

PROCESSES/METHODOLOGIES

Howard Yudkin

HOW WE STAND NOW

- OK For Small Projects, Not So Good For Large Projects
- Not Good For Addressing Iterative Nature Of Requirements Resolution & Implementation (Mostly Based On Waterfall)
- Does Not Address Complexity Issues Of Requirements Stabilization (Based On Functional Decomposition)
- Does Not Explicitly Address Reuse Opportunities
- Does Not Help With People Shortages

**NEED TO DEFINE AND AUTOMATE IMPROVED
SOFTWARE ENGINEERING PROCESSES**

N91-19725

52-61
320489

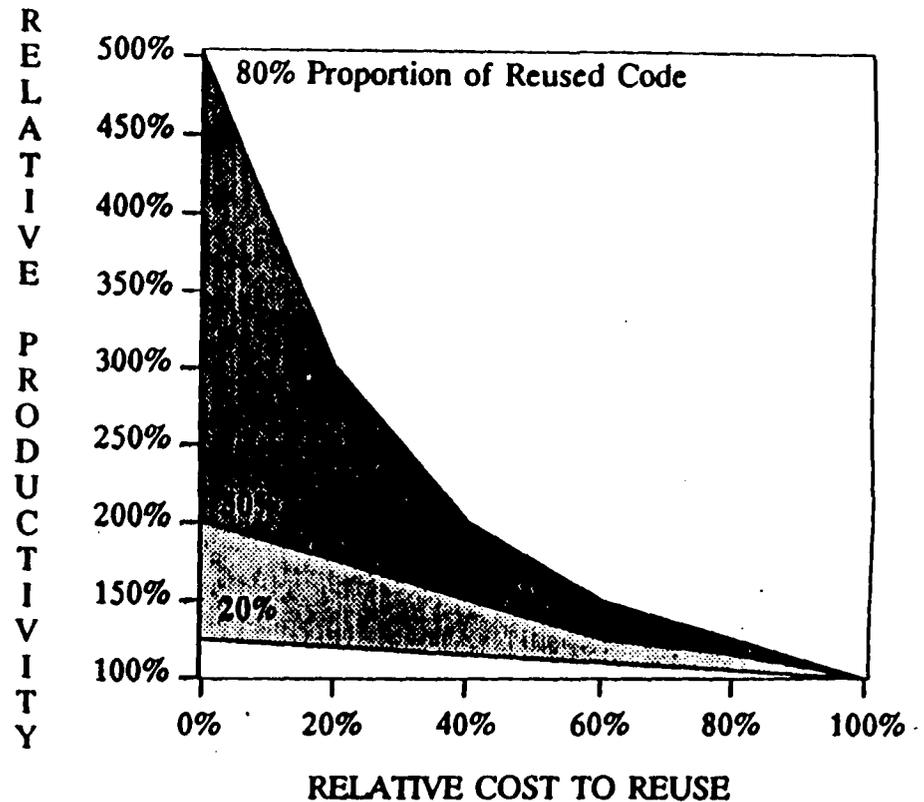
1.20

REUSE AND PROTOTYPING - TWO SIDES OF THE SAME COIN

- Reuse Library Parts Are Used To Generate Good Approximations To Desired Solutions, i.e., Prototypes
- Rapid Prototype Composition Implies Use Of Pre-existent Parts, I.E., Reusable Parts
 - Prototype Quality Depends On Fit Of The Available Parts
 - The Parts Will Often Require Some Adaptation
 - As The Set of Parts Available Becomes Richer The Prototypes Will Better Approximate Acceptable Pieces of Final Systems

REUSE PAY-OFF

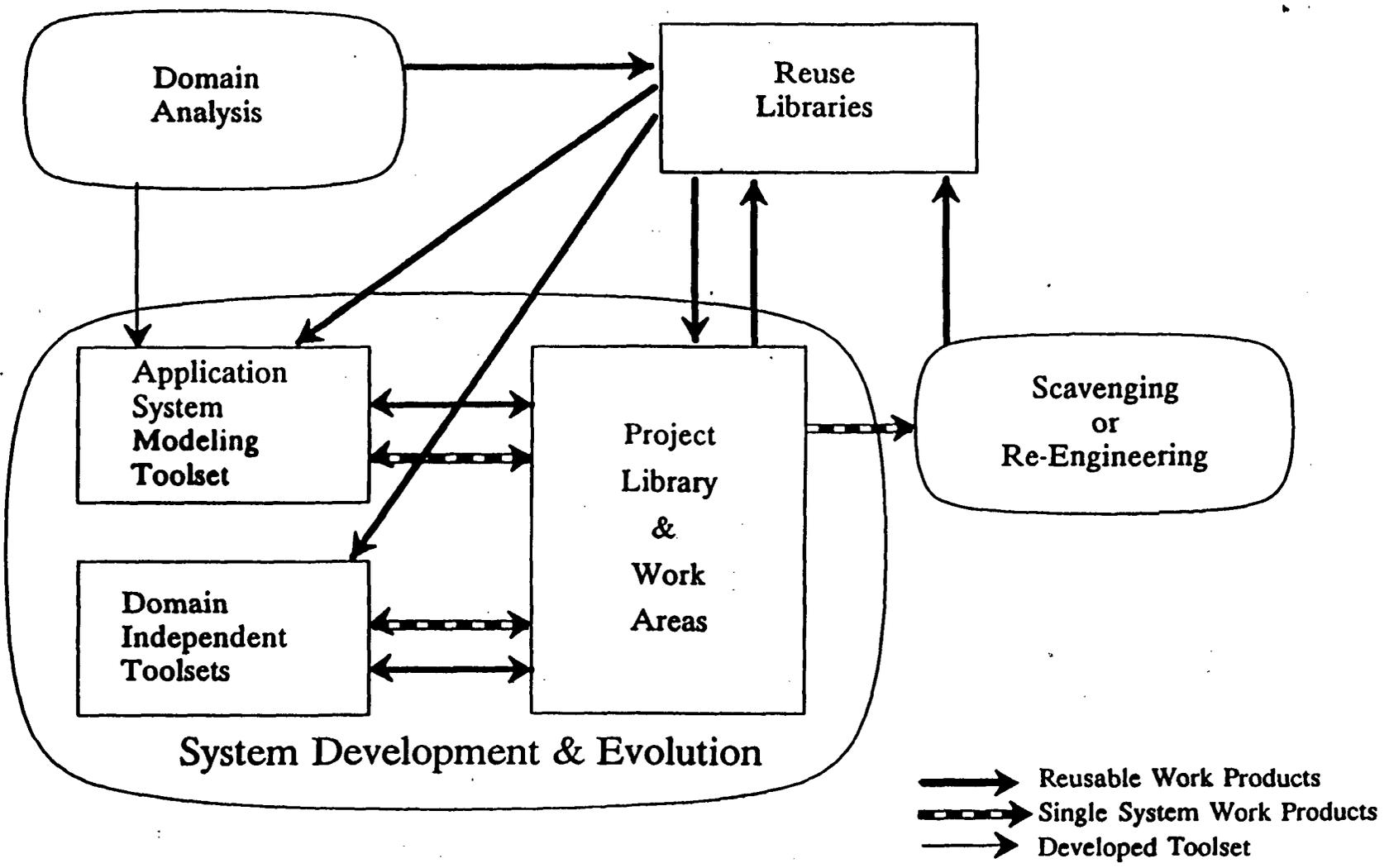
- Big Gains In Productivity Will Come From Reusing Fewer Larger Parts Or Assemblies Of Smaller Parts, Not From Many Unassembled Small Parts.
- Productivity Gain vs Cost Is Acceptable If Assemblies Of Parts Are Reused Frequently.



SYNTHESIS MOTIVATED BY AND ORIENTED TOWARD

- Reuse: Exploit Similarities Across Systems
- Iteration: Feedback and Enhancement
- Composition and Adaptation: Using Standard Schemes, Parts, and Designs
- Specialists: Incorporate Expertise, and Facilitating and Coordinating
- Systems View: Engineering Process
- Applying Synthesis to “Synthesizer”

THREE MAJOR SYNTHESIS SUBPROCESSES

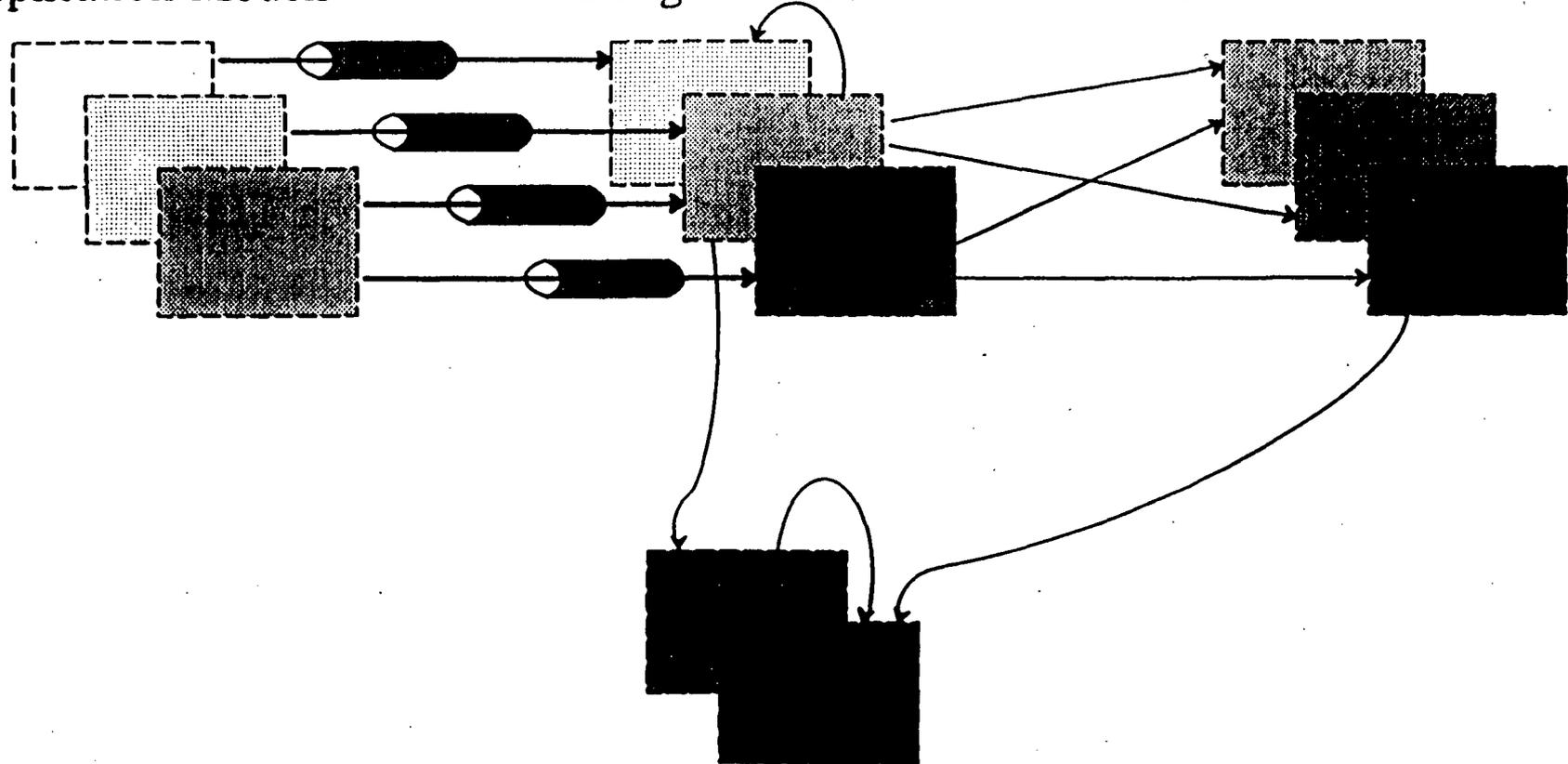


LIBRARY CONTENTS

Application Models

Design Models

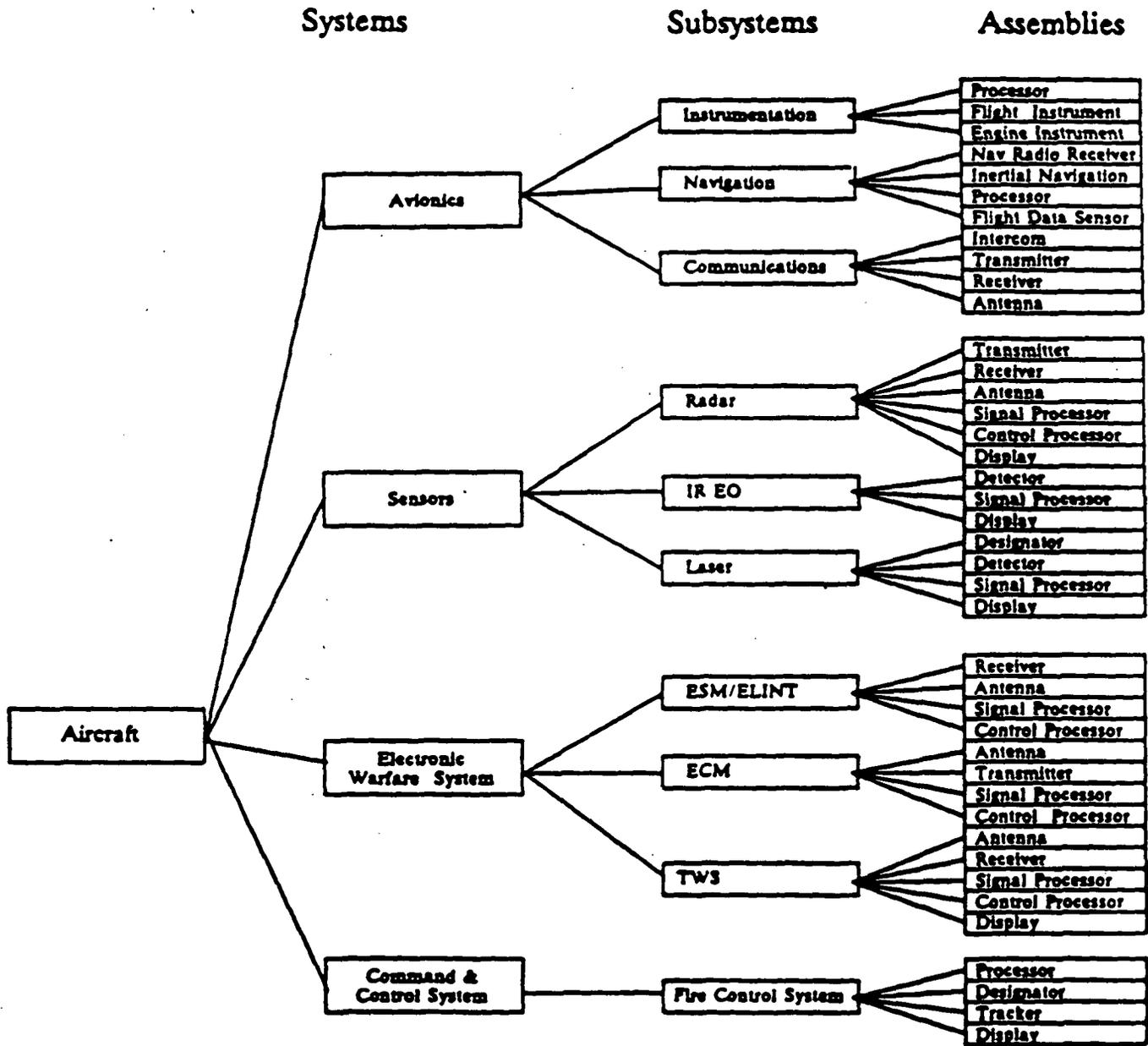
Executable Code



Other Work Products

→ Mappings

TARGET APPLICATIONS FOR DOMAIN ANALYSIS - AIRPLANE EXAMPLE



ESSENCE OF DOMAIN ANALYSIS

- Each application area must be analyzed and characterized by standard *designs* or *architectures* that capture the way that many systems in that area could reasonably be built.
- The application engineer must be able to state his needs in application terms and have those needs mapped appropriately to an instance of the standard design.
- The design instance can be realized by specification of a set of parts from a reuse library and a set of rules for combining those parts.

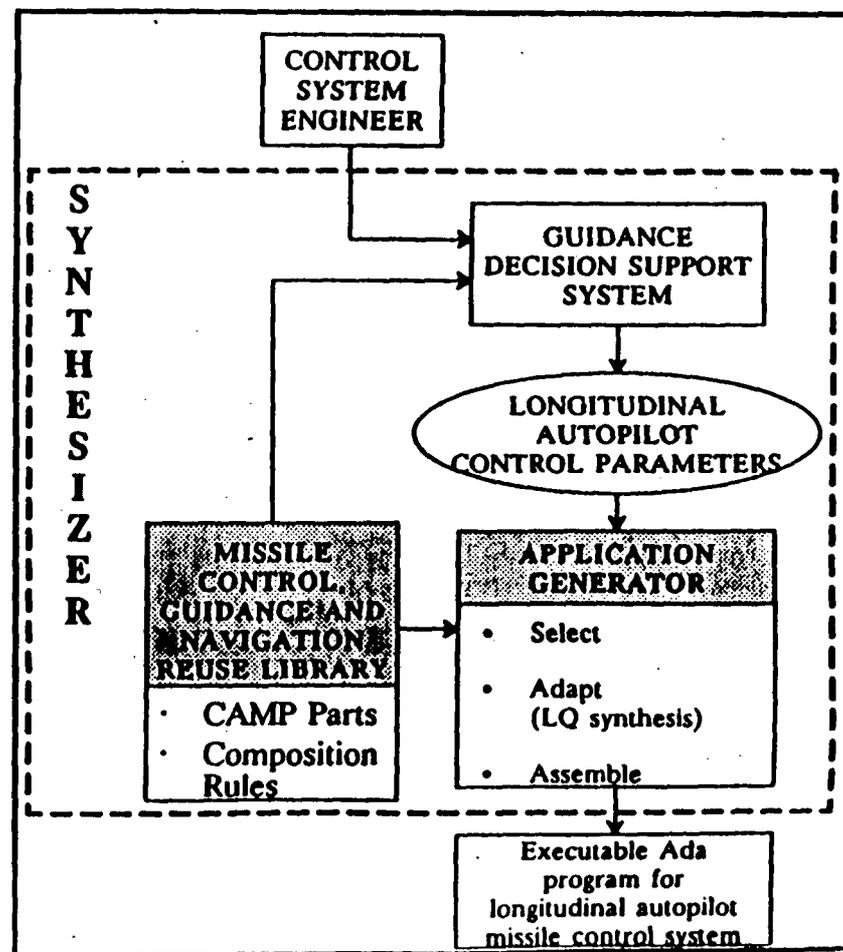
SYNTHESIS SUBPROCESS - SCAVENGING

- Many systems with software have portions amenable to adaptation for reuse.
- Scavenging these systems for reusable parts involves:
 - Extraction
 - Generalization
 - Standardization
 - Certification
 - Cataloging and storing in reuse libraries.

A MISSILE GUIDANCE SYNTHESIS PROTOTYPE TOOL

An example of the application of reuse, prototyping, and synthesis using a reuse library in a specific domain

- Based on U.S. Air Force “Common Ada Missile Packages” (CAMP) parts
- Initially demonstrates a longitudinal autopilot control system
- Aids understanding of the economics of reuse



PARTS OF SYNTHESIS/SYNTHESIZER

User
Inter-
face
Services

Domain
Analysis

Reuse
Libraries

Application
System
Modeling
Toolset

Domain Independent

Design

Dynamics
Assessment

Composition

Traceability

Coding

Verification

Management
& Coordination

OA and
Documentation

Project
Library
&
Work
Areas

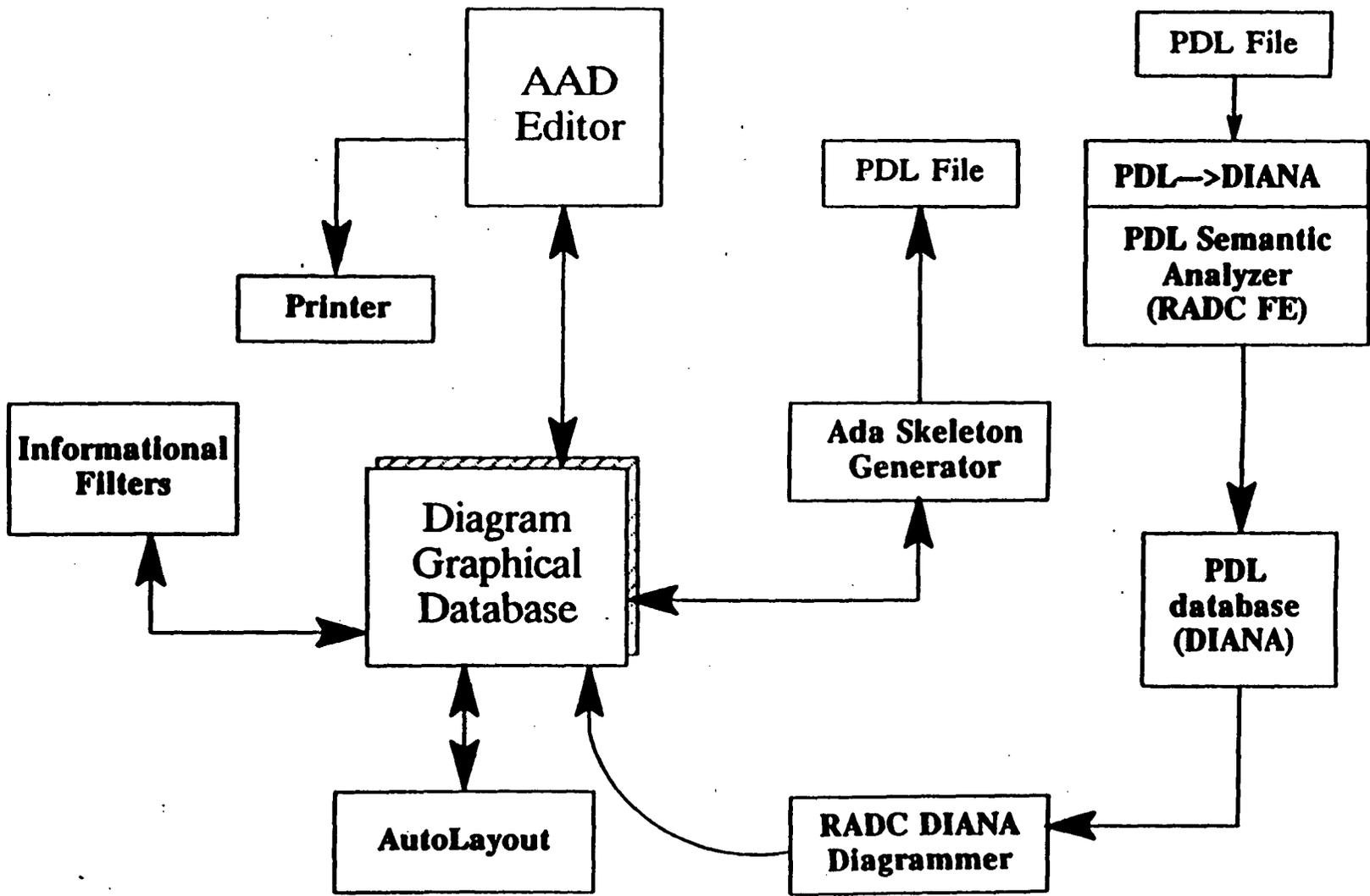
Scavenging

System Development & Evolution

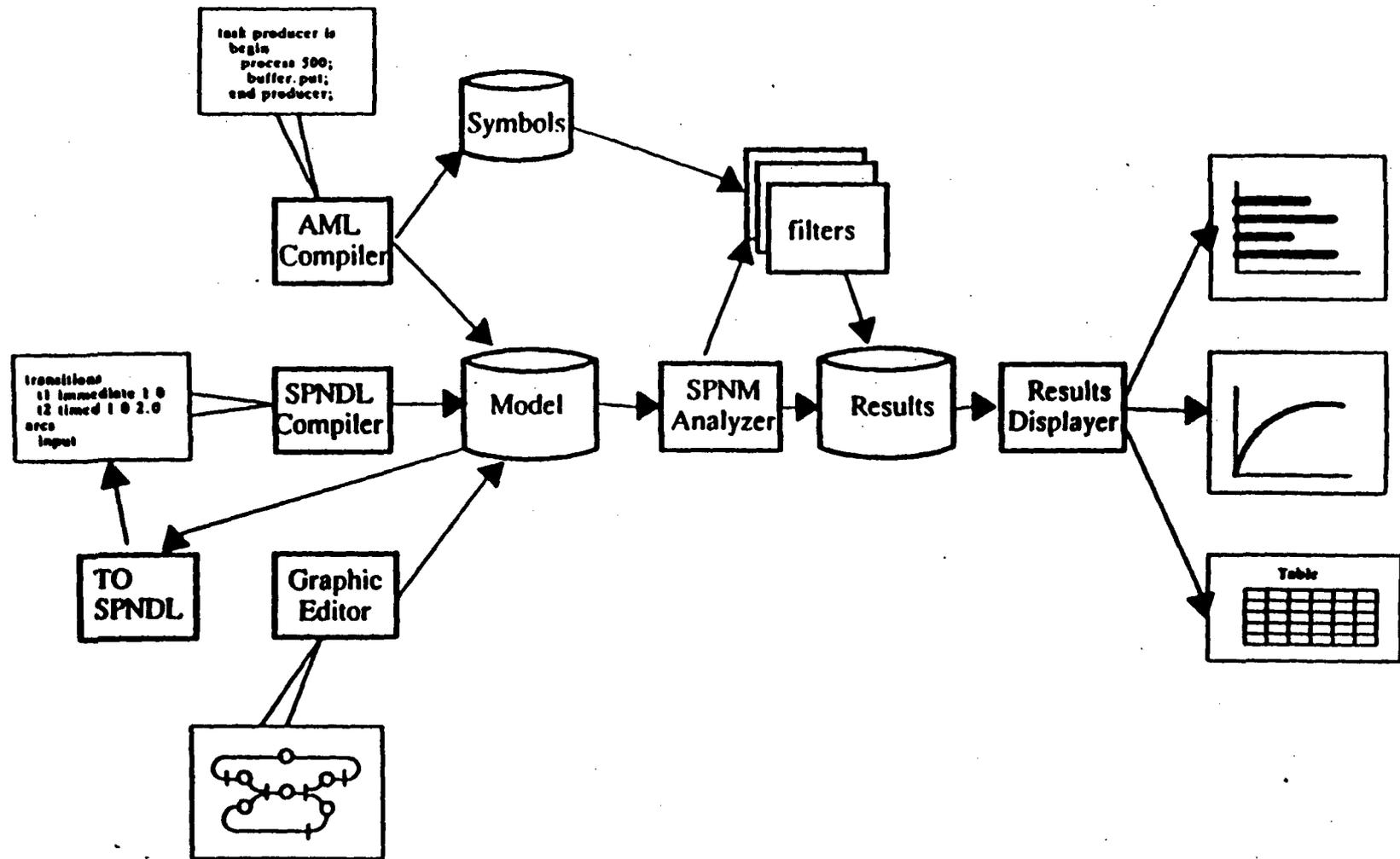
A METHODOLOGY FOR PARTS SPECIFICATION AND MODEL ASSEMBLY IS EVOLVING

- Based On NRL Software Cost Reduction Methodology
 - Information Hiding Module Families
 - Abstract Interfaces
- Accommodates Ada Packaging And Tasking Concepts
 - Tasking Guidelines Evolved (ADARTS)
- Initial Guidebooks Written And In Use

PRODUCT SET 1A STRUCTURE

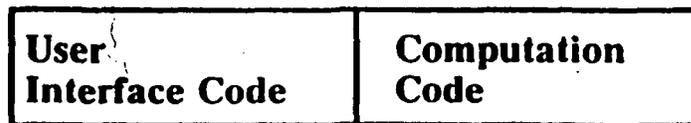


DYNAMICS ASSESSMENT TOOLSET COMPONENTS



UIS ARCHITECTURE

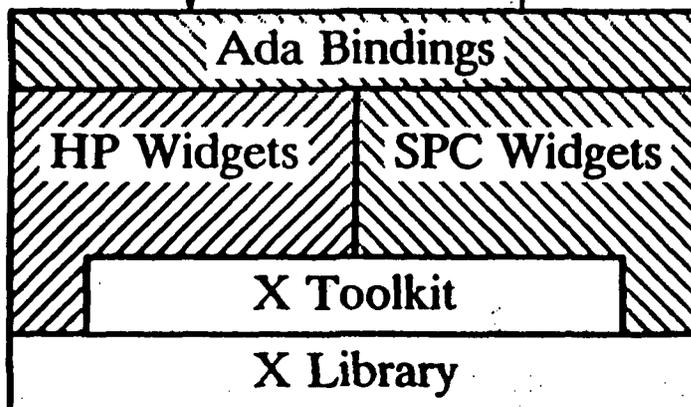
Tool



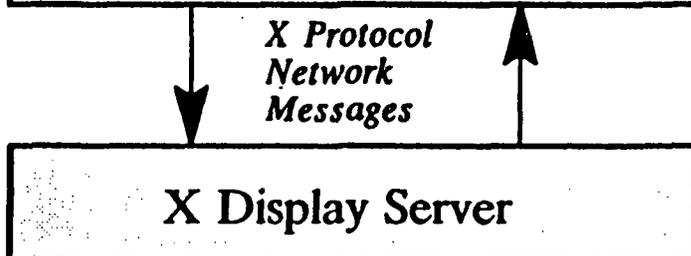
*Widget
Instantiation
Calls*

*Tool
Computation
Callbacks*

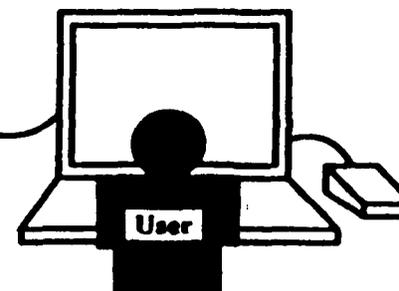
*User
Interface
services*



*X Window
System*



*(or Integrated Server for X
and Native Window System)*



THE LAYERED REPOSITORY CONCEPT

ONE FOR EACH MEMBER
COMPANY NETWORK
LOCATION

REPOSITORY

ONE FOR EACH PROJECT

PROJECT
LIBRARY

ONE OR MORE FOR
EACH DATABASE LOCATION

PARTITION

CREATED BASED ON
PROJECT NEEDS

COLLECTION

(COLLECTIONS CAN CONTAIN DATA OBJECTS
AND/OR OTHER COLLECTIONS)

ACTUAL DATA
OBJECTS.

OBJECT

OBJECT

OBJECT

INCLUDES BOTH RDBMS STORAGE AND COTS CM STORAGE

DISTRIBUTED
DATA
LEVEL

INDIVIDUAL
DATABASE
LOCATION
LEVEL

OBJECT
LEVEL

TYPICAL PROJECT LIBRARY ACCESS

APPLICATIONS CODE:

- DESIGN TOOL
- ASSESSMENT TOOL
- TRACEABILITY
- HARNESS TOOLS
- MC DEVELOPED TOOLS

TOOL K

TYPICAL COMMANDS

DATA OBJECT RELATED

- CREATE DATA OBJECT
- DELETE DATA OBJECT
- CHECK OUT DATA OBJECT BODY
- CHECK IN DATA OBJECT BODY
- GET ATTRIBUTE
- SET ATTRIBUTE
- GET CONTENTS LIST

RELATIONSHIP RELATED

- CREATE RELATIONSHIP
- DELETE RELATIONSHIP
- GET ATTRIBUTE
- SET ATTRIBUTE

UNIQUE ID RELATED

- PATHNAME TO UID
- RELATIONSHIP TO UID

QUERY RELATED

- FETCH BY ATTRIBUTE VALUE
- GET RELATIONSHIP TYPES
- GET RELATIONSHIPS

DMS CODE

DAS/DMS

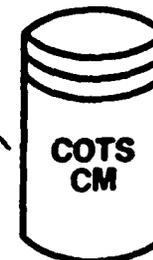
COTS PRODUCTS



RDBMS

ATTRIBUTES
RELATIONSHIPS

DYNAMIC SQL INTERFACE



**COTS
CM**

OBJECT BODIES

TAILORED CODE
INTERFACE

Traceability Navigator 1

NAV - Traceability Navigator

Requirements objects in /xyz/abc/def/jkl 17 total
 Subset: date > 880325, owner = Johnson

Name	Rev	Para number	Type	Date
<input checked="" type="checkbox"/> jlljkljlk	5	3.1.1.2	cap	12Sep88
<input type="checkbox"/> jllkjdsfsj	4	3.1.1.2.1	in	21Sep88
<input checked="" type="checkbox"/> jkjojkldfmk	5	3.1.1.2.2	out	21Sep88
<input checked="" type="checkbox"/> djlojklmkn	2	3.1.1.2.3	cap	17Aug88
<input checked="" type="checkbox"/> jlljkljlk	2	3.1.1.3	cap	
<input type="checkbox"/> jllkjdsfsj	8	3.1.1.3.1	in	
<input checked="" type="checkbox"/> jkjojkldfmk	4	3.1.1.3.2	out	
<input checked="" type="checkbox"/> djlojklmkn	12	3.1.1.3.3	cap	
<input type="checkbox"/> jlljkljlk	6	3.1.1.4	cap	
<input type="checkbox"/> jllkjdsfsj	3	3.1.1.4.1	in	

Tracing relationships to a set of related objects

- change view->
- create new view
- change presenta
- check constraint
- make report->
- create relationships
- deselect all
- tools->
- list objects->
- list relationships->
- trace relationships
- list relationships
- show subset

NAV - Trace Relationships

Tracing from requirements objects in NAV window 1

specify relationship type(s)

- tested by
- implemented by
- derived from
- described in
- sllksj llksdifo

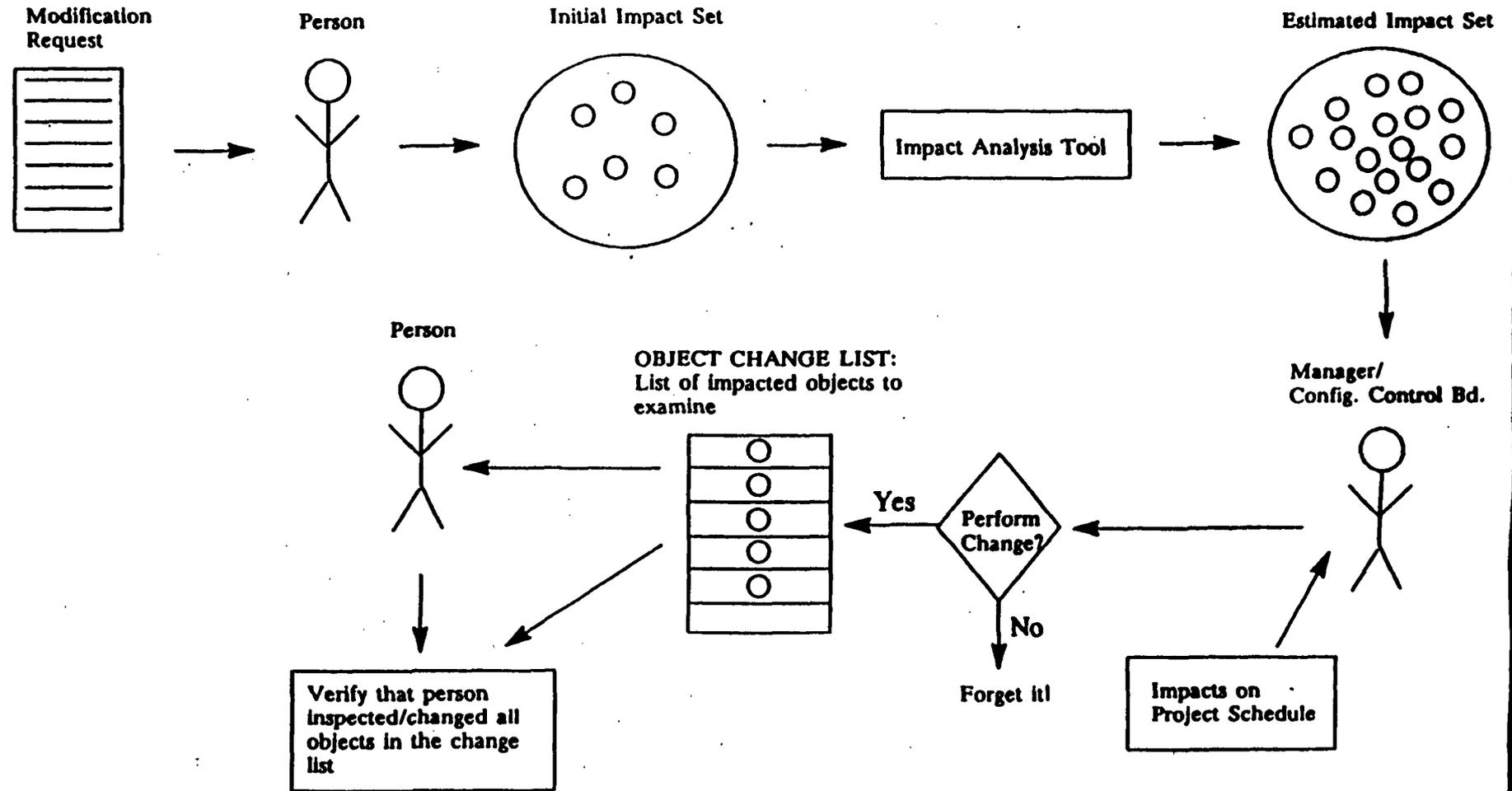
specify scope of tracing

/V23 CCMS

GO

ORIGINAL PAGE IS
OF POOR QUALITY

IMPACT ANALYSIS TOOLSET OVERVIEW



CANDIDATE USER INTERFACE FOR RLT

X-Window
System

