BilSen
Bilkent Software Engineering
Research and Education

Fourth Turkish Aspect-Oriented Software Development Workshop

TAOSD

Bilkent University, Ankara, Turkey
29 December 2009

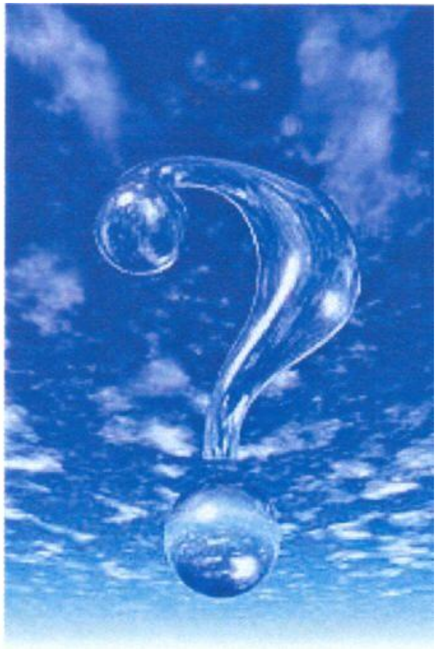
Bedir Tekinerdoğan
Bilkent University
Department of Computer Engineering
bedir@cs.bilkent.edu.tr

Bilkent University

Short Introduction to AOSD

Short Survey…

- Who has ever heard of Aspect-Oriented Software Development ?
- Who has ever used it ?
- Who is planning to use it?



Separation of Concerns

Concern

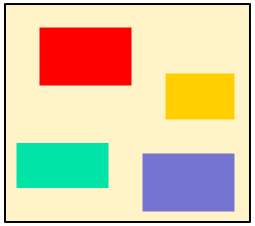
- properties or area of interest

Separation of Concerns Principle

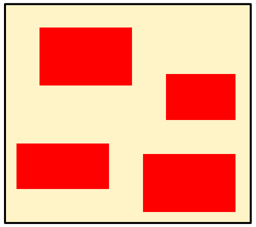
- Identification of the various distinct concerns
- Assignment of single concerns to single modules
- … to support quality factors

Cohesion and Coupling

- **Cohesion**
  - parts of a module are grouped because they all contribute to a single concern of the module

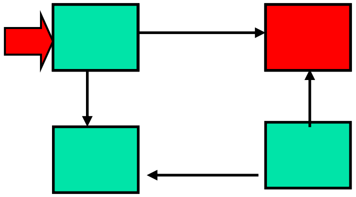


Low Cohesive Module

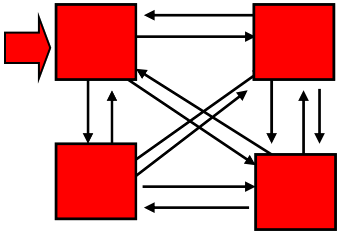


High Cohesive Module

- **Coupling**
  - the degree to which each program module relies on each one of the other modules



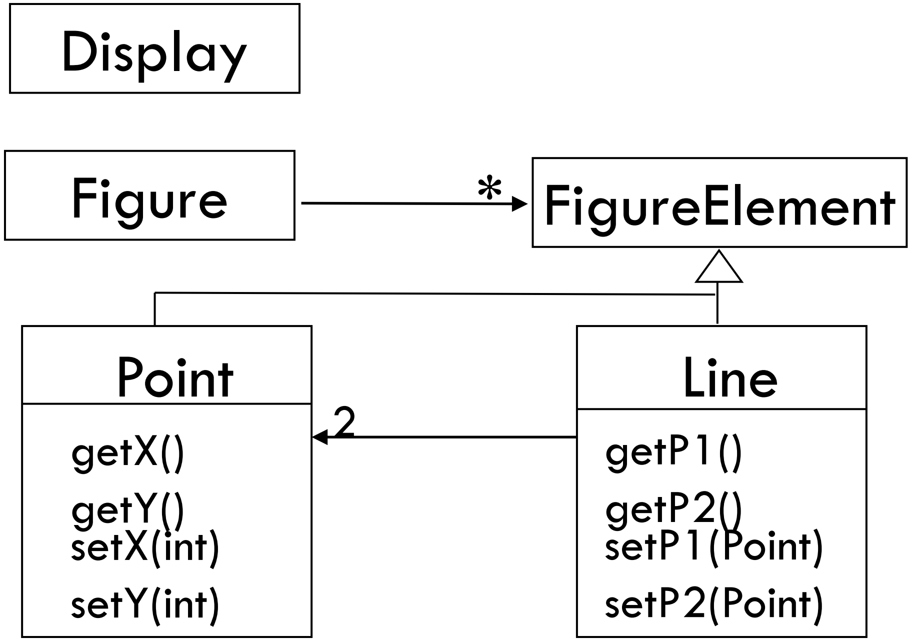
Loosely Coupling



High Coupling

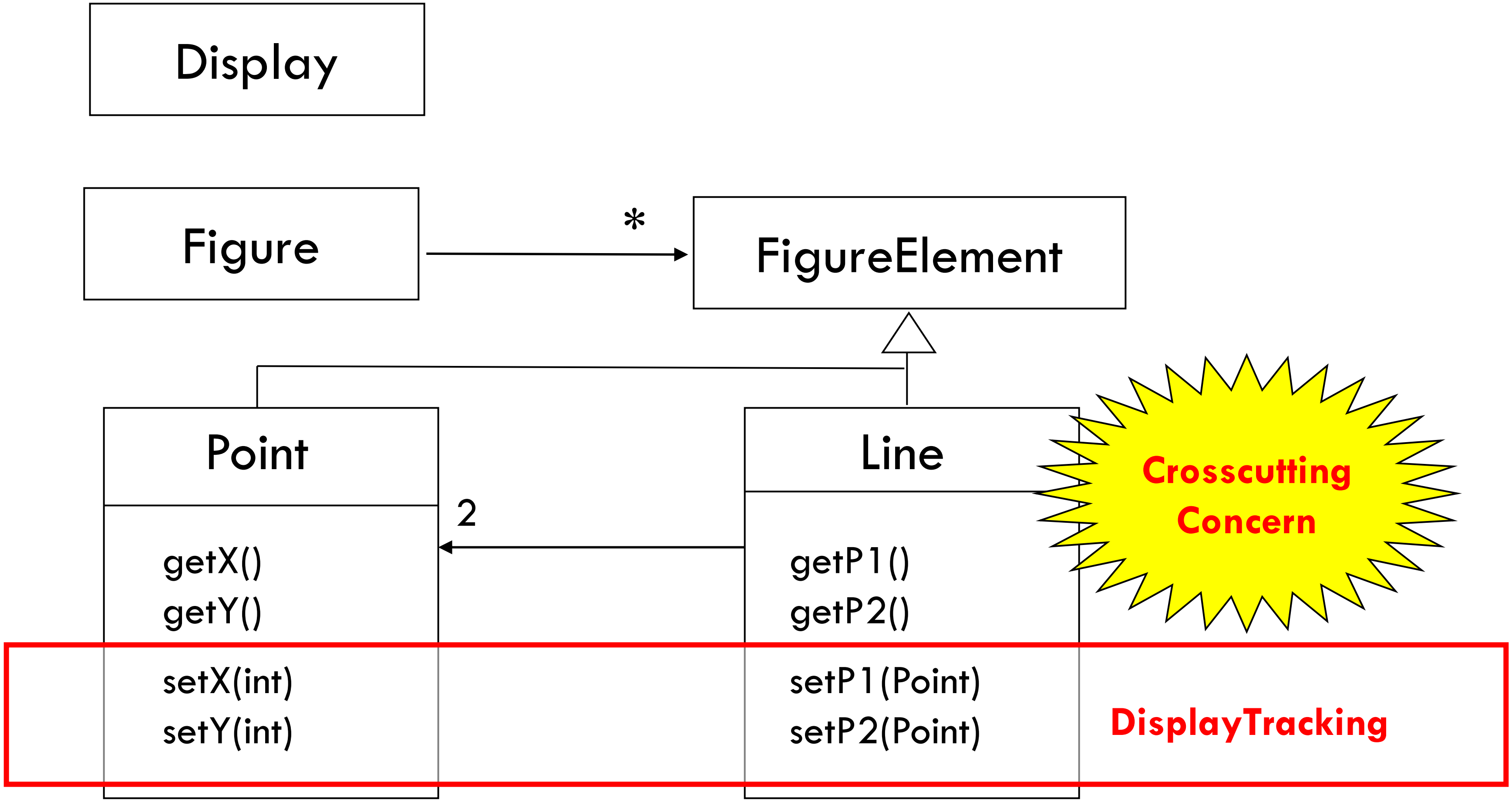
Figure Editor Example

A *figure* consists of several *figure elements*. A figure element is either a *point* or a *line*. Figures are drawn on *Display*. A point includes X and Y coordinates. A line is defined as two points.

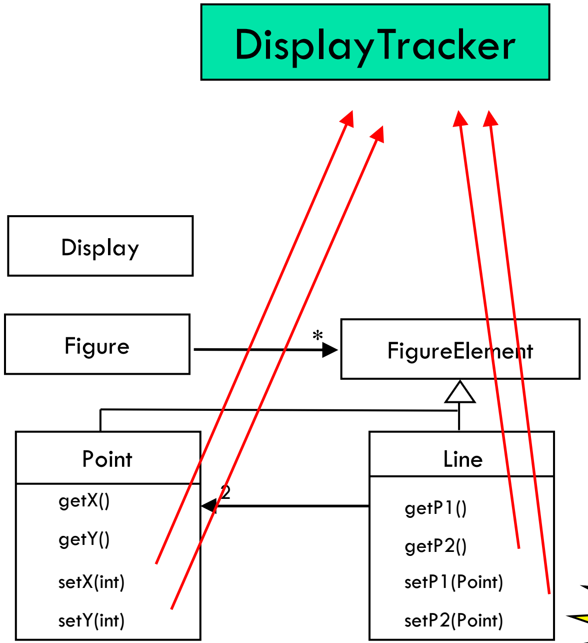


Update Display if elements move → Add Concern tracking

Crosscutting Concern - example



Example - Tracing



```
class DisplayTracker {

  static void updatePoint(Point p)
  {
                this.display(p);
                ....
  }
  static void updateLine(Line l)
  {
                this.display(l);
                ....
```

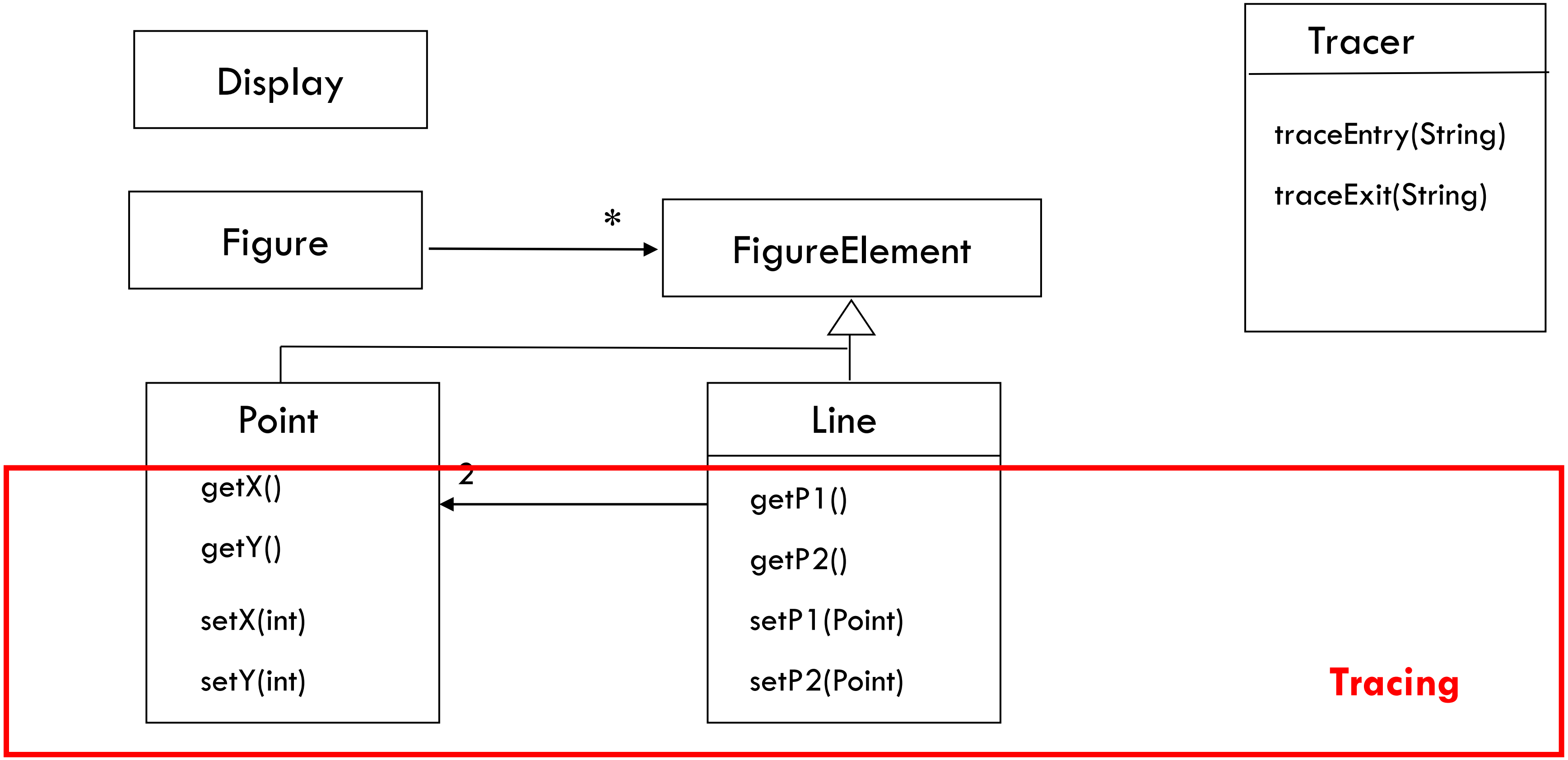
Scattered Concern

```
class Point {
  void set(int x) {
  DisplayTracker.updatePoint(this);
     _x = x;

}
}
```

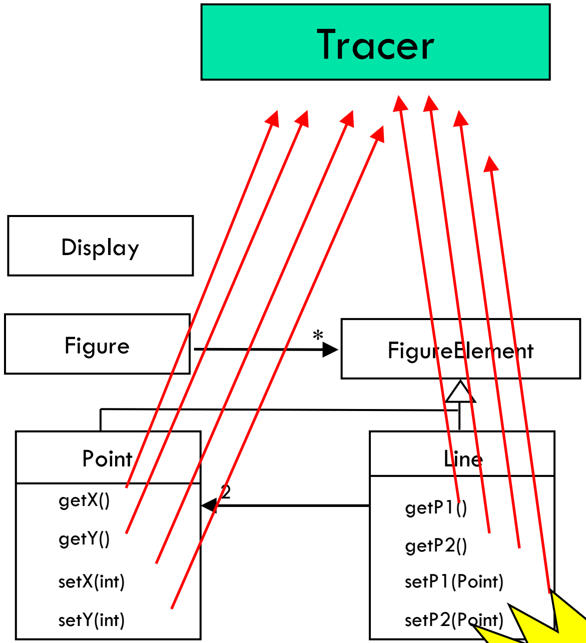
```
class Line {
  void setP1(Point p1 {
   DisplayTracker.updateLine(this);
    _p1 = p1;

}
}
```

Tangling Code

Example - Tracing



Example: Tracing



```
class Tracer {

  static void traceEntry(String str)
  {
                  System.out.println(str);
  }
  static void traceExit(String str)
  {
                  System.out.println(str);
  }
}
```

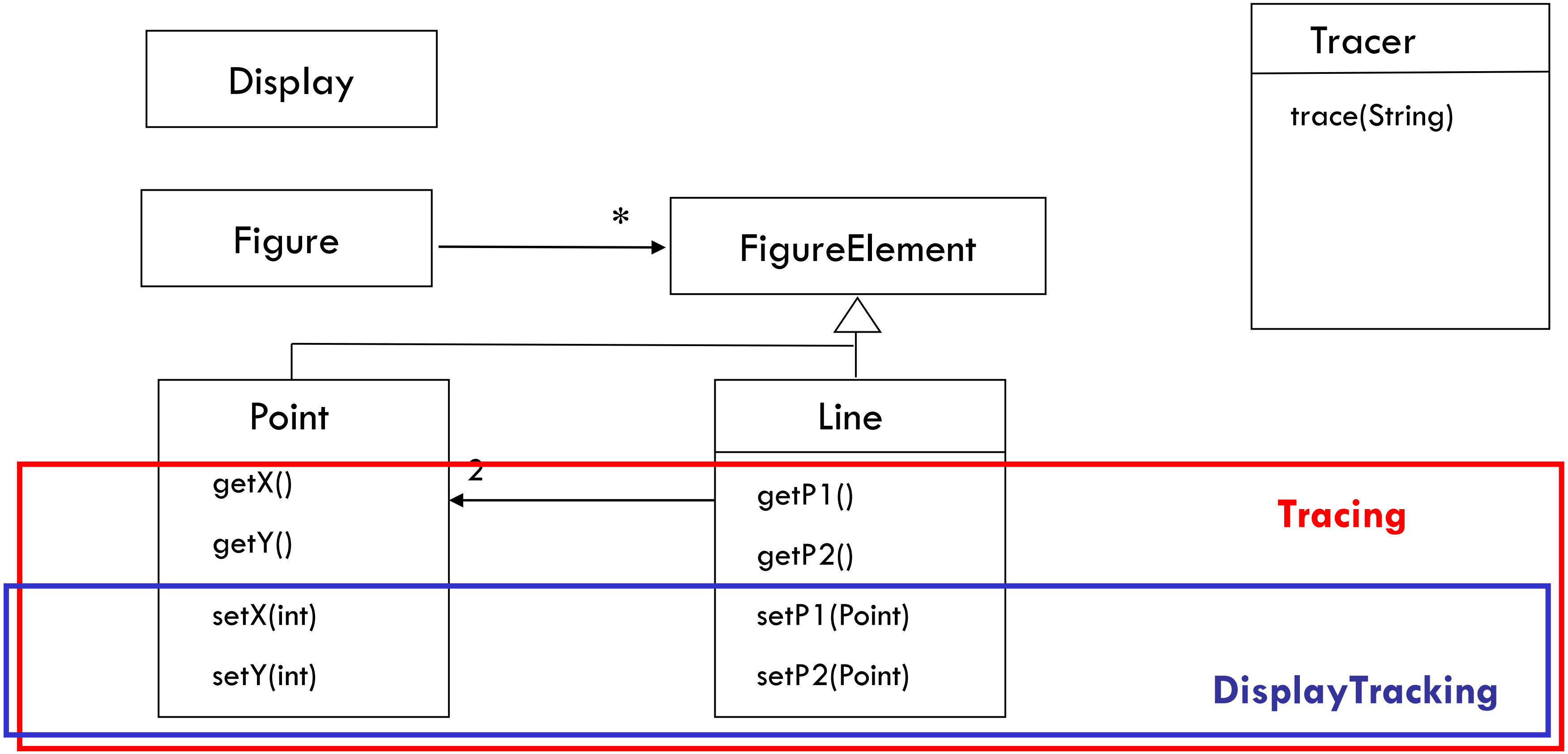
Scattered Concern

Tangling Code

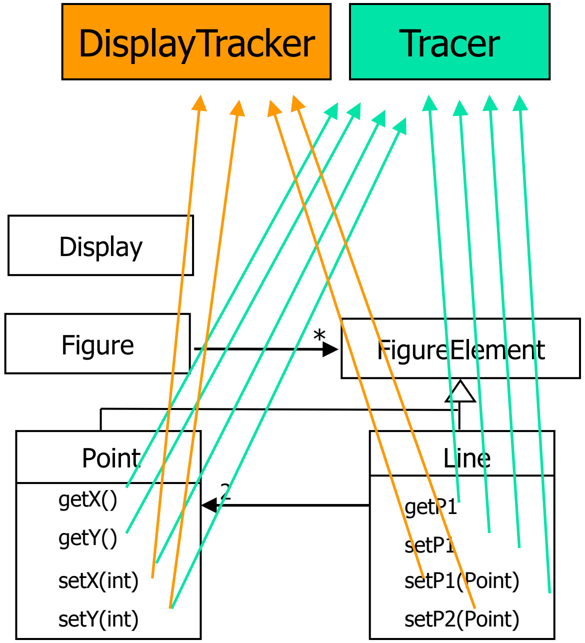
```
class Point {
  void set(int x) {
   Tracer.traceEntry("Point.set");
    _x = x;
      Tracer.traceExit("Point.set");
}
}
```

```
class Line {
  void setP1(Point p1) {
   Tracer.traceEntry("Line.set");
    _p1 = p1;
      Tracer.traceExit("Line.set");
}
}
```

Example – Tracing and Display Tracking



Example: Tracing and Display Tracking

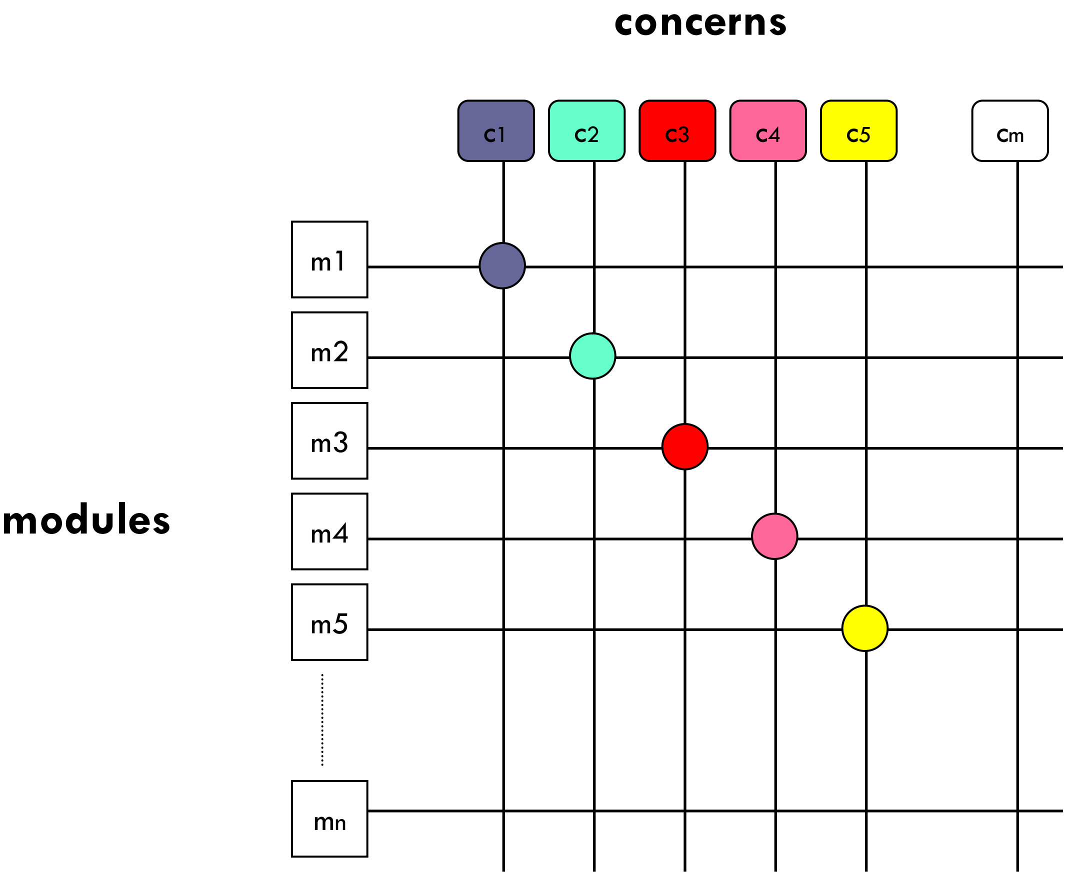


```
class Point {
  void setX(int x) {
     Tracer.traceEntry(“Entry Point.set”);
      _x = x;
     Tracer.traceExit(“Exit Point.set”);
     DisplayTracker.updatePoint(this);
}
}
```

High coupling

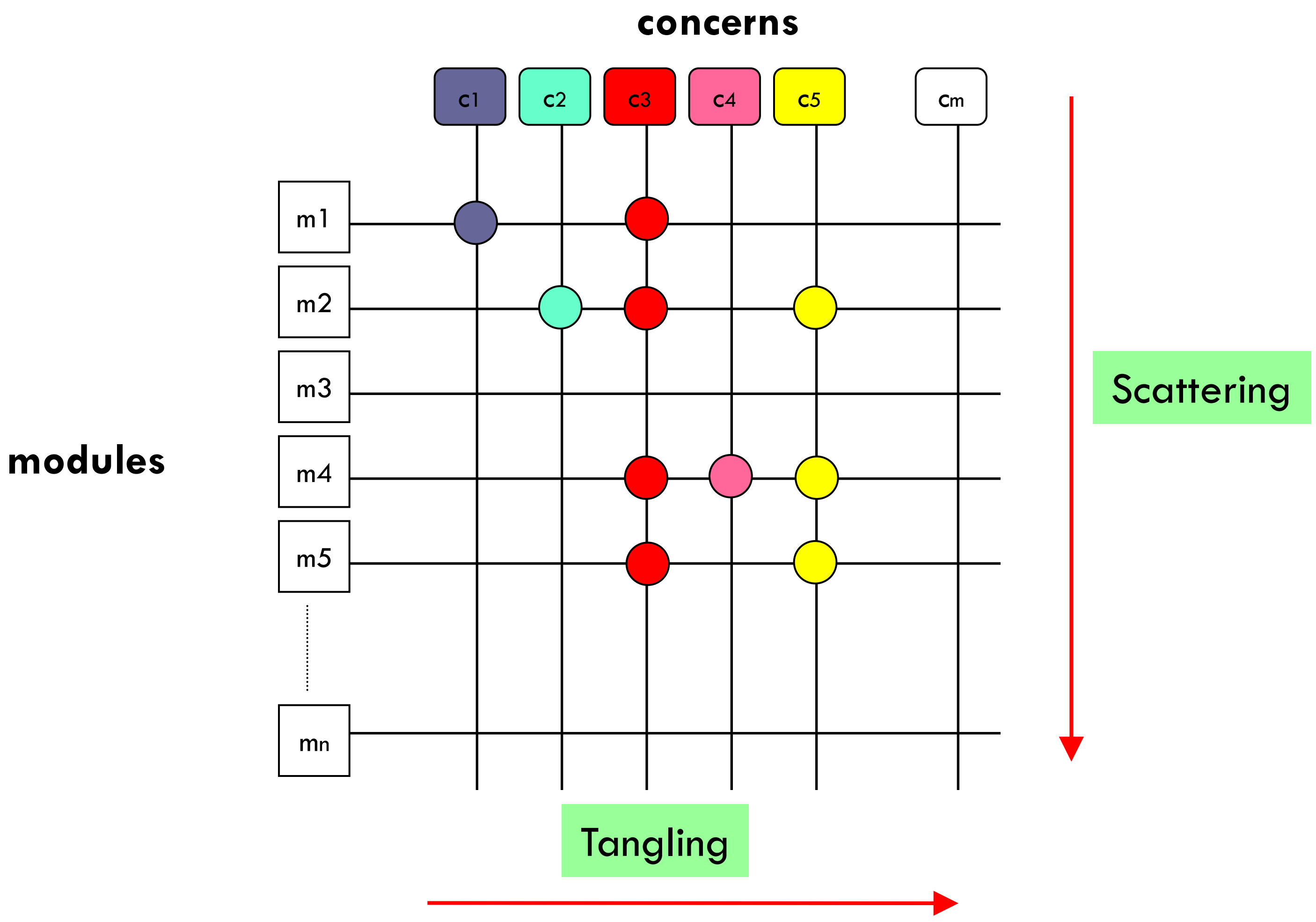
Low Cohesion

Mapping Concerns to Modules…



Separation of Concerns
Managing complexity
Increased Maintainability

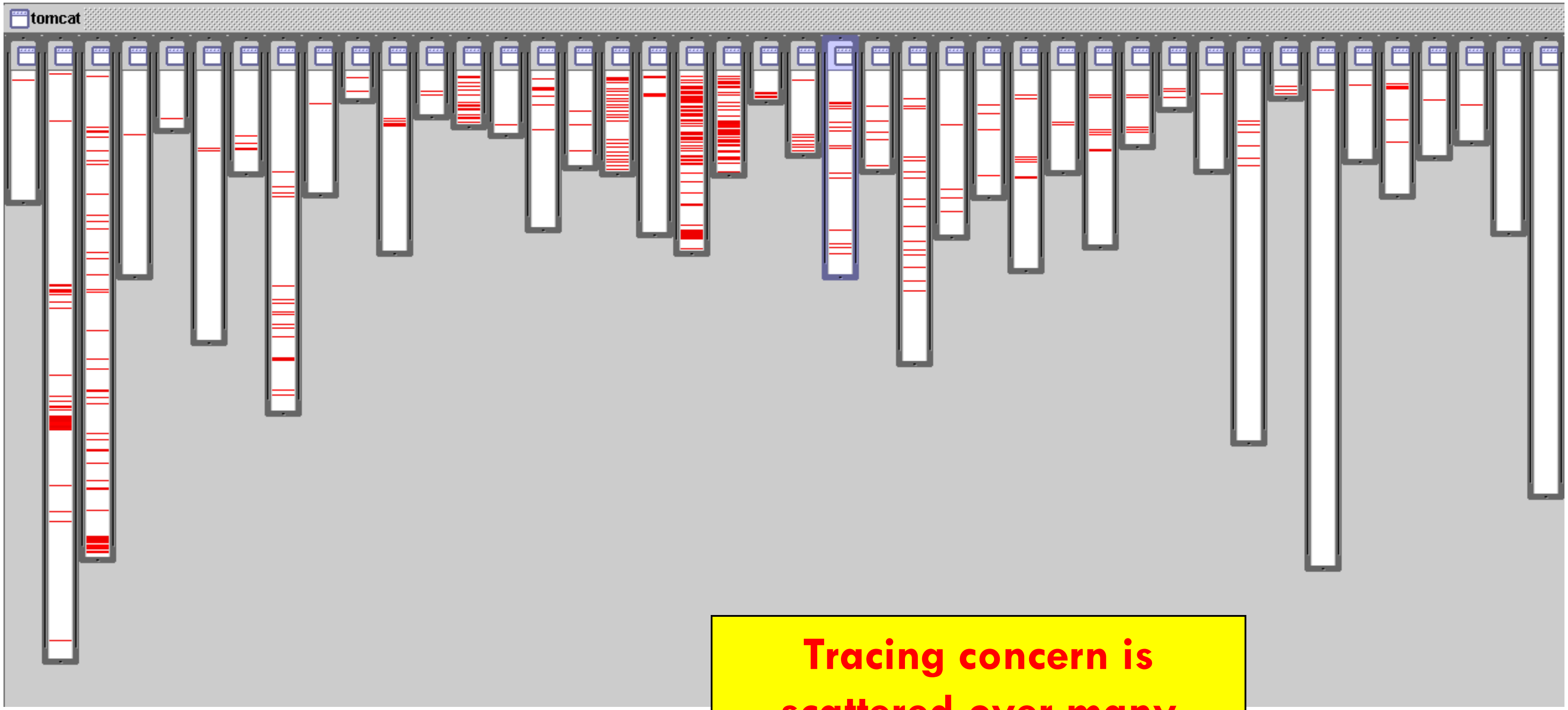
Crosscutting Concerns



Crosscutting, Scattering and Tangling

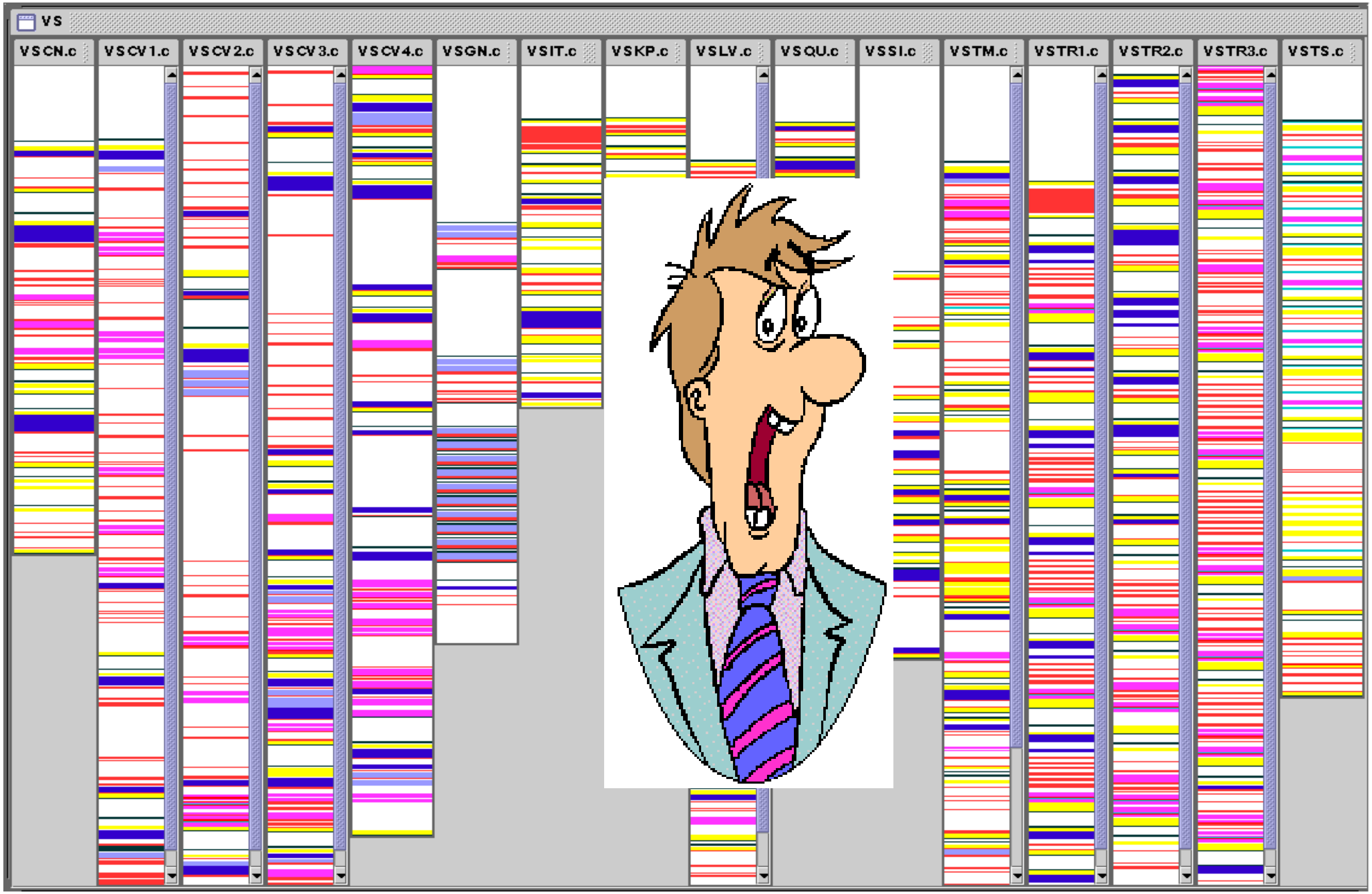
- Crosscutting
  - concern that *inherently* relates to multiple components.
  - results in scattered concern and tangled code
- Scattering
  - Single concern affects multiple modules
- Tangling
  - multiple concerns are interleaved in a single module

Aspects Visualized - Tracing



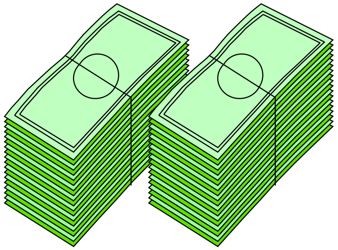
Tracing concern is scattered over many modules.
In every module tangled code.

Many more aspects… ☹



The Cost of Crosscutting Concerns

- Reduced understandability
  - Redundant code in many places
  - non-explicit structure
- Decreased adaptability
  - have to find all the code involved
  - and be sure to change it consistently
  - and be sure not to break it by accident
  - New concerns cannot be easily added
- Decreased reusability
  - component code is tangled with specific tangling code
- Decreased maintainability
  - ‘ripple effect’



Example of Crosscutting Concerns

- Synchronization
- Real-time constraints
- error-checking
- object interaction constraints
- memory management
- persistency
- security
- caching
- logging
- monitoring
- testing
- domain specific optimization
- ...

What to do...?

- improve requirements analysis
- more and better design
- apply coding guidelines
- improve project management
- process improvement (CMMI)
- improve testing
- more documentation
- .....

Aspect-Oriented Software Development

- provides better separation of concerns by explicitly considering crosscutting concerns (as well)
- Does this by providing explicit abstractions for representing crosscutting concerns, i.e. aspects
- and composing these into programs, i.e. aspect weaving or aspect composing.
- As such AOSD improves modularity
- and supports quality factors such as
  - maintainability
  - adaptability
  - reusability
  - understandability
  - ...

Example - AspectJ

```
class Line {
  private Point _p1, _p2;

  Point getP1() { return _p1; }
  Point getP2() { return _p2; }

  void setP1(Point p1) {
    _p1 = p1;
  }
  void setP2(Point p2) {
    _p2 = p2;
  }
}

class Point {
   private int _x = 0, _y = 0;

  int getX() { return _x; }
  int getY() { return _y; }

  void setX(int x) {
    _x = x;
  }
  void setY(int y) {
    _y = y;
  }
}
```

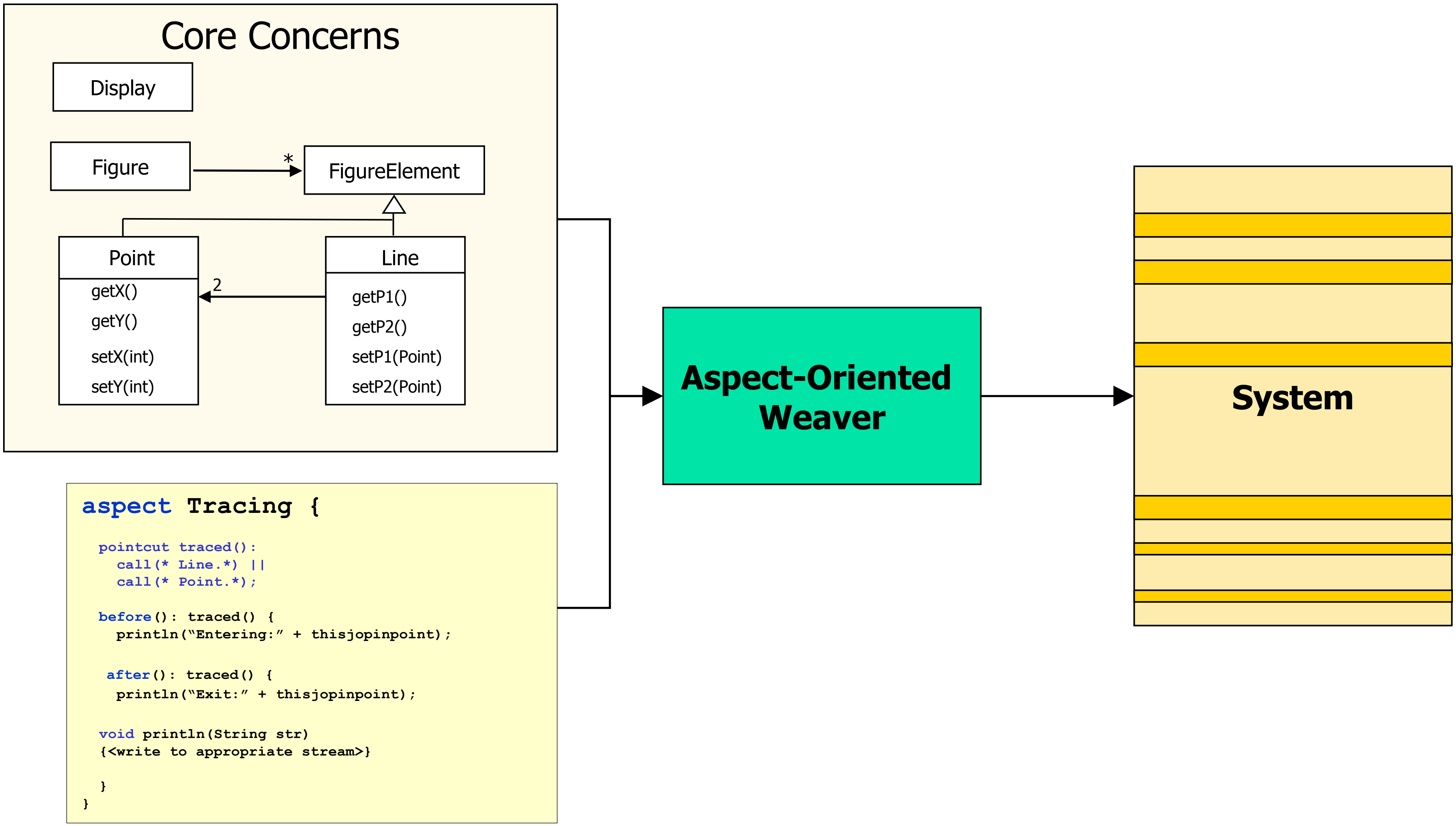
```
aspect Tracing {
   pointcut traced():
     call(* Line.*) ||
     call(* Point.*);

   before(): traced() {
     println("Entering:" +
             thisjopinpoint);

    after(): traced() {
     println("Exit:" +
             thisjopinpoint);


   void println(String str)
   {<write to appropriate stream>}

   }
}
```

aspect

pointcut

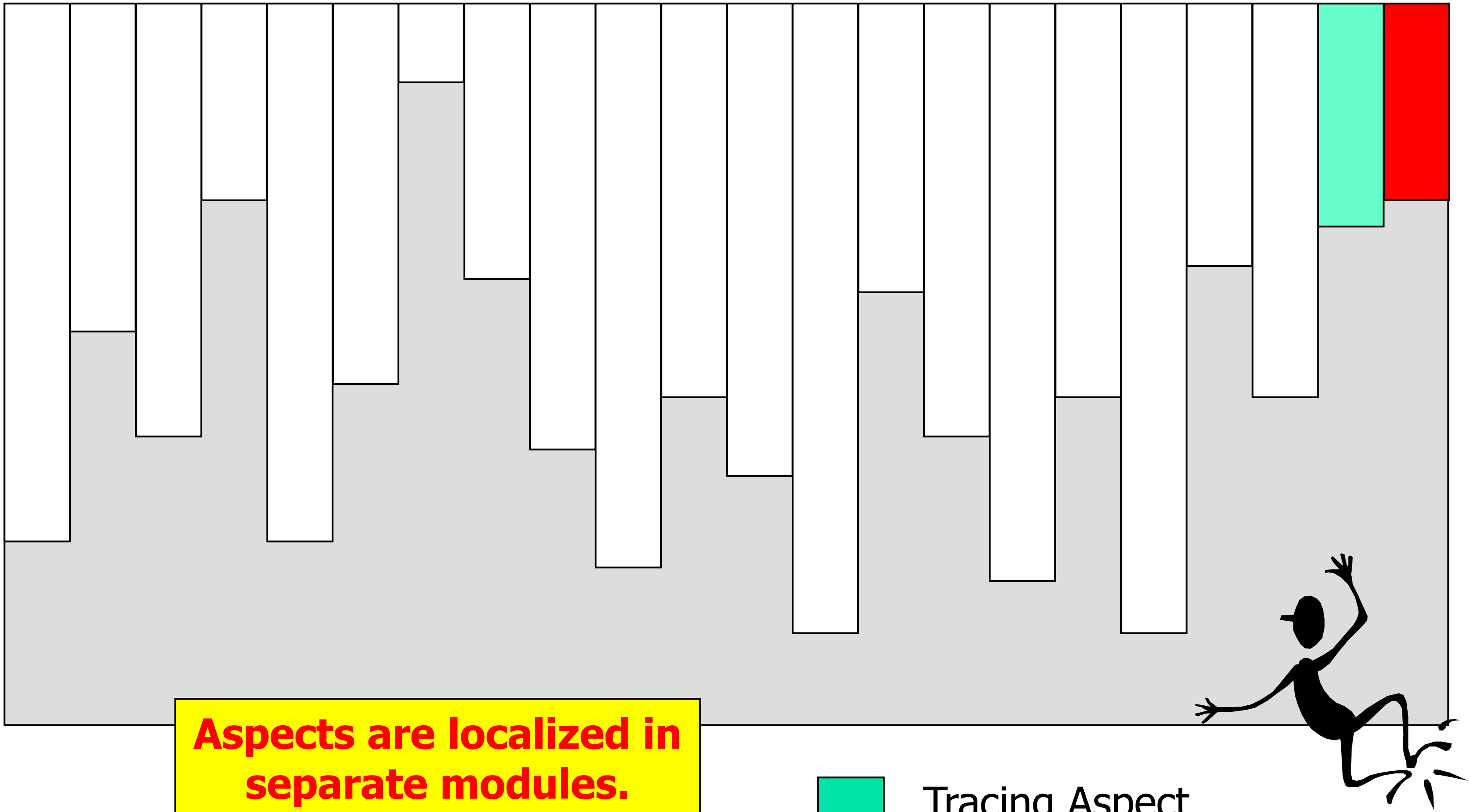
advice

AOP Solution



```
aspect Tracing {

  pointcut traced():
    call(* Line.*) ||
    call(* Point.*);

  before(): traced() {
    println("Entering:" + thisjopinpoint);

   after(): traced() {
    println("Exit:" + thisjopinpoint);

  void println(String str)
  {<write to appropriate stream>}

  }
}
```

Aspects Visualized



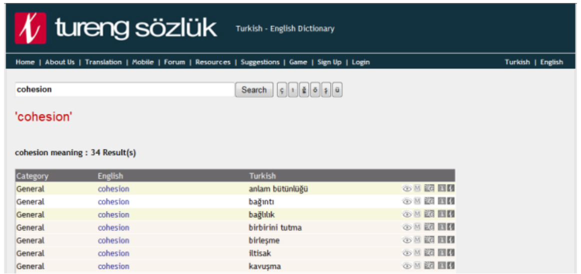
**Aspects are localized in separate modules.
No crosscutting.**

🟩 Tracing Aspect

🟥 Display Tracking Aspect

Birazda Türkçe... ☺

English	Türkçe
concern	ilgi
cohesion	uyuşma
coupling	bağlaşım
scattering	saçılma
crosscutting	enine kesen
aspect	aspekt ☺


tureng sözlük
Turkish - English Dictionary
Home | About Us | Translation | Mobile | Forum | Resources | Suggestions | Game | Sign Up | Login
Turkish | English
Search
ç ı ğ ö ş ü

cohesion

'cohesion'

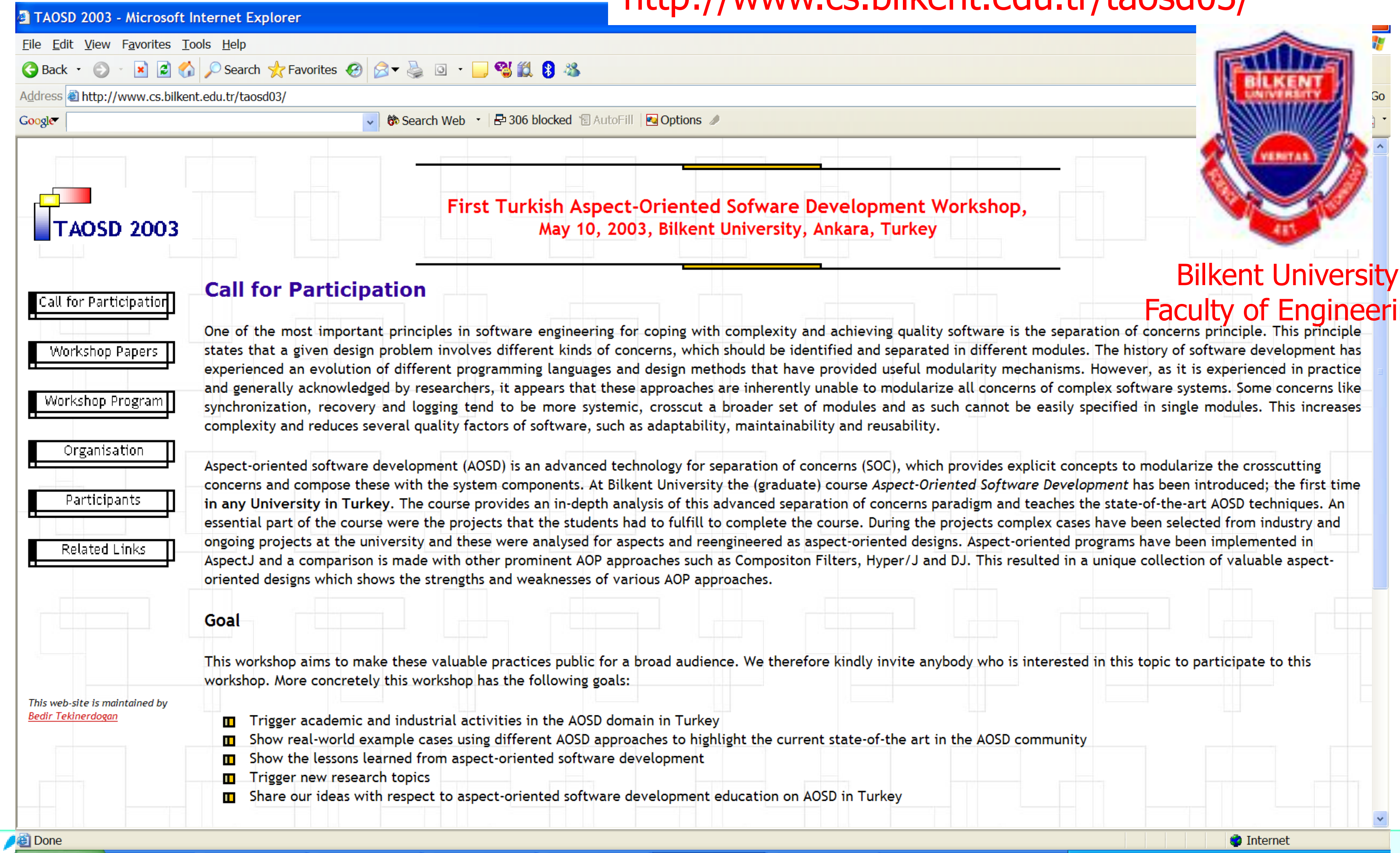
cohesion meaning : 34 Result(s)

Category	English	Turkish
General	cohesion	anlam bütünlüğü
General	cohesion	bağıntı
General	cohesion	bağlılık
General	cohesion	birbirini tutma
General	cohesion	birleşme
General	cohesion	iltisak
General	cohesion	kavuşma

Workshop Organization and Goals

First TAOSD Workshop - 2003

http://www.cs.bilkent.edu.tr/taosd03/



TAOSD 2003 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.cs.bilkent.edu.tr/taosd03/
Google Search Web 306 blocked AutoFill Options
Done
Internet

TAOSD 2003

First Turkish Aspect-Oriented Sofware Development Workshop,
May 10, 2003, Bilkent University, Ankara, Turkey

- Call for Participation
- Workshop Papers
- Workshop Program
- Organisation
- Participants
- Related Links

This web-site is maintained by Bedir Tekinerdogan

Call for Participation

One of the most important principles in software engineering for coping with complexity and achieving quality software is the separation of concerns principle. This principle states that a given design problem involves different kinds of concerns, which should be identified and separated in different modules. The history of software development has experienced an evolution of different programming languages and design methods that have provided useful modularity mechanisms. However, as it is experienced in practice and generally acknowledged by researchers, it appears that these approaches are inherently unable to modularize all concerns of complex software systems. Some concerns like synchronization, recovery and logging tend to be more systemic, crosscut a broader set of modules and as such cannot be easily specified in single modules. This increases complexity and reduces several quality factors of software, such as adaptability, maintainability and reusability.

Aspect-oriented software development (AOSD) is an advanced technology for separation of concerns (SOC), which provides explicit concepts to modularize the crosscutting concerns and compose these with the system components. At Bilkent University the (graduate) course *Aspect-Oriented Software Development* has been introduced; the first time **in any University in Turkey**. The course provides an in-depth analysis of this advanced separation of concerns paradigm and teaches the state-of-the-art AOSD techniques. An essential part of the course were the projects that the students had to fulfill to complete the course. During the projects complex cases have been selected from industry and ongoing projects at the university and these were analysed for aspects and reengineered as aspect-oriented designs. Aspect-oriented programs have been implemented in AspectJ and a comparison is made with other prominent AOP approaches such as Compositon Filters, Hyper/J and DJ. This resulted in a unique collection of valuable aspect-oriented designs which shows the strengths and weaknesses of various AOP approaches.

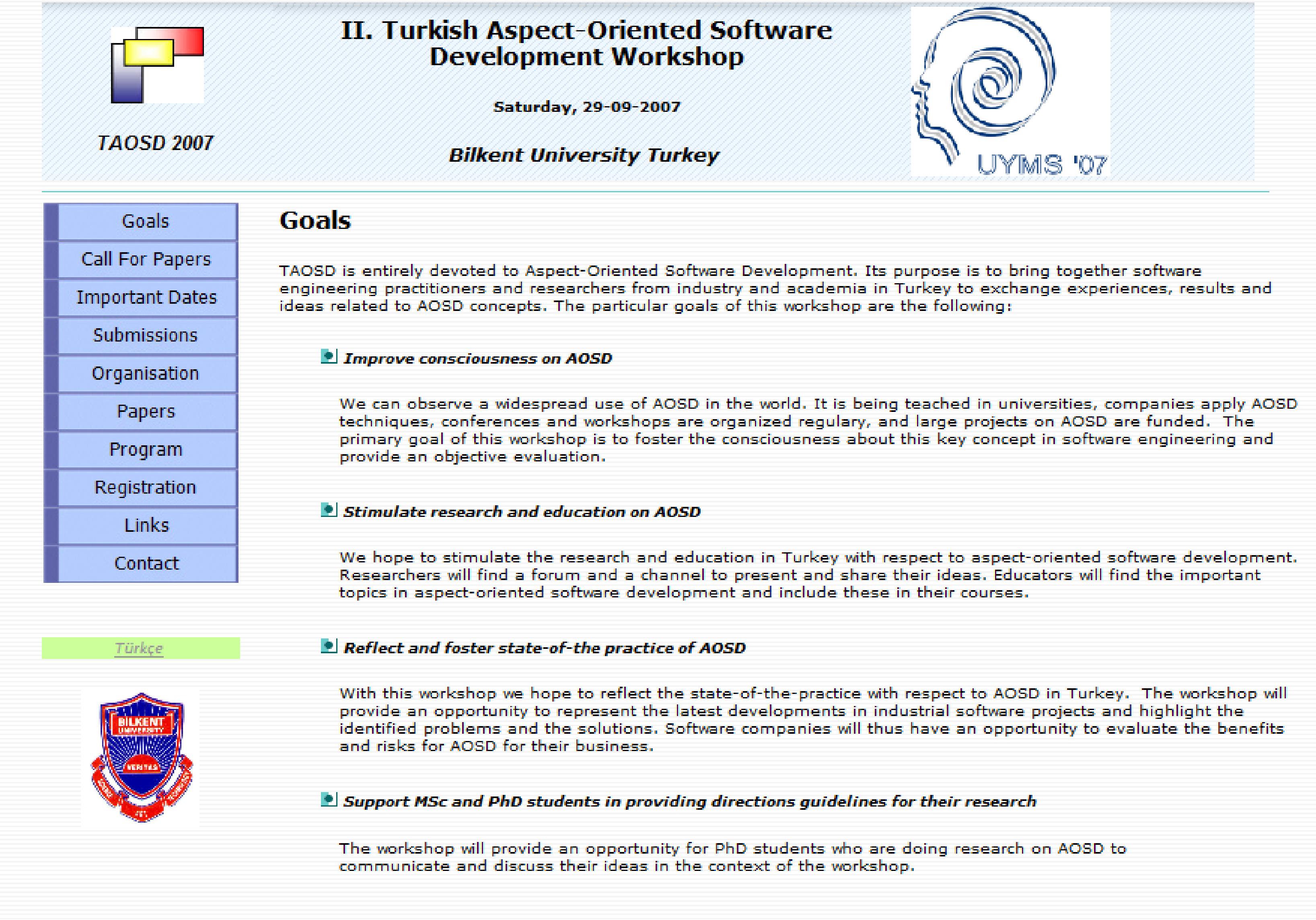
Goal

This workshop aims to make these valuable practices public for a broad audience. We therefore kindly invite anybody who is interested in this topic to participate to this workshop. More concretely this workshop has the following goals:

- Trigger academic and industrial activities in the AOSD domain in Turkey
- Show real-world example cases using different AOSD approaches to highlight the current state-of-the art in the AOSD community
- Show the lessons learned from aspect-oriented software development
- Trigger new research topics
- Share our ideas with respect to aspect-oriented software development education on AOSD in Turkey

Bilkent University
Faculty of Engineering

Second TAOSD Workshop - 2007



II. Turkish Aspect-Oriented Software Development Workshop

Saturday, 29-09-2007

Bilkent University Turkey

Goals

TAOSD is entirely devoted to Aspect-Oriented Software Development. Its purpose is to bring together software engineering practitioners and researchers from industry and academia in Turkey to exchange experiences, results and ideas related to AOSD concepts. The particular goals of this workshop are the following:

Improve consciousness on AOSD

We can observe a widespread use of AOSD in the world. It is being teached in universities, companies apply AOSD techniques, conferences and workshops are organized regulary, and large projects on AOSD are funded. The primary goal of this workshop is to foster the consciousness about this key concept in software engineering and provide an objective evaluation.

Stimulate research and education on AOSD

We hope to stimulate the research and education in Turkey with respect to aspect-oriented software development. Researchers will find a forum and a channel to present and share their ideas. Educators will find the important topics in aspect-oriented software development and include these in their courses.

Reflect and foster state-of-the practice of AOSD

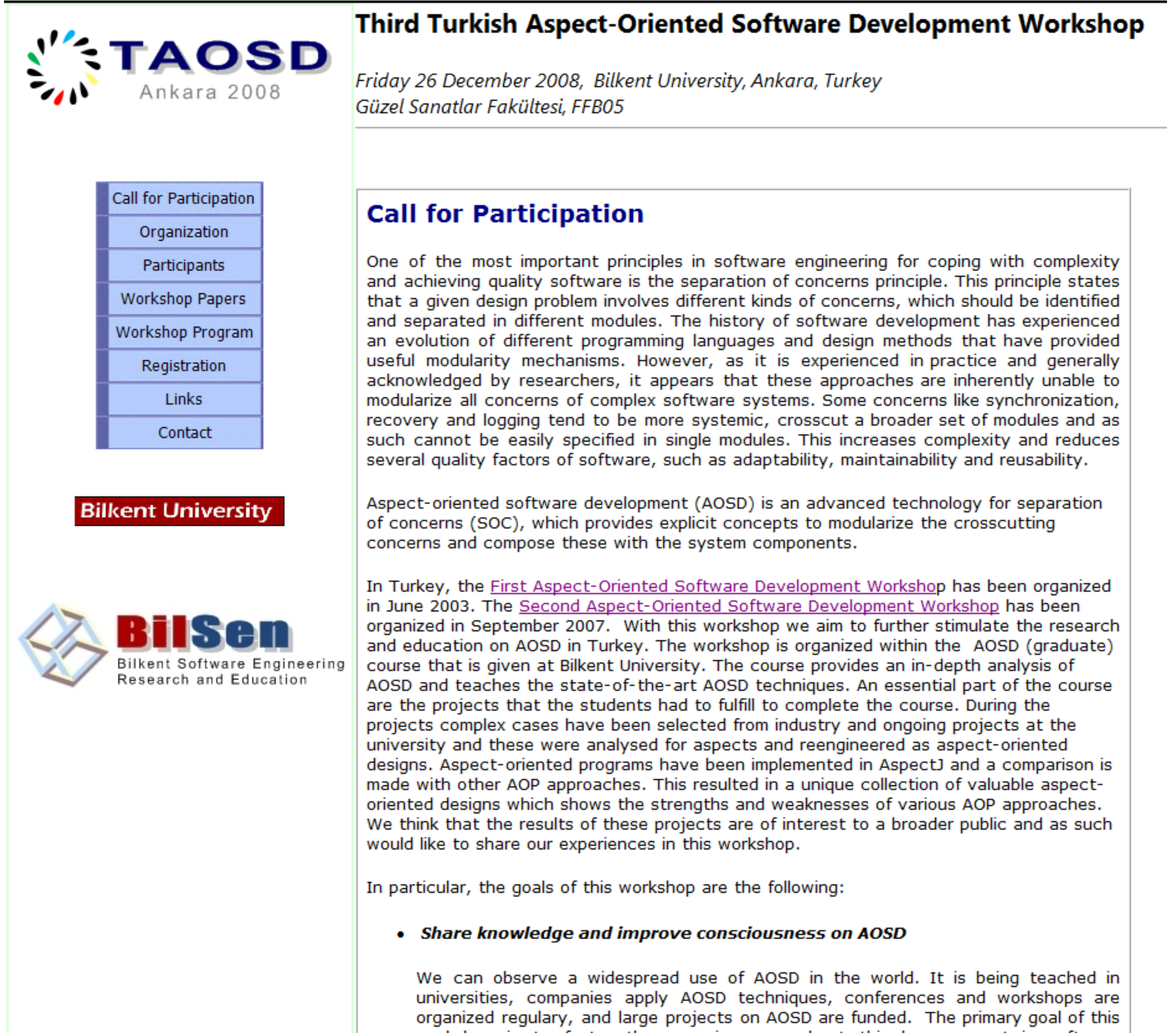
With this workshop we hope to reflect the state-of-the-practice with respect to AOSD in Turkey. The workshop will provide an opportunity to represent the latest developments in industrial software projects and highlight the identified problems and the solutions. Software companies will thus have an opportunity to evaluate the benefits and risks for AOSD for their business.

Support MSc and PhD students in providing directions guidelines for their research

The workshop will provide an opportunity for PhD students who are doing research on AOSD to communicate and discuss their ideas in the context of the workshop.

Third TAOSD Workshop - 2008

http://www.cs.bilkent.edu.tr/Bilsen/TAOSD-2008/



Third Turkish Aspect-Oriented Software Development Workshop

Friday 26 December 2008, Bilkent University, Ankara, Turkey
Güzel Sanatlar Fakültesi, FFB05

Call for Participation

One of the most important principles in software engineering for coping with complexity and achieving quality software is the separation of concerns principle. This principle states that a given design problem involves different kinds of concerns, which should be identified and separated in different modules. The history of software development has experienced an evolution of different programming languages and design methods that have provided useful modularity mechanisms. However, as it is experienced in practice and generally acknowledged by researchers, it appears that these approaches are inherently unable to modularize all concerns of complex software systems. Some concerns like synchronization, recovery and logging tend to be more systemic, crosscut a broader set of modules and as such cannot be easily specified in single modules. This increases complexity and reduces several quality factors of software, such as adaptability, maintainability and reusability.

Aspect-oriented software development (AOSD) is an advanced technology for separation of concerns (SOC), which provides explicit concepts to modularize the crosscutting concerns and compose these with the system components.

In Turkey, the First Aspect-Oriented Software Development Workshop has been organized in June 2003. The Second Aspect-Oriented Software Development Workshop has been organized in September 2007. With this workshop we aim to further stimulate the research and education on AOSD in Turkey. The workshop is organized within the AOSD (graduate) course that is given at Bilkent University. The course provides an in-depth analysis of AOSD and teaches the state-of-the-art AOSD techniques. An essential part of the course are the projects that the students had to fulfill to complete the course. During the projects complex cases have been selected from industry and ongoing projects at the university and these were analysed for aspects and reengineered as aspect-oriented designs. Aspect-oriented programs have been implemented in AspectJ and a comparison is made with other AOP approaches. This resulted in a unique collection of valuable aspect-oriented designs which shows the strengths and weaknesses of various AOP approaches. We think that the results of these projects are of interest to a broader public and as such would like to share our experiences in this workshop.

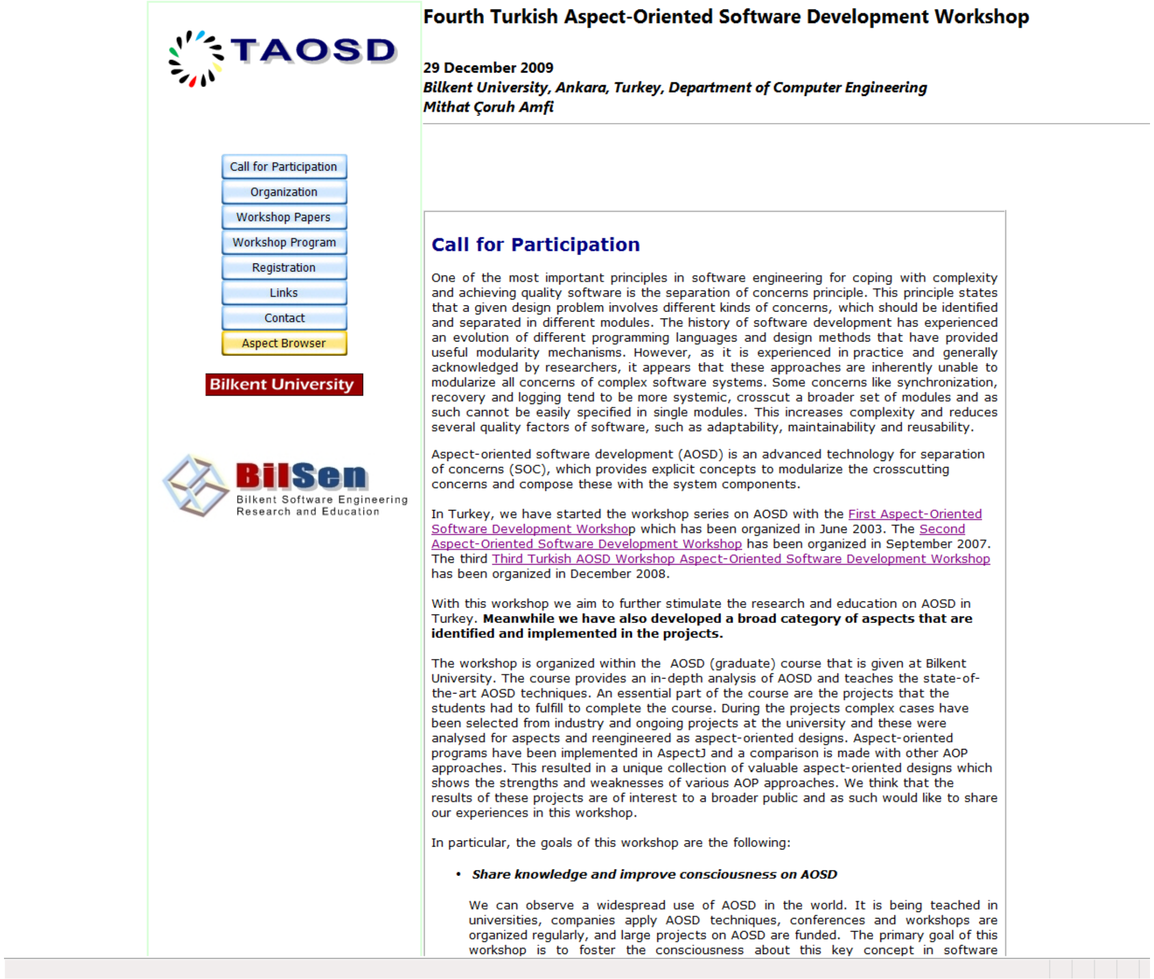
In particular, the goals of this workshop are the following:

- ***Share knowledge and improve consciousness on AOSD***

  We can observe a widespread use of AOSD in the world. It is being teached in universities, companies apply AOSD techniques, conferences and workshops are organized reguIary, and large projects on AOSD are funded. The primary goal of this

Fourth TAOSD Workshop - 2009

http://www.cs.bilkent.edu.tr/Bilsen/TAOSD-2009/



Fourth Turkish Aspect-Oriented Software Development Workshop

29 December 2009
Bilkent University, Ankara, Turkey, Department of Computer Engineering
Mithat Çoruh Amfi

Call for Participation

One of the most important principles in software engineering for coping with complexity and achieving quality software is the separation of concerns principle. This principle states that a given design problem involves different kinds of concerns, which should be identified and separated in different modules. The history of software development has experienced an evolution of different programming languages and design methods that have provided useful modularity mechanisms. However, as it is experienced in practice and generally acknowledged by researchers, it appears that these approaches are inherently unable to modularize all concerns of complex software systems. Some concerns like synchronization, recovery and logging tend to be more systemic, crosscut a broader set of modules and as such cannot be easily specified in single modules. This increases complexity and reduces several quality factors of software, such as adaptability, maintainability and reusability.

Aspect-oriented software development (AOSD) is an advanced technology for separation of concerns (SOC), which provides explicit concepts to modularize the crosscutting concerns and compose these with the system components.

In Turkey, we have started the workshop series on AOSD with the First Aspect-Oriented Software Development Workshop which has been organized in June 2003. The Second Aspect-Oriented Software Development Workshop has been organized in September 2007. The third Third Turkish AOSD Workshop Aspect-Oriented Software Development Workshop has been organized in December 2008.

With this workshop we aim to further stimulate the research and education on AOSD in Turkey. **Meanwhile we have also developed a broad category of aspects that are identified and implemented in the projects.**

The workshop is organized within the AOSD (graduate) course that is given at Bilkent University. The course provides an in-depth analysis of AOSD and teaches the state-of-the-art AOSD techniques. An essential part of the course are the projects that the students had to fulfill to complete the course. During the projects complex cases have been selected from industry and ongoing projects at the university and these were analysed for aspects and reengineered as aspect-oriented designs. Aspect-oriented programs have been implemented in AspectJ and a comparison is made with other AOP approaches. This resulted in a unique collection of valuable aspect-oriented designs which shows the strengths and weaknesses of various AOP approaches. We think that the results of these projects are of interest to a broader public and as such would like to share our experiences in this workshop.

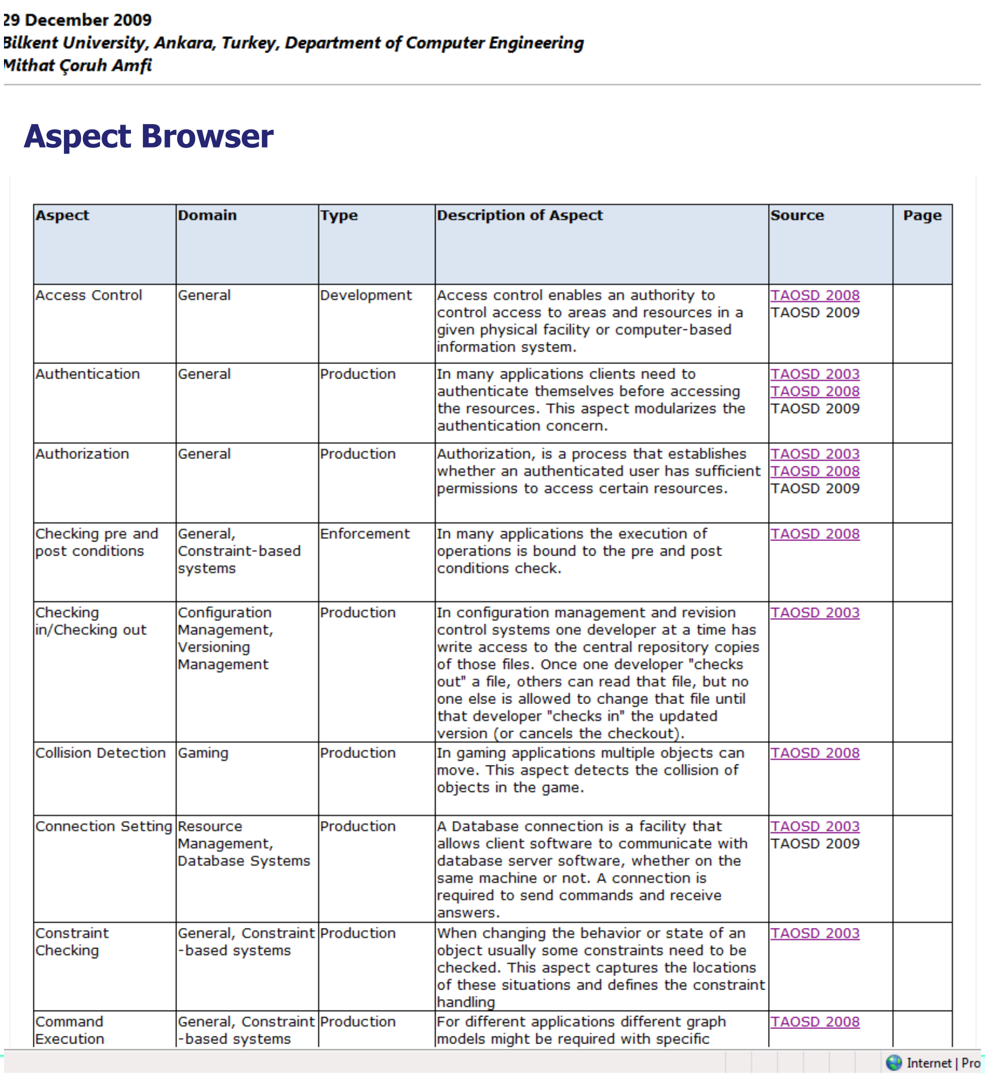
In particular, the goals of this workshop are the following:

- ***Share knowledge and improve consciousness on AOSD***

  We can observe a widespread use of AOSD in the world. It is being teached in universities, companies apply AOSD techniques, conferences and workshops are organized regularly, and large projects on AOSD are funded. The primary goal of this workshop is to foster the consciousness about this key concept in software

Aspect Browser

http://www.cs.bilkent.edu.tr/Bilsen/TAOSD-2009/



29 December 2009
Bilkent University, Ankara, Turkey, Department of Computer Engineering
Mithat Çoruh Amfi

Aspect Browser

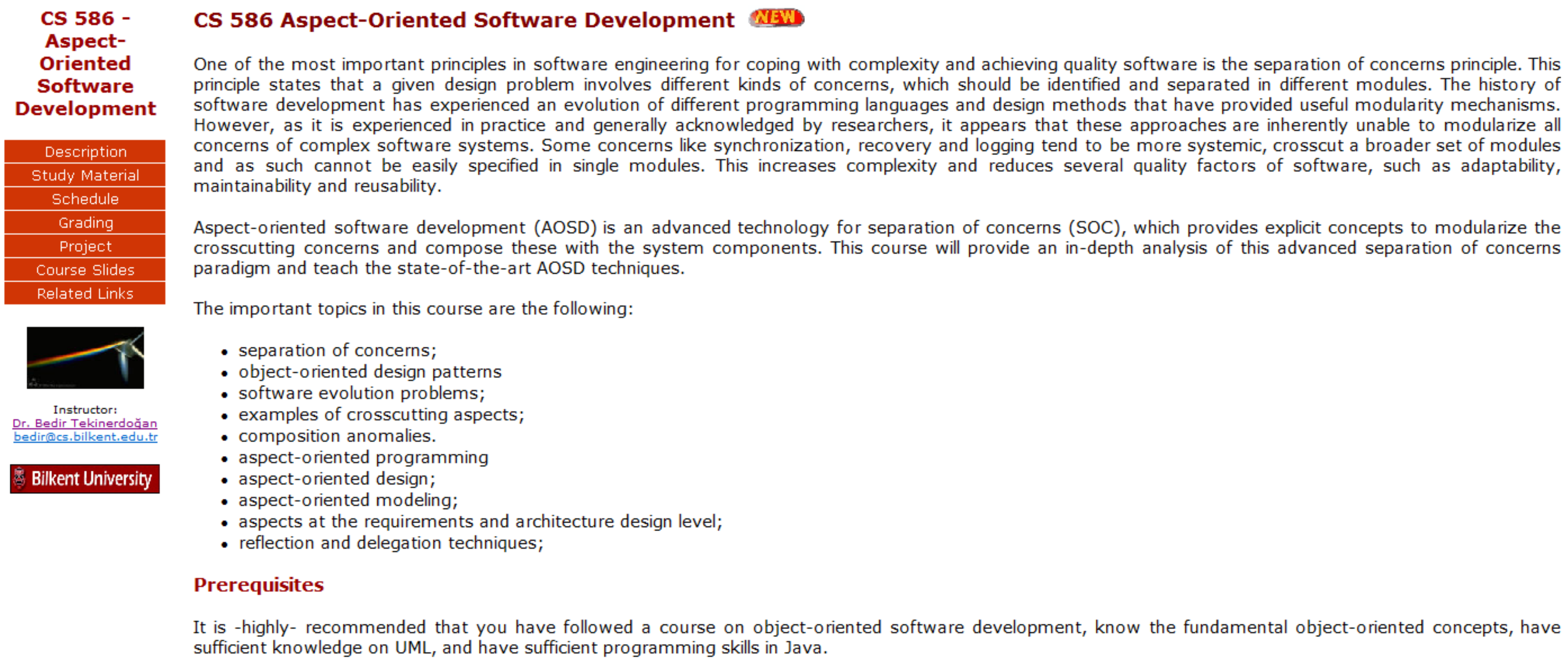
Aspect	Domain	Type	Description of Aspect	Source	Page
Access Control	General	Development	Access control enables an authority to control access to areas and resources in a given physical facility or computer-based information system.	TAOSD 2008 TAOSD 2009	
Authentication	General	Production	In many applications clients need to authenticate themselves before accessing the resources. This aspect modularizes the authentication concern.	TAOSD 2003 TAOSD 2008 TAOSD 2009	
Authorization	General	Production	Authorization, is a process that establishes whether an authenticated user has sufficient permissions to access certain resources.	TAOSD 2003 TAOSD 2008 TAOSD 2009	
Checking pre and post conditions	General, Constraint-based systems	Enforcement	In many applications the execution of operations is bound to the pre and post conditions check.	TAOSD 2008	
Checking in/Checking out	Configuration Management, Versioning Management	Production	In configuration management and revision control systems one developer at a time has write access to the central repository copies of those files. Once one developer "checks out" a file, others can read that file, but no one else is allowed to change that file until that developer "checks in" the updated version (or cancels the checkout).	TAOSD 2003	
Collision Detection	Gaming	Production	In gaming applications multiple objects can move. This aspect detects the collision of objects in the game.	TAOSD 2008	
Connection Setting	Resource Management, Database Systems	Production	A Database connection is a facility that allows client software to communicate with database server software, whether on the same machine or not. A connection is required to send commands and receive answers.	TAOSD 2003 TAOSD 2009	
Constraint Checking	General, Constraint -based systems	Production	When changing the behavior or state of an object usually some constraints need to be checked. This aspect captures the locations of these situations and defines the constraint handling	TAOSD 2003	
Command Execution	General, Constraint -based systems	Production	For different applications different graph models might be required with specific	TAOSD 2008	

Goal

- Improve consciousness on AOSD
- Stimulate research and education on AOSD
- Reflect and foster state-of-the practice of AOSD
- Support MSc and PhD students in providing directions guidelines for their research

AOSD Course at Bilkent University

http://www.cs.bilkent.edu.tr/~bedir/CS586-AOSD/



CS 586 - Aspect-Oriented Software Development

Instructor:
Dr. Bedir Tekinerdoğan
bedir@cs.bilkent.edu.tr

CS 586 Aspect-Oriented Software Development

One of the most important principles in software engineering for coping with complexity and achieving quality software is the separation of concerns principle. This principle states that a given design problem involves different kinds of concerns, which should be identified and separated in different modules. The history of software development has experienced an evolution of different programming languages and design methods that have provided useful modularity mechanisms. However, as it is experienced in practice and generally acknowledged by researchers, it appears that these approaches are inherently unable to modularize all concerns of complex software systems. Some concerns like synchronization, recovery and logging tend to be more systemic, crosscut a broader set of modules and as such cannot be easily specified in single modules. This increases complexity and reduces several quality factors of software, such as adaptability, maintainability and reusability.

Aspect-oriented software development (AOSD) is an advanced technology for separation of concerns (SOC), which provides explicit concepts to modularize the crosscutting concerns and compose these with the system components. This course will provide an in-depth analysis of this advanced separation of concerns paradigm and teach the state-of-the-art AOSD techniques.

The important topics in this course are the following:

- separation of concerns;
- object-oriented design patterns
- software evolution problems;
- examples of crosscutting aspects;
- composition anomalies.
- aspect-oriented programming
- aspect-oriented design;
- aspect-oriented modeling;
- aspects at the requirements and architecture design level;
- reflection and delegation techniques;

Prerequisites

It is -highly- recommended that you have followed a course on object-oriented software development, know the fundamental object-oriented concepts, have sufficient knowledge on UML, and have sufficient programming skills in Java.

AOSD Project

Tasks

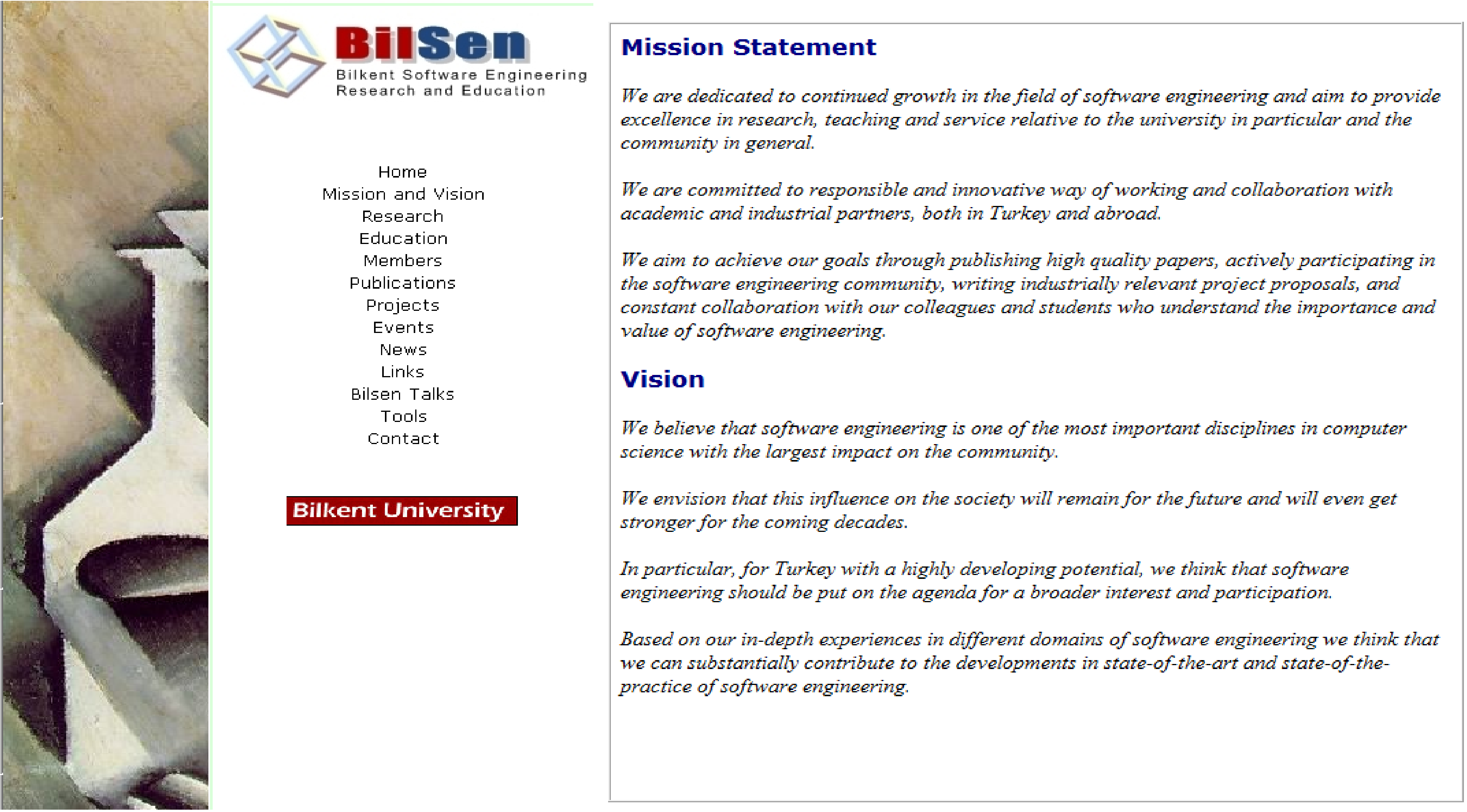
- Select a case (existing or developed from scratch)
- Object-oriented design of the case; using object-oriented design patterns
- Develop change scenarios
- Problem Statement: explain the shortcomings of the object-oriented design and identify crosscutting concerns, aspects
- Aspect-Oriented Programming in AspectJ
- and comparison with another AOP language

Deliverables

- Aspect-oriented program(s) in Java.
- PowerPoint presentation
- Workshop paper

Bilsen - Software Engineering Group

http://www.cs.bilkent.edu.tr/Bilsen



Mission Statement

We are dedicated to continued growth in the field of software engineering and aim to provide excellence in research, teaching and service relative to the university in particular and the community in general.

We are committed to responsible and innovative way of working and collaboration with academic and industrial partners, both in Turkey and abroad.

We aim to achieve our goals through publishing high quality papers, actively participating in the software engineering community, writing industrially relevant project proposals, and constant collaboration with our colleagues and students who understand the importance and value of software engineering.

Vision

We believe that software engineering is one of the most important disciplines in computer science with the largest impact on the community.

We envision that this influence on the society will remain for the future and will even get stronger for the coming decades.

In particular, for Turkey with a highly developing potential, we think that software engineering should be put on the agenda for a broader interest and participation.

Based on our in-depth experiences in different domains of software engineering we think that we can substantially contribute to the developments in state-of-the-art and state-of-the-practice of software engineering.

Organisation

Unique in two perspectives

- Presenters
  - Mostly students
- Audience
  - Students
  - Visitors with probably no explicit/mature knowledge on AOSD

Synergy – Win x Win x Win x ....

- *Synergy*: Final outcome is greater than the sum of its parts…

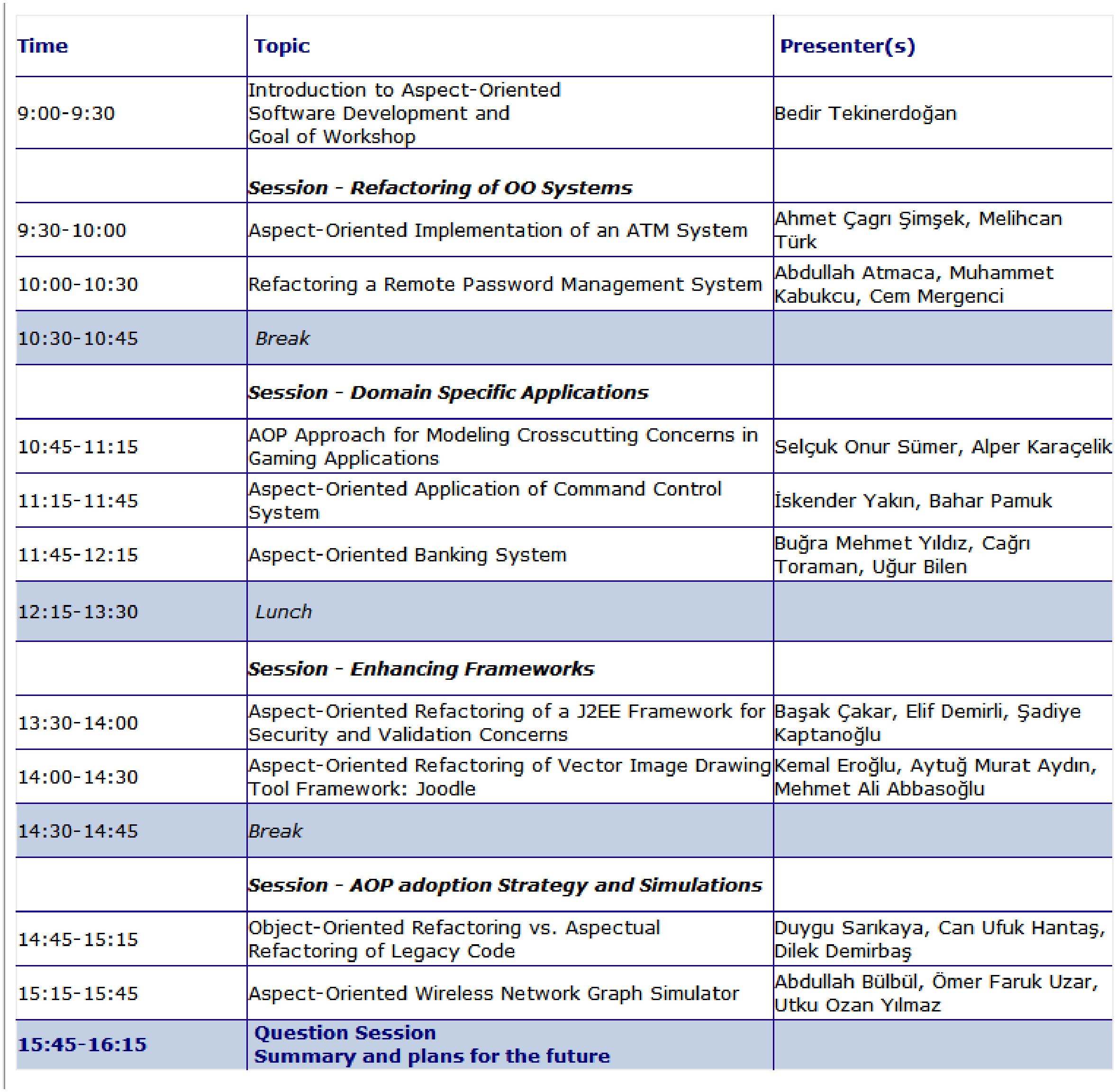
Workshop goal = Win^{n}

- AOSD course students
  - Experience in workshop setting
  - Experience in writing papers
  - Experience in writing AOP program
- Workshop Participants
  - Knowledge of AOSD domain
  - Acquiring new ideas for development and research
- Bilkent University…
- Bilsen ☺
- Software Engineering *Research and Education* in Turkey
- …?

4th Generation AOSD Experts in Turkey… ☺

Abbasoğlu, Mehmet Ali CS	Atmaca, Abdullah CS	Aydın, Aytuğ Murat CS	Pamuk, Bahar METU
Bilen, Uğur CS	Bülbül, Muhammed Abdullah CS	Çakar, Başak CS	Demirbaş, Dilek CS
Demirli, Elif CS	Eroğlu, Kemal CS	Hantaş, Can Ufuk CS	Kabukçu, Muhammet CS
Kaptanoğlu, Şadiye CS	Karaçelik, Alper CS	Mergenci, Cem CS	Sarıkaya, Duygu CS
Sümer, Selçuk Onur CS	Şimşek, Ahmet Çağrı CS	Toraman, Çağrı CS	Türk, Melihcan CS
Uzar, Ömer Faruk CS	Yakın, İskender CS	Yıldız, Buğra Mehmet CS	Yılmaz, Utku Ozan CS

Workshop Program - Morning



Time	Topic	Presenter(s)
9:00-9:30	Introduction to Aspect-Oriented Software Development and Goal of Workshop	Bedir Tekinerdoğan
	Session - Refactoring of OO Systems	
9:30-10:00	Aspect-Oriented Implementation of an ATM System	Ahmet Çagrı Şimşek, Melihcan Türk
10:00-10:30	Refactoring a Remote Password Management System	Abdullah Atmaca, Muhammet Kabukcu, Cem Mergenci
10:30-10:45	*Break*	
	Session - Domain Specific Applications	
10:45-11:15	AOP Approach for Modeling Crosscutting Concerns in Gaming Applications	Selçuk Onur Sümer, Alper Karaçelik
11:15-11:45	Aspect-Oriented Application of Command Control System	İskender Yakın, Bahar Pamuk
11:45-12:15	Aspect-Oriented Banking System	Buğra Mehmet Yıldız, Cağrı Toraman, Uğur Bilen
12:15-13:30	*Lunch*	
	Session - Enhancing Frameworks	
13:30-14:00	Aspect-Oriented Refactoring of a J2EE Framework for Security and Validation Concerns	Başak Çakar, Elif Demirli, Şadiye Kaptanoğlu
14:00-14:30	Aspect-Oriented Refactoring of Vector Image Drawing Tool Framework: Joodle	Kemal Eroğlu, Aytuğ Murat Aydın, Mehmet Ali Abbasoğlu
14:30-14:45	*Break*	
	Session - AOP adoption Strategy and Simulations	
14:45-15:15	Object-Oriented Refactoring vs. Aspectual Refactoring of Legacy Code	Duygu Sarıkaya, Can Ufuk Hantaş, Dilek Demirbaş
15:15-15:45	Aspect-Oriented Wireless Network Graph Simulator	Abdullah Bülbül, Ömer Faruk Uzar, Utku Ozan Yılmaz
15:45-16:15	**Question Session** **Summary and plans for the future**	

Participants

- Around 50-60 participants
- Both from industry and academy

BilSen
Bilkent Software Engineering
Research and Education

TAOSD
Ankara 2008

Fourth Turkish Aspect-Oriented Software Development Workshop

Bilkent University, Ankara, Turkey
29 December 2009

Bilkent University