Lowering the Total Cost of Ownership for Corporations doing Traditional z/OS Development, Maintenance and Support with Rational Developer for System z

Jon Sayles
RDz Technical Enablement
jsayles@us.ibm.com
Agenda

- IBM/Rational "Selective Offloading – Comprehensive Coverage"
- Additional Cost Savings
- Factors in Mainframe Chargeback Savings
- The Return-on-Investment Spreadsheet
- Working session

All performance data contained in this publication was obtained in the specific operating environment and under the conditions described in this publication and is presented as an illustration only.

Performance obtained in other operating environments may vary and customers should conduct their own testing.
With RAA/RDz/PD Tools, you get the benefit of MIPS reduced development for:

- COBOL, PL/I, C/C++, Assembler
- JCL
- BMS/MFS
- SQL, DL/I

- No offloading an entire z/OS run-time platform
  - No Data and file logical extracts
  - No downloading and configuring Resource Definitions
  - No worries about called utilities or Assembler
  - In fact, you do not have to offload any artifacts that are not in the current work requirement set

- No emulation
RDz – Project/Work Request Coverage

- **UI Modernization**
  - Rational Developer for System z
  - Rational Asset Analyzer
  - Problem Determination Tools

- **SOA/XML Web Services Modernization**
  - **Traditional z/OS Development**
    - COBOL, PL/I, HLASM
    - CICS, IMS, DB2
  - **z/OS Application Production Support**
  - **Traditional z/OS Maintenance and Enhancements**
    - COBOL, PL/I, Assembler
    - CICS, IMS, DB2
RDz – Integrated Lifecycle Coverage

Build Process**
- Rational Team Concert
- ClearCase for z

System, QA Acceptance Test
- PD Tools - Code Coverage
- Rational Function Tester for z***
- Rational Performance Tester for z***

Integration Test
- Rational Developer for System z
- Problem Determination Tools

Edit – Compile – Unit Test
- Rational Developer for System z
- Problem Determination Tools

Project and Program Analysis, Scoping, Understanding
- Rational Asset Analyzer/Rational Developer for System z

** Beyond the scope of today's presentation

Build Process**
- Rational Team Concert
- ClearCase for z

System, QA Acceptance Test
- PD Tools - Code Coverage
- Rational Function Tester for z***
- Rational Performance Tester for z***

Integration Test
- Rational Developer for System z
- Problem Determination Tools

Edit – Compile – Unit Test
- Rational Developer for System z
- Problem Determination Tools

Project and Program Analysis, Scoping, Understanding
- Rational Asset Analyzer/Rational Developer for System z

** Beyond the scope of today's presentation
Additional RAA/RDz Development Savings

- **Productivity Benefits:**
  - Even for experienced TSO veterans, but especially for new PC-aware developers:
    - See (view) substantially more code
    - Integrated analyze/edit/compile/debug/upload/download – access to z/Resources
    - Instantaneous response time during develop and test
      - Function/Enter keys do not interrupt z/OS CPU
    - 40+ (over forty) discrete edit/test/debug productivity features available with RDz:
      - Not available using ISPF
      - And only a handful of ISPF features not available on RDz

- **Web Services**
  - RDz provides wizards to automate service creation:
    - Bottom-up:
      - From a CICS program COMMAREA and an IMS IO-AREA
    - Top-down:
      - From a WSDL (RDz generates call-able, "wrapper" programs)
    - Both Bottom-up/Top-down
  - This takes SOA development for traditional mainframe applications from the Stone Age to the 21st Century
    - Anecdotal evidence suggests that doing Web Service development using RDz provides several magnitudes of order productivity increases
What Benefits Will You Get With RAA?

- Save costs for:
  - Contracts and bids
  - Project estimation
  - Project management
  - Analysis
  - Code understanding
  - Development
  - Maintenance
  - Testing and Bug-Fixes
  - Documentation

- Offload TSO/ISPF code analysis:
  - TSO/ISPF 3.4 Dataset searches
  - Edit session internal program "Find commands"
  - File browsing
  - Program compiles:
    - Listings/X-Refs, etc.

- During project estimation/code analysis**
  - MIPS savings from: 30% → 60%
  - Productivity increases for analyzing code from: 30% → 60%

** Reported by RAA users
What Kinds of Mainframe Chargeback Savings Can You Get With RDz?

Three Kinds of Savings**

1. Mainframe compile savings:
   - Batch compiles
   - TSO/Foreground compiles
   - Both include the CICS and DB2 pre-processors
   - Compile cost reduction: ~80%
   - Compile productivity savings: ~25%

2. TSO/CPU Savings - Per user/Per session
   - CPU Costs
   - TSO Session Connect time
   - DASD/Virtual Storage/Swap Space
   - Combined savings: 35% → 75%

3. Productivity gains
   - Analysis (assuming RAA): 20% → 50%
   - Edit: 10% → 25%
   - Debug: 10% → 20%

** Reductions based on RDz customer experiences
Mainframe Compile Savings

- **Reduced number of Compiles** –
  - CPU cycles
  - EXCP

- **RDz provides five levels of reduced compile costs:**
  - Content Assist
    - Code completion – significantly reduces typos
  - Validation
    - Interactive syntax checking – catches typos as they occur
  - Local Syntax Check
    - Offloaded compile
  - Remote Compile
    - z/OS compile without creating binaries – significantly less expensive than…
  - Full compile
    - Enterprise compiler
Mainframe TSO Session Savings

- **TSO/ISPF Development:**
  - Connect/Disconnect/Reconnect (and all ISPF Dataset Allocation/De-Allocation)
  - Browse
  - Edit
  - Search (3.4)
  - Dataset Management
  - Job submission and review
  - Unit testing – PD Tools, Xpeditor, etc.
  - Integration testing

- **DB2/SQL Testing**
  - SPUFI/QMF:
    - SQL statement testing
    - Test data management

- **CICS Testing**

- **IMS Testing**

How are TSO savings possible?
Every Enter and every PF-Key pressed while in a TSO session results in CPU cycles – which end up as charge-back costs***

See Slide Notes for Details***
Key to Lowering TSO/Session CPU Costs

- For those development activities that don't require you to offload your z/OS development sub-system, use RAA/RDz and PD Tools to minimize TSO access and activity.
What Cost-Saving Numbers Do We Have?

- **Three kinds:**
  1. IBM internal study of a "trivial code maintenance" project
     - Detailed SMF data
     - Comparison between TSO/ISPF and RDz – for TSO CPU cycles (translated to Service Units and MIPS)
     - What is important is the relative savings factor (% MIPS reduction) for this workload
  2. Anecdotal data from customers
  3. Your data (and specific inputs)
     - The detailed part of this session
     - You should have with you:
       - Cost per/developer
       - Cost per/batch compile
       - Cost per TSO Session
     - With this, we can predict RDz MIP savings
## IBM "Trivial Edit" Usage Pattern Benchmark – Scenario Details

<table>
<thead>
<tr>
<th>RDz – 30 minute session</th>
<th>TSO/ISPF – 30 minute session</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Connect to RDz MOP system</td>
<td>Logon to TSO</td>
</tr>
<tr>
<td>- Create a remote project and subproject</td>
<td>Enter ISPF</td>
</tr>
<tr>
<td>- Edit a remote program</td>
<td>ISPF Edit Session</td>
</tr>
<tr>
<td>- Use content assist to add new COBOL and CICS statements.</td>
<td>Manually type statements</td>
</tr>
<tr>
<td>- Open a remote copy member</td>
<td>Split Screen – open and edit a copybook</td>
</tr>
<tr>
<td>- Put an error in the program</td>
<td>Edit (put in a syntax error op purpose)</td>
</tr>
<tr>
<td>- Local Syntax Check with Refresh dependencies, with 2 dependant files</td>
<td>Submit a batch compile – Split screen – go to Outlist, View results</td>
</tr>
<tr>
<td>- Local Syntax check again</td>
<td>Submit a batch compile – Split screen – go to Outlist, View results</td>
</tr>
<tr>
<td>- Local syntax check again ➔ no more errors</td>
<td>Submit a batch compile – Split screen – go to Outlist, View results</td>
</tr>
<tr>
<td>- From the remote source program, generate the JCL to Compile and Link</td>
<td>Edit the Compile/Link/GO JCL</td>
</tr>
<tr>
<td>- Edit the JCL</td>
<td>ISPF Edit</td>
</tr>
<tr>
<td>- Submit the JCL</td>
<td>Submit</td>
</tr>
<tr>
<td>- View the Job Output – purge -- JCL error , dataset not found</td>
<td>View the results – Purge the JES Queue (JCL Error)</td>
</tr>
<tr>
<td>- Modify the program properties group to change the dataset name</td>
<td>Edit the JCL – change the DD card</td>
</tr>
<tr>
<td>- Submit the JCL</td>
<td>Submit the job</td>
</tr>
<tr>
<td>- Open the Job Monitor</td>
<td>Navigate to the Outlist facility</td>
</tr>
<tr>
<td>- View the Job Output -- flush -- JCL error, another dataset not found again</td>
<td>View the results – another JCL error – Purge the JES Queue (JCL Error)</td>
</tr>
<tr>
<td>- Edit and Submit the JCL again</td>
<td>Edit JCL – change DD card Submit the Job again</td>
</tr>
<tr>
<td>- Open the Job Monitor</td>
<td>Navigate to the Outlist facility</td>
</tr>
<tr>
<td>- View the Job Output – find that the JOB ended with RC=0</td>
<td>Browse – find the JOB ended with RC=0</td>
</tr>
<tr>
<td>- From the remote connection, open the Host Connection emulator to connect to CICS</td>
<td>Leave TSO – Logon to CICS Subsystem</td>
</tr>
<tr>
<td>- CADP to define the transaction to debug;</td>
<td>CADP to define the transaction to debug;</td>
</tr>
<tr>
<td>The 30 minutes window raised , disconnect from RDz</td>
<td>Logoff CICS</td>
</tr>
</tbody>
</table>
IBM "Trivial Edit" Usage Pattern Benchmark – ROI Calculations

**Savings:**

5,620,300 Service Units/day ***

**CPU Savings:** 42%

*** Based on 10 developers, working 7 hours/day, using RDz instead of TSO/ISPF

**Details:**

- **TSO Session:**
  - CPU = 191,842 SU/per user/per hour
- **RDz Development:**
  - CPU = 111,552 SU/per user/per hour
- **Savings:** 80,290 Service Units/per user

*** See Notes
IBM "Trivial Edit" Usage Pattern Benchmark – Additional Benefits

- Note that the previous benchmark and ROI calculations:
  - Are based on a "Worst Case" usage scenario for RDz
  - Are minus a number of additional returns on investment:
    1. RAA cost savings (substantial)
    2. PD Tools cost savings (substantial savings in the acquisition and license fees)
    3. Productivity Benefits:
      - Assume 
       保守的 benefits broken out by lifecycle phase:

<table>
<thead>
<tr>
<th>Developer time spent in:</th>
<th>Developer time improvements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis/Design/Understanding</td>
<td>35.00%</td>
</tr>
<tr>
<td>Editing</td>
<td>20.00%</td>
</tr>
<tr>
<td>Compile</td>
<td>5.00%</td>
</tr>
<tr>
<td>Debug</td>
<td>15.00%</td>
</tr>
<tr>
<td>Other</td>
<td>25.00%</td>
</tr>
<tr>
<td>Overall Productivity Increase</td>
<td>100.00%</td>
</tr>
</tbody>
</table>

- What does that all add up to?
Anecdotal Data From Customers

- **Large bank in Germany** did an even more precise and detailed study (as on the previous slides) but with a different use case.
  - Benchmark results:
    - 75% Percent CPU Savings
    - With 100 Developers - €3,600,000 savings/Annually ($4,160,000 U.S.)
- **Other European Banks benchmarks and feedback – With RDz:**
  - Reduced TSO consumption by as much as 80%
  - Saved 2 Euros an hour/Per Developer using RDz versus the mainframe
- **U.S. Telecomm company – With RDz:**
  - Reduced costs by more than US$75,000 on a single project
- **European Services Provider** – estimates that by rolling RDz out to 600 users will save ~ €74,000,000 over five years ($102,490,000 U.S.)
- **Financial services provider in Sweden:**
  - Saves approximately **US $1.6 million** on licensing costs & improved ABEND analyses, file alterations and application debugging processes when they replaced existing debugging software with PD Tools
  - Reduced 90% of time spent tuning programs and discovering performance bottlenecks when they implemented Application Performance Analyzer to help diagnose issues in IMS & DB2
Can we guarantee these MIPS reductions?

- Sorry – but:
  - Yours might be more 😊
  - Yours might be equal-to 😊
  - Yours might be less 😐
  - It depends on:
    - The kind of workflow and tools usage:
      - The more analysis/edit/compile/unit test - the bigger the benefits
    - Your developers' willingness to adopt a better development model:
      - Old model:
        - Do some analysis (usually incomplete and cursory)
        - Edit
        - Compile
        - Run the integration stream and verify results
        - Repeat
      - The right – or Best Practices development model:
        - Deep analysis (offline – in less time and with vastly improved results with RDz or RAAi)
        - Edit/Compile using RDz (offline)
        - Repeat (also offline)
        - Compile and integration test using RDz/PD Tools on the host – far fewer test runs
How to Calculate Your Own Savings

- Time to work through the ROI spreadsheet

- Enterable fields are in: [Legend]
  - Blue
  - Yellow

- Gray fields are spreadsheet computations – based solely on:
  - Your enter-able values
  - That's it… no "secret sauce"

- The following slides describe the values and formulae in the ROI calculator
### ROI Spreadsheet – 1 of 4 (Note – values not specific to any customer)

<table>
<thead>
<tr>
<th></th>
<th>Year 1</th>
<th>Year 2</th>
<th>Year 3</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Developers</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cost of one hour of mainframe CPU</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cost of Developer (yearly) (daily)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cost of 1 TSO per month</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cost of 1 TSO per day per developer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cost of Compiles per day</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of compiles per day per developer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average Cost of 1 Compile</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Developers needed in setup phase</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Developers needed each year for admin/mgmt</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

|                           |        |        |        |       |
| Cost of each License      | $5,253.00 |        |        |       |
| Yearly cost of license PPA contract |        |        |        | $1,050.60 |
| Working days in a year    |        |        |        | 220   |
| First year rate of return (<100%) |        |        |        | 80.0% |

**Notes:**
- IBM Sales Team will give you the: Cost of each license and Yearly cost of license PPA contract
- These are conservative TSO costs (in this screen capture) – based on world-wide corporate statistics
- Your values for:
  - Developer costs – fully loaded
  - TSO costs/Compile costs
  - Setup and administration conservative number
## ROI Spreadsheet – 2 of 4

### Cost of Initial Services

<table>
<thead>
<tr>
<th>Year</th>
<th>Cost</th>
<th>Days per Developer</th>
<th>Developers per Training</th>
<th>Cost of One Day Trainer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$50,000.00</td>
<td>14 days</td>
<td>2</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>$0.00</td>
<td>0 days</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$0.00</td>
<td>0 days</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>$50,000.00</strong></td>
<td><strong>14 days</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Direct Support

<table>
<thead>
<tr>
<th>Year</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>0.50</td>
</tr>
<tr>
<td>3</td>
<td>0.50</td>
</tr>
</tbody>
</table>

### HW Growth

<table>
<thead>
<tr>
<th>Year</th>
<th>Cost</th>
<th>Additional HW Cost per Developer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$0.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>2</td>
<td>$0.00</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$0.00</td>
<td></td>
</tr>
</tbody>
</table>

### Developers' Time Spent In:

<table>
<thead>
<tr>
<th>Task</th>
<th>Developer Time</th>
<th>Productivity Improvements</th>
<th>Average Reduced Num. of Compiles</th>
<th>Average TSO Sessions Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis/Design/Understanding</td>
<td>35.00%</td>
<td>42.00%</td>
<td>80%</td>
<td>40%</td>
</tr>
<tr>
<td>Editing</td>
<td>20.00%</td>
<td>28.00%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compile</td>
<td>5.00%</td>
<td>5.00%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Debug</td>
<td>15.00%</td>
<td>5.00%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td>25.00%</td>
<td>0.00%</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Overall</strong></td>
<td>100.00%</td>
<td>21.30%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Education can be zeroed out, if choosing RDz Distance Learning**
- **Productivity improvements are based on:**
  - Customer feedback
  - IBM's laboratory research (measured against green-screen development tools)
- **Percentages for reduced num. of compiles, Average TSO sessions reduction and productivity are conservative**
- **For hardware, assume that you need a fast processor, 2 GIGS/RAM/10 GIGS of free disk space**
### ROI Spreadsheet – 3 of 4

**Costs**

<table>
<thead>
<tr>
<th>Costs</th>
<th>70</th>
<th>0</th>
<th>0</th>
<th>70</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software</td>
<td>$367,710</td>
<td>$0</td>
<td>$0</td>
<td>$367,710</td>
</tr>
<tr>
<td>Subscription</td>
<td>$73,542</td>
<td>$73,542</td>
<td>$73,542</td>
<td></td>
</tr>
<tr>
<td>Education</td>
<td>$50,000</td>
<td>$0</td>
<td>$0</td>
<td>$50,000</td>
</tr>
<tr>
<td>Setup/Admin Staff</td>
<td>$120,000</td>
<td>$0</td>
<td>$0</td>
<td>$120,000</td>
</tr>
<tr>
<td>Support Staff</td>
<td>$0</td>
<td>$60,000</td>
<td>$60,000</td>
<td>$60,000</td>
</tr>
<tr>
<td>Hardware</td>
<td>$0</td>
<td>$0</td>
<td>$0</td>
<td>$0</td>
</tr>
<tr>
<td><strong>Total Costs</strong></td>
<td>$537,710</td>
<td>$133,542</td>
<td>$133,542</td>
<td>$60,000</td>
</tr>
</tbody>
</table>

**Justification**

<table>
<thead>
<tr>
<th>Justification</th>
<th>70</th>
<th>0</th>
<th>0</th>
<th>70</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compile CPU Savings</td>
<td>$325,248</td>
<td>$325,248</td>
<td>$325,248</td>
<td>$325,248</td>
</tr>
<tr>
<td>TSO CPU Savings</td>
<td>$98,560</td>
<td>$98,560</td>
<td>$98,560</td>
<td>$98,560</td>
</tr>
<tr>
<td>Productivity</td>
<td>$1,431,360</td>
<td>$1,789,200</td>
<td>$1,789,200</td>
<td>$1,789,200</td>
</tr>
<tr>
<td><strong>Total Justification</strong></td>
<td>$1,855,168</td>
<td>$2,213,008</td>
<td>$2,213,008</td>
<td>$2,213,008</td>
</tr>
</tbody>
</table>

**Return**

<table>
<thead>
<tr>
<th>Return</th>
<th>70</th>
<th>0</th>
<th>0</th>
<th>70</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$1,317,458</td>
<td>$2,079,466</td>
<td>$2,079,466</td>
<td>$2,153,008</td>
</tr>
</tbody>
</table>

**Notes:**

- These are just #s based on values in previous slide
  - Yours will be entirely different
- **Note the zeros for:**
  - On-site education
    - This does not mean you don't need education – it means you will be taking it via remote, Distance Learning
  - Hardware upgrades
- **Recall – very conservative TSO CPU Savings justification**
Thank You
Appendices:
  - CPU Interrupt data points
Every Enter and every PF-Key pressed while in a TSO session results in a CPU interrupt and charge-back.

This includes (but is not limited to):

- Logging in/Logging out of TSO – ISPPROC Dataset Allocate/De-Allocate
- ISPF Navigation
  - Going anywhere (all ISPF and custom options) =whatever
- Issuing REXX and CLIST commands
- Accessing Help (PF1)
- Accessing ISPF Option 0 customization
- Submitting Jobs
- Managing Jobs:
  - Viewing Job results in TSO:
    - Opening the JES logs
    - Browsing job results:
      - PF7/PF8/PF10/PF11 within the file
      - HEX ON/HEX OFF
  - Purging the JES queue
  - Closing the files
Option 1 – or "browse" function in 3.4, Outlist, SPUFI, etc.

- Opening each file
- PF2
- PF5
- PF7
- PF8
- PF9
- PF10
- PF11

- All ISPF Primary Option commands
  - Ex.
    - Find
    - HEX (on/off)
    - Etc.

- Closing the files
Option 2 – or Edit function of 3.4

(Partial list – common ISPF Edit actions):

- Opening each file
- PF1
- PF2
- PF3
- PF4
- PF5
- PF6
- PF7
- PF8
- PF9
- PF10
- PF11

- Line commands
  - Copy/paste/move/insert/delete/shift/eXclude
  - Tabs
  - Columns
  - Text/Split
Option 2 (Partial list – continued):

- **Primary commands**
  - Reset
  - Find
  - Change
    - All permutations
  - Copy in an external PDS member
  - Create an external PDS member
  - HEX On/Off
  - Locate
  - Nulls
  - RES
  - NUM ON/OFF
  - Other commands…

- Save/Close the file
- Custom ISPF edit macros
- **Dataset Management**
  - Allocate datasets
  - Copy datasets
  - Delete datasets
  - Create Like
  - Work with PDS/PDSE datasets
    - Copy members
    - Compress
    - Browse PDS members – especially source files in 3.4
  - Find statistics on datasets
Working with DB2 SPUFI/QMF

- Navigation
- TSO/DB2 connect (IKJEFT01)
- The SPUFI and QMF dialogs
  - Especially QMF
    (Forget about QBE)
- SQL statement code/test:
  - All editing costs from the Edit slides
- Test data man:
  - Insert
  - Update
  - Delete
- All browse operations - PF7/PF8/PF10/PF11/PF2/PF9
- Test table maintenance:
  - DDL
  - DCL

CICS Testing
- All Transaction execution
Unit Testing with Xpeditor -- versus PD Tools and RDz

- Running the Debugging tool – Costs are same as on the host
- Running the Program – Costs are same as on the host

Debugging the program - every mainframe debug session is a series of CPU interrupts - caused by PF Keys and <Enter>.

All source file navigation that is not an action passed up to PD Tools is a CPU cost saving:

- Browsing/Scrolling the file
  - PF7/PF8
  - PF9/PF10
- Find
- Opening an edit session in another window and modifying the source
- Etc.