

UNIVERSITA' DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

## Part III: structured analysis

- previous lectures:
  - a structured language (c), structured programming principles
  - structured design: an abstraction for design of structured programs, based on structure chart, coupling and cohesion, with a clear transformation to a structured code
- this lecture:
  - structured analysis: an abstraction on structured design, based on Data Flow Diagrams and Transformations Analysis, with a clear transformation to a structured design
- Model Driven Engineering (MDE)
  - models as core artefacts of development, models transformation, models enrichment, automated code generation (Model Driven Development)

31

UNIVERSITA' DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

## Structured analysis

- Structured analysis makes a further step of abstraction
  - abstracts from the hierarchy of functions
  - to emphasize the flow of information and its transformation
- relies on a modeling artifact: the Data Flow Diagram
- and a method: transform analysis (and transactions analysis)

32

UNIVERSITA' DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

## Data flow diagram

- A data flow diagram documents information flows and transformations
  - Data flow: a flow of information
  - Process: produces output flows from input flows
  - Data store: information that persists (for some time)
  - Terminator: a component out of the system boundary

```

graph TD
    subgraph Top
        Input[nome] --> Process1([ottieni codice])
        Process1 <--> DataStore1[clienti]
    end
    subgraph Bottom
        Cliente[cliente] -- nome --> Process2([ottieni codice])
        Process2 -- "codice cliente" --> Output[ ]
        Process2 --> DataStore2[clienti]
    end
  
```

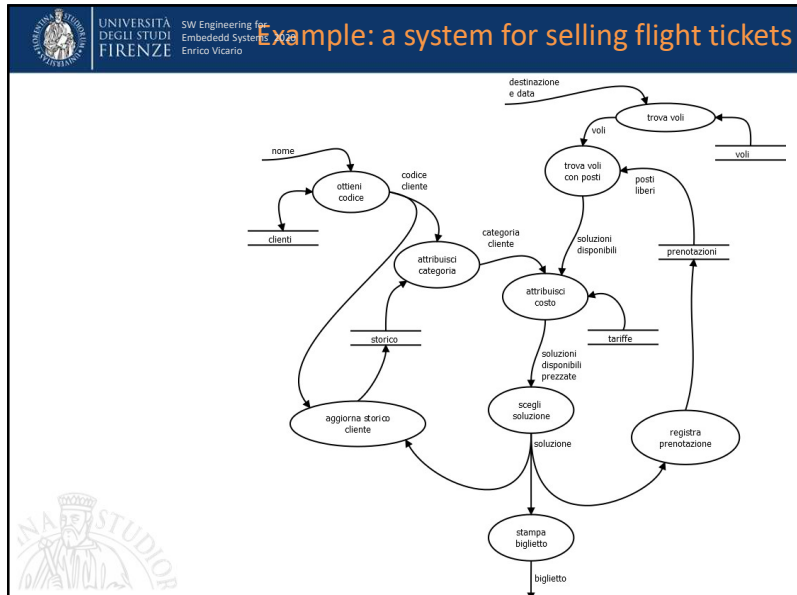
33

UNIVERSITA' DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

## Example: selling flight tickets (repeated)

- Functional requirements
  - Receives date, destination, customer name, price rates, flights, and available places
  - Books and reserves a place, and prints a ticket
  - While so doing, applies a pricing policy based on the customer history.

34



35

UNIVERSITA' DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

### Remarks and Heuristics

- Il dfd è un grafo e non un albero
  - And thus misses an intrinsic order
- Costruzione incrementale
  - È parallelo nel concetto pur non potendolo essere nella costruzione
  - Costruzione dal centro o dall'interfaccia?
- Non ha semantica di "triggering"
  - Non definisce prima/dopo
  - Né condizioni
  - Né ripetizioni
- Non rappresenta informazione di controllo
  - I segnali o le condizioni di attivazione sono trasparenti

36/44

36

UNIVERSITA' DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

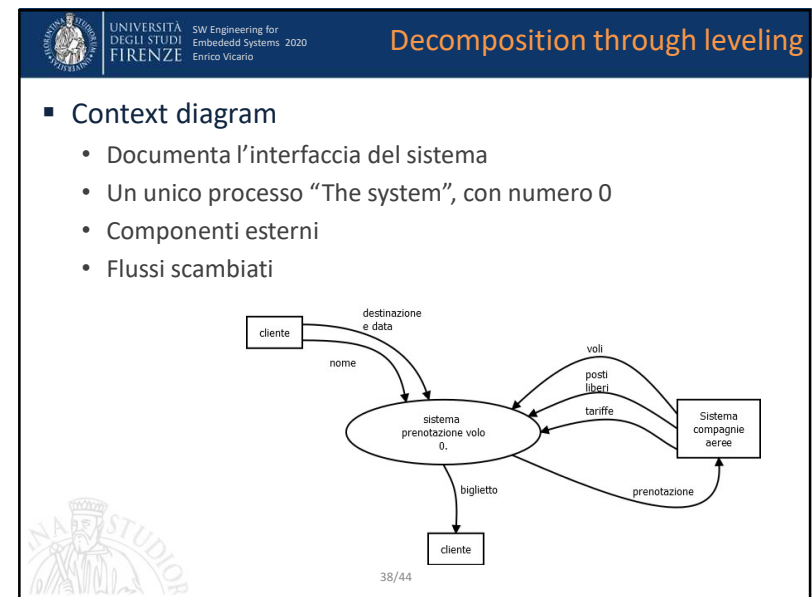
### Remarks and Heuristics

- Naming**
  - Ciò a cui non si riesce a dare un nome, spesso non rispetta la semantica prevista
  - Processo: predicato (transitivo attivo) con complemento
  - Flusso: sostantivo, spesso il complementi oggetto del processo
  - Ogni termine esprime un concetto specifico
    - Un singolo concetto "atomico"
    - Non un termine neutrale (file, archivio, dato, elabora)
- Rappresentare il processo di elaborazione senza distinguere la parte automatizzata rispetto a quella manuale
 

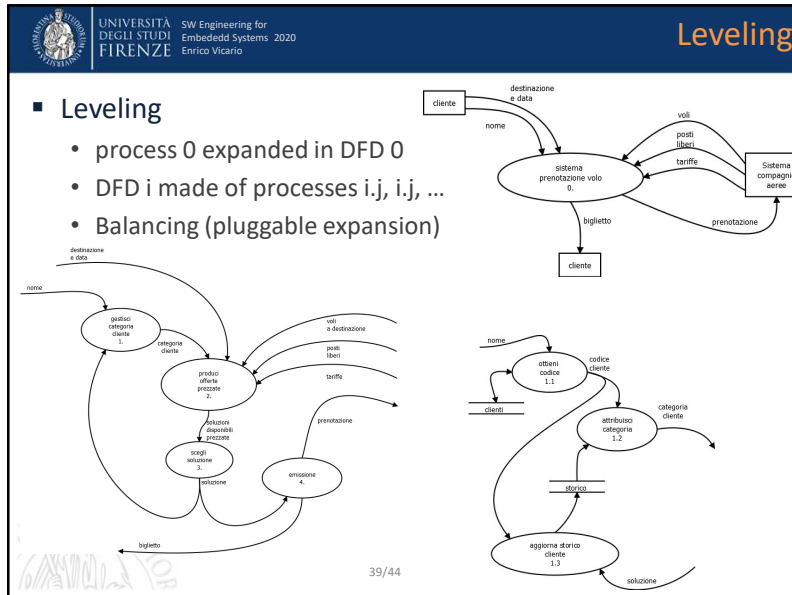
TBD: e.g. `select_flight` sara' probabilmente un dialogo con l'utente, con filtri e selezione

37/44

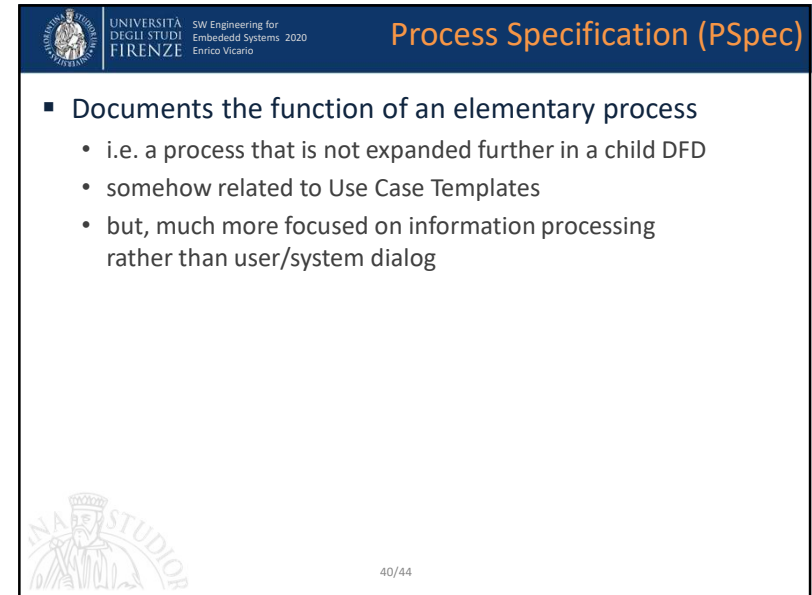
37



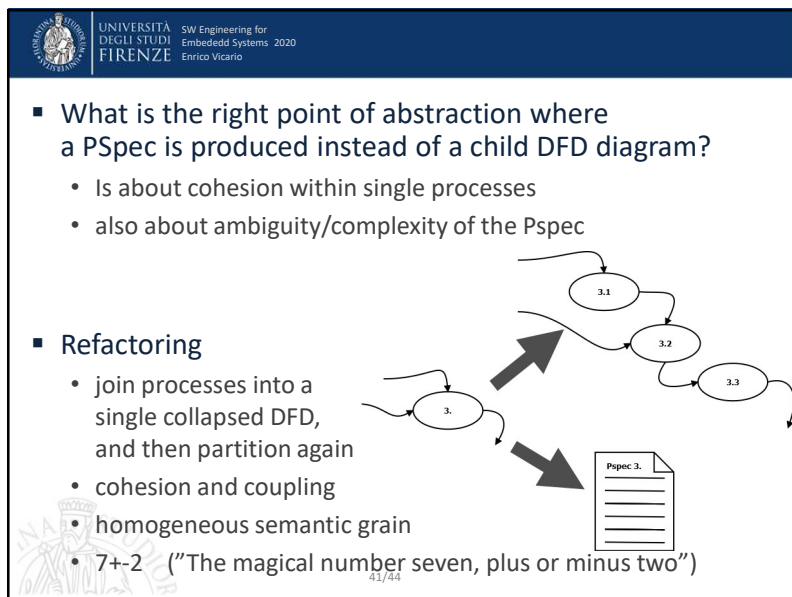
38



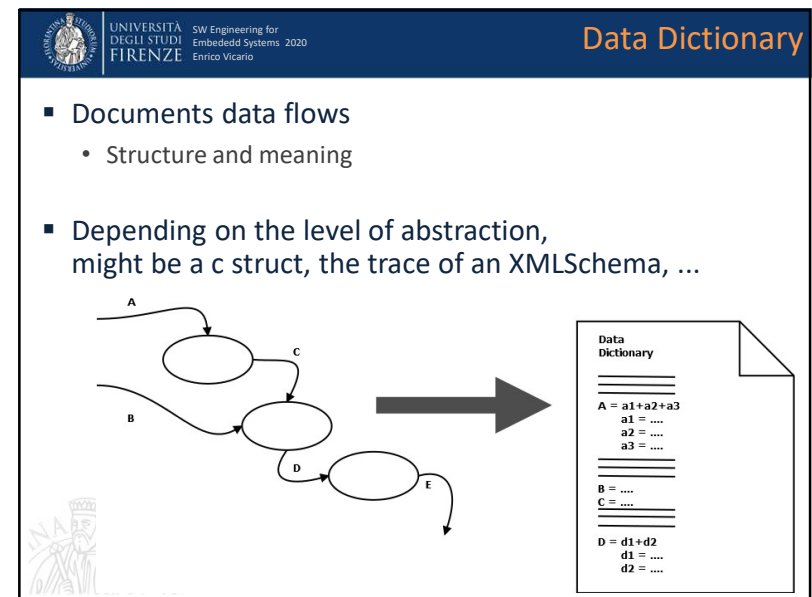
39



40



41



42

UNIVERSITA' DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

## Overall model

- Data Flow diagram: context diagram and hierarchical expansion
- Process Specifications: document the function of elementary processes
- Data dictionary: documents flows structure and meaning

43

UNIVERSITA' DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

## Analisi delle trasformazioni

- Direzione di raffinamento dell'informazione
- Modulo afferente: trasforma informazione nella direzione dal livello fisico a quello logico
  - Dal nome e cognome all'identificativo
- Modulo efferente: viceversa
  - Dagli estremi di un volo al biglietto
- Ramo afferente/efferente
  - Sequenza di moduli afferenti/efferenti
- Trasformazione centrale
  - Il punto di trasformazione al massimo livello logico
  - Alimentata in ingresso da rami afferenti e preliminare a rami efferenti in uscita
- l'analisi si basa sulla semantica dei processi e non solo sulla loro topologia.

44

UNIVERSITA' DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

## Example: a system for selling flight tickets

- A flat representation

45

UNIVERSITA' DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

## Fattorizzazione

- Un executive module (main) coordina:
  - un modulo per la trasformazione centrale
    - riceve dall'executive i flussi di ingresso e gli restituisce quelli di uscita
  - un modulo get per ogni ramo afferente, i.e. per ogni flusso di ingresso alla trasformazione
    - Fornisce all'executive i flussi di ingresso alla trasformazione centrale
  - un modulo put per ogni ramo efferente
    - Riceve dall'executive i flussi di ingresso alla trasformazione centrale

46

UNIVERSITÀ DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

## Espansione della trasformazione centrale

- Se ha una pspec questa definisce la funzione da implementare nel modulo
- Se ha un dfd figlio si ripete con la trasformazione centrale come executive del dfd figlio
- Topologia di trasformazione

47

UNIVERSITÀ DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

## Espansione di un modulo get

- Il modulo finale del ramo afferente opera come trasformazione centrale
- Il modulo get la alimenta con i dati ricevuti da un insieme di moduli get
- Topologia afferente

48

UNIVERSITÀ DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

## Espansione di un modulo put

- Il primo modulo del ramo afferente opera come trasformazione centrale
- Il modulo put produce i flussi per un insieme di moduli put
- Topologia efferente

49

UNIVERSITÀ DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

## Limiti

50/44

50

UNIVERSITA' DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

## Limiti

- **TBD: discuss here cases where the translation is imperfect**
  - the perfect case
  - bypass
  - feedback
  - multiple transformations
- TBD: remark that all this is about semantics, not just about graph topology

51/44

51

UNIVERSITA' DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

## Structured development: synthesis

- structured programming -> structured design -> structured analysis
- ... and back

```

.....
int x;
int y;
while( )
{
.....
}
.....
.....

```

52/44

52

UNIVERSITA' DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

## Discussione

- **Iperstrutturazione**
  - Può diventare un limite nel mantenere visione
- **Efficienza**
  - Non è il problema
  - È più facile far andare veloce un sistema che funziona piuttosto che fare funzionare un sistema veloce
  - L'ottimizzazione è un particolare tipo di manutenzione
    - Focalizzata sulle parti calde evidenziate nel profiling
- **Limite rispetto allo sviluppo ad oggetti**
  - È basato sul concetto di coesione funzionale
  - Comunque rigido rispetto alla evoluzione dei requisiti
  - Scarsamente orientato al riuso
- **Fondamentale per la analisi delle procedure e dei flussi**
  - Incorporato come parte dello sviluppo orientato agli oggetti

53/44

53

UNIVERSITA' DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

## References

- [1] T.De Marco, "Structured Analysis and System specification", Yourdon Press, Englewood Cliffs, New jersey, 1979.
- [2] E.Yourdon, L.Constantine, "Structured design: fundamentals of a discipline of Computer programming and system design," Prentice Hall, September 1986.
- [3] any concise book on the c language, e.g.: S.Berretti, L.Carnevali, E.Vicario, "Fondamenti di programmazione: linguaggio c, strutture dati e algoritmi elementari, c++" Esculapio Editore, Bologna, 2017.

54/44

54

UNIVERSITA' DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

### an example about learning, diagnosis, prediction and scheduling

- the agenda of analytics maturity models (by Gartner)

Value

Difficulty

Gartner.

- example on a specific implementation of the concept
  - using Hidden Markov Models and Discrete Time Markov Chains
  - for on-line failure prediction and just in time maintenance

55/54

55

UNIVERSITA' DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

### ground concepts: on line failure prediction

- on line failure prediction at time  $t$ 
  - given observations in a *past data window*  $D_{td}$
  - predict some event happening in a *prediction period*  $D_{tp}$
  - starting at some *lead time*  $D_{tl}$
  - not earlier than a *warning time*  $D_{tw}$
  - F.Salfner, M.Lenk, M.Malek, "A survey of online failure prediction methods," *ACM Computing Surveys*, 2010.

time

- Just in Time Maintenance
  - concept from inventory management
  - emphasized in the agenda of Industry 4.0

56/54

56

UNIVERSITA' DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

### ground concepts: DTMC

- Discrete Time Markov Chain (DTMC)
  - states, transition probabilities, initial probability
  - transient probabilities, Chapman Kolmogorov, computation
  - steady state probabilities, existence, computation

57/54

57

UNIVERSITA' DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

### ground concepts: HMM

- Hidden Markov Models
  - a hidden DTMC with an Observation Model
  - a set of symbols and an emission probability distr. per state
  - L.R.Rabiner, "a Tutorial on Hidden Markov Models and selected applications in speech recognition," *Proc.IEEE*, 1989.
- 3 problems
  - #1 evaluation:  $P(\text{ObsSeq} | \text{Model})$   
(transient analysis on the DTMC, forward backward)
  - #2 inference:  $\text{Likelihood} = P(\text{StateSeq} | \text{Model}, \text{ObsSeq})$   
(Viterbi, forward backward)
  - #3 learning:  $\text{Model} = \arg \text{Max}\{P(\text{ObsSeq} | \text{Model})\}$   
(Expectation Maximization)

58/54

58



UNIVERSITA DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

## learning, diagnosis, prediction and scheduling

- a 4 stages process for Maintenance Just in Time
  - descriptive: learn a model from a dataset, offline
  - diagnostic: by inference on observations @runtime
  - predictive: transient analysis, partially prepared offline
  - prescriptive: schedule timed actions or defer

Gartner. 59/54

59

UNIVERSITA DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

## learning, diagnosis, prediction and scheduling

- a DFD description of the overall transformation process

60/54

60

UNIVERSITA DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

## an example about learning, diagnosis, prediction and scheduling

- Structure Chart
  - by Transform Analysis

61/54

61

UNIVERSITA DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

## an example about learning, diagnosis, prediction and scheduling

- problem and solution can be extended in many ways
  - generating and splitting data across testing and training, leave one out cross validation
  - classify data and learn different models
  - let the model evolve @runtime
  - repeat for different values of hyper-parameters
  - restart form different initial model estimations
  - repeat on different datasets
  - ...
- a good structure helps maintenance

62/54

62



UNIVERSITA' DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

## can Data Flow Diags fit OO Development?

- J.Rumbaugh, M.Blaho, W.Premierani, F.Eddy, W.E.Lorensen, *"Object-oriented modeling and design,"* 1991.
  - decomposes development across 3 models
- Object Model:
  - a static model, Class Diags and Object Diags
- Dynamic Model
  - about state behaviour
  - Sequence and Activity diagrams, State Charts
- Functional Model:
  - about information processing, transformations, algorithms
  - Data Flow Diagrams, Use Case Diagrams
  - drives identification of methods to be allocated to classes

63/54

63

UNIVERSITA' DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

## can Data Flow Diags fit OO Development?

- ... various other attempts, e.g.:
  - M.Dahan, P.Shoval, P., A.Sturm, "Comparing the impact of the OO-DFD and the Use Case methods for modeling functional requirements on comprehension and quality of models: a controlled experiment," *Requirements Engineering*, 2014.
  - data stores in the DFD become classes and objects ...
- no perfect solution,
- ... yet, common sense can drive integration
  - identify classes as common in OO Analysis and Design
  - implement DFD transformations as methods and allocate them
  - data stores are classes, but this is not the core concept

64/54

64

UNIVERSITA' DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

65/54

65


UNIVERSITA' DEGLI STUDI FIRENZE SW Engineering for Embedded Systems 2020 Enrico Vicario

## Esempio: il problema dell'orario

- Produzione dell'orario
  - assegnamento dei moduli ad <aule x ore>
  - analisi della qualità, identificazione di problemi
  - pubblicazione e comunicazione alle portinerie
  - interazione con singoli docenti
- Definizione dei corsi
  - produzione del manifesto
    - elenco dei corsi di laurea (cdl)
    - per ciascun cdl elenco degli insegnamenti
    - regolamento (periodo didattico, precedenza, scelte)
  - definizione dei corsi attivati
  - accorpamento di insegnamenti condivisi su più cdl
  - assegnamento coperture
    - bando
    - ricezione domande
    - valutazione consiglio di facoltà
    - contratti
  - identificazione dei "moduli attuali"

66/44

66

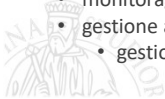


UNIVERSITA'  
DEGLI STUDI  
FIRENZE


SW Engineering for  
Embedded Systems 2020  
Enrico Vicario

Esempio: il problema dell'orario

- **Relazione con i docenti**
  - acquisizione dati (telefono, mail, URL)
  - ricezione requisiti vincolanti
  - ricezione requisiti di attrezzatura
  - comunicazione dell'orario
  - aggiornamento sulle modifiche
  
- **Gestione aule**
  - previsione del carico
    - a partire dal manifesto, dall'orario dell'anno precedente, da osservazioni ricevute dal cdl
  - richiesta aule al polo amministrativo
    - ricezione aule dal polo
    - comunicazione ex-post delle aule impegnate
  - monitoraggio sull'uso e produzione di statistiche di occupazione
  - gestione aule per impegni sporadici (esami, seminari,...)
    - gestione attrezzatura tecnica


67/44

67




UNIVERSITA'  
DEGLI STUDI  
FIRENZE

SW Engineering for  
Embedded Systems 2020  
Enrico Vicario

Esempio: il problema dell'orario

- **Per produrre l'orario occorrono:**
  - elenco dei CDL,
  - Insegnamenti in ciascun CDL, ciascuno con numero di ore settimanali (determinato in base a CFU e periodi didattici)
  - Aule assegnate alla scuola (sulla base delle richieste, dell'allocazione storica, dell'astatistica di uso nell'anno passato) e in grado di ospitare ciascun insegnamento (stima basata su numero di immatricolati o su studenti dell'anno passato)
  - Vincoli per condivisione di insegnamenti fra CDL (desunto dal manifesto), per richieste personali di docenti, per docenti impegnati su più insegnamenti (desunto dal piano di copertura del manifesto, a sua volta determinato da incarichi interni e contratti).
  
- **La produzione dell'orario**
  - è un processo iterativo nel quale ripetutamente si allocano ore di insegnamenti ad aule, producendo eventuali conflitti (sulla concorrenza di insegnamenti per CDL, o di docenti, o di uso dell'aula, o di allocazione ad una aula non adeguata ...)
  - Tipicamente si basa sulla soluzione dell'anno precedente
  - Qualcuno in giro per l'Italia ha scritto codice che lo produce automaticamente (?). Ha delle metriche di qualità.
  
- **quando l'orario è fatto**
  - deve essere pubblicato sul web, notificato ai docenti, integrato in MRBS con le occupazioni sporadiche, e nel caso le occupazioni sporadiche impattate devono essere notificate.



68