Industrial Software Rejuvenation using Open-Source Parsers

Arjan Mooij (TNO-ESI)
Mabel Joy (TNO-ESI)
Gernot Eggen (Philips Healthcare)
Paul Janson (FEI Company)
Andrei Rădulescu (FEI Company)

Juventas, 19th April 2016
FEI Case
FEI Case: Calibration Procedures

- Calibrator started as a research tool

- And evolved into a part of the product

**Goal**: Redesign that preserves the valuable domain logic.

Electron microscope
FEI Case: Rejuvenation approach

Current design
- Models (parse tree)
- Parse (DGrok)
- Implementation (Delphi code)
- Core logic (per procedure) → 1 routine, ~1000 lines

New design
- Models (parse tree)
- Serialize (C#)
- Implementation (Delphi code)
- Core logic (per procedure) → Multiple subroutines

Transform (C#)
- Eliminate implementation details
- Highlight the structure (based on UI)

Manual migration
- Library with supporting routines
- General infrastructure

Manual migration
- Library with supporting routines
- General infrastructure
FEI Case: Eliminate implementation details

procedure TestBeamImages();
begin
  if (not InSimulation) or (chkDeepSimulation.checked) then
    begin
      pst := 'Executing AC beam-shift calibration.' + crlf;
      SetBeamPosition(0, 0);
      SetInstructions(pst + 'Acquiring image 1.');
      CheckResults.AcquireCodImage(1, FCodeAcq, Binning, FCodeAcq, StartX, FCodeAcq, EndX, FCodeAcq, StartY, FCodeAcq, EndY, FCodeAcq, ExposureTime), 'Failed to acquire image 1.';
      (now try estimated 0.2 * image size)
      SetBeamPosition(0.2 * CodImageSize * CalFactor * UncalibratedStemPixel, 0);
      SetInstructions(pst + 'Acquiring image 2.');
      CheckResults.AcquireCodImage(2, FCodeAcq, Binning, FCodeAcq, StartX, FCodeAcq, EndX, FCodeAcq, StartY, FCodeAcq, EndY, FCodeAcq, ExposureTime), 'Failed to acquire image 2.';
      (now try estimated 0.4 * image size)
      SetBeamPosition(0.4 * CodImageSize * CalFactor * UncalibratedStemPixel, 0);
      SetInstructions(pst + 'Acquiring image 3.');
      CheckResults.AcquireCodImage(3, FCodeAcq, Binning, FCodeAcq, StartX, FCodeAcq, EndX, FCodeAcq, StartY, FCodeAcq, EndY, FCodeAcq, ExposureTime), 'Failed to acquire image 3.';
      SetBeamPosition(0, 0);
      (check that the results for 0.1 and 0.2 * image size are consistent)
      SetInstructions(pst + 'Measuring shift between images 1 and 2.');
      CheckResults.FilteredCrossCorrelation(FImage[1], FImage[2], Shift1, 0, 0, false), 'Failed to save intermediate (modal distortion, trial)';
      SetInstructions(pst + 'Measuring shift between images 1 and 3.');
      CheckResults.FilteredCrossCorrelation(FImage[1], FImage[3], Shift2, 0, 0, false), 'Failed to determine suitable image shift'!
    end;
  if InSimulation then
    begin
      Shift1 := InitializeVector(198, 23);
      Shift2 := InitializeVector(399, 44);
    end;
  AddToLog(‘Trial measurement 1, vector ‘ + VecToStr(Shift1));
  AddToLog(‘Trial measurements 2, vector ‘ + VecToStr(Shift2));
  CheckResult(Veolength(Shift2) <> 0), 'Failed to determine suitable image shift';
end;

procedure TestBeamImages();
begin
  pst := 'Executing AC beam-shift calibration.' + crlf;
  FAquisition.SetBeamPosition(0, 0);
  FU1.SetInstructions(pst + 'Acquiring image 1.');
  FAquisition.AcquireCodImage(1, FAquisition.FCodeAcq);
  { now try estimated 0.2 * image size }
  FAquisition.SetBeamPosition(0.2 * CodImageSize * CalFactor * UncalibratedStemPixel, 0);
  FU1.SetInstructions(pst + 'Acquiring image 2.');
  FAquisition.AcquireCodImage(2, FAquisition.FCodeAcq);
  { now try estimated 0.4 * image size }
  FAquisition.SetBeamPosition(0.4 * CodImageSize * CalFactor * UncalibratedStemPixel, 0);
  FU1.SetInstructions(pst + 'Acquiring image 3.');
  FAquisition.AcquireCodImage(3, FAquisition.FCodeAcq);
  FAquisition.SetBeamPosition(0, 0);
  { check that the results for 0.1 and 0.2 * image size are consistent }
  FU1.SetInstructions(pst + 'Measuring shift between images 1 and 2.');
  FProcessing.FilteredCrossCorrelation(FImage[1], FImage[2], Shift1, 0, 0, false);
  FU1.SetInstructions(pst + 'Measuring shift between images 1 and 3.');
  FProcessing.FilteredCrossCorrelation(FImage[1], FImage[3], Shift1, 0, 0, false);
  Results.SetIntermediate(modal distortion, trial);
  FU1.CheckResult(VeoLength(Shift2) <> 0), 'Failed to determine suitable image shift';
end;
FEI Case: Highlight the structure (based on UI)

- Graph structure: visually nice, but yields non-local jumps in the code

- Three-level code structure, by outsourcing code fragments:
  - 1 top: overall structure (try-catch, if-then-else)
  - several flows: order of user interactions (WaitForNext)
  - many steps: computations and control (no user interaction)
FEI Case: Summary

• Automated transformations to reduce the amount of human work
  - Incrementally introduce more abstractions into the code
  - Possibly a stepping stone towards domain-specific models
  - New code structure is natural and understandable for the domain expert

• Automated code analysis to give insight to human developers
  - Variables: routines with read/write access, shadowing declarations
  - Routines: call graphs
  - Exceptions: exception handling structure
PHC Case
PHC Case: COM Removal

- Software is decomposed into components
  - using Microsoft COM technology

- Typical component design:

  - Presentation (COM → C++)
  - Functional (C++)
  - Abstraction/Fascade (C++ → COM)

**Goal**: Reduce amount of glue code and platform dependencies.

Interventional X-ray scanner
PHC Case: Merge a pair of COM components

Presentation (COM → C++)

Functional (C++)

Abstraction/Fascade (C++ → COM)

Presentation (COM → C++)

Functional (C++)

Abstraction/Fascade (C++ → COM)
PHC Case: Approach

- Model reasoning (Xtext+Xtend)
  - Glue model (Domain-specific)
  - Glue model (Domain-specific)
  - Glue model (Domain-specific)

- Model extraction (Java)
  - Parse tree model (AST for C++)
    - Parse tree model (AST for C++)

- Code parsing (Eclipse CDT)
  - Implementation (C++ code)
    - Implementation (C++ code)
    - Implementation (C++ code)

- Implementation (C++ code)
PHC Case: Extract glue models

```cpp
BOOL CASCExamEpxAdapter::ApplyExamData(
    IUnknown* pApplARDExaminationData,
    IID IID_IApplARDExaminationData)
{
    CXAGEN_TRACE_INTERNAL_CALL(ApplyExamData, (_T("")));

    INFRA_CHECK(NULL != m_pCASCExamEpxAdapterATL);

    IUnknown* pCallback = m_pCASCExamEpxAdapterATL->GetInterface();
    INFRA_CHECK(NULL != pCallback);

    HRESULT hR = GetStateController().SelectExam(
        pCallback, pApplARDExaminationData, IID_IApplARDExaminationData);

    CGenUtilities::ReleaseAndNull(pApplARDExaminationData);
    CGenUtilities::ReleaseAndNull(pCallback);

    return SUCCEEDED(hR);
}
```

source "CASCExamEpxAdapter".ApplyExamData(inout,in) -> target m_pIApplicationStateController.SelectExam

target#1 = m_pCASCExamEpxAdapterATL->GetInterface
target#2 = source#1
target#3 = source#2
PHC Case: Combine and simplify glue models

```
source "CASCApplicationStateControllerATL".EpXSelected(in,inout) -> target m_pIASCApplicationStateControllerEpXSelectedCB.EpXSelected
target#1 = source#1
target#2 = CComBSTR(source#2)

source "CASCExamEpXAdapterATL".EpXSelected(in,in) -> target m_pIASCExamEpXPartCBInterface.EpXSelected
decoupled
target#1 = source#1
target#2 = CString(source#2)

source "CCAASCApplicationStateControllerExamEpxCB".EpXSelected(in,inout)
if source#1
  -> target m_pIASCExamEpxCB.EpXDataApplied
else
  -> target m_pIASCExamEpxCB.ApplyEpXDataFailed
  target#1 = source#2
fi

source "CASCApplicationStateControllerATL".EpXSelected(in, inout)
if source#1
  then
  -> target "IASCExamEpxCB".EpXDataApplied
decoupled
else
  -> target "IASCExamEpxCB".ApplyEpXDataFailed
decoupled
  target#1 = source#2
fi
```
PHC Case: Summary

- Glue domain model that is more abstract than the parse tree (AST)
  - Glue code has limited functionality
  - Glue models are based on what is needed for the model reasoning

- 26 glue layers have been simplified into a very thin glue layer
  - Revealed some unexpected functionality inside the glue layers

- Refactoring other COM-related glue code (not isolated in layers)
  - Based on parse tree models (similar to the Calibration Procedures case)

- Tedious to put the two COM components into a single process
  - Due to the incremental removal, COM cannot be removed completely
Concluding remarks
Semi-automated techniques

Abstract Syntax Tree (AST)

(Extended) Visitor pattern

Brains

if _knob0 = nil then
    begin
        _knob1 := _class.Create;
        _knob2.Init(_string1, _string2);
        __stat1;
    end;

Code snippets with placeholders
Conclusions and further work

• Open-source parsers applied in two industrial rejuvenation cases:
  - Calibration procedures (Delphi) redesign while preserving domain logic
  - COM-related glue code (C++) reduce code and platform dependency

• Separate **valuable domain logic** from implementation details:
  - Limited functionality: extract valuable knowledge in domain models
  - Otherwise: eliminate implementation details from code

• Exploit the abstractions introduced by the original developers
  - File dependencies, macros/pre-processor, blank lines, code comments

• Further work:
  - Integrated frameworks for the basic rejuvenation infrastructure
  - Semi-automated extraction of valuable domain logic from legacy code
  - Interactive analysis techniques to get human insight in the legacy code