

**The CÆSAR Code:  
Software Design Issues**

Michael L. Hall  
Radiation Transport Methods Group (XTM)  
Los Alamos National Laboratory  
Email: `hall@lanl.gov`

X-Division External  
Review Committee Presentation  
3 / 10 / 99

Available on-line at  
<http://www.lanl.gov/Caesar/>

# Outline

- Background
  - CÆSAR Description
  - Diffusion Discretization References
- Documentation
  - Why Document A Program?
  - Levels of Documentation
  - Literate Programming
  - Simplified Approach: **Document**
  - A Simple Example
  - CÆSAR Documentation Features
- Unit Testing / Levelized Design
  - Basic Ideas
  - Preliminary CÆSAR Levelized Design
  - Unit Testing Implementation
- Design By Contract / Verification
- Summary

## CÆSAR Description

- 3-T Photonics Diffusion ( $P_1$ ) Code
- Multiple Dimensionality (1-D, 2-D, 3-D)
- Unstructured Hexahedral Cells in 3-D
- Second-Order Convergent Diffusion Discretizations
- Parallel, written in Fortran 90
- Based on earlier Augustus ( $P-1$ ) and Spartan ( $SP_N$ ) codes
- Future: Polyhedral Meshes, Multigroup, Tensor Diffusion, Mixed Cells, Transport

## Diffusion Discretization References

- Morel-Hall Asymmetric Method
  - Described in

Michael L. Hall, and Jim E. Morel. A Second-Order Cell-Centered Diffusion Differencing Scheme for Unstructured Hexahedral Lagrangian Meshes. In *Proceedings of the 1996 Nuclear Explosives Code Developers Conference (NECDC)*, UCRL-MI-124790, pages 359–375, San Diego, CA, October 21–25 1996. LA-UR-97-8.

which is an extension of

J. E. Morel, J. E. Dendy, Jr., Michael L. Hall, and Stephen W. White. A Cell-Centered Lagrangian-Mesh Diffusion Differencing Scheme. *Journal of Computational Physics*, 103(2):286–299, December 1992.

to 3-D unstructured meshes, with an alternate derivation.

- Support Operator Symmetric Method:
  - Described in

Michael L. Hall, and Jim E. Morel. Diffusion Discretization Schemes in Augustus: A New Hexahedral Symmetric Support Operator Method. In *Proceedings of the 1998 Nuclear Explosives Code Developers Conference (NECDC)*, Las Vegas, NV, October 26–30 1998. LA-UR-98-3146.

which is an extension of

Mikhail Shashkov and Stanly Steinberg. Solving Diffusion Equations with Rough Coefficients in Rough Grids. *Journal of Computational Physics*, 129:383–405, 1996.

to 3-D unstructured meshes, with an alternate derivation.

# Why Document A Program?

## For Others:

- To Demonstrate Progress in Coding
- To Encourage Use of the Package
- To Reduce “Hit-By-A-Bus” Syndrome
- To Facilitate Technical Review

## For Yourself:

- To Understand Global Logical Code Structure
- To Facilitate Computer Code “Re-Entry” For Debugging, Maintenance, and Enhancement
- To Explain Things Once, not Multiple Times, to Users
- To Allow Quick Code Access via Hypertext
- To Be Proud of Your Work

# Levels of Documentation

A code can be rated according to where it falls on this sequential list:

## 0. Layout

0-a. Consistency

0-b. Logical Block Structure (Few or No Branches)

0-c. Indentation to Show Logical Structure

0-d. Blank Lines and Spaces for Readability

0-e. Statements Grouped Semantically

## 1. Descriptive Variable and Routine Names

## 2. Comments throughout the Code

## 3. Routine Headers with

3-a. Purpose

3-b. Input/Output Variable Descriptions

3-c. Internal Variable Descriptions

3-d. Methods Employed

## Levels of Documentation (cont)

### 4. Hardcopy Documentation

- 4-a. Code Listing
- 4-b. Code Manual
- 4-c. User's Manual
- 4-d. Method Discussion

### 5. Hypertext Documentation

- 5-a. Code Listing
- 5-b. Code Manual
- 5-c. User's Manual
- 5-d. Method Discussion
- 5-e. External Links

### 6. Literate Programming:

Source Code and Documentation are Generated from the Same File

# Literate Programming

- Basic Idea: Combine Documentation and Source Code
- Original: WEB (Donald Knuth, of T<sub>E</sub>X fame)
  - Weave: web file  $\longrightarrow$  documentation (T<sub>E</sub>X)
  - Tangle: web file  $\longrightarrow$  source code (Pascal)
- Many WEB-related packages exist — my opinion: most are too complex or don't support my situation (F90, L<sub>A</sub>T<sub>E</sub>X, Unix)



# The Document Package:

## *A Simplified Approach to Literate Programming*

- Eliminate “tangle” step – files are compilable source
- Documentation is included in comments
- Small set of commands to direct output
- Formatting language independent
- Source code language independent (almost — just need to know comment characters)
- Implementation via a short perl script: **Document**
- Source and documentation for the **Document** Package are available online at:

<http://www.lanl.gov/Document>

## A Simple Example

This input file:

```
! Begin_Doc
! Some documentation for standard out.
! End_Doc
!
! This line doesn't get output by Document.
! Begin_Doc file.tex
! This output goes to the file named file.tex.
! Comment characters are stripped by default.
!
! Begin_Verbatim
! Comment characters are included in verbatim
! environments, which are often used for code:
do i = 1, 100
  j = j+1
end do
! End_Verbatim
! End_Doc
```

when processed by **Document**, outputs this to standard out:

```
Some documentation for standard out.
```

and this to **file.tex**:

```
This output goes to the file named file.tex.
Comment characters are stripped by default.
```

```
! Comment characters are included in verbatim
! environments, which are often used for code:
do i = 1, 100
  j = j+1
end do
```

## CÆSAR Documentation

Making use of the capabilities of **Document**, L<sup>A</sup>T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X2HTML, the CÆSAR Code documentation has these features:

- Hardcopy *and* HTML versions from a single source, which is collocated with the source code
- Graphics, equations, code listings easily included
- Automatic table of contents and semi-automatic indexing (hyperlinked in HTML)
- Automatic navigation tools for HTML (Next, Up, Previous, Contents, and Index links on every page)
- Hyper references and external HTML links

Bottom Line: This satisfies **Level 6 Documentation** requirements — User’s Manual, Code Manual, Methods Discussion and Code Listing in Hardcopy and Hyperlinked HTML via Literate Programming

# Unit Testing / Levelized Design

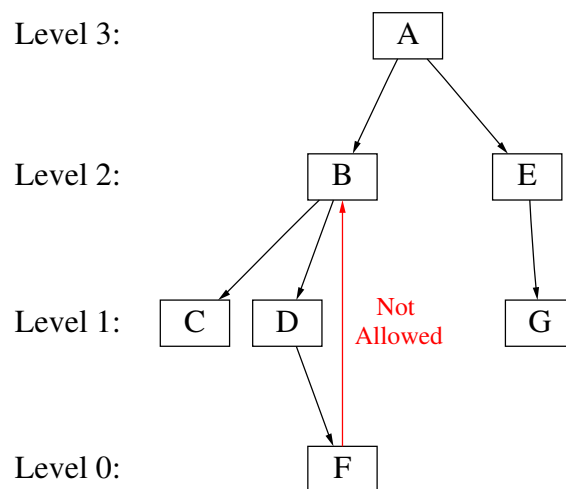
## Basic Idea of Unit Testing:

Each component is tested in isolation – only components that have been previously tested may be included.

## Basic Idea of Levelized Design:

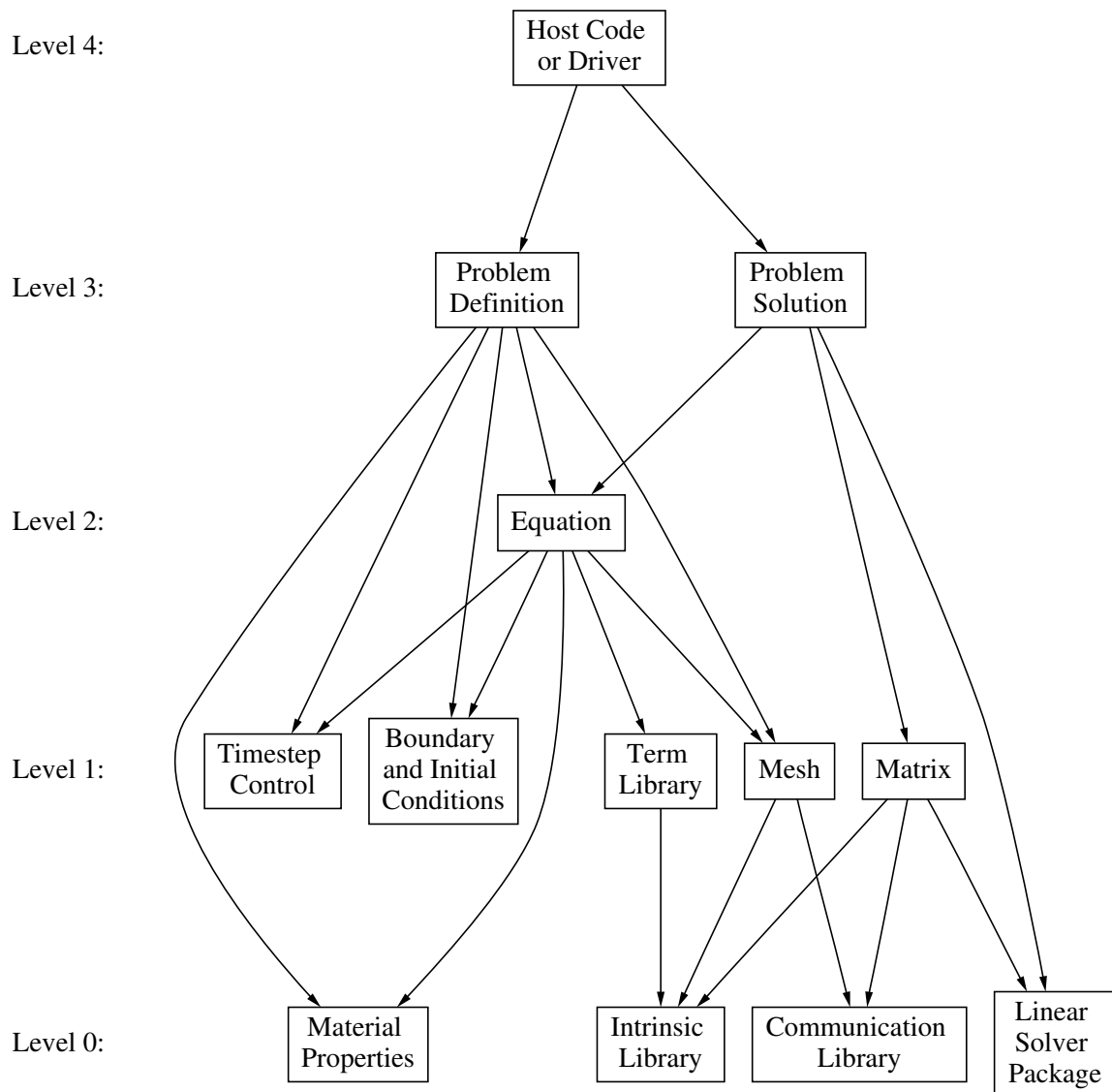
Each component depends only on components that are at a lower level – no feedback or circular designs.

Example:



Bottom Line: Levelized Design is desirable because it makes Unit Testing possible.

# Preliminary Levelized Design for CÆSAR



## Unit Testing Implementation

- Every component contains its own specific driver routine for unit testing.
- Unit test driver routine is only compiled in when certain **gm4** macro preprocessor flags are set.
- CÆSAR uses **Document** to extract and run a unit test script imbedded in each component.

# Design By Contract / Verification

## Basic Idea of Verification:

Statements that verify that specified conditions are true are conditionally compiled into the code, allowing error checking that can be turned off completely for fast execution.

In CÆSAR, [verification](#) is implemented via `gm4` macros.

## Basic Idea of Design by Contract:

Routines satisfy a contract when they are called – input requirements are verified upon entry and output guarantees are verified prior to exit.

[Design by Contract](#) does nothing more than specify where and what to [verify](#).

## Summary

The CÆSAR 3-T photonics package employs many of the latest ideas in software design:

- [Literate Programming](#) — source and documentation stored together.
- The [Document](#) Package is used to extract documentation from code source, which is processed by L<sup>A</sup>T<sub>E</sub>X into hardcopy and L<sup>A</sup>T<sub>E</sub>X2HTML into hyperlinked HTML.
- A [Levelized Design](#) is used to facilitate [Unit Testing](#), which is accomplished using the **gm4** preprocessor and the self-test feature of the **Document** Package.
- [Verification gm4](#) macros are used to implement [Design By Contract](#).