Session
VS040

Case Studies in OpenVMS Shareable Libraries

Robert Gezelter Software Consultant
35 – 20 167th Street, Suite 215
Flushing, New York 11358 – 1731
United States of America

+1 718 463 1079
gezelter@rlgsc.com

Thursday, June 6, 1996
11:00 – 11:50
Room 124

Spring 1996 US DECUS Symposium
America's Center
St. Louis, Missouri

Robert Gezelter +1 718 463 1079
Software Consultant 35 – 20 167th Street, Suite 215, Flushing, New York 11358 – 1731 USA

Why Use Shareable Libraries?

Maintenance
Speed/Efficiency
Leave no stone unturned
(or program un-relinked)
Dynamic code generation

Robert Gezelter
Software Consultant

When and Why?

It is well known that shareable libraries make sense in heavily used applications. For example, the VMS Run-Time library is implemented as a series of Shareable Libraries.

Robert Gezelter
Software Consultant

NOTES

When and Why? (cont'd)

Not as well known are the benefits realized in program development and applications implementation. These benefits are completely user realizeable. and are separate from the traditional system–wide benefits of using shareable libraries.

Robert Gezelter
Software Consultant

Maintenance

No need to re-link entire program for change in one routine.

Ability to quickly switch between new and old versions of routines.

Robert Gezelter
Software Consultant

NOTES

Speed/Efficiency

- ***INSTALLed shareable image***
- ***Read-only pages shared by many processes***
- ***Significant reduction in memory requirements***
- ***Significant reduction in disk storage requirements***

Robert Gezelter
Software Consultant

Leave No Stone Unturned

- ***Changes in object libraries require relinking to take effect***
- ***Relinking is a major task in a medium/large facility (tens or hundreds of programs)***

Robert Gezelter
Software Consultant

NOTES

Dynamic Code Generation

- ***Permits execution time customization***
- ***Highly efficient***
- ***Simplifies code***
- ***Old tactic; but not well known***

Robert Gezelter
Software Consultant

Cases from our Files:

- ***We will present two case studies:***
- ***Development advantages***
- ***Applications tool for dynamic code generation***

Robert Gezelter
Software Consultant

NOTES

Case 1 – Development

Symptom:

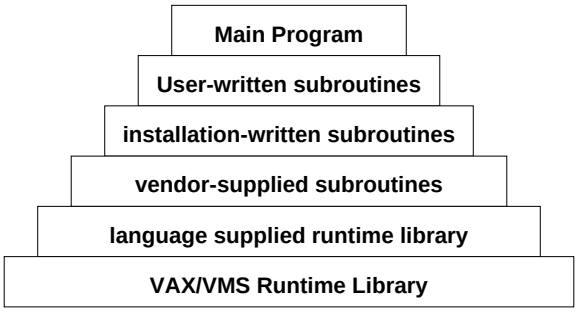
Large Program – Slow Links

Linking this program takes up to 20 minutes on a VAX–11/780

Robert Gezelter
Software Consultant

Problem:

Program is like a pyramid – very large foundation



Robert Gezelter
Software Consultant

NOTES

Solution:

Create one or more user shareable images containing most of the foundation elements.

Result:

Link time reduced to 15 seconds!

Robert Gezelter
Software Consultant

Mechanics of Shareable Libraries

Define Transfer Vector:

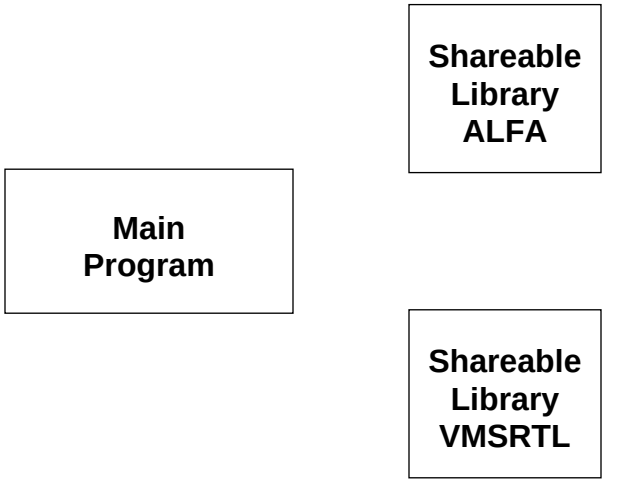
```
.TRANSFER     TEKPLT
.MASK         TEKPLT
JMP           L^TEKPLT+2
.END
```

Assemble transfer vector.

Robert Gezelter
Software Consultant

NOTES

The support code, which is the bulk of the image, is in the shareable libraries!



Robert Gezelter
Software Consultant

Specify the shareable library at execution time

Use logical names.

No privileges required!

```
$ ASSIGN –
$_ $1$DUA2:[GEZELTER]TEKPLT.EXE –
$_ TEKPLT
$ RUN program
```


Robert Gezelter
Software Consultant

NOTES

Case 2 – Dynamic Linking a.k.a. Power Tools with Interchangeable Heads/Bits

Most programs are written to do a particular job.

How does one write a program to do many different jobs?

With Shareable Libraries, of course!

Robert Gezelter
Software Consultant

Programming By Chinese Menu

Pick:

1 from Column A

1 from Column B

3 from Column C

Robert Gezelter
Software Consultant

NOTES

Conventional Programming

Column A: 5 possible choices

Column B: 7 possible choices

Column C: 30 possible choices

TOTAL: 1050 programs
(5 * 7 * 30)

Robert Gezelter
Software Consultant

Programming By Chinese Menu

5 Group A subroutine packages

7 Group B subroutine packages

30 Group C subroutine packages

1 Main Program

TOTAL: 43 programs / packages
(5 + 7 + 30 + 1)

Robert Gezelter
Software Consultant

NOTES

Conventional Programming vs. Chinese Menu – The Difference

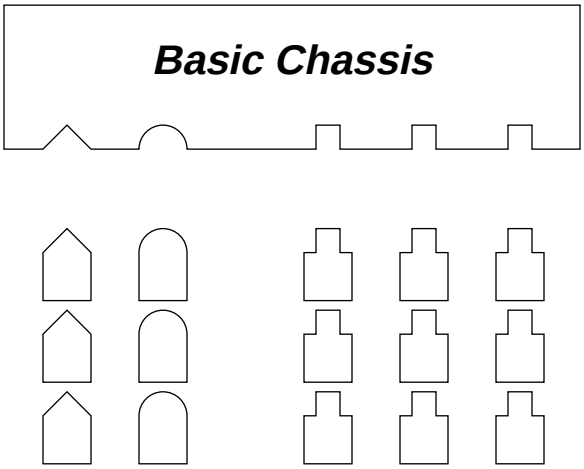
Conventional:
1050 programs

Chinese Menu:
43 modules/packages
3 interfaces

The Difference:
1007 programs!
(or combinations of options)

Robert Gezelter
Software Consultant

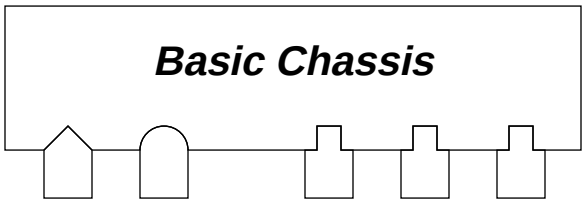
Key Concept: Programming by Chassis



Robert Gezelter
Software Consultant

NOTES

Programming by Chassis: Operation



Robert Gezelter
Software Consultant

Goal:

Develop a large family of related programs with minimal effort

Maintain separation between different applications

Robert Gezelter
Software Consultant

NOTES

Case Study: Mailing List System

Must generate:

- ***Labels***
- ***Envelopes***
- ***Form letters***
- ***Invitations***
- ***Listings***
- ***Attendee Lists***
- ***...***

Robert Gezelter
Software Consultant

Problem: Complexity

Program complexity grows as an exponential (n**m) of the number of different options AND the number of different values of the options

Robert Gezelter
Software Consultant

NOTES

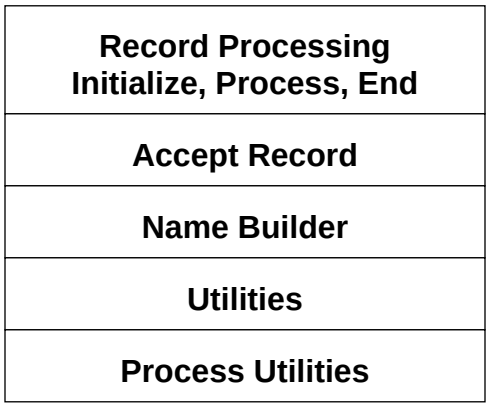
Complexity

Research has shown that correctness of code is endangered by large numbers of nested IF statemets

By hanging different applications components on the same chassis, we are able to achieve a wide variety of options WITH NO INCREASE IN APPLICATIONS COMPLEXITY

Robert Gezelter
Software Consultant

Progamming by Components



Robert Gezelter
Software Consultant

NOTES

Result:

Shareable libraries permit us to achieve the effect of multiple levels of nested IF statements without increasing program complexity.

Robert Gezelter
Software Consultant

Production Environment

The selection of components is driven by the menu system. There is little need for multiple levels of IF statements.

Robert Gezelter
Software Consultant

NOTES

Another view:

This way of building applications is conceptually similar to genetics. You build applications (organisms) out of simple building blocks.

Questions?

Robert Gezelter Software Consultant
35 – 20 167th Street, Suite 215
Flushing, New York 11358 – 1731
United States of America

+1 718 463 1079
gezelter@rlgsc.com

NOTES