ICRTCCM17 Invited Talk

Programming vs. Component Development

Evolution of Enterprise Computing and its Challenges to University Education

Dr. Vassil T. Vassilev

Senior Manager of Cyber Security Research Centre

London Metropolitan University – London, UK
Content

1 Past and present – my own experience in programming
2 My bibles – the books I have used to teach programming
3 Evolution of Programming – from Algorithms to Frameworks
4 Challenges to university education posed by programming
1 Past and present – my own programming experience

- Initially fascinated by the idea of *modelling physical processes* – studied Control Theory for BSc and programmed numerical methods.
- Hooked by the idea of *modelling abstract processes* – switched to Computer Science at my MSc and programmed language processors.
- Always seriously attracted by the idea of *modelling intelligent behaviour* – did AI in my PhD years and programmed knowledge-based systems.
More recent focus

- Dived deeper into *data representation* – did databases in industry, designing databases and developing information systems using Oracle
- Dived deeper into *data modelling* – did semantic modelling at university, developing ontologies and semantic applications
- Recently dived deeper into *data processing* – did BigData and data analytics industrial training for data analytics
Programming and Languages

- Numerical calculations: **Fortran** (1979-1983)
- Language processors: **Pascal** (1984-1985)
- Symbolic computation: **Lisp** (1986-1993)
- Database applications: **PL/SQL** (1994-1999)
- Network programming: **Java** (2002-)
- Real-time data processing: **Python** (2014-)
Academic roles

◆ **Senior Lecturer in Computer Science**
  (teaching Introduction to Computer Science, Formal Specification, Distributed and Internet Systems, Enterprise Components and Systems and BigData Management)

◆ **Course Leader at the School of Computing**
  (BSc Computer Science, MSc Computer Science, MSc Professional Engineering and Technology, MSc Computing and Information Systems)

◆ **Senior Manager of the Cyber Security Research Centre of Londonmet**
  (projects for Cross-channel fraud detection in banking, Online customer analysis using Web microservices and Individual and Group Dynamic Behaviour Analysis in CCTV)
2 My bibles — the books I have used to teach programming

- Programming as implementing algorithms for processing data (Pascal)

• Programming as implementing models for solving problems (Scheme Lisp)

• Programming as constructing objects for exchanging information (Java)

• Programming as a quick fix (Python)

3 Evolution of Programming – from Algorithms to Frameworks

Algorithm Specification and Coding
Program Design and Implementation
Software Components and Development
Software Systems and Integration
Software Frameworks and Configuration
Example 1: Algorithms
The insertion sort algorithm

define Sort(List):
    initialize N = 2
    while (N <= length of List) repeat
        Pivot = Nth entry in List
        Remove Nth entry leaving a hole in List
        while (there is an Entry above the hole and Entry > Pivot) repeat
            Move Entry down into the hole
        stop repeating
        Move Pivot into the hole
        increment N
    stop repeating

finish
Example 2: Programs
Hierarchical Streamlined Text Processing

def main():
    dataOut(
        dataProcess(
            dataIn()))
Example 3: Software Components
Event logger with observer pattern

```
«abstract»
Subject

setOfObservers[1..*]
attach(Observer)
detach(Observer)
notify()

observed by

«abstract»
Observer

update()

ConcreteSubject

subjectState
getState()
setState()

notify() invokes update() on all observer references when the state changes

observes

ConcreteObserver

observerState
notify()

update() invokes getState() on the subject to keep observerState consistent with the subject

attach() and detach() add and remove observers to/from the list

getState() returns the subjectState
```
Example 4: Enterprise Systems
Java EE application with remote desktop client
Example 5: **Software Frameworks**

Oozie workflow for Pig processing on Hadoop

```xml
<workflow-app xmlns="uri:oozie:workflow" name="whitehouse">
  <start to="transform_whitehouse_visitors"/>
  <action name="transform_whitehouse_visitors">
    <pig>
      <job-tracker>${resourceManager}</job-tracker>
      <name-node>${nameNode}</name-node>
      <prepare>
        <delete path="wh_visits"/>
      </prepare>
      <script>whitehouse.pig</script>
    </pig>
    <ok to="end"/>
    <error to="fail"/>
  </action>
  <kill name="fail">
    <message>Job failed, error message
      [{${wf:errorMessage(wf:lastErrorNode())}}]
    </message>
  </kill>
  <end name="end"/>
</workflow-app>
```
4 Challenges to university education posed by programming

Challenges to the students: (Examples: model-driven vs. ad-hoc, command line vs. visual drag-end-drop, procedural vs. object-oriented)

Challenges to the lecturers: technological advances require insight (Examples: Big Data vs. Small Data, Database vs. Data Lake, vs. Data Fog)

Challenges to the technical teams: new versions require annual software installations, limited resources force centralized maintenance, software interactions requires custom-tailored configurations

Challenges to the management body for managing the changes
Conclusion: Trends and Recommendations

**Computing Infrastructure:** from native client installation to running within a sandbox on the client to cloud-based

**Programming Languages:** Introductory (Python), Further (Java/.Net), Specialized (JavaScript, C++, .Net, Java EE)

**Programming Environments:** BlueJ-NetBeans-Eclipse (Java), Visual Studio (.Net), Idle (Python)

**Software Libraries:** Using programming APIs and software Libraries vs. ad hoc programming

**Software Methodologies:** more and more agile development, less and less classical approaches and code generation