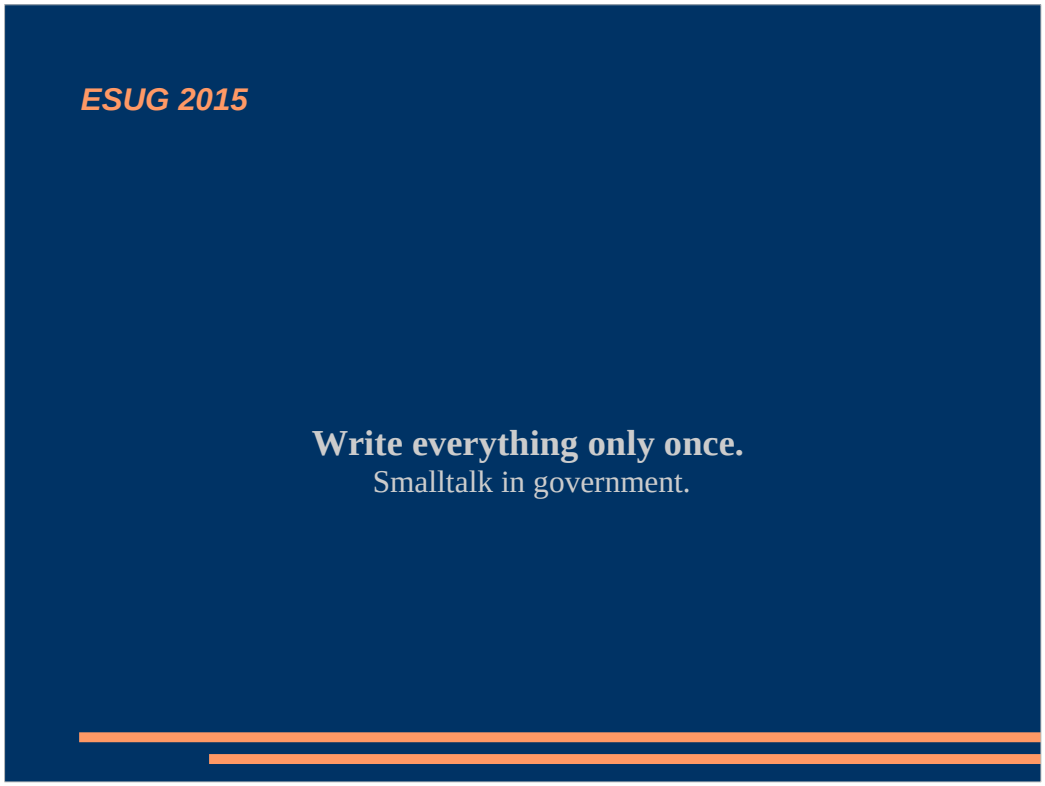
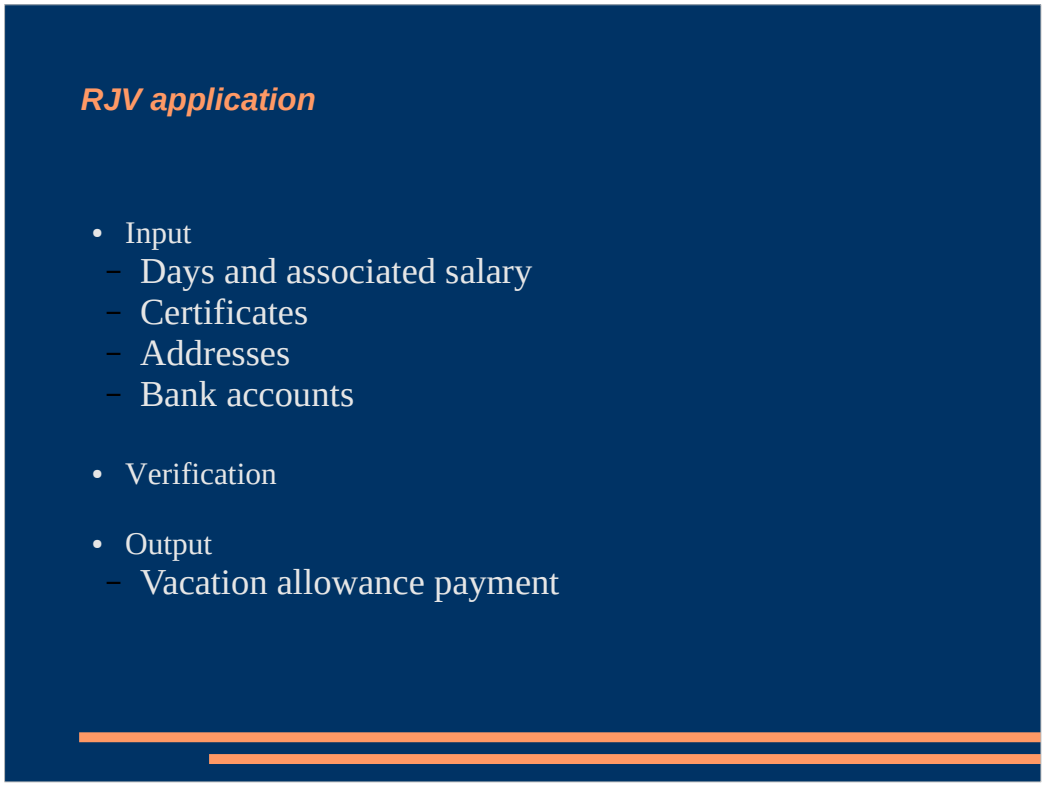


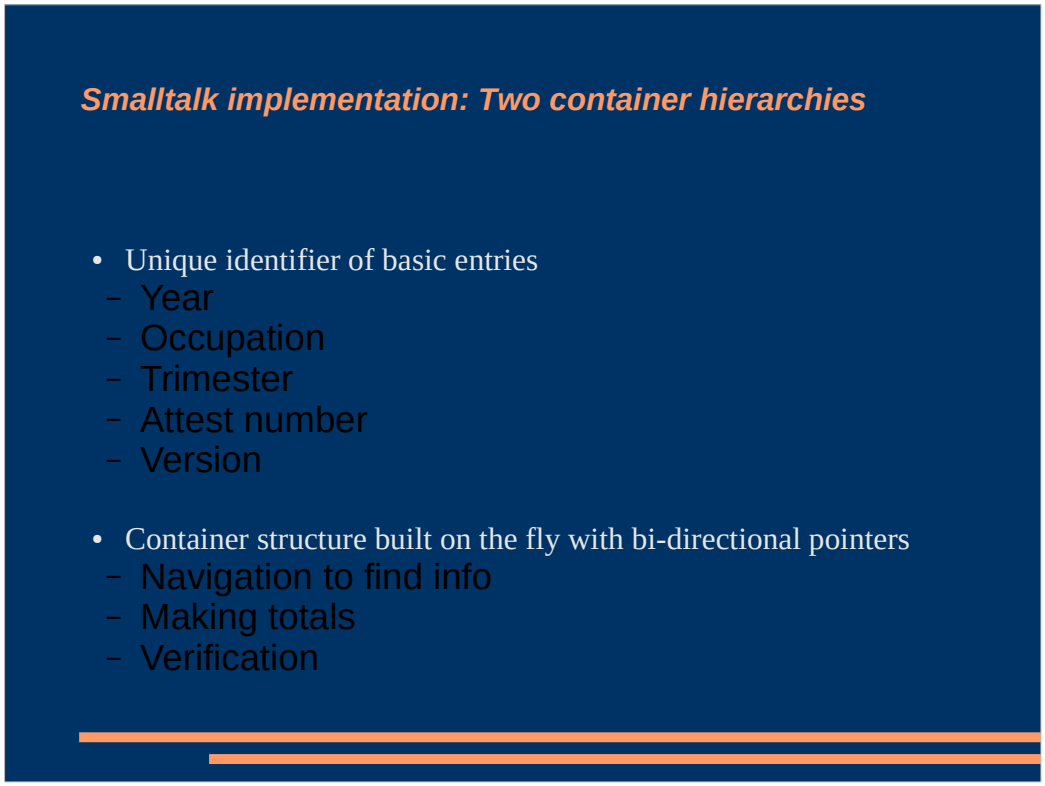
The subject om my talk is to illustrate the power of chosing “Write everything only once” as the guiding thought for development. This means always look out for duplicate code and eliminating it.
My discovery came while over a period of five years I spent on and off probably two years in the (re)development of an application for the RJV. The RJV is a Belgian government agency that pays the vacation allowance for the workers.
In fact I threw away much moreaway than what was finaly left.

Overview

- RJV application highlights
- Smalltalk implementation
- Metrics of different implementations
- How to be successfull



Salary and days are reported each trimester.
Certificates (for example illness, technical unemployment, …) confirm validity of salary or days reported.
Addresses and bank accounts are needed to execute payments.



– Two container hierarchies:
  – Salary and days
  – Certificates
– Navigation is mainly to retrieve information from higher in the container hierarchy. Some info is available at the trimester, occupation, year or worker level.
– Totals are required at all container levels, worker, year, occupation, trimester.
– Verification uses both hierarchies.

Show container structure

- Entity
  - BasicEntry
  - Container
- Identifier
  - LatgIdentifier
- VersionedEntry with history
  - AttestEntry
  - CarreerEntry
- Navigation

Verification – Delegation pattern squared

LatgEntity implements:

```
exceptions
    ^ RJVExceptionFinder new returnExceptionsOn: self.
```

RJVExceptionFinder implements:

```
possibleExceptions
    ^ self target  possibleExceptions collect:
     [:exception | exception asClass new target: self target; yourself].

returnExceptionsOn: anRJVLatgSubset
    self target:  anRJVLatgSubset.
    ^self possibleExceptions select: [:exception | exception isValid] .
```

Every real LatgEntity subclass implements:

```
possibleExceptions
    ^ #(RJVExceptionB005 RJVExceptionB006 RJVExceptionB010)
```

Show implementation

- Show code Exceptions B0036, B0038, B0039, B0048, B0049

Show implementation

- Show code Exceptions B0036, B0038, B0039, B0048, B0049

Making totals: CarreerTotal

- CarreerTotal collects required totals over container structure
- Both container and each subclass of basicEntry implement carreerTotal method:

```
carreerTotal
    ^self entries inject: RJVCarreerTotal new into: [:a :b | a +  b carreerTotal]


carreerTotal
    RJVCarreerTotal new
        daysWorked: self days;
        daysVacation: self days;
        daysVacationLengthCalculation: self daysVacationLengthCalculation;
        daysCalculationFictif: self days;
        daysFictifSalaryPaid: self days;
        salaryCalculationBrut: self salary;
        salaryCalculationFictif: self salary;
        yourself
```

Each real subclass of basicEntry implements its own version of carreerTotal which is a reduced copy of the one shown.

ErrorFreeCarreerTotal

Making total of error free entries

```
errorFreeCarreerTotal
|errorFreeEntries |
errorFreeCarreerTotal isNil ifTrue:
[
errorFreeEntries := self entries select: [:entry | entry isErrorFree].
self errorFreeCarreerTotal: (errorFreeEntries inject:
RJVCarreerTotal new into: [:a :b | a +  b errorFreeCarreerTotal])
].
^errorFreeCarreerTotal
```

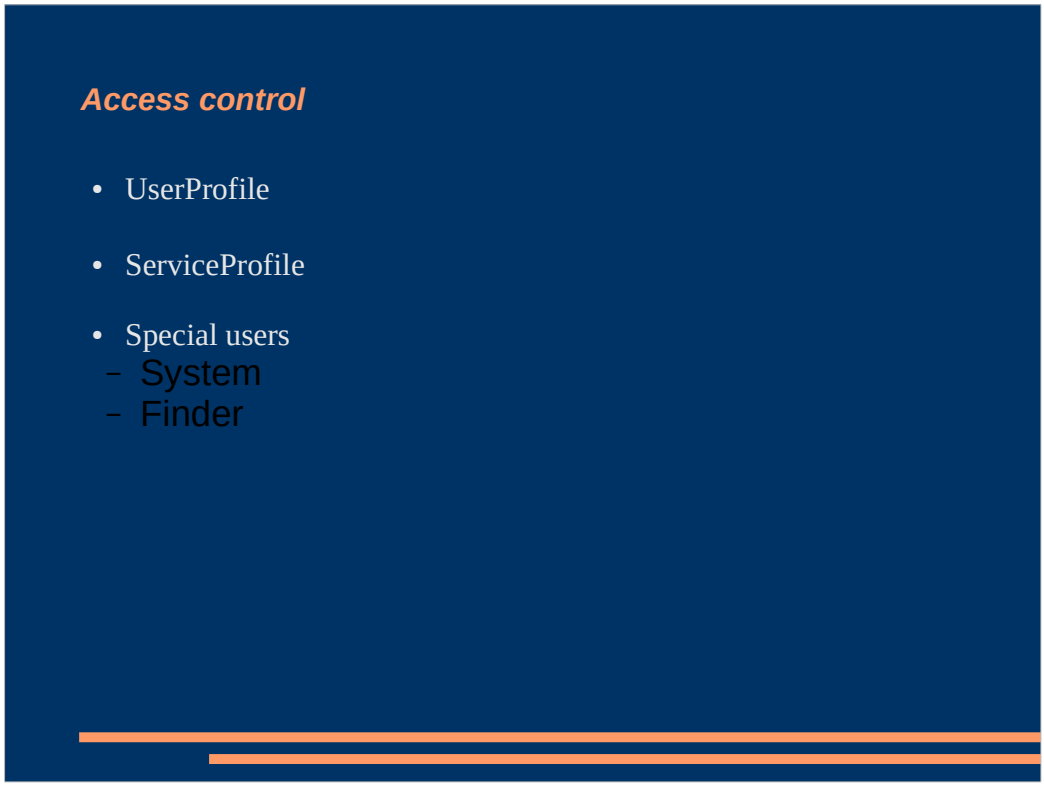
This message is sent repeatedly in the vacation allowance calculation.
For performance reasons errorFreeCarreerTotal uses lazy initialization.
Adding an entry to one of the container structures resets this total.

Advantages of implementation

- Duplicate code eliminated
- Simple navigation
- Yearly change limited
  - New days
  - New exceptions

Command pattern – Access control

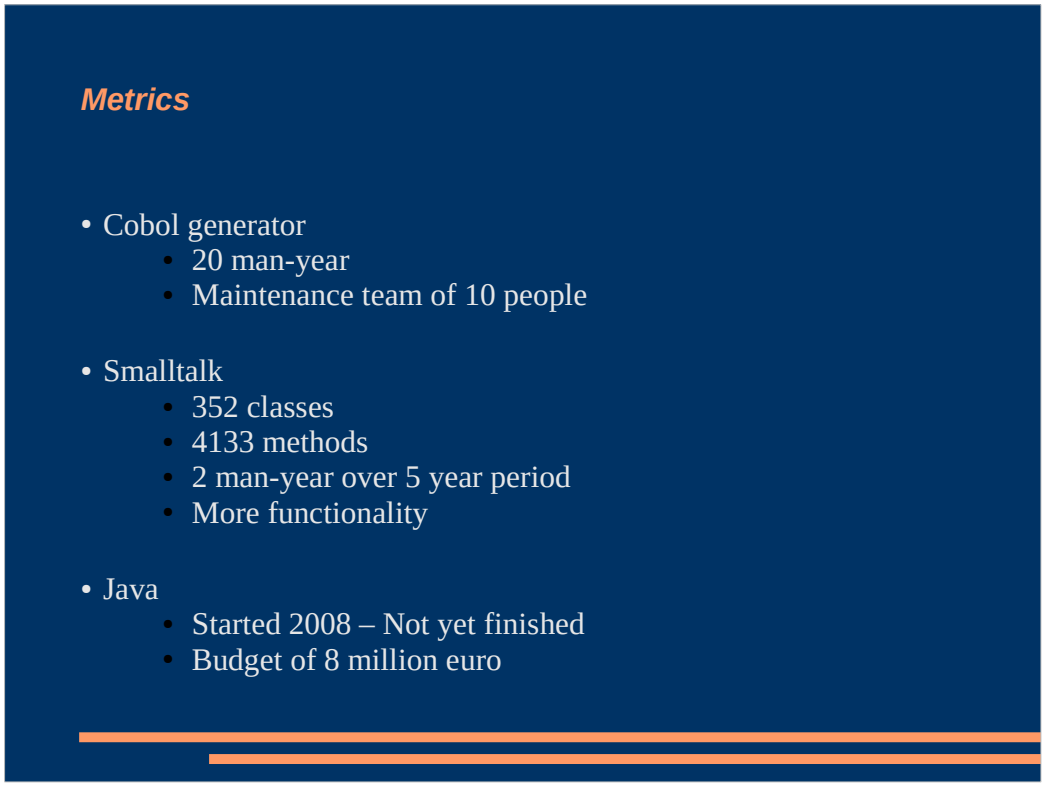
- Command pattern supports undo-redo
- Commands used for access control
- Commands used in DB support



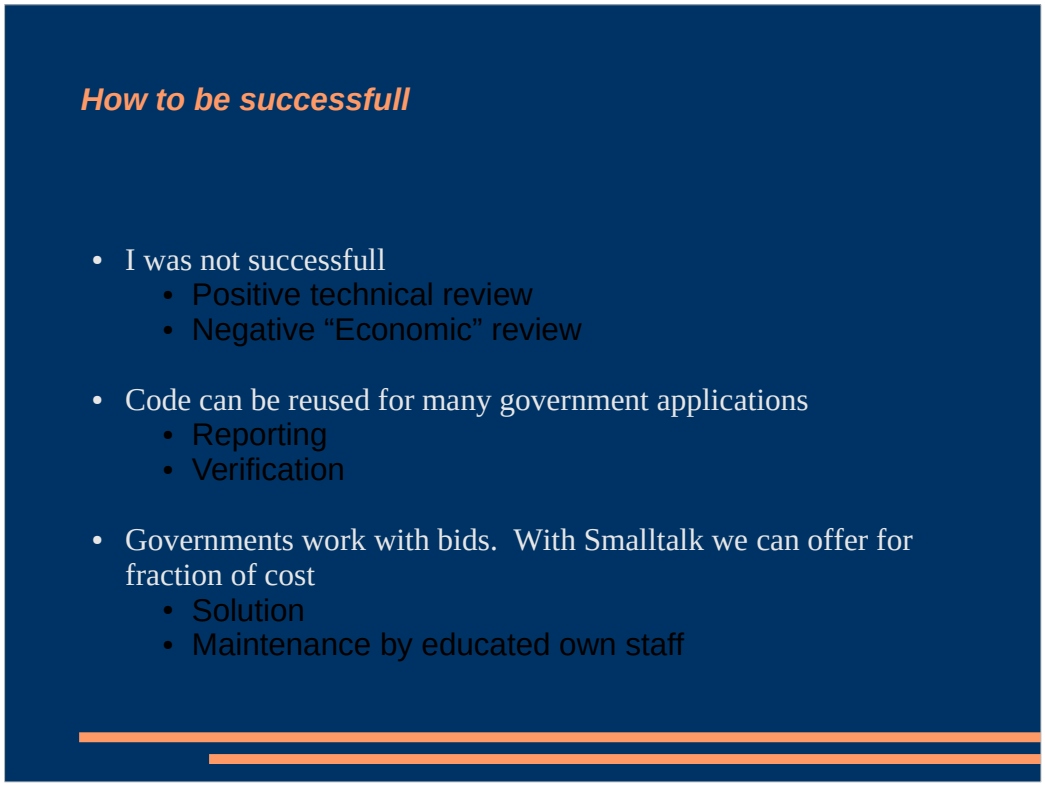
– A user profile is a collection of service profiles
– A service profile is a collection of commands
– A user can execute a command if it is included in one of its service profiles
– User System, User Profile System and Service Profile System. Service Profile System contains all commands.
– User Finder, User Profile Finder and Service Profile Finder. Service Profile System contains all Find commands.

Database support

- Object Extender
- Glorp



– Started with punched cards
– Cobol Generator Implementation was done under time pressure with sliced salami approach. Cut application into small pieces each with their own developer. Big maintencance problem.
– Smalltalk implementation was done by one developer.
– The JAVA implementation is done on a fixed price contract. A conservative extimate sets the effort at 40 man year.



– I was not. IBM consultant reviewed implementation and called it an example of what could be done with good OO design.

– An economy professor told management that the application should be written in a mainstream language – Java. It would only take two years