Validated theories allow to predict the future pretty much like in SPC, though we have an even greater problem with hidden variables.

A knowledge base of valid KPI theories



The KB contains the equations encoding valid theories.

How to maintain the theories when the DW changes?
How trustable is a theory?
What about non-linear dependencies?

Summary

- KPI' s are closely linked to the multi-dimensional model of DW's

- KPI's are based on observations (fact table of DW)

- The observations are taken from running processes

- Making the process explicit helps to understand how the facts can be collected

To do

- create a language for specifying KPIs such that the DW schema and the queries can be derived from it