Contract Req
Software Requirements Analysis
Version 1: A C-BASS Component

by Denis McGurin

ARL-MR-458

August 1999

Approved for public release; distribution is unlimited.
The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.
Contract Req Software Requirements Analysis Version 1: A C-BASS Component

Denis McGurin
Corporate Information and Computing Directorate, ARL

Approved for public release; distribution is unlimited.
Abstract

This document contains the software requirements analysis for a prototype of Contract Request Version 1.0 (Contract Req). As a component of the Corporate Business Application Software System (C-BASS), this application automates the preparation of the requester's Procurement Data Package (PDP). The document follows the process of structured analysis, or step-wise refinement of requirements, as applied to the development of Contract Req. The "environmental model" includes a high-level system description, followed by a context diagram and a list of events to which the system must respond. The "behavioral model" includes a data flow diagram (DFD) for each of the five Contract Req subsystems. From this representation, the basic functional specifications are derived and represented in structured English (or program design language). The final segment of the document includes a data dictionary.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Figures</td>
<td>v</td>
</tr>
<tr>
<td>List of Tables</td>
<td>v</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Contract Req Prototype</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Development Plan and Project Schedule</td>
<td>1</td>
</tr>
<tr>
<td>1.3 Contents of the Report</td>
<td>1</td>
</tr>
<tr>
<td>2. Structured Analysis Overview</td>
<td>2</td>
</tr>
<tr>
<td>3. System Overview</td>
<td>3</td>
</tr>
<tr>
<td>3.1 Purpose and Required Functionality</td>
<td>3</td>
</tr>
<tr>
<td>3.2 Event List</td>
<td>5</td>
</tr>
<tr>
<td>4. System Requirements</td>
<td>5</td>
</tr>
<tr>
<td>5. Functional Specifications</td>
<td>9</td>
</tr>
<tr>
<td>5.1 Contract Req Subsystems</td>
<td>10</td>
</tr>
<tr>
<td>5.2 Subsystems Data Flow Diagrams</td>
<td>11</td>
</tr>
<tr>
<td>5.3 Processing Narration</td>
<td>14</td>
</tr>
<tr>
<td>6. Data Dictionary</td>
<td>22</td>
</tr>
<tr>
<td>7. References</td>
<td>33</td>
</tr>
<tr>
<td>Distribution List</td>
<td>35</td>
</tr>
<tr>
<td>Report Documentation Page</td>
<td>37</td>
</tr>
</tbody>
</table>

iii
INTENTIONALLY LEFT BLANK.
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Context Diagram for Contract Req</td>
<td>4</td>
</tr>
<tr>
<td>2.</td>
<td>Major Subsystems of Contract Req</td>
<td>10</td>
</tr>
<tr>
<td>3.</td>
<td>Prepare Contract Request Process</td>
<td>12</td>
</tr>
<tr>
<td>4.</td>
<td>Approve Funds Process</td>
<td>12</td>
</tr>
<tr>
<td>5.</td>
<td>Approvals Process</td>
<td>13</td>
</tr>
<tr>
<td>6.</td>
<td>Edit Contract Request Process</td>
<td>13</td>
</tr>
<tr>
<td>7.</td>
<td>Inquiries Process</td>
<td>14</td>
</tr>
</tbody>
</table>

List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>High-Level Requirements for the Contract Req Prototype</td>
<td>4</td>
</tr>
</tbody>
</table>
INTENTIONALLY LEFT BLANK.
1. Introduction

Contract Request Vision 1.0 is a component of the Corporate Business Application Software System (C-BASS) cluster of applications, an integrated set of Lotus Notes and Web-based software to support U.S. Army Research Laboratory (ARL) electronic workflow and task automation. (Throughout this report, Contract Request Version 1.0 is abbreviated as Contract Req.) The motivating force behind this work effort has been ARL downsizing and the findings contained in the Business Process Reengineering (BPR) report on the contracting process published in 1995 [1]. The BPR "To-Be Model: Formal Contracts" [1] identified potential process improvements — some of which require computer automation — that would increase productivity of contracting operations at ARL. Development of a full-production Contract Req will proceed in phases, using an incremental, evolutionary approach. The system described in this document is the first version in this process.

1.1 Contract Req Prototype. The purpose of the Contract Req prototype is to provide a secure client/server system that provides for the processing of contract requests. This proof-of-principle prototype will alleviate some of the risks involved in implementing new technologies used to build the ARL Intranet. The project will also refine requirements described in the ARL BPR To-Be Model [1].

1.2 Development Plan and Project Schedule. The "Contract Req Software Development Plan" [2] states a definition of the problem; gives an overview of technical, management, and reliability issues; and provides a detailed project schedule. The report given herein covers the work accomplished to solidify user requirements (primarily drawn from existing high-level design documents) and the analytical expansions used to derive a data flow model, a pseudocode representation of processing, and a data dictionary.

1.3 Contents of the Report. This document presents the results of a structured analysis used to derive the software requirements for Contract Req, starting with the baseline given in the BPR "To-Be Model: Formal Contracts" [1]. The body of the report contains five sections:
• "Structured Analysis Overview" - briefly explains the methodology used to extract the functional specifications.

• "System Overview" - delineates the basic Contract Req concept and outlines the high-level requirements.

• "System Requirements" - breaks the more generic statements into low-level, derived requirements and describes each in detail.

• "Functional Specifications" - discusses the products of the structured analysis (i.e., the data flow diagrams and the structured English narrative) for each subsystem of Contract Req.

• "Data Dictionary" - lists each of the Contract Req data elements, giving a full description and type for the data model.

2. Structured Analysis Overview

Modern software engineering utilizes structured analysis [3] as a powerful methodology for developing system specifications. Through a series of step-wise refinements, detailed delineation of the system's components and their behavior are extracted from high-level descriptions of system features and functions. In other words, primary system elements are broken down into progressively more detailed levels of processes, and the data paths between these processes are defined. Three modeling tools facilitate this decomposition: (1) data flow diagrams (DFDs), (2) structured English process narratives (either pseudocode or program design language [PDL]), and (3) a data dictionary. Accuracy and precision in progressively expanding design definitions are critical to successful system development.

The results of this analytical approach are systematic elaborations of product requirements, typically expressed as parts of two separate models:
• An environmental model that defines the system’s interfaces to the outside world (see section 3, “System Overview”).

• A behavioral model that defines the internal behavior the system must exhibit in order to deal with the environment (see section 4, “System Requirements,” and section 5, “Functional Specifications”).

3. System Overview

The environmental model typically consists of three components: (1) a concise statement of the system’s purpose or required functionality, (2) a context diagram, and (3) an event list. The context diagram is the highest level DFD. It shows the system as a single process, including user interaction and communication with external systems, as well as data flow input and output. The event list provides an index of outside stimuli to which the system responds.

3.1 Purpose and Required Functionality. As a system, the Contract Req prototype provides a secure, automated means for the preparation, submission, approval, and tracking of certain aspects of the contracting process. The complete contract life cycle is complex and multistaged. Contract Req currently automates only the preparation of the requester’s Procurement Data Package (PDP). See “Contract Req Software Development Plan” [2] for an explanation of the exact portions of the contract process automated by this product.

Table 1 lists the high-level requirements for the Contract Req prototype and gives a general description of what the requirement involves.

Figure 1 shows the context diagram for the prototype. This diagram displays the external entities (e.g., users, functional areas, and legacy systems), represented by the squares, with which Contract Req communicates. The data flowing into and out of Contract Req are represented by the arrows. A few elements on the DFD need additional explanation. First, the external entity “User” represents all users of the system. The data flow associated with this box is limited to
Table 1. High-Level Requirements for the Contract Req Prototype

<table>
<thead>
<tr>
<th>Requirement</th>
<th>General Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security</td>
<td>Provide security measures to prevent unauthorized access to the system and its data, and keep authorized users from performing tasks not allowed in their roles.</td>
</tr>
<tr>
<td>Contract request preparation</td>
<td>Provide a means for the requesters and functional users to input/edit relevant information pertaining to a contract request.</td>
</tr>
<tr>
<td>Automated request routing</td>
<td>Automate the process of routing contract requests to the various functional areas.</td>
</tr>
<tr>
<td>Electronic approval</td>
<td>Provide a means for approving officials and functional users to electronically approve or reject a contract request.</td>
</tr>
<tr>
<td>Request tracking</td>
<td>Allow users to track the status of active contract requests currently in the system.</td>
</tr>
<tr>
<td>Legacy system interface</td>
<td>Implement automated interfaces to Standard Operations and Maintenance Army Research Development System (SOMARDS).</td>
</tr>
<tr>
<td>Reporting</td>
<td>Provide users and management with a means for reporting cycle times and costs.</td>
</tr>
</tbody>
</table>

Figure 1. Context Diagram for Contract Req.
display of information. All the other external entities (e.g., "Requester," "Supervisor," etc.), and their corresponding data flows, show the specific information that is passed to Contract Req, either by that user or by the system. The data store EMPLOYEE contains user information such as name, phone number, address, office symbol, etc.

3.2 Event List. The following are the events to which the system must respond.

- Requester initiates new contract request.
- Requester prepares request.
- Requester corrects rejected request.
- Requester, supervisor, or budget analyst completes fund source.
- Requester or supervisor cancels request.
- Budget analyst certifies fund source.
- SOMARDS certifies and commits funds.
- Special approving official(s) approves request item(s).
- Property book officer approves request.
- Contracting officer assigns buyer to request.
- Supervisor approves actual costs.
- User submit status inquiry.
- User request report.

4. System Requirements

Antecedent studies and legacy systems also contribute to Contract Req’s requirements. The “To-Be Model: Formal Contracts” [1] was produced during the BPR study in 1995. However, for some areas, this document lacks detail, making it necessary to derive missing elements. Additionally, the limited scope of a prototype necessitates leaving out some of the more complicated or vague automation requirements of the “To-Be Model” or where the requirement already exists in a Department of Defense (DOD) standard system (e.g., functionalities handled by the Standard Army Automated Contract System [SAACONS]). The “Contract Req Software
Development Plan" [2] more fully addresses the constraints on the prototype and the requirements of the legacy systems.

The following elaboration more fully defines the proscribed system tasks enumerated in Table 1.

E1

E2  Security

E21  Prevent unauthorized access

*Description* — Prevent unauthorized access to the system and its data.

*Source* — Derived, due to the nature of the system.

*Interfaces to major functions and external entities:*

  User

E22  Enforce role restrictions

*Description* — Prevent users from performing tasks or accessing/editing data that are out of the scope of their role.

*Source* — BPR “To-Be Model” document, requirement A151.

*Interfaces to major functions and external entities:*

  User
  Approvals
  Edits
  Employee address book (for roles).

E3  Contract request preparation

E31  Create new contract request

*Description* — Allow the requester to create a new contract request with preliminary user information filled in.

*Source* — BPR “To-Be Model” document, Automation Requirements section, requirement A151

*Interfaces to major functions and external entities:*
User
Security
Employee address book (for user info)

E32 Select items

*Description* — Provide a means for the requester to enter item descriptions, specifications, quantities, and estimated costs.

*Source* — BPR “To-Be Model” document, “Automation Requirements” section, requirement A151.

*Interfaces to major functions and external entities:*

User
Security

E33 Complete contract request

*Description* — Provide a means for the requester and/or approving supervisor to complete the contract request.

*Source* — BPR “To-Be Model” document, Automation Requirements section, requirements A151.

*Interfaces to major functions and external entities:*

User
Security

E34 Item tag input

*Description* — Provide a means for the Property Book Officer to enter item tags.

*Source* — BPR “To-Be Model” document, Automation Requirements section, requirements A151.

*Interfaces to major functions and external entities:*

User
Security

E35 Edit contract request

*Description* — Provide a means for users to edit certain request details as needed.

*Source* — Derived, due to the need for making corrections to rejected contract request.

*Interfaces to major functions and external entities:*

7
User
Security

E4

E5 Automated routing

Description — Automate the process of routing contract requests to the various functional areas and approving officials.

Source — BPR “To-Be Model” document, Automation Requirements section, requirements A151.

Interfaces to major functions and external entities:

Security
Employee address book (for default routing)

E6 Electronic approval

Description — Provide a means for approving officials and functional users to approve or reject a contract request.

Source — BPR “To-Be Model” document, Automation Requirements section, requirements A151.

Interfaces to major functions and external entities:

User
Security

E7 Request tracking

Description — Allow users to track the status of active contract requests currently in the system.

Source — User requested.

Interfaces to major functions and external entities:

User
E8 Legacy system interfaces

E81 Budget legacy system interface

Description — Provide an electronic interface to the legacy financial system (SOMARDS) that automates the certification and commitment of funds.

Source — “To-Be Model” section, process model diagram A12, process A153.

E9

E10 Reporting

Description — Provide users and management with a means for reporting cycle times and costs.

Source — Derived from ARL corporate experience. Not specifically noted in BPR Formal Contracts documentation.

Interfaces to major functions and external entities:

User

Security

E11 Navigation

Description — Provide users with a means for navigating to the various functional areas within the system.

Source — Derived from the requirements listed previously.

Interfaces to major functions and external entities:

User

5. Functional Specifications

The behavioral model expands the results from the environmental model to more fully define how the system performs prescribed tasks. Typical representations in this mode are (1) concise flow charts showing how information is transformed as it moves through the system and subsystems; (2) a set of structured English statements forming a processing narration based on
data types, control structures, and transformations; and (3) a data dictionary defining each data item.

5.1 Contract Req Subsystems. The eight functions listed in the previous section identify the major components of Contract Req: (1) security, (2) contract request preparation, (3) automated routing, (4) electronic approval, (5) request tracking, (6) legacy system interfaces, (7) reporting, and (8) navigation. These categories consolidate into five subsystems: (1) prepare contract request, (2) approve funds, (3) obtain approvals, (4) edit contract requests, and (5) inquire about status.

Figure 2 is a DFD showing the major functional subsystems of Contract Req. Each of the five bubbles in the diagram represents a major subsystem or process of Contract Req, with the arrows showing the data flowing into and out of the processes.

Figure 2. Major Subsystems of Contract Req.
The data store ACTIVE—a Lotus Notes database located in the center of the diagram—holds all the active contract requests, each waiting for the various users to perform their functions on them. The CLOSED and CANCELED data stores contain contract requests that have been closed out and canceled, respectively. The small squares along the outer edges of this DFD are interfaces to the outside world.

No process bubble for security appears at this level because the Lotus Notes application environment handles security and verifies user authorization. Additionally, enforcement of role restrictions are handled within each subsystems, as detailed in the following sections.

5.2 Subsystems Data Flow Diagrams. System objects and operations can be coherently represented as DFDs. A DFD can be used to capture system concepts and components at any level of abstraction. Each of the following five DFDs (Figures 3–7) provides more detail for the information flow and functionality of each of the identified Contract Req subsystems.

Figure 3 shows the DFD for the “Prepare Contract Request” subsystem. The major inputs to this process (and its basic functions) are the requester information (derived from the user supplied userid and the EMPLOYEE data store), item details, vendor information (if known at this time), and the date by which the items are required. The fund source completes the information for the request, and the supervisory approval puts the request into the contracting cycle.

Figure 4 shows the DFD for the “Approve Funds” subsystem. This subsystem, besides having interfaces to users for approvals, also has connections to the SOMARDS legacy system. The “Build Block” process is executed at the start of the day and creates the transaction block that will be used by Contract Req for the remainder of the day. As eligible contract requests are created during the course of the day, the “Certify Funds” process queries SOMARDS and grabs the returning message. Depending on the results of this query, the request is either certified or rejected (with explanation). At the end of the day, the “Reconcile” process is executed to balance the transaction block.
Figure 3. Prepare Contract Request Process.

Figure 4. Approve Funds Process.
The "Approvals" process is shown in Figure 5. At this point, the various approving officials attach their approval or rejection to the request. The property book officer also attaches item tags to the individual items in order to flag them during receipt of the shipment.

Figure 5. Approvals Process.

Figure 6 diagrams the "Edit Contract Request" process. The rejected contract request is displayed for the requester, who then enters the corrections. What fields within the request the user can edit depends on where the rejection came from and how far along in the approval process the request has traveled.

Figure 6. Edit Contract Request Process.

The "Inquiries" process is diagramed in Figure 7. The processes shown in this figure are used to display to the user (1) pending actions (inbox), (2) contract request details, (3) the status of requests, and reports.
5.3 Processing Narration. Having captured the flow of information and identified data objects, each transformation can be further expanded by using the stylized notation of structured English. Basic procedural constructs are combined with English phrases to give a concise description for each major operation listed in the prescribed tasks analysis given in section 4.

E1 Security
    E11 Login
        Input — userid, passwd
        Process:
        REPEAT
        GET from user the userid, passwd
        UNTIL VALID userid, passwd
        ALLOW login
        Output — Access on Success, Error Message on Failure
    E12 Role
        Input — role, action
        Process
        GET from EMPLOYEE the role using requester_userid
IF VALID action for role THEN
    EXECUTE action
ELSE
    NULL
ENDIF

Output — action_results

E2 Prepare contract request for ordering
E21 Create new contract request
    Input — pr, requester_userid, user_info
    Process 1.1
        GET from EMPLOYEE the user_info using
                requester_userid
        SET in pr the requester_userid
        SET in pr the user_info using user_info
    Output — blank_pr
E22 Fill in contract request
    Input — blank_pr, priority_code, items, vendors, date_required
    Process 1.2
        GET from user the priority_code
        SET in pr the priority_code
        DO WHILE there is another item to add
            GET from user the item
            SET in pr the item
        ENDDO
        GET from user the specifications
        SET in pr the specifications
        DO WHILE there is another vendor to add
            GET from user the vendor
            SET in pr the vendor
        ENDDO
GET from user the **date_required**
SET in **pr** the **date_required**

*Output* — **partial_pr**

**E23** Fill in fund source

*Input* — **partial_pr**, **fund_source**

*Process 1.3*

GET from user the **fund_source**
SET in **pr** the **fund_source**

*Output* — **funded_pr**

**E24** Correct contract request

*Input* — **rejected_pr**, **corrections**

*Process 4.1*

DISPLAY to user **rejected_pr** and explanation
DO WHILE there are more **corrections**
GET from user the **corrections**
SET in **pr** the **corrections**
ENDDO

*Output* — **corrected_pr**

**E25** Cancel request

*Input* — **active_pr**, **cancel_order**

*Process 4.2*

DISPLAY to user the **active_pr**
GET from user the **cancel_order**
PUT **canceled_pr** into CANCELED

*Output* — **cancelled_pr**

**E3** Routing

**E31** Automated routing

*Input* — **active_pr**

*Process* — **TBD**

*Output* — **active_pr**
E32   Manual routing

Input — active_pr

Process — TBD

Output — active_pr

E33   Assign Buyer

Input — orderable_pr

Process 5.1

DISPLAY to user the orderable_pr

GET from user the buyer_assignment

SET in pr the buyer_assignment

SET in pr the inbox_location to buyer

Output — assigned_pr

E4   Approvals

E41   Supervisory approval

Input — funded_pr, supervisory_approval

Process 1.4

DISPLAY to user the funded_pr

GET from user the supervisory_approval

IF supervisory_approval is ‘Yes’ THEN

SET in pr the supervisory_approval to ‘Yes’

SET in pr the request_date to today’s date

SET in pr the inbox_location to budget

ELSE

GET from user the explanation

SET in pr the supervisory_approval to ‘No’

SET in pr the explanation

SET in pr the inbox_location to requester

ENDIF

Output — completed_pr, rejected_pr
**E42 Fund source approval**

*Input* — completed_pr, fund_source_approval

*Process 2.1*

DISPLAY to user the completed_pr

GET from user the fund_source_approval

IF fund_source_approval is ‘Yes’ THEN

SET in pr the fund_source_approval to ‘Yes’

SET in pr the inbox_location to certification

ELSE

GET from user the explanation

SET in pr the fund_source_approval to ‘No’

SET in pr the explanation

SET in pr the inbox_location to requester

ENDIF

*Output* — certifiable_pr, rejected_pr

**E43 Special approval**

*Input* — certified_pr, special_approval

*Process 3.1*

DISPLAY to user the certified_pr

GET from user the special_approval

IF special_approval is ‘Yes’ THEN

SET in pr the special_approval to ‘Yes’

SET in pr the inbox_location to property_book_officer

ELSE

GET from user the explanation

SET in pr the special_approval to ‘No’

SET in pr the explanation

SET in pr the inbox_location to requester

ENDIF

*Output* — special_pr, rejected_pr
E44  Property approval

Input — special_pr, property_approval

Process 3.3

DISPLAY to user special_pr
GET from user property_approval
IF property_approval is 'Yes' THEN
  SET in pr the property_approval to 'Yes'
  SET inbox_location to contracting_officer
ELSE
  GET from user explanation
  SET in pr the property_approval to 'No'
  SET in pr the explanation
  SET in pr the inbox_location to requester
ENDIF

Output — orderable_pr, rejected_pr

E45  Actual cost approval

Input — actual_pr, actual_cost_approval, explanation

Process 2.2

DISPLAY to user actual_pr
GET from user actual_cost_approval
IF actual_cost_approval is 'Yes' THEN
  SET in pr the actual_cost_approval to 'Yes'
  SET in pr the inbox_location to buyer
ELSE
  GET from user explanation
  SET in pr the actual_cost_approval to 'No'
  SET in pr the explanation
  SET in pr the inbox_location to requester
ENDIF

Output — certifiable_pr, rejected_pr
Buyer approval

Input — assigned_pr, buyer_approval, explanation

Process 2.2

DISPLAY to user assigned_pr
GET from user the buyer_approval
IF buyer_approval is ‘Yes’ THEN
   SET in pr the buyer_approval to ‘Yes’
ELSE
   GET from user explanation
   SET in pr the buyer_approval to ‘No’
   SET in pr the explanation
   SET in pr the inbox_location to requester
ENDIF

Output — approved_pr, rejected_pr

Interface with legacy systems

Build block

Input — N/A

Process 2.3

PUT to SOMARDS the block
GET from SOMARDS the return_message

Output — N/A

Certify funds

Input — certifiable_pr

Process 2.4

GET from certifiable_pr the funds
PUT to SOMARDS the block, funds
GET from SOMARDS the return_message
IF return_message is ‘OK’ THEN
   SET in pr the certification to ‘Yes’
SET in pr the inbox_location to special_approval
ELSE
SET in pr the to certification to ‘No’
SET in pr the explanation to return_message
SET in pr the inbox_location to requester
ENDIF

Output — certified_pr, rejected_pr

E53  Reconcile

Input — N/A

Process 2.5

PUT to SOMARDS the block, reconcile
GET from SOMARDS the return_message

Output — N/A

E54  Upload to SAACONS

Input — assigned_pr

Process 5.1

GET from assigned_pr the upload
PUT to SAACONS the upload

Output — N/A

E6  Status inquires

Input — active_pr

Process 7.3

GET current status from ACTIVE
DISPLAY to user the status

Output — status

E7  Generate reports

Input — report_type

Process 7.4 — TBD

Output — report
6. Data Dictionary

While DFDs and pseudocode (structured English) are important to system specifications, additional information is required for a complete analytical model. The content of each data or control item should be more fully identified. A data dictionary is a quasi-formalism for describing content of information as it flows through the system. The standard notation conventions are

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>is composed of</td>
</tr>
<tr>
<td>+</td>
<td>and</td>
</tr>
<tr>
<td>[ ]</td>
<td>either - or</td>
</tr>
<tr>
<td>{ }^n</td>
<td>n repetitions of</td>
</tr>
<tr>
<td>( )</td>
<td>optional data</td>
</tr>
<tr>
<td>* *</td>
<td>comments</td>
</tr>
</tbody>
</table>

Contract Req prototype’s data dictionary appears in following text. Each left-hand element is taken from the DFD and the PDL representations of the system. These data items are then given an expanded, unambiguous definition in the right-hand column.
acceptance =  
*requester accepts or returns shipment item*
["Yes" | "No"]

ACTIVE =  
{active_pr}

action =  
**
TBD

action_results =  
**
TBD

active_pr =  
*purchase request at some point in the approval cycle*

actual_cost =  
*actual cost of an item*
*units: dollars*

actual_cost_approval =  
*supervisory approval of the total actual cost of the request*
["Yes" | "No"]

actual_pr =  
*purchase request where the total actual cost is greater than the total estimated cost*
approved_pr + {actual_cost} + {item_tot_act_cost} + total_actual_cost

address =  
**
street_address + (mail_stop) + city + state + postal_code + (country)

approved_pr =  
*purchase request that has procurement buyer approval*
assigned_pr + buyer_approval

assigned_pr =  
*purchase request that has been assigned to a buyer by the contracting officer*
orderable_pr + buyer_assignment

bar_code_no =  
*property book bar code number*
{alphanumeric}

batch_no =  
*SOMARDS batch number*
"Contract Req"

blank_pr =  
*purchase request with the requester and delivery info filled*
requester_userid + user_info
bldg_no =  *building number*  
{alphanumeric}

blk_no =  *SOMARDS block number*  
"ARL"

blk_tkt_dt =  *SOMARDS block ticket date*  
*format: MM/DD/YY*  
date

block =  *SOMARDS build block data*  
trns_cd + user_auth_key + cmd_dsg + update_code +  
blk_no + blk_tkt_dt + tot_blk + batch_no + tot_batch

buyer_approval =  *buyer approval of purchase request*  
["Yes" | "No"]

buyer_assignment =  **  
buyer_userid + saacons_buyer_code

buyer_userid =  **  
userid

cancel_order =  *order to cancel purchase request*  
["Yes" | "No"]

CANCELED =  {canceled_pr}

canceled_pr =  *purchase request that has been canceled*  
active_pr + cancel_order

certifiable_pr =  *purchase request that has fund source approval or actual costs approved*  
[completed_pr + fund_source Approval | actual_pr +  
actual_cost_approval]

certification =  *SOMARDS certification*  
["Yes" | "No"]

certified_pr =  *purchase request that has been certified by SOMARDS*  
certifiable_pr + certification

city =  **  
{alphanumeric_character}
CLOSED = {closed_pr}
closed_pr = *purchase request that has been accepted by the requester*
tagged_pr + {acceptance}
cmd_dsg = *SOMARDS CMD-DSG* "T"
company_address = **
address
company_email = **
email_address
company_fax_no = **
phone_no
company_name = **
{legal_character}
company_phone_no = **
phone_no
company_poc = **
name
completed_pr = *purchase request that has been approved by the supervisor* 
@doc_ref_no + funded_pr + request_date + supervisory_approval
comt_ref_no = *SOMARDS document reference number* 
doc_ref_no
corrected_pr = *purchase request that has been corrected by the requester* 
rejected_pr + corrections
corrections = *corrections to a rejected purchase request* 
TBD
country = **
{alphabetic_character}
cum_btc_h_value = *SOMARDS cumulative batch total for the days certification*
*units: dollars*

date_required = *date shipment is required by*
date

delivery_date = *estimated date for delivery from vendor*
date

description = *item description*
{legal_character}

doc_ref_no = *purchase request document reference number* "W." + **TBD

email_address = **TBD

EMPLOYEE = {employee}

employee = *employee information - the bare minimum should contain*
user info + {roles}

eor = *funding element of resource*
{alphabetic_character}

estimated_cost = *estimated cost of an item*
*units: dollars*

explanation = *rejection, cancelation, or return explanation*
{legal_character}

finished_pr = *purchase request where the total actual cost is less than or
equal to the total estimated cost*
approved_pr + {actual_cost} + {item_tot_act_cost} +
total_actual_cost

first_name = *a person's first name*
{alphabetic_character}

fund_source = **jo_no + eor
fund_source_approval = *budget analyst approval of fund source*
["Yes" | "No"]

funded_pr = *purchase request with a fund source* partial_pr + fund_source

funds = *funding information for SOMARDS certification*
trns_cd + user_auth_key + cmd_dsg + update_code + blk_no + blk_tkt_dt + batch_no + rej_rept_director + doc_ref_no + jo_no + eor + act_amt

inbox = *purchase requests requiring action from user* {active_pr}

inbox_inquiry = **
TBD

inbox_location = *current purchase request location* TBD

item = **
@line_item_no + description + unit_of_issue + qty + estimated_cost + actual_cost + item_tag + acceptance + item_tot_est_cost + item_tot_act_cost

items = **
{item} + specifications

item_tag = *property book officer inputs ‘yes’ item is taggable, receiving overwrites with bar_code_no* ["Yes" | bar_code_no]

item_tags = {item_tags}

item_tot_act_cost = *line item total actual cost* *units: dollars*

item_tot_est_cost = *line item total estimated cost* *units: dollars*

jo_no = *funding job number* {alphanumeric}

last_name = *a person’s last name* {alphabetic_character}

27
line_item_no = *line item number*
   {numeric_digit}

mail_stop = *mail stop or department*
   {legal_character}

name = **
   first_name + last_name

nomenclature = *SOMARDS certification comment field*
   {alphanumeric}

office_symbol = *ARL office symbol*
   "AMSRL-" + {alphabetic_character} + "-" +
   {alphabetic_character}

orderable_pr = *purchase request that can be ordered by procurement*
   taggable_pr + property_approval

ordered_pr = *purchase request that has been ordered*
   finished_pr + delivery_date + vendor + po_number

partial_pr = *purchase request with items and vendors filled in*
   blank_pr + date_required + priority_code + items + vendors

phone_no = *a phone number*
   {numeric_digit}

po_number = *purchase order number*
   TBD

postal_code = *postal/zip code*
   {numeric_digit}

pr_details = *details about the purchase request*
   TBD

priority_code = *request priority code*
   *range:01 - 15, 99*
   {numeric_digit}

property_approval = *property book officer approval*
   ["Yes" | "No"]
qty = *quantity requested* 
{numeric_digit}

receipt = *shipment receipt* 
["Yes" | "No"]

received_pr = *purchase request that has been received* 
ordered_pr + receipt

reconcile = *end of the day SOMARDS reconcile info* 
trns_cd + user_auth_key + cmd_dsg + update_code + 
blk_no + blk_tkt_dt + batch_no + totblk + tot_batch + 
ty_act_cd + cum_bch_value + variance

rejected_pr = *purchase request that has been rejected* 
active_pr + explanation

rej_rept_director = *SOMARDS REJ-REPT-DIRECTOR* 
"R"

report = ** 
TBD

report_type = *type of report to generate* 
TBD

request_date = *date purchase request was approved by supervisor* 
date

requester_userid = ** 
userid

return_message = *message returned from SOMARDS process* 
["processing complete" | "bad user_auth_key" | "wrong 
update code" | "blk_no/blk_tkt_dt already exists" | 
"accounting class displayed" | "blk_no/blk_tkt_dt doesn’t 
exist" | "invalid jo_no" | "invalid eor" | "insufficient funds" 
| "duplicate comt_ref_no" | "cum_bch_value" | "make 
changes" | "variance" ]

role = *user role* 
{alphanumeric}

room_no = *room number* 
{alphanumeric}
sole_source_just = *justification for using a single vendor* 
{legal_character}

special_approval = *approval from a special approving officials* 
["Yes" | "No"]

special_pr = *purchase request with special approvals* 
certified_pr + special_approval

state = *state or province* 
{legal_character}

status = **
TBD

status_inquiry = **
TBD

street_address = ** 
{legal_character}

supervisory_approval = *approval from supervisor* 
["Yes" | "No"]

taggable_pr = *purchase request that has had item tags attached by property book officer* 
special_pr + item_tags

tagged_pr = *purchase request that has been received and taggable items have been appropriately tagged* 
received_pr + item_tags

total_actual_cost = *the total actual cost of the purchase request* 
*units: dollars*

total_estimated_cost = *the total estimated cost of the purchase request* 
*units: dollars*

tot_batch = *SOMARDS batch number* 
*units: dollars* 
["0.00" | cum_btch_value]

tot_blk = *SOMARDS total block* 
*units: dollars* 
["0.00" | cum_btch_value]
trns_cd = *SOMARDS transaction code*
["003" | "004" | "310"]

ty_act_cd = *SOMARDS action code*
"C"

unit_of_issue = **
TBD

update_code = *SOMARDS update code*
["CM" | "NM"]

user_auth_key = *SOMARDS user authorization key*
{alphanumeric}

user_info = *user information*
name + office_symbol + phone_no + bldg_no + room_no

userid = **
{alphanumeric}

variance = *SOMARDS variance between tot_batch and
cum_butch_value - should be 0.00*
*units: dollars*

vendor = *vendor information*
company_name + company_address + company_phone_no
+ company_fax_no + company_poc + company_email

vendors = *up to three suggested vendors*
{vendor} + sole_source_justification
INTENTIONALLY LEFT BLANK.
7. References


INTENTIONALLY LEFT BLANK.
<table>
<thead>
<tr>
<th>NO. OF COPIES</th>
<th>ORGANIZATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>DEFENSE TECHNICAL INFORMATION CENTER DTIC DDA 8725 JOHN J KINGMAN RD STE 0944 FT BELVOIR VA 22060-6218</td>
</tr>
<tr>
<td>1</td>
<td>HQDA DAMO FDQ D SCHMIDT 400 ARMY PENTAGON WASHINGTON DC 20310-0460</td>
</tr>
<tr>
<td>1</td>
<td>OSD OUSD(A&amp;T)/ODDDR&amp;E(R) R J TREW THE PENTAGON WASHINGTON DC 20301-7100</td>
</tr>
<tr>
<td>1</td>
<td>DPTY CG FOR RDE HQ US ARMY MATERIEL CMD AMCRD MG CALDWELL 5001 EISENHOWER AVE ALEXANDRIA VA 22333-0001</td>
</tr>
<tr>
<td>1</td>
<td>INST FOR ADVANCED TCHNGLY THE UNIV OF TEXAS AT AUSTIN PO BOX 202797 AUSTIN TX 78720-2797</td>
</tr>
<tr>
<td>1</td>
<td>DARPA B KASPAR 3701 N FAIRFAX DR ARLINGTON VA 22203-1714</td>
</tr>
<tr>
<td>1</td>
<td>NAVAL SURFACE WARFARE CTR CODE B07 J PENNELLA 17320 DAHLGREN RD BLDG 1470 RM 1101 DAHLGREN VA 22448-5100</td>
</tr>
<tr>
<td>1</td>
<td>US MILITARY ACADEMY MATH SCI CTR OF EXCELLENCE DEPT OF MATHEMATICAL SCI MAJ M D PHILLIPS THAYER HALL WEST POINT NY 10996-1786</td>
</tr>
<tr>
<td>1</td>
<td>DIRECTOR US ARMY RESEARCH LAB AMSRL DD J J ROCCHIO 2800 POWDER MILL RD ADELPHI MD 20783-1145</td>
</tr>
<tr>
<td>1</td>
<td>DIRECTOR US ARMY RESEARCH LAB AMSRL CS AS (RECORDS MGMT) 2800 POWDER MILL RD ADELPHI MD 20783-1145</td>
</tr>
<tr>
<td>3</td>
<td>DIRECTOR US ARMY RESEARCH LAB AMSRL CI LL 2800 POWDER MILL RD ADELPHI MD 20783-1145</td>
</tr>
<tr>
<td>4</td>
<td>DIR USARL AMSRL CI LP (305)</td>
</tr>
</tbody>
</table>

ABERDEEN PROVING GROUND
8
DIRECTOR
US ARMY RESEARCH LAB
AMSRL IS
J GRILLS
AMSRL CI A
R ROSEN (2 CPS)
AMSRL CS
COL K LOGAN
AMSRL CI E
W JERNIGAN
W MCCOY
B SCHALLHORN
AMSRL SE S
A SINDORIS
2800 POWDER MILL RD
ADELPHI MD 20783-1145

1
DIRECTOR
US ARMY RESEARCH LAB
AMSRL EA
P STAY
WSMR NM 88002-5501

ABERDEEN PROVING GROUND

14
DIR USARL
AMSRL CI
   N RADHAKRISHNAN
AMSRL CI E
   D ULERY (10 CPS)
AMSRL HR SF
   B AMREIN
AMSRL WM IM
   R MCGEE
AMSRL CS AP
   D ORE
### Report Documentation Page

**Title and Subtitle:** Contract Req Software Requirements Analysis Version 1: A C-BASS Component

**Author(s):** Denis McGurin

**Performing Organization Name(s) and Address(es):** U.S. Army Research Laboratory, ATTN: AMSRL-CI-E, Aberdeen Proving Ground, MD 21005-5067

**Funding Numbers:** AC9UA4EC

**Performer Organization Report Number:** ARL-MR-458

**Sponsoring/Monitoring Agency Name(s) and Address(es):**

**Supplementary Notes:**

**Distribution/Availability Statement:** Approved for public release; distribution is unlimited.

**Abstract (Maximum 200 words):**

This document contains the software requirements analysis for a prototype of Contract Request Version 1.0 (Contract Req). As a component of the Corporate Business Application Software System (C-BASS), this application automates the preparation of the requester's Procurement Data Package (PDP). The document follows the process of structured analysis, or step-wise refinement of requirements, as applied to the development of Contract Req. The "environmental model" includes a high-level system description, followed by a context diagram and a list of events to which the system must respond. The "behavioral model" includes a data flow diagram (DFD) for each of the five Contract Req subsystems. From this representation, the basic functional specifications are derived and represented in structured English (or program design language). The final segment of the document includes a data dictionary.
INTENTIONALLY LEFT BLANK.
USER EVALUATION SHEET/CHANGE OF ADDRESS

This Laboratory undertakes a continuing effort to improve the quality of the reports it publishes. Your comments/answers to the items/questions below will aid us in our efforts.

1. ARL Report Number/Author  ARL-MR-458 (McGurin)  Date of Report  August 1999

2. Date Report Received

3. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which the report will be used.)

4. Specifically, how is the report being used? (Information source, design data, procedure, source of ideas, etc.)

5. Has the information in this report led to any quantitative savings as far as man-hours or dollars saved, operating costs avoided, or efficiencies achieved, etc? If so, please elaborate.

6. General Comments. What do you think should be changed to improve future reports? (Indicate changes to organization, technical content, format, etc.)

Organization

CURRENT ADDRESS
Name
Street or P.O. Box No.
City, State, Zip Code

7. If indicating a Change of Address or Address Correction, please provide the Current or Correct address above and the Old or Incorrect address below.

Organization

OLD ADDRESS
Name
Street or P.O. Box No.
City, State, Zip Code

(Remove this sheet, fold as indicated, tape closed, and mail.)
(DO NOT STAPLE)