ECU Configuration Management Exam Solutions

Automotive Software Engineering

May 1, 2025

1 XML Configuration Management (20 pts)

1.1 Solution

```xml
<!-- ECU_Config.xml -->
<ECUConfig>
  <Version>1.0</Version>
  <Parameters>
    <Parameter name="EngineRPM" pin="PA5" type="Analog" unit="RPM"/>
    <Parameter name="CoolantTemp" pin="PB2" type="Analog" unit="C"/>
  </Parameters>
</ECUConfig>
```

```cpp
// ConfigManager.h
class ConfigManager {
public:
  bool loadConfig(const QString& filePath);
  bool saveConfig(const QString& filePath);

private:
  QDomDocument m_doc;
};
```

2 Physical-Functional I/O Mapping (25 pts)

2.1 Solution

```
// PinMappingWidget.cpp
void PinMappingWidget::loadFromXml(const QDomElement& root) {
  setColumnCount(3);
  setHorizontalHeaderLabels({"Physical Pin", "Functional Name
    ", "Type"});

  QDomNodeList params = root.elementsByTagName("Parameter");
  for(int i = 0; i < params.count(); i++) {
    QDomElement el = params.at(i).toElement();
    insertRow(i);
    setItem(i, 0, new QTableWidgetItem(el.attribute("pin")));
    setItem(i, 1, new QTableWidgetItem(el.attribute("name")))
    ;
    setItem(i, 2, new QTableWidgetItem(el.attribute("type")))
    ;
  }
}
```

3 Code Generation (15 pts)

3.1 Solution

```
// CodeGenerator.cpp
QString CodeGenerator::generateHeader(const QDomElement& root
    ) {
  QString code = "// Auto-generated\n#pragma once\n\n";

  QDomNodeList params = root.elementsByTagName("Parameter");
  for(int i = 0; i < params.count(); i++) {
    QDomElement el = params.at(i).toElement();
    code += QString("#define %1_PIN %2\n")
      .arg(el.attribute("name"))
      .arg(el.attribute("pin"));
  }

  return code;
}
```

4 HMI Preview (15 pts)

4.1 Solution

```
// HMIPreviewWidget.cpp
void HMIPreviewWidget::paintEvent(QPaintEvent*) {
  QPainter painter(this);

  // Draw RPM gauge
  painter.setBrush(Qt::white);
  painter.drawRect(10, 10, 50, 100);

  int gaugeHeight = qMin(m_rpm / 100, 100);
  painter.setBrush(Qt::red);
  painter.drawRect(10, 110 - gaugeHeight, 50, gaugeHeight);

  painter.drawText(70, 30, QString("RPM: %1").arg(m_rpm));
}
```

5 Diagnostic Data Management (15 pts)

5.1 Solution

```
// DiagnosticDialog.cpp
void DiagnosticDialog::addDtc() {
  QDomElement root = m_doc.documentElement();
  QDomElement dtc = m_doc.createElement("DTC");
  dtc.setAttribute("code", m_codeEdit->text());
  dtc.setAttribute("description", m_descEdit->toPlainText());
  root.appendChild(dtc);
}
```

6 Export Functionality (10 pts)

6.1 Solution

```
// Exporter.cpp
bool Exporter::exportToZip(const QString& configPath,
                           const QString& codePath,
                           const QString& outputPath) {
```

```
  // Implementation using QuaZip
  QFile configFile(configPath);
  QFile codeFile(codePath);

  QuaZip zip(outputPath);
  if(!zip.open(QuaZip::mdCreate))
    return false;

  // Add files to zip
  // ...

  zip.close();
  return true;
}
```

MainWindow Integration

```
// MainWindow.cpp
MainWindow::MainWindow(QWidget* parent) : QMainWindow(parent)
    {
  // Setup UI
  m_pinWidget = new PinMappingWidget(this);
  m_hmiWidget = new HMIPreviewWidget(this);

  // Connect signals/slots
  connect(ui->actionLoad, &QAction::triggered,
          this, &MainWindow::loadConfig);
}
```