

Application-Specific Models and Pointcuts using a Logic Meta Language

JOHAN BRICHAU

ANDY KELLENS

KRIS GYBELS

KIM MENS

ROBERT HIRSCHFELD

THEO D'HONDT



Crosscutting Concerns

```
import java.io.*;
import java.util.zip.*;

// Author: Steve Nouri
// Version: 1.0
// Command line program to copy a file to another directory.
// Usage: java CopyFile.java <sourceFile> <targetDir>

public class CopyFile {
    // command line arguments
    public static void main(String[] args) {
        if (args.length < 2) {
            System.out.println("Usage: java CopyFile.java <sourceFile> <targetDir>");
            return;
        }
        String sourceFile = args[0];
        String targetDir = args[1];

        // Check if source file exists
        File source = new File(sourceFile);
        if (!source.exists()) {
            System.out.println("Source file does not exist: " + sourceFile);
            return;
        }

        // Check if target directory exists
        File target = new File(targetDir);
        if (!target.exists()) {
            System.out.println("Target directory does not exist: " + targetDir);
            return;
        }

        // Copy file
        try {
            File destFile = new File(target, source.getName());
            FileInputStream in = new FileInputStream(sourceFile);
            FileOutputStream out = new FileOutputStream(destFile);
            byte[] buffer = new byte[1024];
            int bytesRead;
            while ((bytesRead = in.read(buffer)) != -1) {
                out.write(buffer, 0, bytesRead);
            }
            in.close();
            out.close();
        } catch (IOException e) {
            System.out.println("Error copying file: " + e.getMessage());
        }

        // Verify file
        File dest = new File(target, source.getName());
        if (!dest.exists()) {
            System.out.println("Destination file does not exist: " + dest);
        }
    }
}
```

```
public class FileDownload {
    public static void download(String address, String localFileName) {
        URL url = new URL(address);
        HttpURLConnection conn = null;
        InputStream in = null;
        OutputStream out = null;

        try {
            conn = (HttpURLConnection) url.openConnection();
            in = conn.getInputStream();
            out = new FileOutputStream(localFileName);

            byte[] buffer = new byte[1024];
            int numRead;
            long numWritten = 0;
            while ((numRead = in.read(buffer)) != -1) {
                out.write(buffer, 0, numRead);
                numWritten += numRead;
            }
            System.out.println("Downloaded " + numWritten + " bytes");
        } catch (IOException e) {
            System.out.println("Error downloading file: " + e.getMessage());
        } finally {
            try {
                if (in != null) {
                    in.close();
                }
                if (out != null) {
                    out.close();
                }
            } catch (IOException e) {}
        }
    }

    public static void download(String address) {
        int lastSlashIndex = address.lastIndexOf('/');
        if (lastSlashIndex >= 0 && lastSlashIndex < address.length() - 1) {
            download(address, address.substring(lastSlashIndex + 1));
        } else {
            System.err.println("Could not figure out local file name for " + address);
        }
    }

    public static void main(String[] args) {
        for (int i = 0; i < args.length; i++) {
            download(args[i]);
        }
    }
}
```

```
public class HappyNewYear implements Runnable {
    private static NumberFormat formatter = NumberFormat.getInstance();
    private JFrame frame;
    private JLabel label;
    private long newYearMillis;
    private String message;

    public HappyNewYear(JFrame frame, JLabel label) {
        this.frame = frame;
        this.label = label;
        // compute beginning of next year
        Calendar cal = new GregorianCalendar();
        int nextYear = cal.get(Calendar.YEAR) + 1;
        cal.set(Calendar.YEAR, nextYear);
        cal.set(Calendar.MONTH, Calendar.JANUARY);
        cal.set(Calendar.HOUR_OF_DAY, 0);
        cal.set(Calendar.MINUTE, 0);
        cal.set(Calendar.SECOND, 0);
        newYearMillis = cal.getTime().getTime();
        // prepare a message
        message = "Happy " + nextYear + "!";
    }

    public static int determineFontSize(JFrame frame,
                                       int componentWidth, String fontName, int fontStyle,
                                       String text) {
        int fontSize = componentWidth * 2 / text.length();
        Font font = new Font(fontName, fontStyle, fontSize);
        FontMetrics fontMetrics = frame.getFontMetrics(font);
        int stringWidth = fontMetrics.stringWidth(text);
        return (int)(fontSize * 0.9 * componentWidth / stringWidth);
    }

    public static void main(String[] Args) {
        JFrame frame = new JFrame();
        frame.addKeyListener(new KeyListener() {
            public void keyPressed(KeyEvent event) {}
            public void keyReleased(KeyEvent event) {}
            public void keyTyped(KeyEvent event) {}
        });

        frame.setUndecorated(true);
        JLabel label = new JLabel("");
        label.setBackground(Color.BLACK);
        label.setForeground(Color.WHITE);
        label.setFont(font);
        label.setText(message);
        GraphicsEnvironment env = GraphicsEnvironment.getLocalGraphicsEnvironment();
        final int screenHeight = env.getHeight();
        final String fontName = "Monospace";
        final int fontSize = determineFontSize(frame,
                                             screenHeight, fontName, Font.PLAIN, message);
        int fontSizeText = determineFontSize(frame,
                                             screenHeight, fontName, Font.PLAIN, message);
        label.setFont(new Font(fontName, Font.PLAIN, fontSizeText));
        label.setText(message);
        new HappyNewYear(frame, label).run();
    }

    public void run() {
        boolean newYear = false;
        do {
            long time = System.currentTimeMillis();
            long remaining = (newYearMillis - time) / 1000;
            String output = remaining < 1 ? "" : remaining + "s";
            // new year!
            newYear = time > newYearMillis;
            output = message;
        } while (true);

        // make a String from the number of seconds
        output = formatter.format(remaining);
        label.setText(output);
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {}
    }
}
```

Crosscutting Concerns

Synchronization

```
import java.io.*;
import java.util.*;

/**
 * Command line program to copy a file to another directory.
 * Author: Steve Nouri
 */
public class CopyFile {
    // constant values for the overwrite option
    public static final int overwriteNever = 0;
    public static final int overwriteOnce = 1;
    public static final int overwriteAlways = 2;

    // program options
    private static boolean quiet = true;
    private static boolean verbose = true;
    private static boolean overwriteOption = overwriteAlways;

    public static long copyFile(String srcFile, String destDir)
        throws IOException {
        // check if destDir is a directory
        File destDirFile = new File(destDir);
        if (!destDirFile.isDirectory())
            throw new IOException("Destination directory is not a directory.");

        // check if srcFile exists
        File srcFile = new File(srcFile);
        if (!srcFile.exists())
            throw new IOException("Source file does not exist.");

        // get the file name
        String fileName = srcFile.getName();

        // get the destination file name
        String destFileName = destDirFile + fileName;

        // check if the destination file exists
        File destFile = new File(destFileName);
        if (destFile.exists()) {
            // check if we should overwrite
            if (overwriteOption == overwriteNever)
                return 0;
            else if (overwriteOption == overwriteOnce)
                if (!destFile.isDirectory())
                    return 0;
            else // overwriteAlways
                return 1;
        }

        // copy the file
        try {
            FileInputStream in = new FileInputStream(srcFile);
            FileOutputStream out = new FileOutputStream(destFile);
            byte[] buffer = new byte[1024];
            int numRead;
            while ((numRead = in.read(buffer)) != -1)
                out.write(buffer, 0, numRead);
            out.close();
            in.close();
        } catch (IOException e) {
            System.err.println("Error: " + e.getMessage());
        }

        // return the number of bytes copied
        return destFile.length();
    }

    // main method
    public static void main(String[] args) {
        if (args.length != 2)
            System.err.println("Usage: java CopyFile <srcFile> <destDir>");
        else {
            String srcFile = args[0];
            String destDir = args[1];
            try {
                long bytesCopied = copyFile(srcFile, destDir);
                System.out.println("Copied " + bytesCopied + " bytes from " + srcFile + " to " + destDir);
            } catch (IOException e) {
                System.err.println("Error: " + e.getMessage());
            }
        }
    }
}
```

```
public class FileDownload {
    public static void download(String address, String localFileName)
        throws IOException {
        URL url = new URL(address);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("GET");
        conn.setDoOutput(true);
        OutputStream out = new FileOutputStream(localFileName);
        InputStream in = conn.getInputStream();
        byte[] buffer = new byte[1024];
        int numRead;
        long numWritten = 0;
        while ((numRead = in.read(buffer)) != -1)
            out.write(buffer, 0, numRead);
        numWritten += numRead;
        System.out.println("Downloaded " + numWritten + " bytes from " + address);
        out.close();
        in.close();
    }

    public static void download(String address)
        throws IOException {
        int lastSlashIndex = address.lastIndexOf('/');
        if (lastSlashIndex >= 0 && lastSlashIndex < address.length() - 1)
            download(address, address.substring(lastSlashIndex + 1));
        else
            System.err.println("Could not figure out local file name for " + address);
    }

    public static void main(String[] args) {
        for (int i = 0; i < args.length; i++)
            download(args[i]);
    }
}
```

```
public class HappyNewYear implements Runnable {
    private static NumberFormat formatter = NumberFormat.getInstance();
    private JFrame frame;
    private JLabel label;
    private long newYearMillis;
    private String message;

    public HappyNewYear(JFrame frame, JLabel label) {
        // set the window title
        this.frame = frame;
        this.label = label;

        // compute beginning of next year
        Calendar cal = new GregorianCalendar();
        int nextYear = cal.get(Calendar.YEAR) + 1;
        cal.set(Calendar.YEAR, nextYear);
        cal.set(Calendar.MONTH, Calendar.JANUARY);
        cal.set(Calendar.DAY_OF_MONTH, 1);
        cal.set(Calendar.HOUR_OF_DAY, 0);
        cal.set(Calendar.MINUTE, 0);
        cal.set(Calendar.SECOND, 0);
        newYearMillis = cal.getTime().getTime();

        // prepare a message
        message = "Happy " + nextYear + "!";
    }

    public static int determineFontSize(JFrame frame,
        int componentWidth, String fontName, int fontStyle,
        String text) {
        int fontSize = componentWidth * 2 / text.length();
        Font font = new Font(fontName, fontStyle, fontSize);
        FontMetrics fontMetrics = frame.getFontMetrics(font);
        int stringWidth = fontMetrics.stringWidth(text);
        return (int) (fontSize * 0.9 * componentWidth / stringWidth);
    }

    public static void main(String[] args) {
        JFrame frame = new JFrame();
        frame.addKeyListener(new KeyListener() {
            public void keyPressed(KeyEvent event) {
                public void keyReleased(KeyEvent event) {}
                public void keyTyped(KeyEvent event) {}
            }
        });
        frame.setTitle("Happy New Year");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 300);
        frame.setVisible(true);
        label = new JLabel("Happy New Year!");
        label.setFont(new Font("Serif", Font.BOLD, 24));
        label.setForeground(Color.RED);
        label.setHorizontalAlignment(JLabel.CENTER);
        label.setVerticalAlignment(JLabel.CENTER);
        frame.getContentPane().add(label);
        GraphicsEnvironment env = GraphicsEnvironment.getLocalGraphicsEnvironment();
        GraphicsDevice gd = env.getDefaultScreenDevice();
        GraphicsConfiguration gc = gd.getDefaultConfiguration();
        final String fontName = "SansSerif";
        final int fontStyle = Font.PLAIN;
        final Font font = new Font(fontName, fontStyle, determineFontSize(frame,
            frame.getWidth(), fontName, fontStyle, message));
        int fontSize = font.getSize();
        int fontHeight = fontMetrics.getHeight();
        int fontBaseline = fontMetrics.getBaseline();
        label.setFont(font);
        label.setText(message);
        label.setBounds(0, 0, frame.getWidth(), fontHeight);
        new HappyNewYear(frame, label).run();
    }

    public void run() {
        boolean newYear = false;
        do {
            long time = System.currentTimeMillis();
            long remaining = (newYearMillis - time) / 1000;
            String output = "Time until New Year: " + remaining + " seconds";
            if (remaining < 1)
                output = "New Year!";
            else if (remaining < 10)
                output = "Counting down...";
            label.setText(output);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {}
        } while (true);
    }
}
```


Crosscutting Concerns

Synchronization

UI dependency

```
import java.io.*;
import java.util.zip.*;
import java.util.*;

/** Command line program to copy a file to another directory.
 * Author: Steve Nouri
 */
public class CopyFile {

    // constant values for the overwrite option
    public static final int overwriteNever = 0;
    public static final int overwriteAlways = 1;
    public static final int overwriteIfNewer = 2;

    public static final int overwrite = 0;

    // program options initialized to default values
    private static boolean quiet = false;
    private static boolean overwriteMessage = true;
    private static boolean overwrite = overwriteIfNewer;
    private static int overwriteMode = overwriteIfNewer;

    public static long copyFile(String srcFile, File destDir)
        throws IOException {
        File destFile = new File(destDir, srcFile);
        long srcSize = srcFile.length();
        long destSize = destFile.length();

        if (srcSize == destSize) {
            // check if src is newer than dest
            if (srcFile.lastModified() > destFile.lastModified()) {
                // overwrite if newer
                if (overwriteMode == overwriteIfNewer) {
                    overwrite = true;
                }
            } else {
                // src is not newer than dest
                overwrite = false;
            }
        } else if (srcSize > destSize) {
            // src is larger than dest
            overwrite = true;
        } else {
            // src is smaller than dest
            overwrite = false;
        }

        // check if overwrite is allowed
        if (overwrite) {
            // delete dest file
            destFile.delete();
        }

        // copy src file to dest
        try {
            FileInputStream srcStream = new FileInputStream(srcFile);
            FileOutputStream destStream = new FileOutputStream(destFile);
            byte[] buffer = new byte[4096];
            int bytesRead;
            while ((bytesRead = srcStream.read(buffer)) != -1) {
                destStream.write(buffer, 0, bytesRead);
            }
            srcStream.close();
            destStream.close();
        } catch (IOException e) {
            System.err.println("Error: " + e.getMessage());
        }

        // print message
        System.out.println("Copied " + srcFile + " to " + destDir);
    }

    // main method
    public static void main(String[] args) {
        if (args.length != 2) {
            System.out.println("Usage: java CopyFile <srcFile> <destDir>");
            return;
        }
        String srcFile = args[0];
        File destDir = new File(args[1]);
        try {
            copyFile(srcFile, destDir);
        } catch (IOException e) {
            System.err.println("Error: " + e.getMessage());
        }
    }
}
```

```
public class FileDownload {
    public static void download(String address, String localFileName)
        throws IOException {
        URL url = new URL(address);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("GET");
        conn.setDoInput(true);
        conn.setDoOutput(true);
        conn.setConnectTimeout(10000);
        conn.setReadTimeout(10000);
        InputStream inputStream = conn.getInputStream();
        OutputStream outputStream = new FileOutputStream(localFileName);
        byte[] buffer = new byte[4096];
        int numRead;
        long numWritten = 0;
        while ((numRead = inputStream.read(buffer)) != -1) {
            outputStream.write(buffer, 0, numRead);
            numWritten += numRead;
        }
        System.out.println("Downloaded " + localFileName + " (" + numWritten +
            " bytes)");
        inputStream.close();
        outputStream.close();
    }

    public static void download(String address) {
        int lastSlashIndex = address.lastIndexOf('/');
        if (lastSlashIndex >= 0 && lastSlashIndex < address.length() - 1) {
            download(address, address.substring(lastSlashIndex + 1));
        } else {
            System.err.println("Could not figure out local file name for " +
                address);
        }
    }

    public static void main(String[] args) {
        for (int i = 0; i < args.length; i++) {
            download(args[i]);
        }
    }
}
```

```
public class HappyNewYear implements Runnable {
    private static NumberFormat formatter = NumberFormat.getInstance();
    private JFrame frame;
    private JLabel label;
    private long newYearMillis;
    private String message;

    public HappyNewYear(JFrame frame, JLabel label) {
        this.frame = frame;
        this.label = label;
        // compute beginning of next year
        Calendar cal = new GregorianCalendar(1);
        int nextYear = cal.get(Calendar.YEAR) + 1;
        cal.set(Calendar.YEAR, nextYear);
        cal.set(Calendar.MONTH, Calendar.JANUARY);
        cal.set(Calendar.DAY_OF_MONTH, 1);
        cal.set(Calendar.HOUR_OF_DAY, 0);
        cal.set(Calendar.MINUTE, 0);
        cal.set(Calendar.SECOND, 0);
        newYearMillis = cal.getTime().getTime();
        // prepare a message
        message = "Happy " + nextYear + "!";
    }

    public static int determineFontSize(JFrame frame,
        int componentWidth, String fontName, int fontStyle,
        String text) {
        int fontSize = componentWidth * 2 / text.length();
        Font font = new Font(fontName, fontStyle, fontSize);
        FontMetrics fontMetrics = frame.getFontMetrics(font);
        int stringWidth = fontMetrics.stringWidth(text);
        return (int) (fontSize * 0.9 * componentWidth / stringWidth);
    }

    public static void main(String[] args) {
        JFrame frame = new JFrame();
        frame.addKeyListener(new KeyListener() {
            public void keyPressed(KeyEvent event) {
                public void keyReleased(KeyEvent event) {}
                public void keyTyped(KeyEvent event) {}
            }
        });
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 300);
        frame.setVisible(true);
        new HappyNewYear(frame, frame.getContentPane().getLabel()).run();
    }

    boolean newYear = false;
    do {
        long time = System.currentTimeMillis();
        long remaining = (newYearMillis - time) / 1000;
        String output;
        if (remaining < 5) {
            // new year!
            newYear = true;
            output = message;
        } else {
            // make a string from the number of seconds
            output = formatter.format(remaining);
            label.setText(output);
        }
        Thread.sleep(1000);
    } while (true);
}
```



AOP

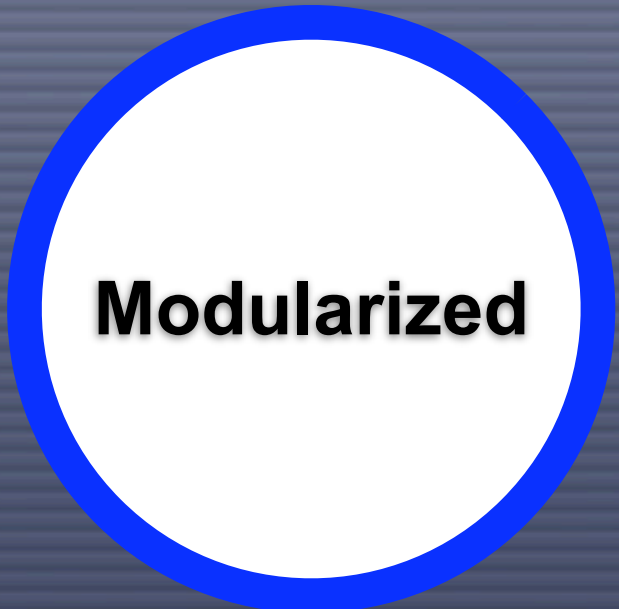
```
import java.io.*;
import java.util.zip.*;
/**
 * Command line program to copy a file to another directory.
 * @author Bruce Eckel
 */
public class CopyFile {
    // constants values for the override option
    public static final int OVERWRITE_MODE = 1;
    public static final int NO_OVERRIDE_MODE = 2;
    // program options initialized to default values
    private static int overwriteMode = 1;
    private static boolean silent = true;
    public static long copyRecursiveDir(String srcDir, File destDir)
        throws IOException {
        byte[] buffer = new byte[1024];
        while (FileUtils.exists(srcDir)) {
            if (FileUtils.isDirectory(srcDir)) {
                checkAndCopyDir(srcDir, destDir);
            } else {
                return new long[checkAndCopyFile(srcDir, destDir)];
            }
        }
        public static long copyRecursiveDir(String srcDir, File destDir)
            throws IOException {
                while (FileUtils.exists(srcDir)) {
                    if (FileUtils.isDirectory(srcDir)) {
                        checkAndCopyDir(srcDir, destDir);
                    } else {
                        return new long[checkAndCopyFile(srcDir, destDir)];
                    }
                }
            }
        public static void main(String[] args) {
            String srcDir = "src";
            String destDir = "dest";
            if (args.length > 0) {
                srcDir = args[0];
            }
            if (args.length > 1) {
                destDir = args[1];
            }
            copyRecursiveDir(srcDir, destDir);
        }
    }
}
```

```
public class FileDownload {
    public static void download(String address, String localFileName) {
        OutputStream out = null;
        URLConnection conn = null;
        InputStream in = null;
        try {
            conn = conn.openConnection();
            byte[] buffer = new byte[1024];
            int numRead;
            long numWritten = 0;
            while ((numRead = in.read(buffer)) != -1) {
                out.write(buffer, 0, numRead);
                numWritten += numRead;
            }
        } finally {
            if (in != null) {
                in.close();
            }
            if (out != null) {
                out.close();
            }
        } catch (IOException ioe) {
            System.err.println("Could not figure out local file name for " + address);
        }
    }
    public static void download(String address) {
        int lastSlashIndex = address.lastIndexOf('/');
        System.err.println("Could not figure out local file name for " + address);
    }
    public static void main(String[] args) {
        for (int i = 0; i < args.length; i++) {
            download(args[i]);
        }
    }
}
```

```
public static long createChecksumFile(String srcDir, File destDir)
    throws IOException {
    byte[] buffer = new byte[1024];
    while ((numRead = in.read(buffer)) != -1) {
        out.write(buffer, 0, numRead);
        numWritten += numRead;
    }
    return new long[checksum.getBytes().length];
}
```

```
public static long createChecksumFile(String srcDir, File destDir)
    throws IOException {
    byte[] buffer = new byte[1024];
    while ((numRead = in.read(buffer)) != -1) {
        out.write(buffer, 0, numRead);
        numWritten += numRead;
    }
    return new long[checksum.getBytes().length];
}
```

```
public class HappyNewYear implements Runnable {
    private static NumberFormat formatter = NumberFormat.getInstance();
    private JFrame frame;
    private JLabel label;
    private long newYearMillis;
    private String message;
    public HappyNewYear(JFrame frame, JLabel label) {
        cal.set(Calendar.YEAR, nextYear);
        cal.set(Calendar.MONTH, Calendar.JANUARY);
        cal.set(Calendar.DAY_OF_MONTH, 1);
        cal.set(Calendar.HOUR_OF_DAY, 0);
        cal.set(Calendar.MINUTE, 0);
        cal.set(Calendar.SECOND, 0);
        newYearMillis = cal.getTime().getTime();
        // prepare a message
        message = "Happy " + nextYear + "!";
    }
    public static int determineFontSize(JFrame frame,
        int componentWidth, String fontName, int fontStyle,
        String text) {
        return (int)(fontSize * 0.95 *
            componentWidth / stringWidth);
    }
    public static void main(String[] args) {
        JFrame frame = new JFrame();
        frame.addKeyListener(new KeyListener() {
            public void keyPressed(KeyEvent event) {
                public void keyReleased(KeyEvent event) {
                    if (event.getKeyChar() == KeyEvent.VK_ESCAPE) {
                        System.exit(0);
                    }
                }
            }
        });
        public void keyTyped(KeyEvent event) {
            // frame.setBackground(Color.BLACK);
            JLabel label = new JLabel(" ");
            label.setBackground(Color.BLACK);
            label.setForeground(Color.WHITE);
            label.setOpaque(true);
            label.setHorizontalAlignment(SwingConstants.CENTER);
            frame.getContentPane().add(label);
            GraphicsEnvironment.getLocalGraphicsEnvironment().
                getConfiguration().setFullScreenWindow(frame);
            final int fontStyle = Font.BOLD;
            label.setFont(new Font(fontName, fontStyle,
                determineFontSize(frame, label.getWidth(),
                    new HappyNewYear(frame, label).run());
        }
    }
    public void run() {
        boolean newYear = false;
        do {
            newYear = true;
            output = message;
        }
        else {
            // make a String from the number of seconds
            output = formatter.format(remaining);
            label.setText(output);
            try {
                Thread.sleep(1000);
            }
        }
    }
}
```



Pointcuts

Often point to details in source code (e.g. names)

```
import java.io.*;
import java.util.zip.*;

/** Command line program to copy a file to another directory.
 * Author: Bruce Eckel
 */
public class CopyFile {
    // constant values for the describe option
    public static final int COMMANDS_DESCRIBE = 1;
    public static final int COMMANDS_COPY = 2;
    // program options initialized to default values
    private static int destination = 1;
    private static boolean silent = false;

    public static long copyRecursive(String srcFile, String destDir)
        throws IOException {
        // create a new byte[] buffer
        byte[] buffer = new byte[4096];
        int bytesRead;
        while ((bytesRead = in.read(buffer)) != 0) {
            // checksum update
            checksum.update(buffer, 0, bytesRead);
        }
        // close
        in.close();
        if (silent) {
            millis = System.currentTimeMillis() - millis;
            System.out.println("seconds: " + millis/1000);
        }
        return new Long(checksum.getValue());
    }

    // determine if data is to be copied to given file.
    // Returns true if data is to be copied to file, false if not
    public static boolean copyFile(String srcFile, String destDir)
        throws IOException {
        // check if destination exists
        File destFile = new File(destDir);
        if (!destFile.exists()) {
            System.out.println("destination does not exist");
            return false;
        }
        // check if source file exists
        File srcFileObj = new File(srcFile);
        if (!srcFileObj.exists()) {
            System.out.println("source file does not exist");
            return false;
        }
        // check if source file is a directory
        if (srcFileObj.isDirectory()) {
            // create a new directory in destination
            File destDirObj = new File(destDir);
            if (!destDirObj.isDirectory()) {
                System.out.println("destination is not a directory");
                return false;
            }
            // check if copying is desired given overwrite option
            if (overwrite == 0) {
                return false;
            }
            // copy timestamp of last modification
        }
        // return user answer, true for yes, false for no
        public static boolean readUserResponse(String message) {
            System.out.println(message);
            BufferedReader in = new BufferedReader(new InputStreamReader(
                System.in));
            String answer = null;
            while ((answer = in.readLine()) != null) {
                if (answer.equals("y") || answer.equals("Y")) {
                    return true;
                }
                if (answer.equals("n") || answer.equals("N")) {
                    return false;
                }
                System.out.println("could not understand answer ('Y' =
                    yes, 'N' = no, please use y for yes or n for no.");
            }
        }
    }
}
```

```
public class FileDownload {
    public static void download(String address, String localFileName) {
        OutputStream out = null;
        URLConnection conn = null;
        InputStream in = null;
        try {
            in = conn.getInputStream();
            byte[] buffer = new byte[1024];
            int numRead;
            long numWritten = 0;
            while ((numRead = in.read(buffer)) != -1) {
                out.write(buffer, 0, numRead);
                numWritten += numRead;
            } finally {
                try {
                    if (in != null) in.close();
                    if (out != null) out.close();
                } catch (IOException ioe) {
                }
            }
        }
        public static void download(String address) {
            int lastSlashIndex = address.lastIndexOf("/");
            System.err.println("Could not figure out local file name for " +
                address);
        }
        public static void main(String[] args) {
            for (int i = 0; i < args.length; i++) {
                download(args[i]);
            }
        }
    }
}
```

```
public static long createChecksumFile(String srcFile) throws IOException {
    long millis = System.currentTimeMillis();
    HttpServletResponse httpServletResponse = new HttpServletResponse();
    byte[] buffer = new byte[4096];
    while ((bytesRead = in.read(buffer)) != 0) {
        // checksum update
        checksum.update(buffer, 0, bytesRead);
    }
    // close
    in.close();
    if (silent) {
        millis = System.currentTimeMillis() - millis;
        System.out.println("seconds: " + millis/1000);
    }
    return new Long(checksum.getValue());
}
```

```
public static long createChecksumFile(String srcFile) throws IOException {
    public HappyNewYear(JFrame frame, JLabel label) {
        cal.set(Calendar.YEAR, nextYear);
        cal.set(Calendar.MONTH, Calendar.JANUARY);
        cal.set(Calendar.DAY_OF_MONTH, 1);
        cal.set(Calendar.HOUR_OF_DAY, 0);
        cal.set(Calendar.MINUTE, 0);
        cal.set(Calendar.SECOND, 0);
        newYearMillis = cal.getTime().getTime();
        // prepare a message
        message = "Happy " + nextYear + "!";
    }
}
```

```
public class HappyNewYear implements Runnable {
    private static NumberFormat formatter = NumberFormat.getInstance();
    private JFrame frame;
    private JLabel label;
    private long newYearMillis;
    private String message;

    public HappyNewYear(JFrame frame, JLabel label) {
        cal.set(Calendar.YEAR, nextYear);
        cal.set(Calendar.MONTH, Calendar.JANUARY);
        cal.set(Calendar.DAY_OF_MONTH, 1);
        cal.set(Calendar.HOUR_OF_DAY, 0);
        cal.set(Calendar.MINUTE, 0);
        cal.set(Calendar.SECOND, 0);
        newYearMillis = cal.getTime().getTime();
        // prepare a message
        message = "Happy " + nextYear + "!";
    }

    public static int determineFontSize(JFrame frame,
        int componentWidth, String fontName, int fontStyle,
        String text) {
        return (int)(fontStyle * 0.95 *
            componentWidth / stringWidth);
    }

    public static void main(String[] args) {
        JFrame frame = new JFrame();
        frame.addKeyListener(new KeyListener() {
            public void keyPressed(KeyEvent event) {
                public void keyReleased(KeyEvent event) {
                    if (event.getKeyChar() == KeyEvent.VK_ESCAPE) {
                        System.exit(0);
                    }
                }
                public void keyTyped(KeyEvent event) {
                }
            }
        });
        frame.setUndecorated(true);
        JLabel label = new JLabel("");
        label.setBackground(Color.BLACK);
        label.setForeground(Color.WHITE);
        label.setOpaque(true);
        label.setHorizontalAlignment(SwingConstants.CENTER);
        frame.getContentPane().add(label);
        GraphicsEnvironment graphicsEnvironment =
            GraphicsEnvironment.getLocalGraphicsEnvironment();
        final int fontStyle = Font.BOLD;
        label.setFont(new Font(fontName, fontStyle,
            Math.min(fontStyleNumber, fontStyle));
        new HappyNewYear(frame, label).run();
    }

    public void run() {
        boolean newYear = false;
        do {
            newYear = true;
            output = message;
        }
        else {
            // make a String from the number of seconds
            output = formatter.format(remaining);
            label.setText(output);
            try {
                Thread.sleep(1000);
            }
        }
    }
}
```

Pointcuts

Often point to details in source code (e.g. names)

```
import java.io.*;
import java.util.zip.*;

/** Command line program to copy a file to another directory.
 * Author: Steve Nouri
 */
public class CopyFile {
    // constant values for the override option
    public static final int COMMENTS_OVERRIDE = 1;
    public static final int COMMENTS_COPY = 2;
    // program options initialized to default values
    private static boolean override = 0;
    private static long timestamp = 0;

    public static long copyFile(String srcFile, String destDir) {
        // create new BufferedInputStream
        try {
            FileInputStream in = new FileInputStream(srcFile);
            // create new BufferedOutputStream
            File destFile = new File(destDir, srcFile);
            FileOutputStream out = new FileOutputStream(destFile);
            // copy the file
            byte[] buffer = new byte[1024];
            int bytesRead;
            while ((bytesRead = in.read(buffer)) != -1) {
                out.write(buffer, 0, bytesRead);
            }
            in.close();
            out.close();
            // copy timestamp if last modified
            timestamp = new File(srcFile).lastModified();
            // return user answer, true if override for no.
            return true;
        } catch (IOException e) {
            System.err.println("Could not copy file: " + e.getMessage());
            return false;
        }
    }

    public static boolean readUserResponse(String message) {
        System.out.println(message);
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        boolean answer = null;
        while ((answer = in.readLine()) != null) {
            if ("y".equalsIgnoreCase(answer)) {
                answer = true;
            } else if ("n".equalsIgnoreCase(answer)) {
                answer = false;
            } else {
                System.out.println("Could not understand answer (" + answer + "). Please use y for yes or n for no.");
            }
        }
        return answer;
    }

    public static void main(String[] args) {
        if (args.length != 2) {
            System.out.println("Usage: java CopyFile <srcFile> <destDir>");
            return;
        }
        String srcFile = args[0];
        String destDir = args[1];
        // copy timestamp of last modified
        timestamp = new File(srcFile).lastModified();
        // copy the file
        copyFile(srcFile, destDir);
        // return user answer, true if override for no.
        boolean answer = readUserResponse("Do you want to override existing file? (y/n)");
        if (answer) {
            copyFile(srcFile, destDir);
        }
    }
}
```

```
public class FileDownload {
    public static void download(String address, String localFileName) {
        try {
            URL url = new URL(address);
            HttpURLConnection conn = (HttpURLConnection) url.openConnection();
            conn.setRequestMethod("GET");
            conn.setDoInput(true);
            conn.setDoOutput(true);
            conn.connect();
            InputStream in = conn.getInputStream();
            OutputStream out = new FileOutputStream(localFileName);
            byte[] buffer = new byte[1024];
            int bytesRead;
            while ((bytesRead = in.read(buffer)) != -1) {
                out.write(buffer, 0, bytesRead);
            }
            in.close();
            out.close();
        } catch (IOException e) {
            System.err.println("Error downloading file: " + e.getMessage());
        }
    }

    public static void download(String address) {
        download(address, "downloadedFile.txt");
    }

    public static void main(String[] args) {
        if (args.length != 1) {
            System.err.println("Could not figure out local file name from arguments");
            return;
        }
        String address = args[0];
        download(address);
    }
}
```

```
public static long createChecksum(File file) throws IOException {
    long millis = System.currentTimeMillis();
    FileInputStream in = new FileInputStream(file);
    try {
        MessageDigest md = MessageDigest.getInstance("MD5");
        byte[] buffer = new byte[1024];
        int bytesRead;
        while ((bytesRead = in.read(buffer)) != -1) {
            md.update(buffer, 0, bytesRead);
        }
        in.close();
        if (in != null) {
            in.close();
        }
        return new long(checksum.getValue());
    } catch (Exception e) {
        System.err.println("Error creating checksum: " + e.getMessage());
    }
}
```

```
public static long createChecksum(File file) throws IOException {
    public HappyNewYear(JFrame frame, JLabel label) {
        cal.set(Calendar.YEAR, nextYear);
        cal.set(Calendar.MONTH, Calendar.JANUARY);
        cal.set(Calendar.DAY_OF_MONTH, 1);
        cal.set(Calendar.HOUR_OF_DAY, 0);
        cal.set(Calendar.MINUTE, 0);
        cal.set(Calendar.SECOND, 0);
        newYearMillis = cal.getTime().getTime();
        // prepare a message
        message = "Happy " + nextYear + "!";
    }
}
```

```
public class HappyNewYear implements Runnable {
    private JLabel label;
    private long newYearMillis;
    private String message;

    public HappyNewYear(JFrame frame, JLabel label) {
        cal.set(Calendar.YEAR, nextYear);
        cal.set(Calendar.MONTH, Calendar.JANUARY);
        cal.set(Calendar.DAY_OF_MONTH, 1);
        cal.set(Calendar.HOUR_OF_DAY, 0);
        cal.set(Calendar.MINUTE, 0);
        cal.set(Calendar.SECOND, 0);
        newYearMillis = cal.getTime().getTime();
        // prepare a message
        message = "Happy " + nextYear + "!";
    }

    public static int determineFontSize(JFrame frame,
        int componentWidth, String fontName,
        int fontStyle) {
        int fontWidth = frame.getWidth();
        int fontHeight = frame.getHeight();
        return (int)(fontStyle * 0.95 *
            componentWidth / stringWidth);
    }

    public void main(String[] args) {
        JFrame frame = new JFrame("Happy New Year");
        frame.addKeyListener(new KeyListener() {
            public void keyPressed(KeyEvent event) {
                public void keyReleased(KeyEvent event) {
                }
                if (event.getKeyChar() == KeyEvent.VK_ESCAPE) {
                    System.exit(0);
                }
            }
            public void keyTyped(KeyEvent event) {
            }
        });
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        label = new JLabel("");
        label.setBackground(Color.BLACK);
        label.setForeground(Color.WHITE);
        label.setFont(new Font(fontName, fontStyle,
            determineFontSize(frame, frame.getWidth(), fontName, fontStyle)));
        frame.getContentPane().add(label);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
        final int fontStyle = Font.BOLD;
        label.setFont(new Font(fontName, fontStyle,
            Math.min(determineFontSize(frame, frame.getWidth(), fontName, fontStyle),
                frame.getWidth())));
        new HappyNewYear(frame, label).run();
    }

    public void run() {
        boolean newYear = false;
        do {
            newYear = true;
            output = message;
            label.setText(output);
        } while (newYear);
        // make a String from the number of seconds
        output = formatter.format(remaining);
        label.setText(output);
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            System.out.println("Error sleeping: " + e.getMessage());
        }
    }
}
```


Aspects in Smalltalk

AspectS

- Framework based
- Pointcut using Smalltalk
- Advice using block

AsBeforeAfterAdvice

qualifier:...

pointcut: [WindowSensor withAllSubclasses

select:

[:each includesSelector: #eventDoubleClick:]

thenCollect: [:each | AsJoinPointDescriptor

targetClass: each

targetSelector: #eventDoubleClick:]]

beforeBlock: [:receiver :args :aspect :client |
...]

Aspects in Smalltalk

AspectS

- Framework based
- Pointcut using Smalltalk
- Advice using block

AsBeforeAfterAdvice

```
qualifier:...
pointcut: [WindowSensor withAllSubclasses
  select:
    [:each includesSelector: #eventDoubleClick:]
  thenCollect:[:each | AsJoinPointDescriptor
    targetClass: each
    targetSelector:#eventDoubleClick:]]
beforeBlock: [:receiver :args :aspect :client |
  ...]
```

Pointcut

Aspects in Smalltalk

AspectS

- Framework based
- Pointcut using Smalltalk
- Advice using block

AsBeforeAfterAdvice

```
qualifier:...
pointcut: [WindowSensor withAllSubclasses
  select:
    [:each includesSelector: #eventDoubleClick:]
  thenCollect:[:each | AsJoinPointDescriptor
    targetClass: each
    targetSelector:#eventDoubleClick:]]
beforeBlock: [:receiver :args :aspect :client |
  ...]
```

Pointcut

Advice

Aspects in Carma

Carma

- Pointcut language
- Declarative Meta Programming
- Based on SOUL

```
after ?jp matching  
    reception(?jp, #eventDoubleClick: ,?args),  
    within(?jp, ?class, ?selector),  
    classInHierarchyOf(?class, [WindowSensor])  
do  
    Transcript show: ?class.  
    ...
```

Aspects in Carma

Carma

- Pointcut language
- Declarative Meta Programming
- Based on SOUL

- Declarative
- Unification
- Recursion

```
after ?jp matching
  reception(?jp, #eventDoubleClick: ,?args),
  within(?jp, ?class, ?selector),
  classInHierarchyOf(?class, [WindowSensor])
do
  Transcript show: ?class.
  ...
```

Aspects in Carma

Carma

- Pointcut language
- Declarative Meta Programming
- Based on SOUL

- Declarative
- Unification
- Recursion

```
after ?jp matching
  reception(?jp, #eventDoubleClick: ,?args),
  within(?jp, ?class, ?selector),
  classInHierarchyOf(?class, [WindowSensor])
```

```
do
```

```
  Transcript show: ?class.
```

```
  ...
```

Pointcut

Aspects in Carma

Carma

- Pointcut language
- Declarative Meta Programming
- Based on SOUL

- Declarative
- Unification
- Recursion

```
after ?jp matching
  reception(?jp, #eventDoubleClick: ,?args),
  within(?jp, ?class, ?selector),
  classInHierarchyOf(?class, [WindowSensor])
```

Pointcut

```
do
  Transcript show: ?class.
```

Advice

AspectSOUL

AspectS

- Framework

Carma

- Pointcuts

AspectSOUL

AspectS

- Framework

Carma

- Pointcuts

```
AsCARMABeforeAfterAdvice
```

```
qualifier:...
```

```
pointcutQuery: 'reception(?jp, #eventDoubleClick: ,?args),
```

```
  within(?jp, ?class, ?selector),
```

```
  classInHierarchyOf(?class, [WindowSensor])'
```

```
beforeBlock: [:receiver :args :aspect :client |
```

```
  ...]
```

AspectSOUL

AspectS

- Framework

Carma

- Pointcuts

AsCARMABeforeAfterAdvice

qualifier:...

```
pointcutQuery: 'reception(?jp, #eventDoubleClick  
  within(?jp, ?class, ?selector),  
  classInHierarchyOf(?class, [WindowSensor])'
```

```
beforeBlock: [:receiver :args :aspect :client |  
  ...]
```

Pointcut

AspectSOUL

AspectS

- Framework

Carma

- Pointcuts

AsCARMABeforeAfterAdvice

qualifier:...

```
pointcutQuery: 'reception(?jp, #eventDoubleClick  
within(?jp, ?class, ?selector),  
classInHierarchyOf(?class, [WindowSensor])'
```

Pointcut

```
beforeBlock: [:receiver :args :aspect :client |  
...]
```

Advice

Fragility

- 1.accessing protocol
- 2.selector corresponds to variable name

```
Person>>name
```

```
^name
```

```
Person>>name: anObject
```

```
name := anObject
```


Fragility

- 1.accessing protocol
- 2.selector corresponds to variable name

```
Person>>name  
  ^name  
  
Person>>name: anObject  
  name := anObject
```

Synchronization

```
class(?class),  
methodNameInClass(?method,?accessor,?class),  
instanceVariableInClassChain(?accessor,?class),  
methodInProtocol(?method,accessing),  
reception(?joinpoint,?accessor,?args),  
withinClass(?joinpoint,?class)
```

Fragility

- 1.accessing protocol
- 2.selector corresponds to variable name

```
Person>>name  
  ^name  
  
Person>>name: anObject  
  name := anObject
```

```
Person>>getName  
  ^name  
  
Person>>setName: anObject  
  name := anObject
```

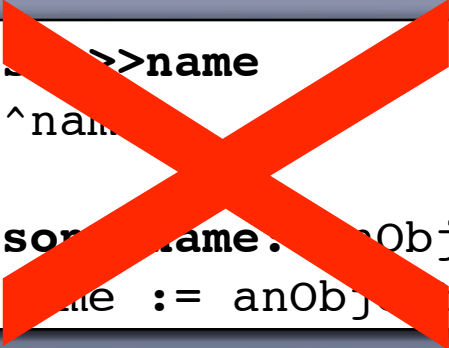
Synchronization

```
class(?class),  
methodNameInClass(?method,?accessor,?class),  
instanceVariableInClassChain(?accessor,?class),  
methodInProtocol(?method,accessing),  
reception(?joinpoint,?accessor,?args),  
withinClass(?joinpoint,?class)
```

Fragility

- 1.accessing protocol
- 2.selector corresponds to variable name

```
Person >> name  
  ^name  
  
Person name. anObject  
  name := anObject
```



```
Person >> getName  
  ^name  
  
Person >> setName: anObject  
  name := anObject
```



Synchronization

```
class(?class),  
methodNameInClass(?method,?accessor,?class),  
instanceVariableInClassChain(?accessor,?class),  
methodInProtocol(?method,accessing),  
reception(?joinpoint,?accessor,?args),  
withinClass(?joinpoint,?class)
```

Complexity

- 1.accessor: return instance var
- 2.mutator: assign instance var

```
Person>>name
```

```
  ^name
```

```
Person>>name: anObject
```

```
  name := anObject
```

Complexity

- 1.accessor: return instance var
- 2.mutator: assign instance var

```
Person>>name
```

```
^name
```

```
Person>>name: anObject
```

```
name := anObject
```

Synchronization

```
class(?class),  
methodNameInClass(?method,?accessor,?class),  
instanceVariableInClassChain(?accessor,?class),  
returnStatement(?method,variable(?var)),  
reception(?joinpoint,?accessor,?args),  
withinClass(?joinpoint,?class)
```

Complexity

- 1.accessor: return instance var
- 2.mutator: assign instance var

```
Person>>name
```

```
  ^name
```

```
Person>>name: anObject
```

```
  name := anObject
```

```
Person>>friends
```

```
  ^friends isNil
```

```
    ifTrue:[friends := Set new]
```

```
    ifFalse:[friends]
```

Synchronization

```
class(?class),  
methodNameInClass(?method,?accessor,?class),  
instanceVariableInClassChain(?accessor,?class),  
returnStatement(?method,variable(?var)),  
reception(?joinpoint,?accessor,?args),  
withinClass(?joinpoint,?class)
```


Complexity

- 1.accessor: return instance var
- 2.mutator: assign instance var

Person>>name

^name

Person>>name: anObject

name := anObject

Person>>friends

^friends isNil

ifTrue:[friends := Set new]

ifFalse:[friends]

Synchronization

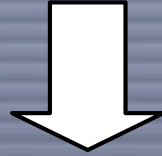
```
class(?class),  
methodNameInClass(?method,?accessor,?class),  
instanceVariableInClassChain(?accessor,?class),  
returnStatement(?method,variable(?var)),  
reception(?joinpoint,?accessor,?args),  
withinClass(?joinpoint,?class)
```

```
withinClass(?joinpoint,?class)
```

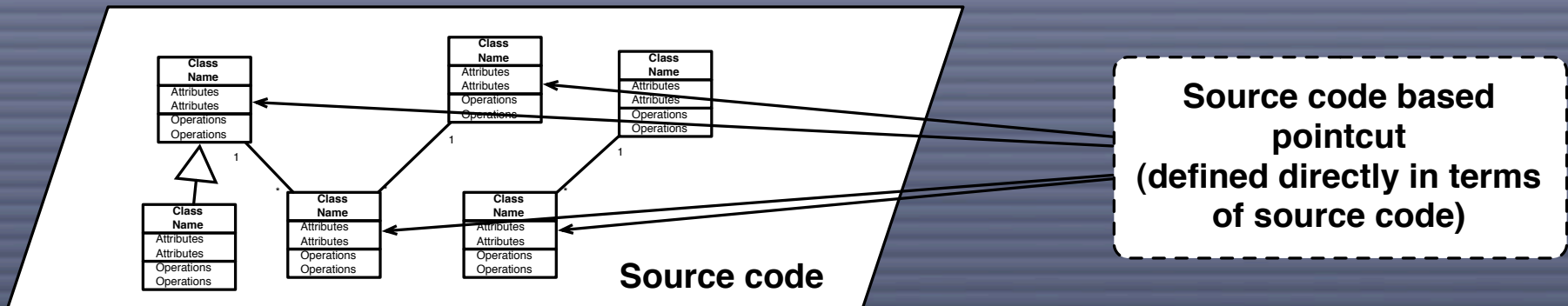
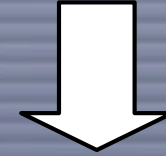
```
withinClass(?joinpoint,?class)
```

Problem Analysis

Base program developer

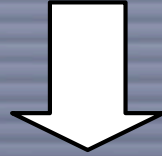


Aspect developer

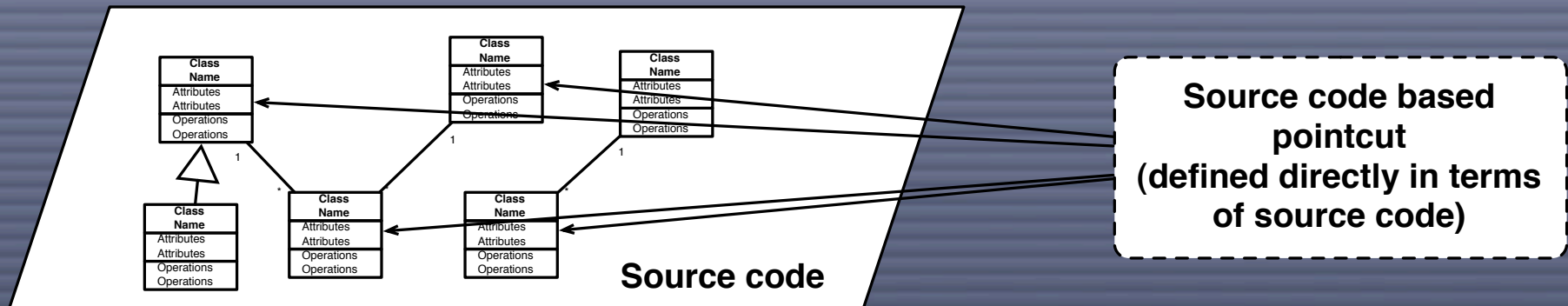
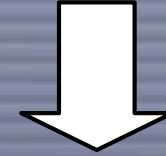


Problem Analysis

Base program developer



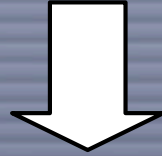
Aspect developer



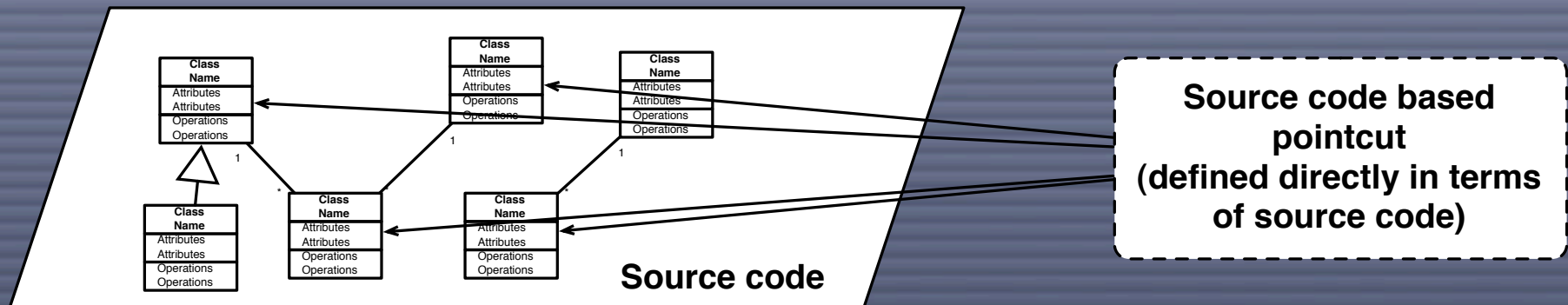
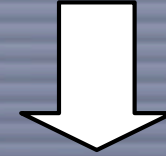
Tight coupling

Problem Analysis

Base program developer



Aspect developer

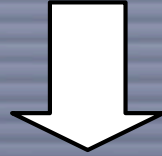


Tight coupling

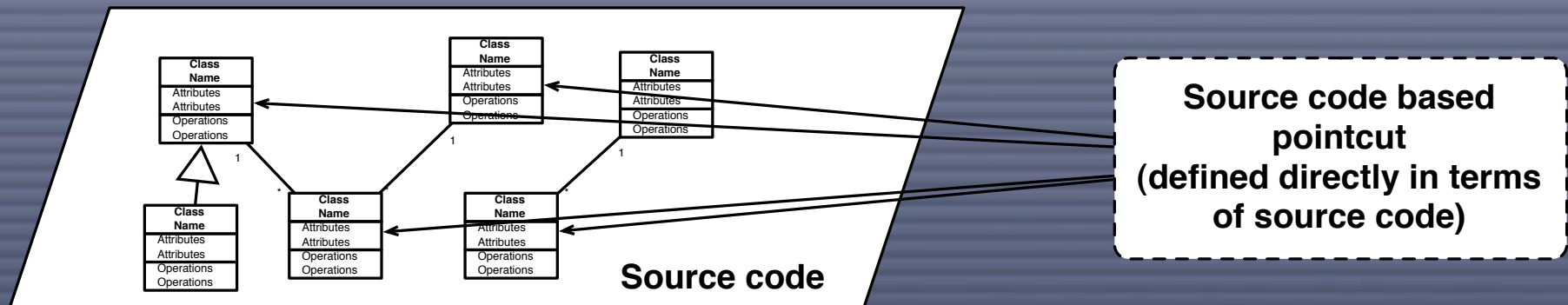
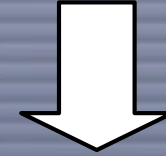
Complex

Problem Analysis

Base program developer



Aspect developer



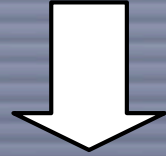
**Tight
coupling**

Complex

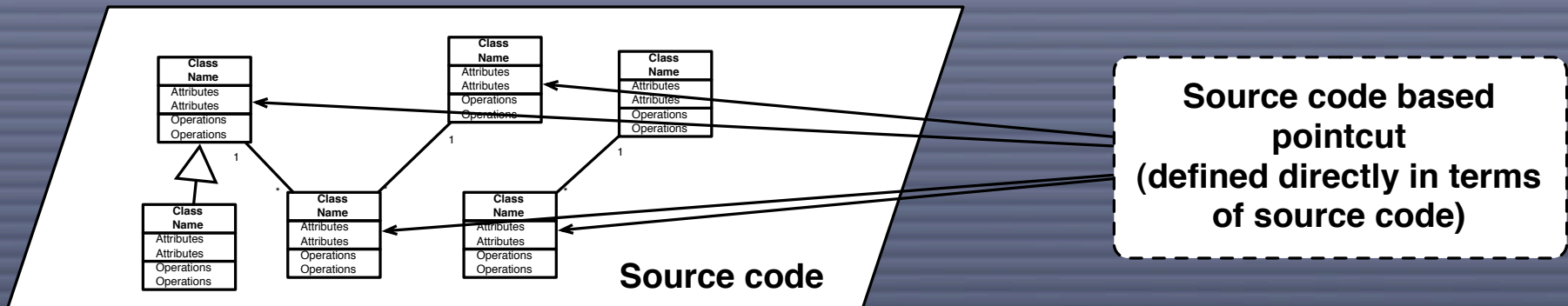
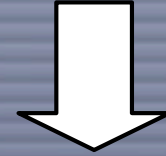
Fragile

Application-specific

Base program developer



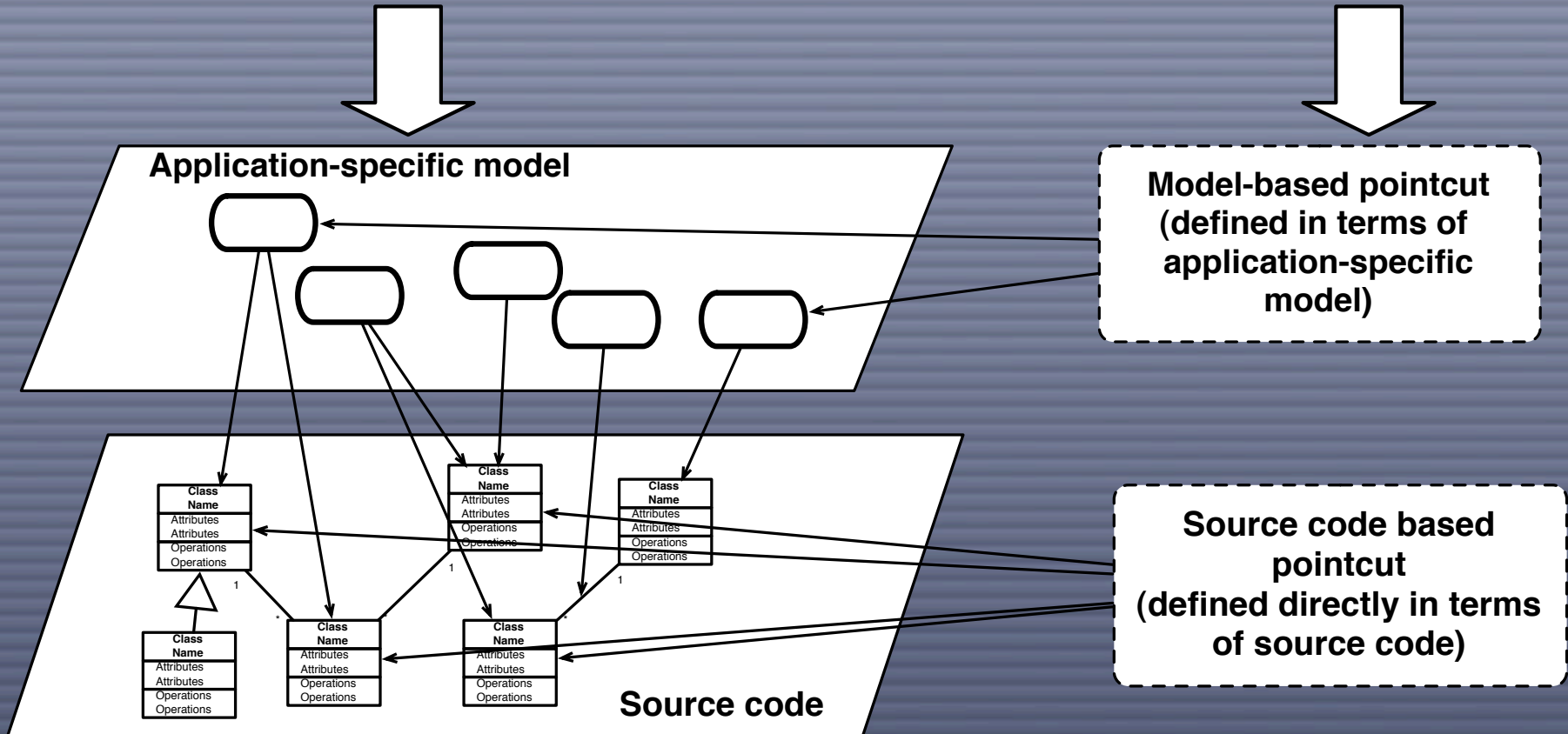
Aspect developer



Application-specific

Base program developer

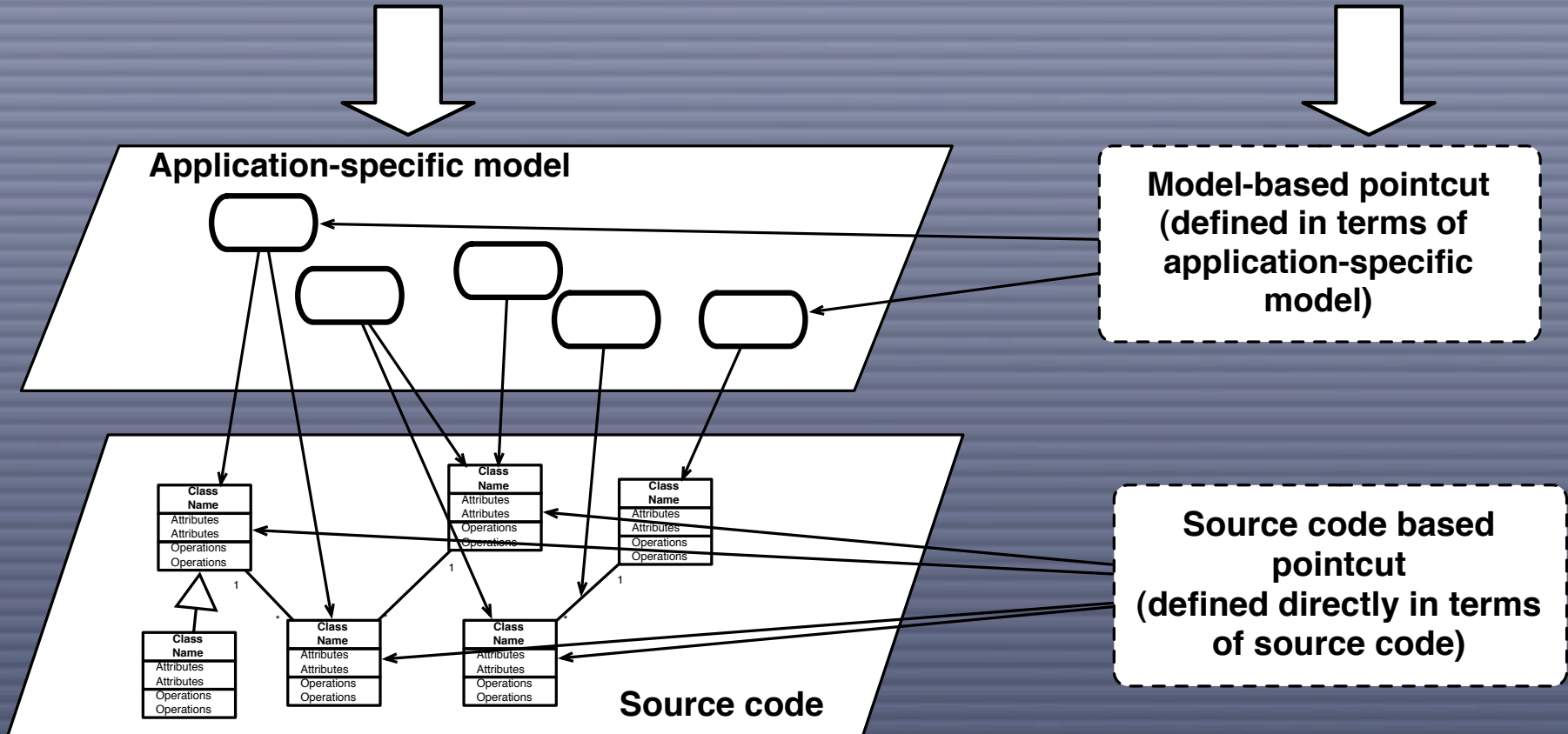
Aspect developer



Application-specific

Base program developer

Aspect developer

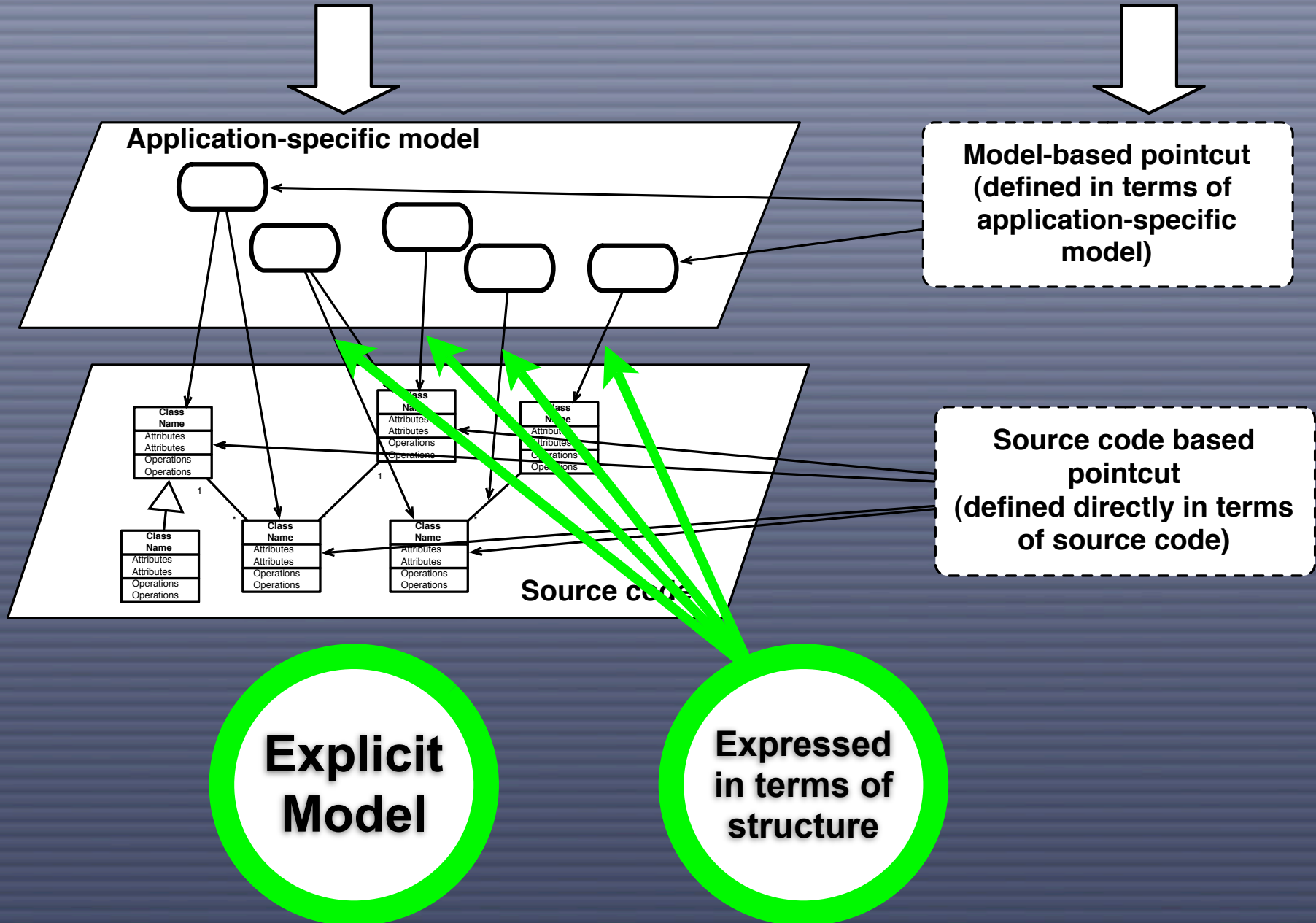


Explicit Model

Application-specific

Base program developer

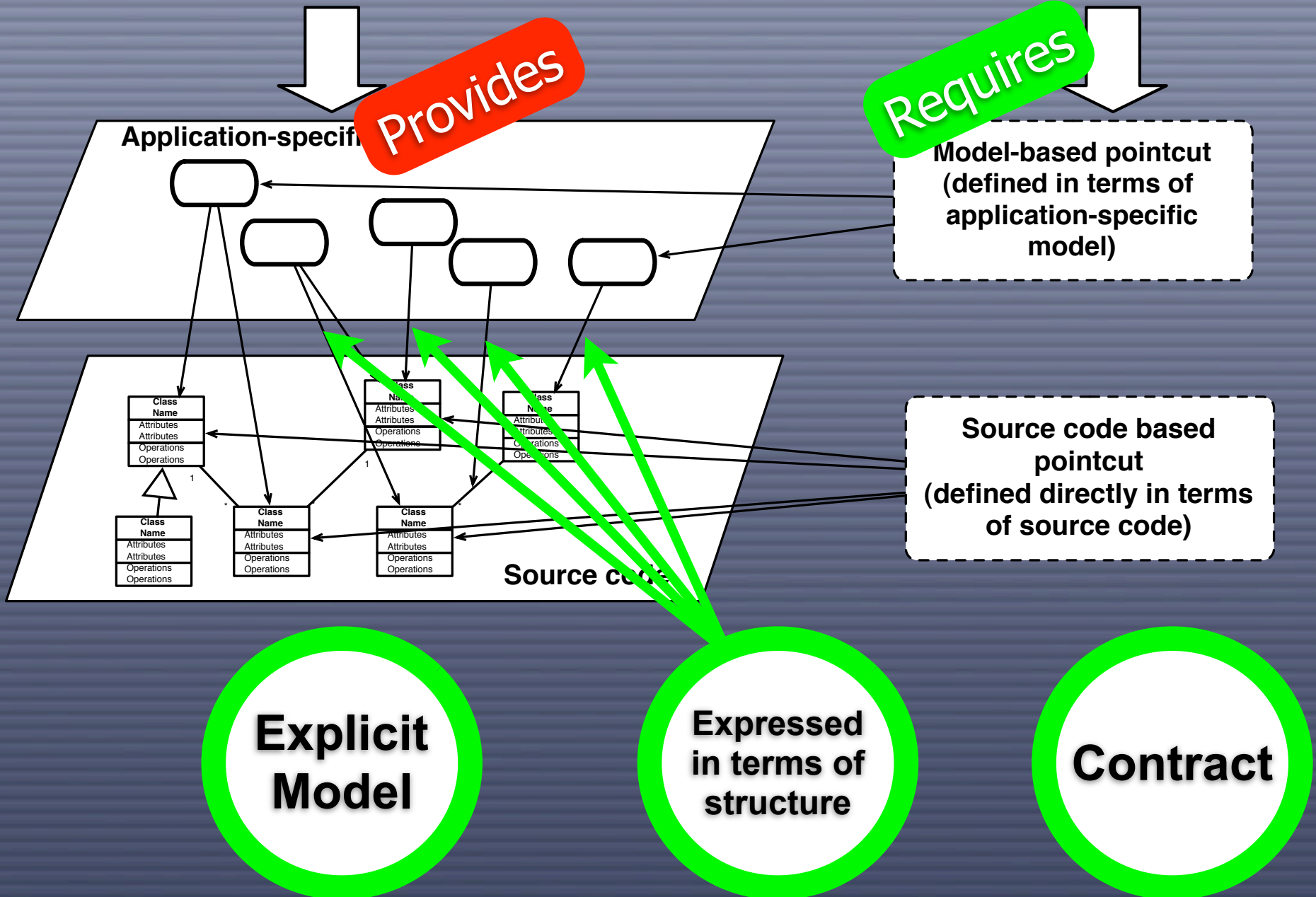
Aspect developer



Application-specific

Base program developer

Aspect developer



Application-specific pointcuts in AspectSOUL

Logic Pointcuts

- Extensible pointcut language

Application-specific model

- Keep in sync with code
- Logic representation
 - specialisation
 - parameters & unification

Application-specific pointcuts in AspectSOUL

Logic Pointcuts

- Extensible pointcut language

Application-specific model

- Keep in sync with code
- Logic representation
 - specialisation
 - parameters & unification

Application-specific pointcuts in AspectSOUL

Logic Pointcuts

- Extensible pointcut language

Application-specific model

- Keep in sync with code
- Logic representation
 - specialisation
 - parameters & unification



Managing the Evolution of Aspect-Oriented Software with Model-based Pointcuts

Andy Kellens, Kim Mens, Johan Brichau, Kris Gybels

In "Proceedings of the 20th European Conference on Object-Oriented Programming (ECOOP)", 2006

Accessors

Source

```
Person>>name
```

```
  ^name
```

```
Person>>name: anObject
```

```
  name := anObject
```

Pointcut

Model

Accessors

Source

```
Person>>name  
  ^name  
  
Person>>name: anObject  
  name := anObject
```

Pointcut

Synchronization

```
reception(?joinpoint,?selector,?args),  
accessor(?class,?selector,?var),  
withinClass(?joinpoint,?class)
```

Model

Accessors

Source

```
Person>>name
  ^name

Person>>name: anObject
  name := anObject
```

Synchronization

Pointcut

```
reception(?joinpoint,?selector,?args),
accessor(?class,?selector,?var),
withinClass(?joinpoint,?class)
```

Model

```
accessor(?class,?method,?varname) if
  class(?class),
  instanceVariableInClassChain(?varName,?class),
  methodWithNameInClass(?method,?varName,?class),
  methodInProtocol(?method,accessing),
  accessorForm(?method,?varname)
```

```
accessorForm(?method,?var) if
  returnStatement(?method,variable(?var))
```

Model specialisation

Source

```
Person>>name
```

```
  ^name
```

```
Person>>name: anObject
```

```
  name := anObject
```

Pointcut

Model

Model specialisation

Source

```
Person>>name
```

```
  ^name
```

```
Person>>name: anObject
```

```
  name := anObject
```

```
Person>>friends
```

```
  ^friends isNil
```

```
    ifTrue:[friends := Set new]
```

```
    ifFalse:[friends]
```

Pointcut

Model

Model specialisation

Source

```
Person>>name
```

```
  ^name
```

```
Person>>name: anObject
```

```
  name := anObject
```

```
Person>>friends
```

```
  ^friends isNil
```

```
    ifTrue:[friends := Set new]
```

```
    ifFalse:[friends]
```

Synchronization

Pointcut

```
reception(?joinpoint,?selector,?args),
```

```
accessor(?class,?selector,?var),
```

```
withinClass(?joinpoint,?class)
```

Model

Model specialisation

Source

```
Person>>name
  ^name
Person>>name: anObject
  name := anObject
```

```
Person>>friends
  ^friends isNil
  ifTrue:[friends := Set new]
  ifFalse:[friends]
```

Synchronization

Pointcut

```
reception(?joinpoint,?selector,?args),
accessor(?class,?selector,?var),
withinClass(?joinpoint,?class)
```

```
accessor(?class,?method,?varname) if
...
```

```
accessorForm(?method,?var) if
  returnStatement(?method,send(?var))
```

```
accessorForm(?method,?var) if
  returnStatement(?method,send(?check),<?true,?false>),
  nilCheckStatement(?check),
  statementsOfBlock(assign(?var,?varinit),?true),
  statementsOfBlock(<?var>,?false)
```

Model

Parameters & Unification

Parameters & Unification

Synchronization

```
reception(?joinpoint,?selector,?args),  
accessor(?class,?selector,?var),  
withinClass(?joinpoint,?class)
```


Parameters & Unification

Synchronization

```
reception(?joinpoint,?selector,?args),  
accessor(?class,?selector,?var),  
withinClass(?joinpoint,?class)
```

Synchronization

```
reception(?joinpoint,?selector,?args),  
accessor(Array,at:put:?,?var),  
withinClass(?joinpoint,?class)
```

Parameters & Unification

Synchronization

```
reception(?joinpoint,?selector,?args),  
accessor(?class,?selector,?var),  
withinClass(?joinpoint,?class)
```

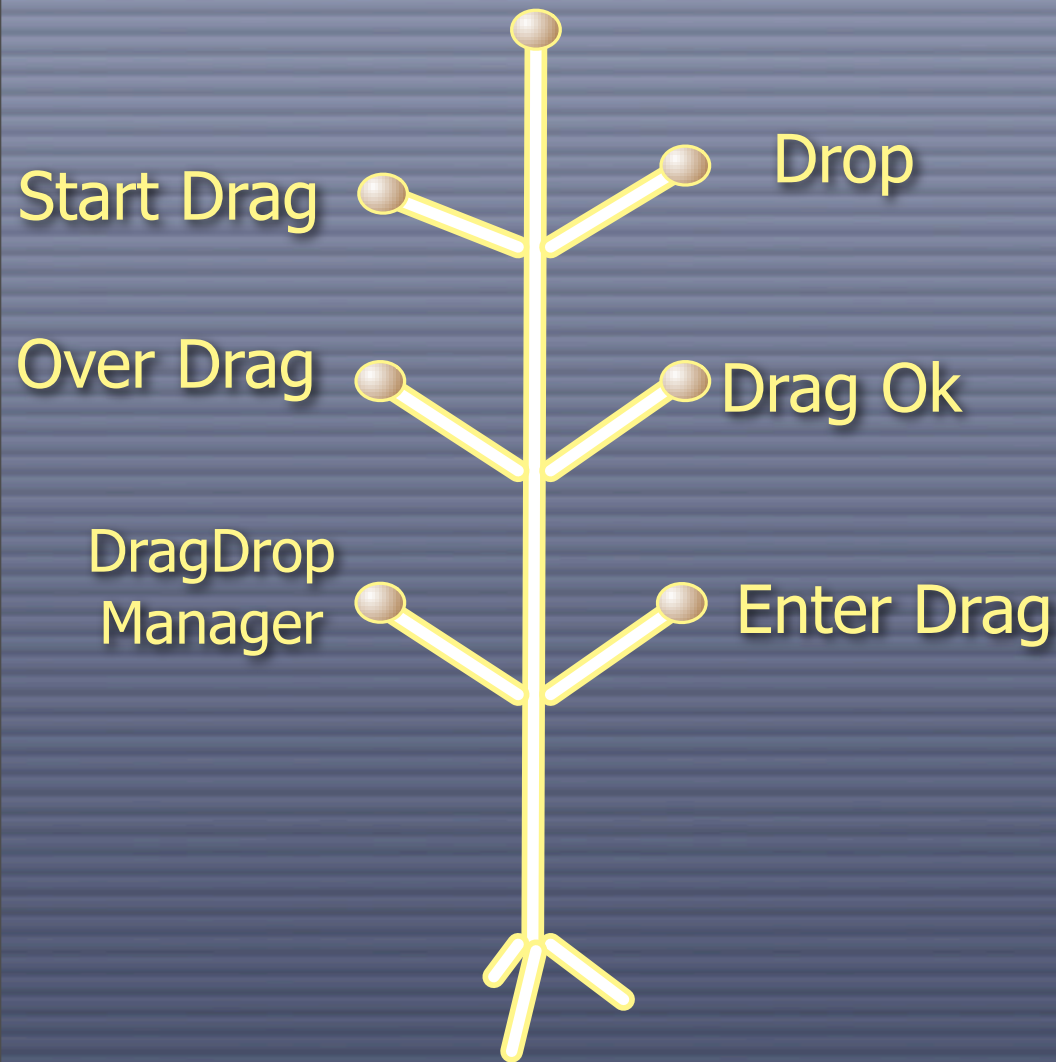
Synchronization

```
reception(?joinpoint,?selector,?args),  
accessor(Array,at:put:?,?var),  
withinClass(?joinpoint,?class)
```

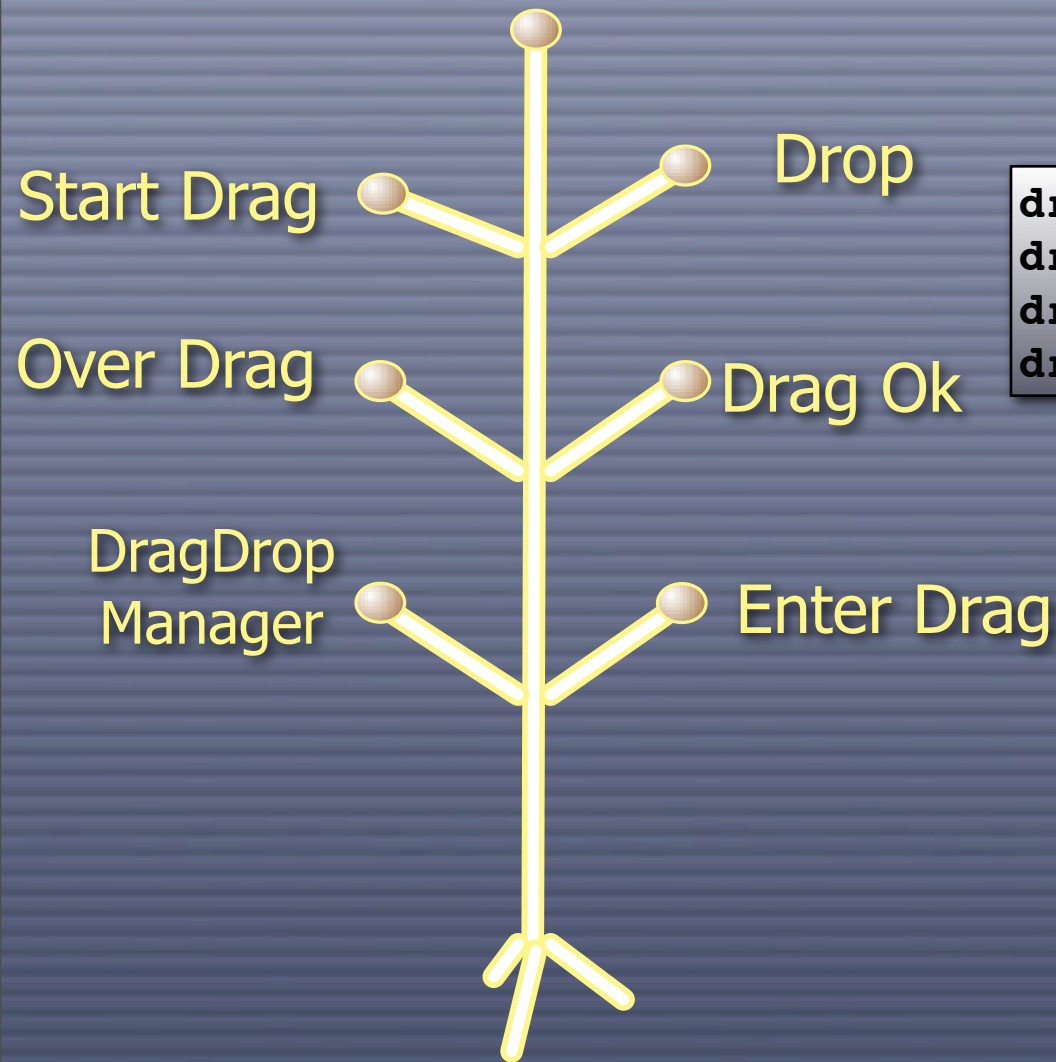
Synchronization

```
reception(?joinpoint,?selector,?args),  
accessor(?class,?selector,address),  
withinClass(?joinpoint,?class)
```

Drag & Drop

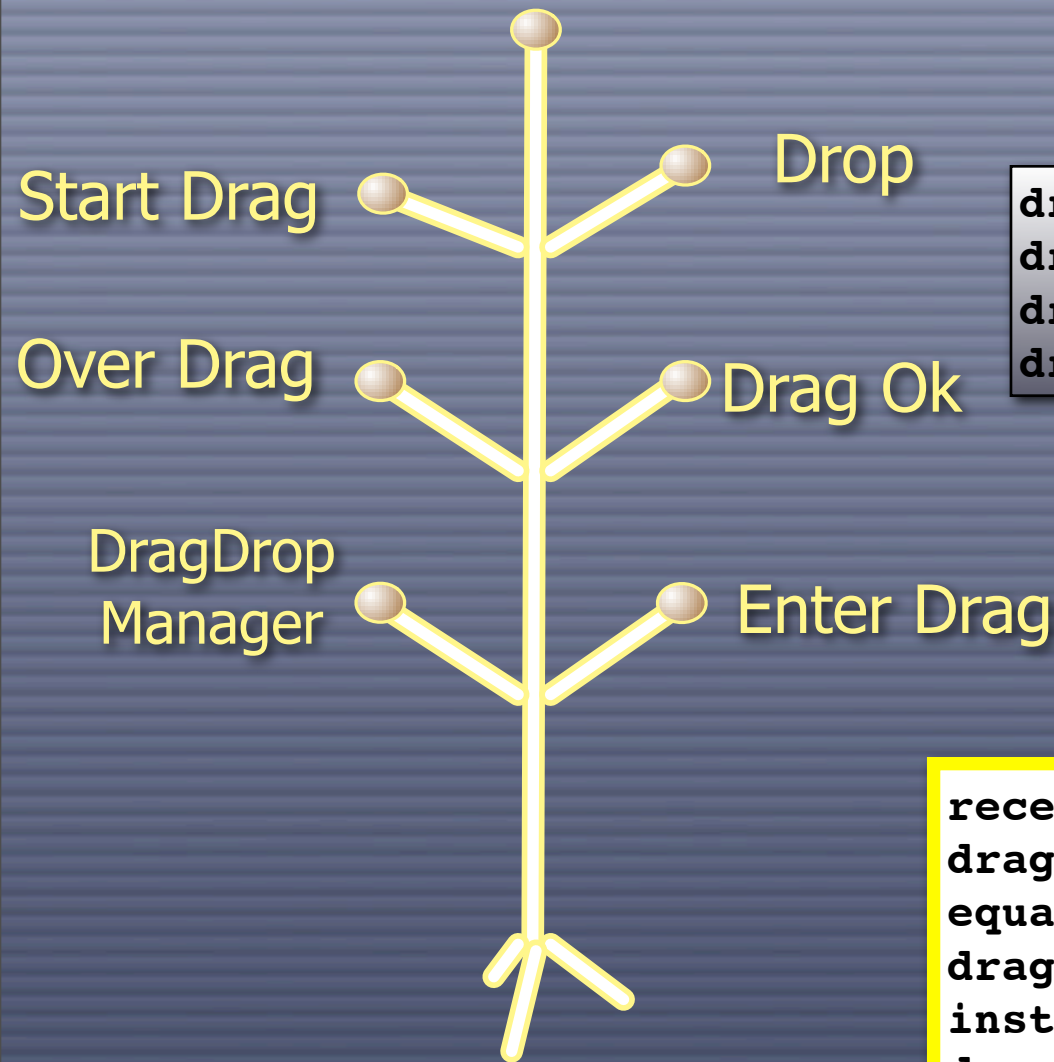


Drag & Drop



```
dragOkMethod(?class,?sel,?component)  
dragEnterMethod(?class,?sel,?component)  
dragSource(?dragdropmanager,?source)  
draggedObject(?dragdropmanager,?object)
```

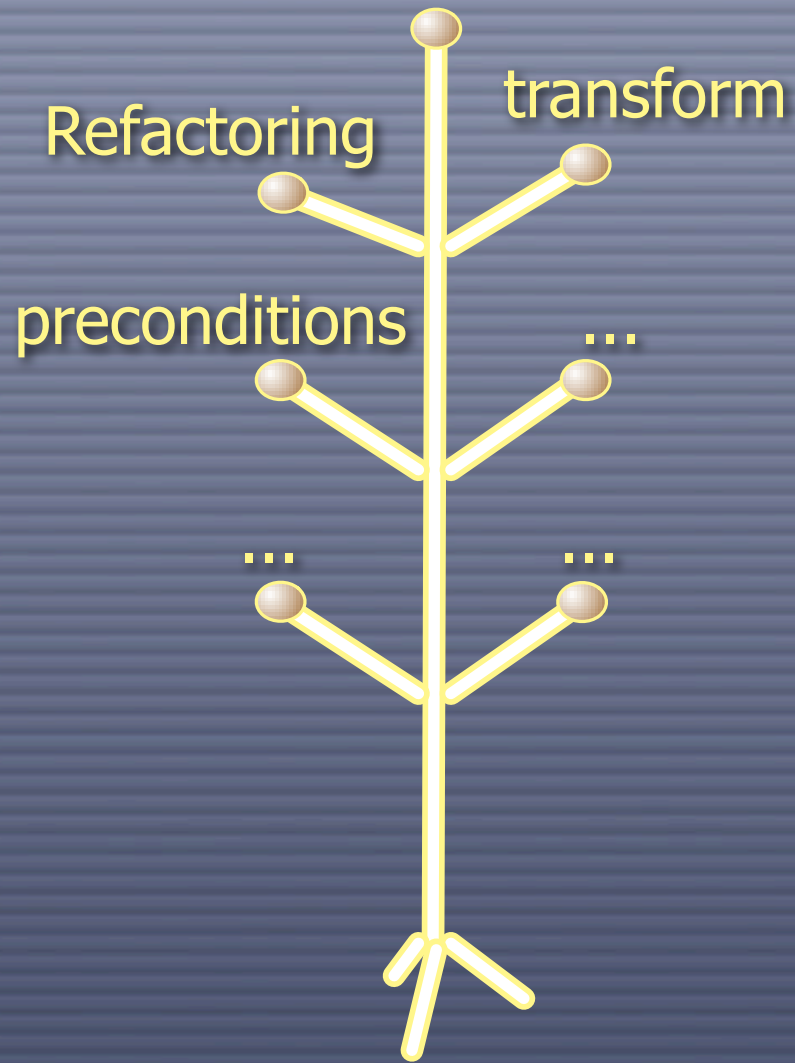
Drag & Drop



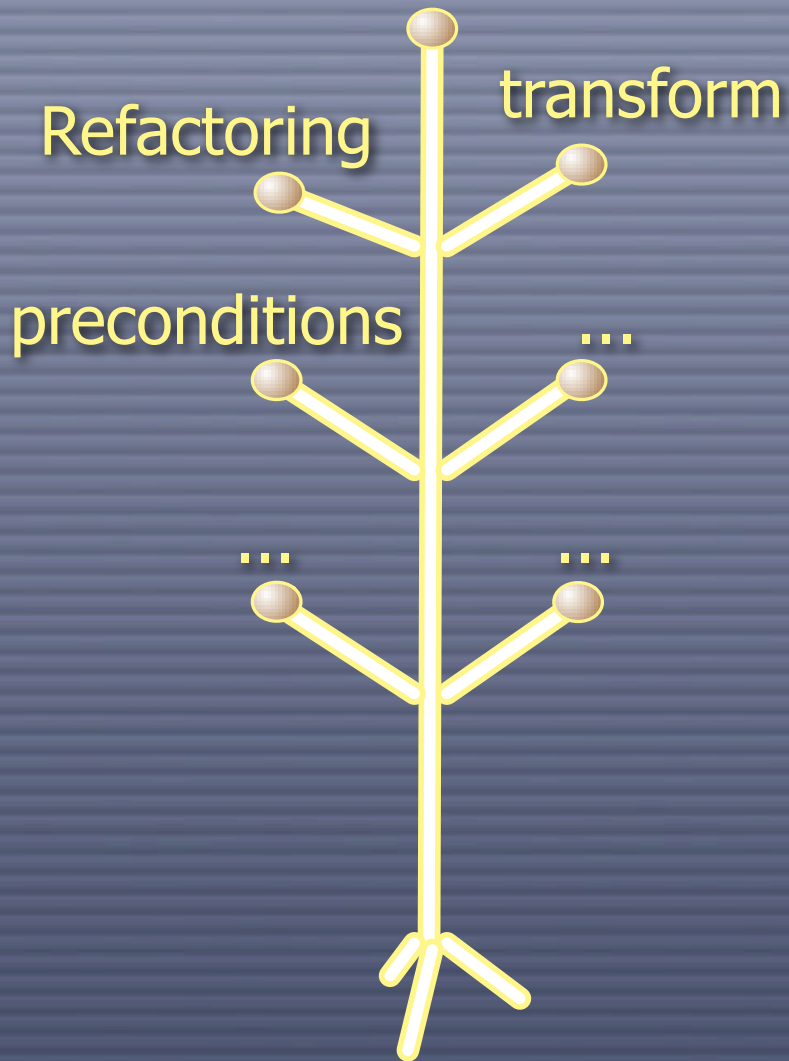
```
dragOkMethod(?class,?sel,?component)  
dragEnterMethod(?class,?sel,?component)  
dragSource(?dragdropmanager,?source)  
draggedObject(?dragdropmanager,?object)
```

```
reception(?jp,?sel,?args),  
dragEnterMethod(?class,?sel,?component),  
equals(?args,<?dragdropmanager>),  
dragSource(?dragdropmanager,?source),  
instanceOf(?source,FigureManager),  
draggedObject(?dragdropmanager,?object),  
instanceOf(?object,Line)
```

Refactoring

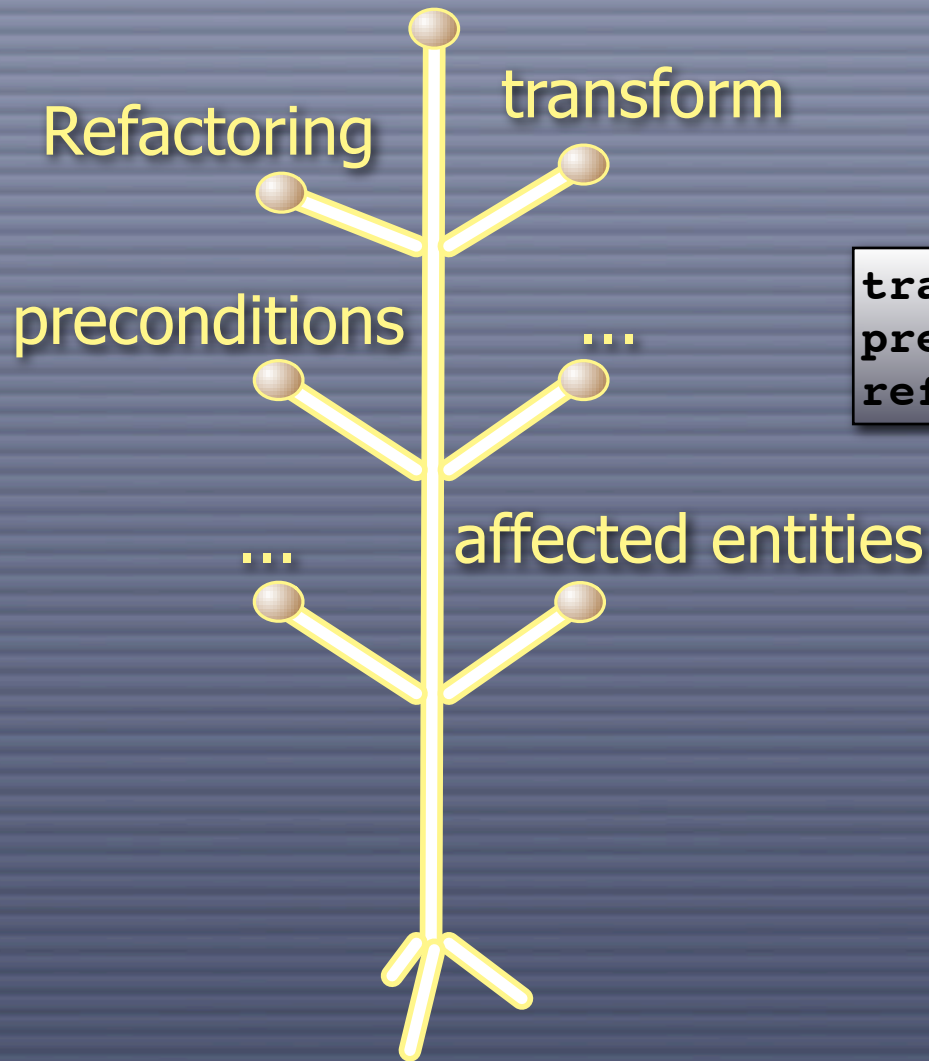


Refactoring



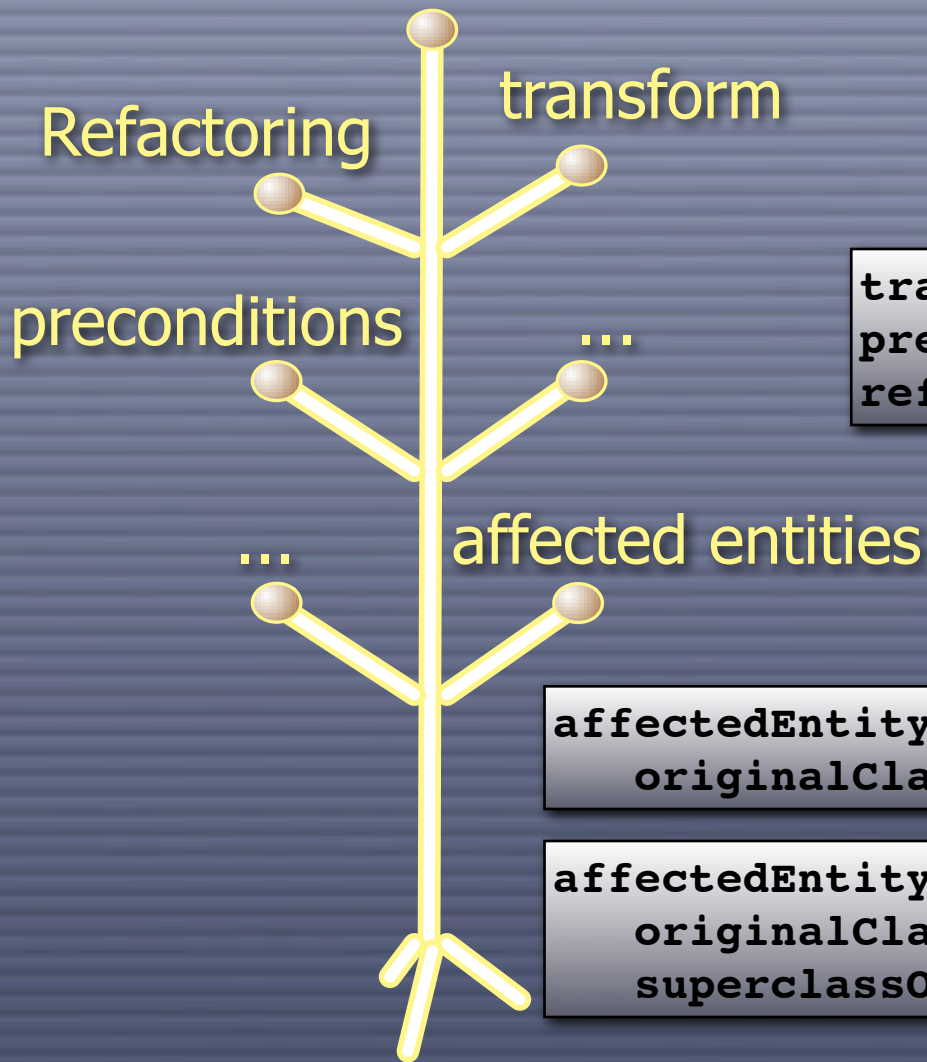
```
transformMethod(?class,?sel,?refactoring)  
preconditions(?refactoring,?preconditions)  
refactoring(?refactoring,?class,?method)
```

Refactoring



```
transformMethod(?class,?sel,?refactoring)  
preconditions(?refactoring,?preconditions)  
refactoring(?refactoring,?class,?method)
```


Refactoring



```
transformMethod(?class,?sel,?refactoring)  
preconditions(?refactoring,?preconditions)  
refactoring(?refactoring,?class,?method)
```

```
affectedEntity(?ref,[PushUpMethod],?input,?entity) if  
originalClassOfPushUpMethod(?input,?entity)
```

```
affectedEntity(?ref,[PushUpMethod],?input,?entity) if  
originalClassOfPushUpMethod(?input,?class),  
superclassOf(entity,?class)
```

Summary

- AspectS + Carma = AspectSOUL
 - Pointcut in terms of application-specific model
 - Extensible pointcut language
 - Logic language to express model
-
- Further integration of AspectS/Carma
 - Integration with support for fragile pointcuts

Questions



 More information:

 <http://prog.vub.ac.be/carma/>