FIG. 1

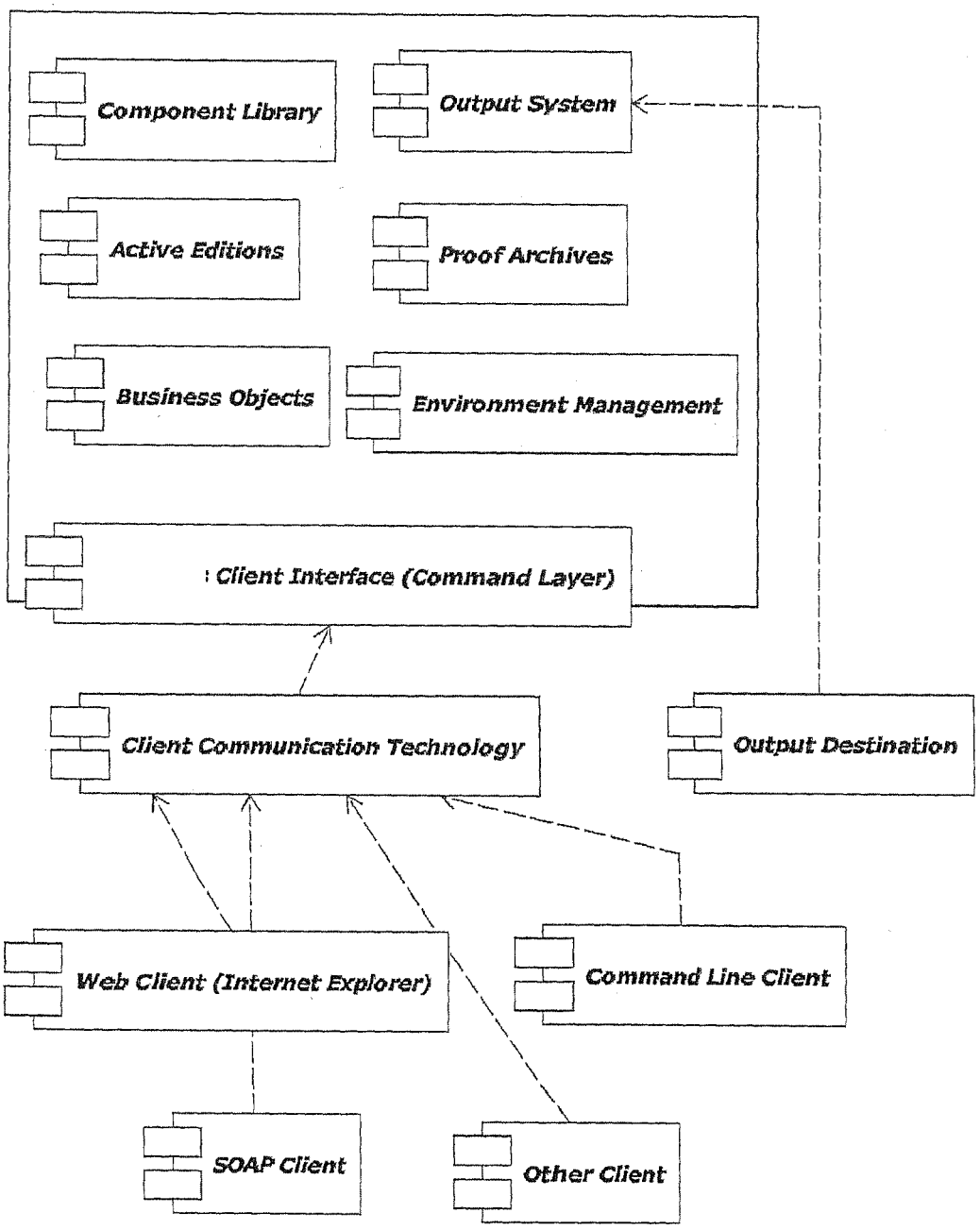

Component Library
Output System
Active Editions
Proof Archives
Business Objects
Environment Management
: Client Interface (Command Layer)
Client Communication Technology
Output Destination
Web Client (Internet Explorer)
Command Line Client
SOAP Client
Other Client

FIG. 2

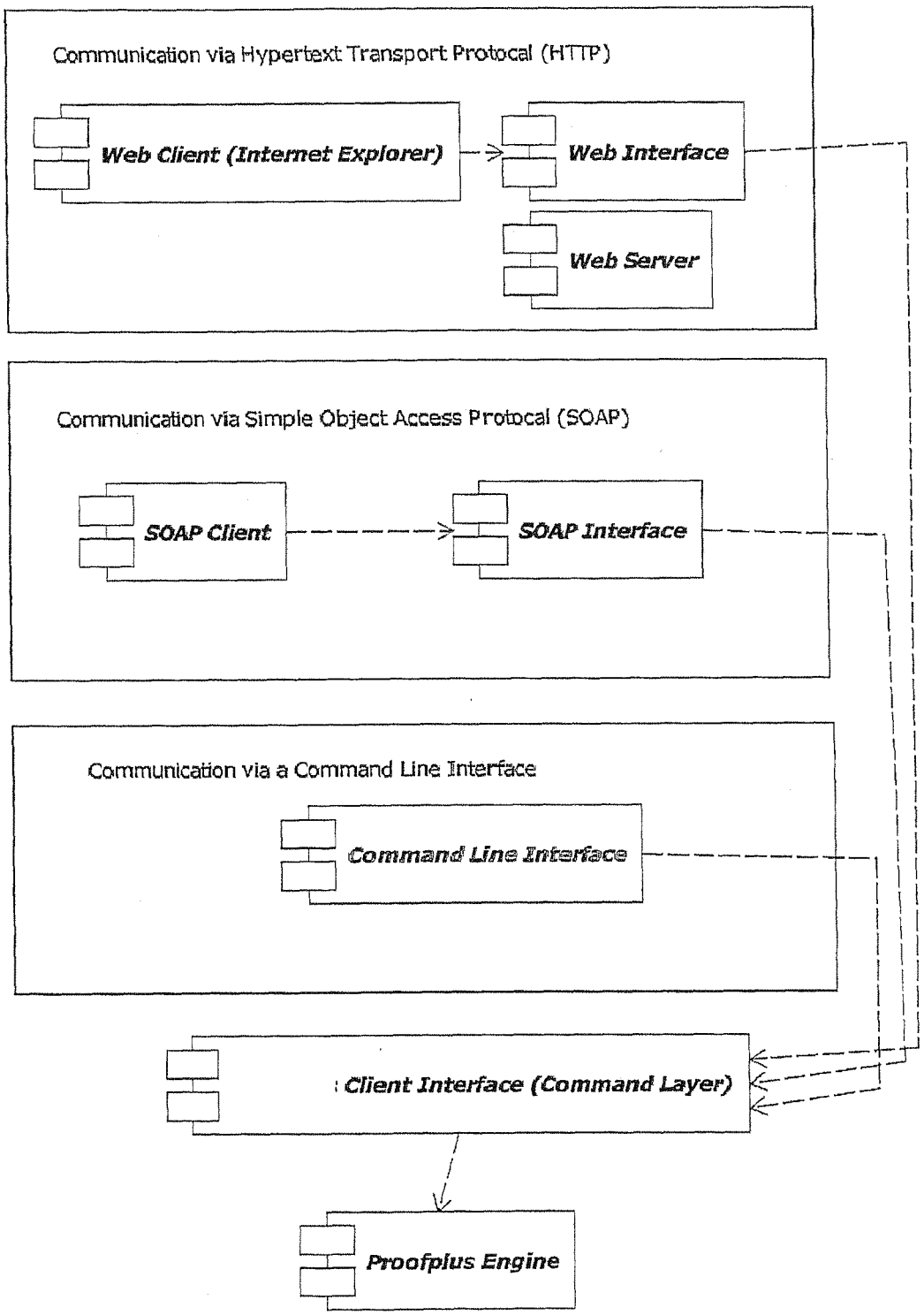

Communication via Hypertext Transport Protocal (HTTP)
Web Client (Internet Explorer)
Web Interface
Web Server
Communication via Simple Object Access Protocal (SOAP)
SOAP Client
SOAP Interface
Communication via a Command Line Interface
Command Line Interface
Client Interface (Command Layer)
Proofplus Engine

FIG. 3

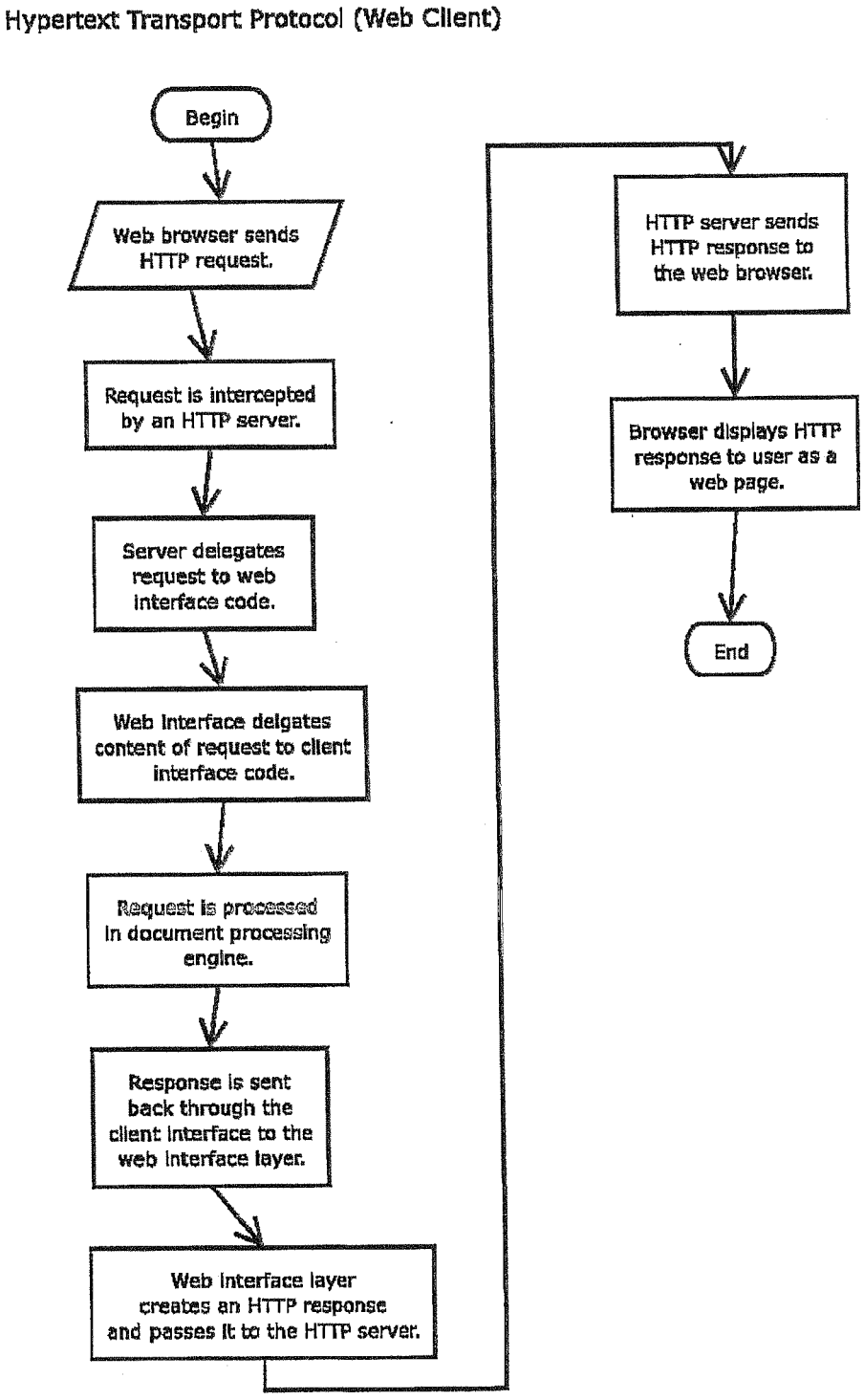

Hypertext Transport Protocol (Web Client)
Begin
Web browser sends HTTP request.
Request is intercepted by an HTTP server.
Server delegates request to web interface code.
Web Interface delgates content of request to client interface code.
Request is processed in document processing engine.
Response is sent back through the client interface to the web interface layer.
Web interface layer creates an HTTP response and passes it to the HTTP server.
HTTP server sends HTTP response to the web browser.
Browser displays HTTP response to user as a web page.
End

FIG. 4

Simple Object Access Protocol (SOAP)

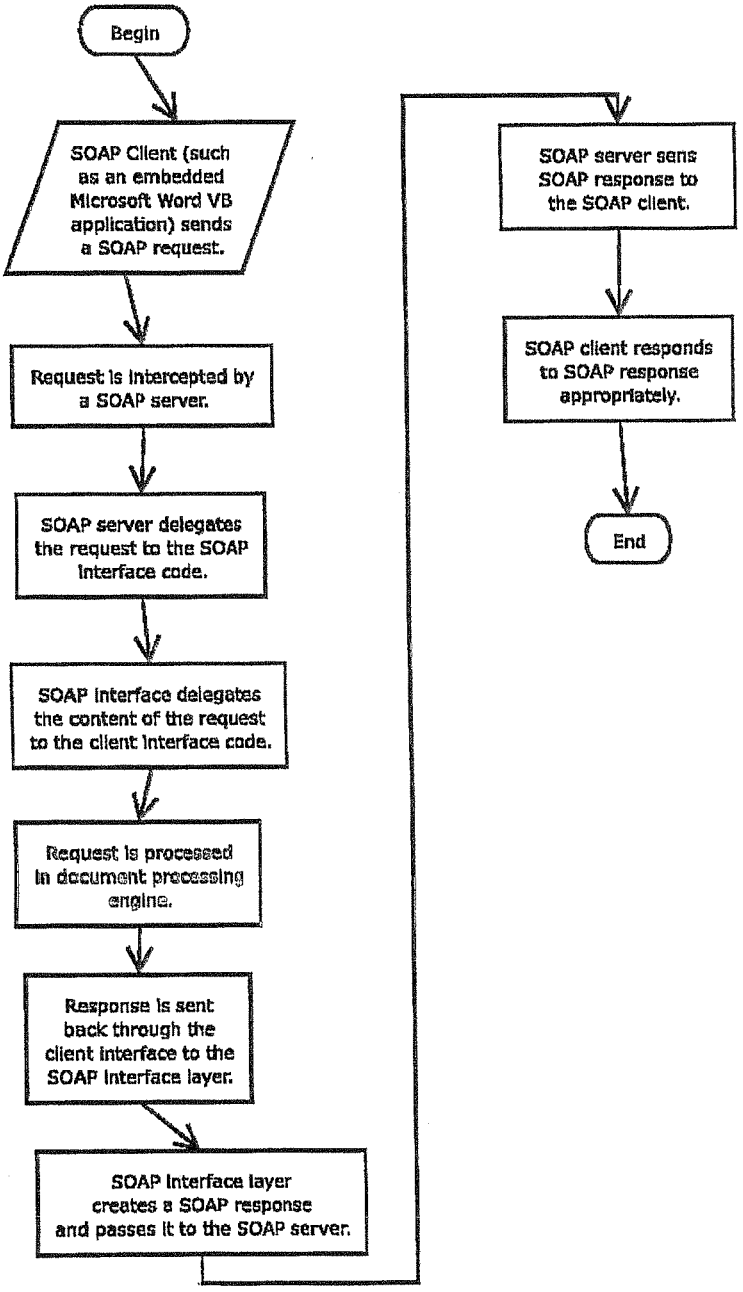


FIG. 5

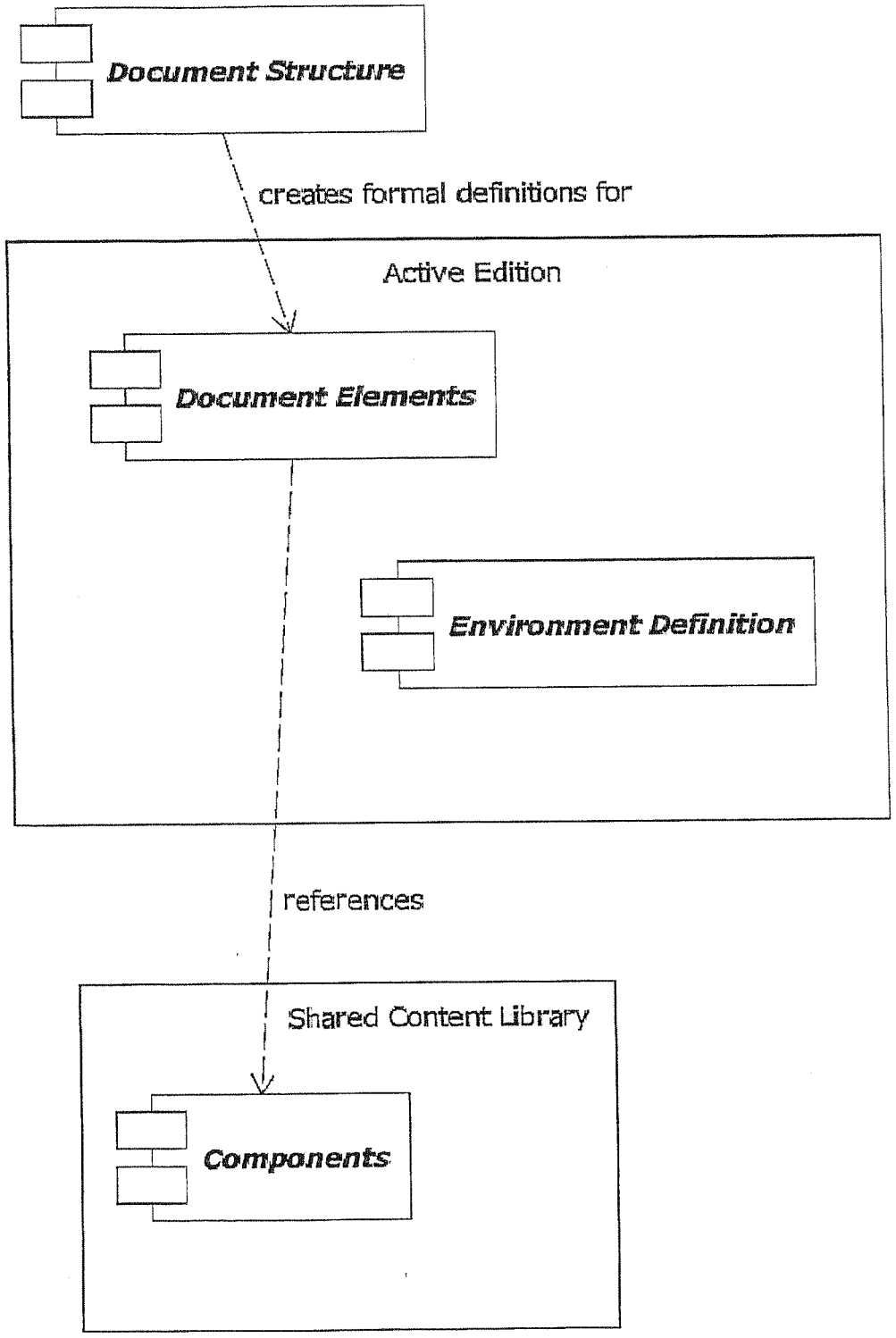

Document Structure
creates formal definitions for
Active Edition
Document Elements
Environment Definition
references
Shared Content Library
Components

FIG. 6

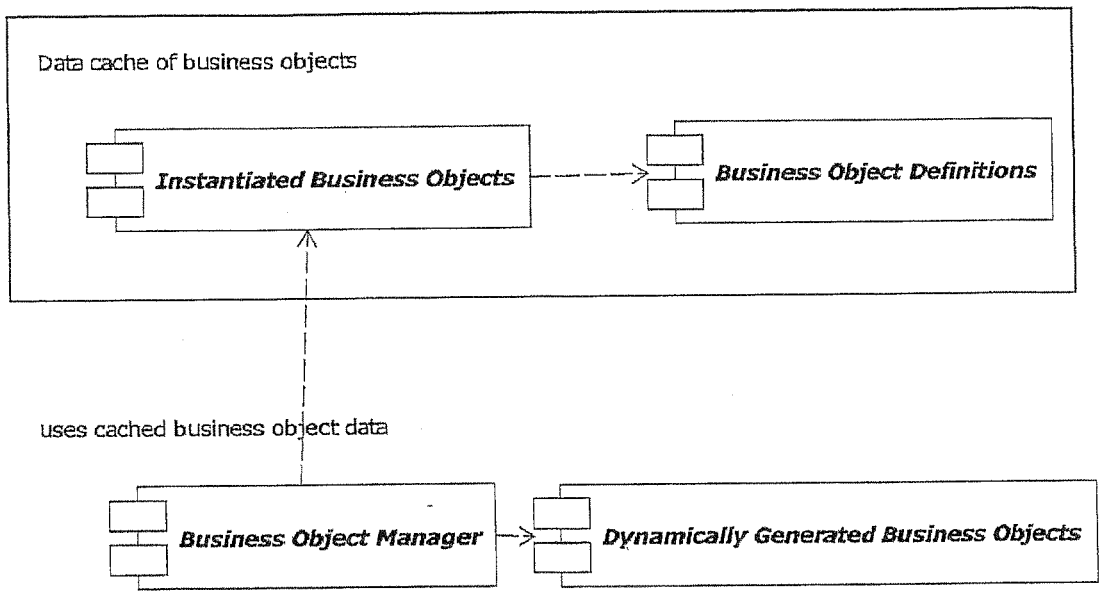

Data cache of business objects
Instantiated Business Objects
Business Object Definitions
uses cached business object data
Business Object Manager
Dynamically Generated Business Objects
dynamically creates complex business objects

FIG. 7

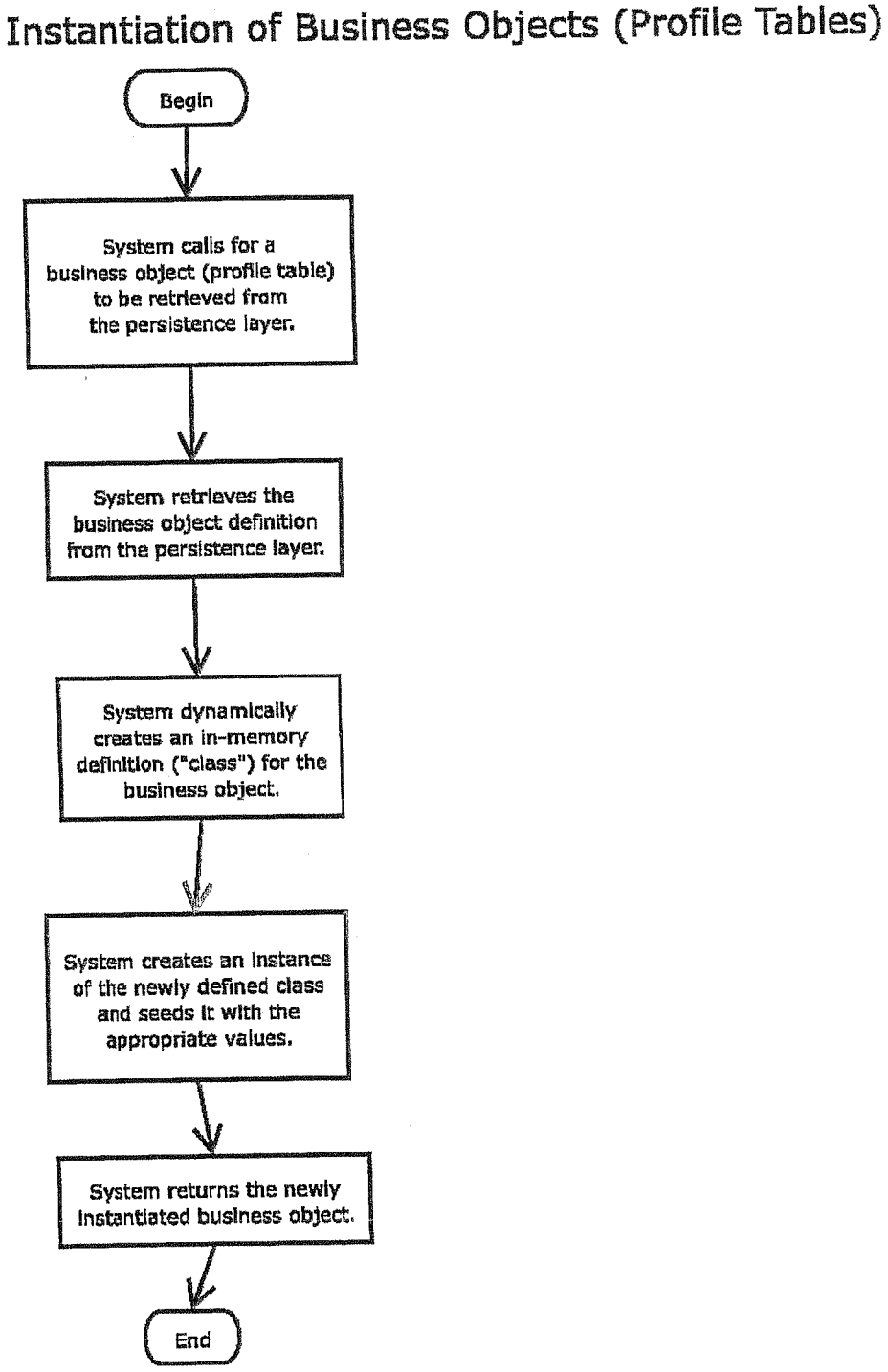

Instantiation of Business Objects (Profile Tables)
Begin
System calls for a business object (profile table) to be retrieved from the persistence layer.
System retrieves the business object definition from the persistence layer.
System dynamically creates an in-memory definition ("class") for the business object.
System creates an instance of the newly defined class and seeds it with the appropriate values.
System returns the newly instantiated business object.
End

FIG. 8

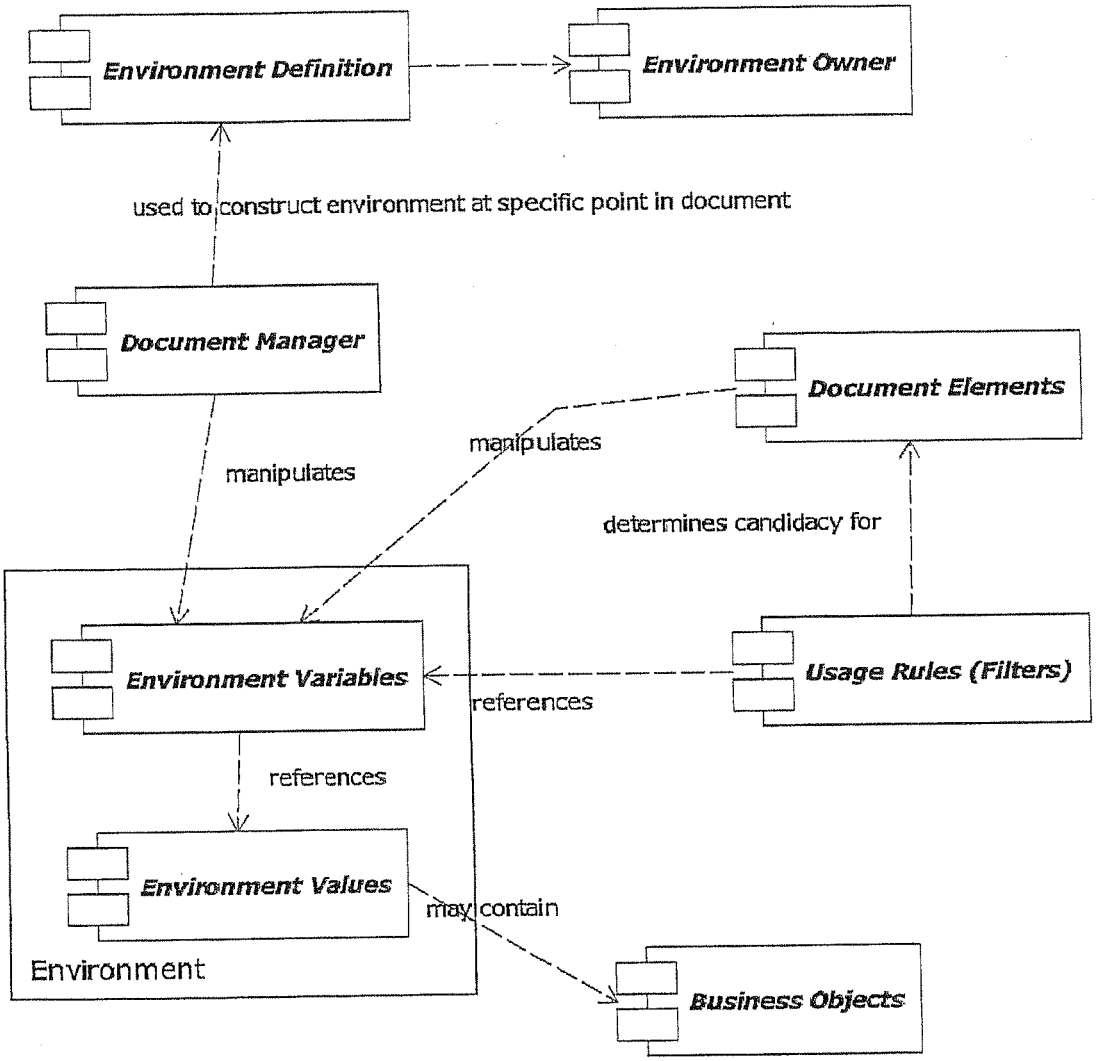

Environment Definition
Environment Owner
used to construct environment at specific point in document
Document Manager
Document Elements
manipulates
manipulates
determines candidacy for
Environment Variables
references
Usage Rules (Filters)
references
Environment Values
may contain
Environment
Business Objects

FIG. 9

Environment Management for Active Editions

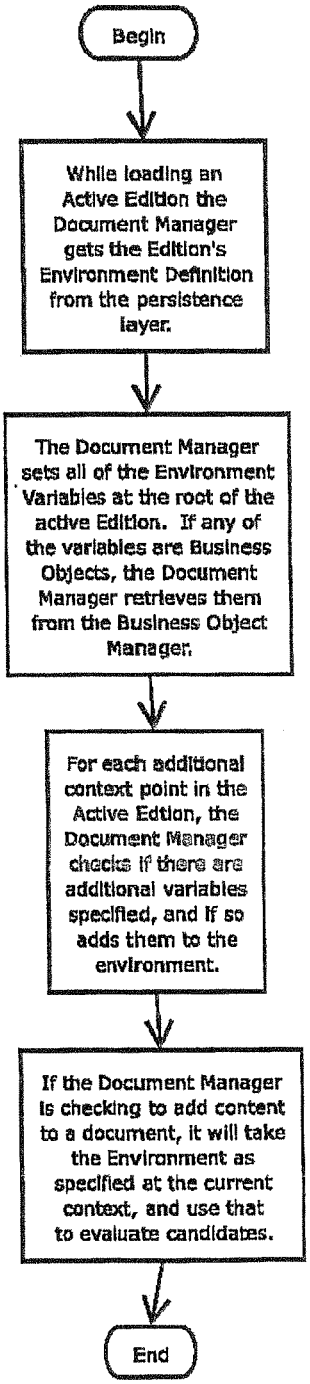


**FIG. 10**

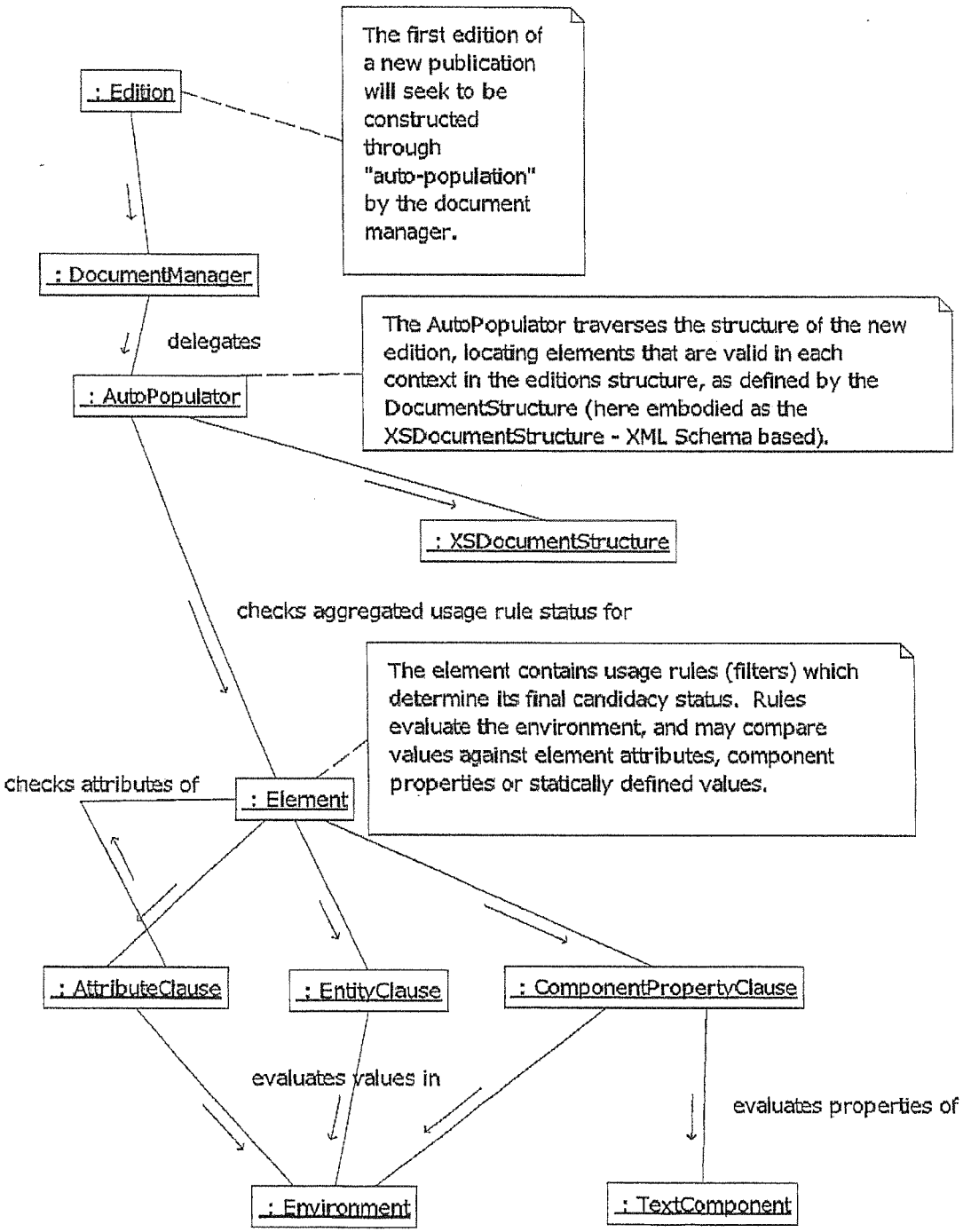

The first edition of a new publication will seek to be constructed through "auto-population" by the document manager.
: Edition
: DocumentManager
delegates
: AutoPopulator
The AutoPopulator traverses the structure of the new edition, locating elements that are valid in each context in the editions structure, as defined by the DocumentStructure (here embodied as the XSDocumentStructure - XML Schema based).
: XSDocumentStructure
checks aggregated usage rule status for
The element contains usage rules (filters) which determine its final candidacy status. Rules evaluate the environment, and may compare values against element attributes, component properties or statically defined values.
checks attributes of
: Element
: AttributeClause
: EntityClause
: ComponentPropertyClause
evaluates values in
evaluates properties of
: Environment
: TextComponent

FIG. 11

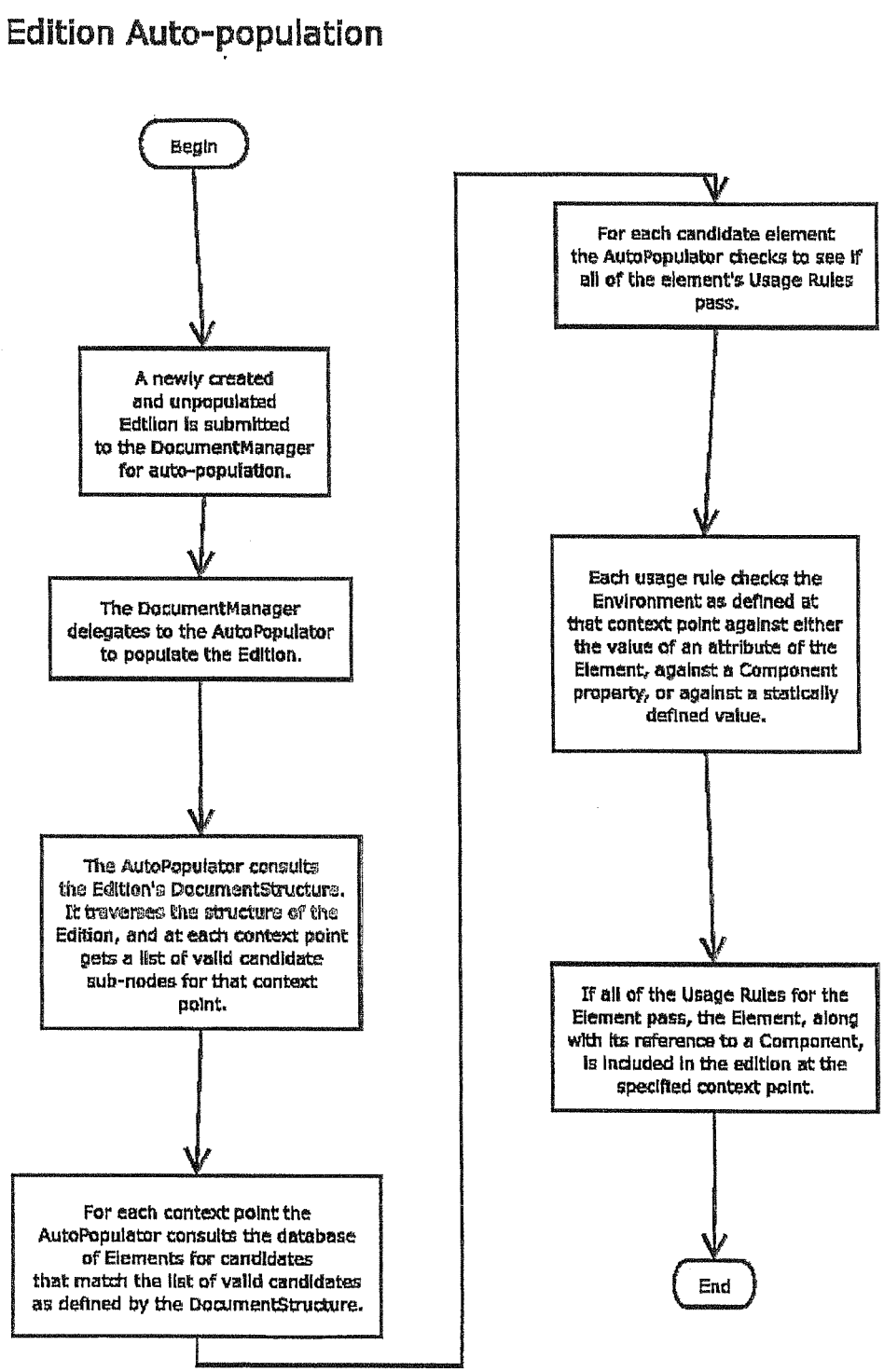

Edition Auto-population
Begin
A newly created and unpopulated Edition is submitted to the DocumentManager for auto-population.
The DocumentManager delegates to the AutoPopulator to populate the Edition.
The AutoPopulator consults the Edition's DocumentStructure. It traverses the structure of the Edition, and at each context point gets a list of valid candidate sub-nodes for that context point.
For each context point the AutoPopulator consults the database of Elements for candidates that match the list of valid candidates as defined by the DocumentStructure.
For each candidate element the AutoPopulator checks to see if all of the element's Usage Rules pass.
Each usage rule checks the Environment as defined at that context point against either the value of an attribute of the Element, against a Component property, or against a statically defined value.
If all of the Usage Rules for the Element pass, the Element, along with its reference to a Component, is included in the edition at the specified context point.
End

**FIG. 12**

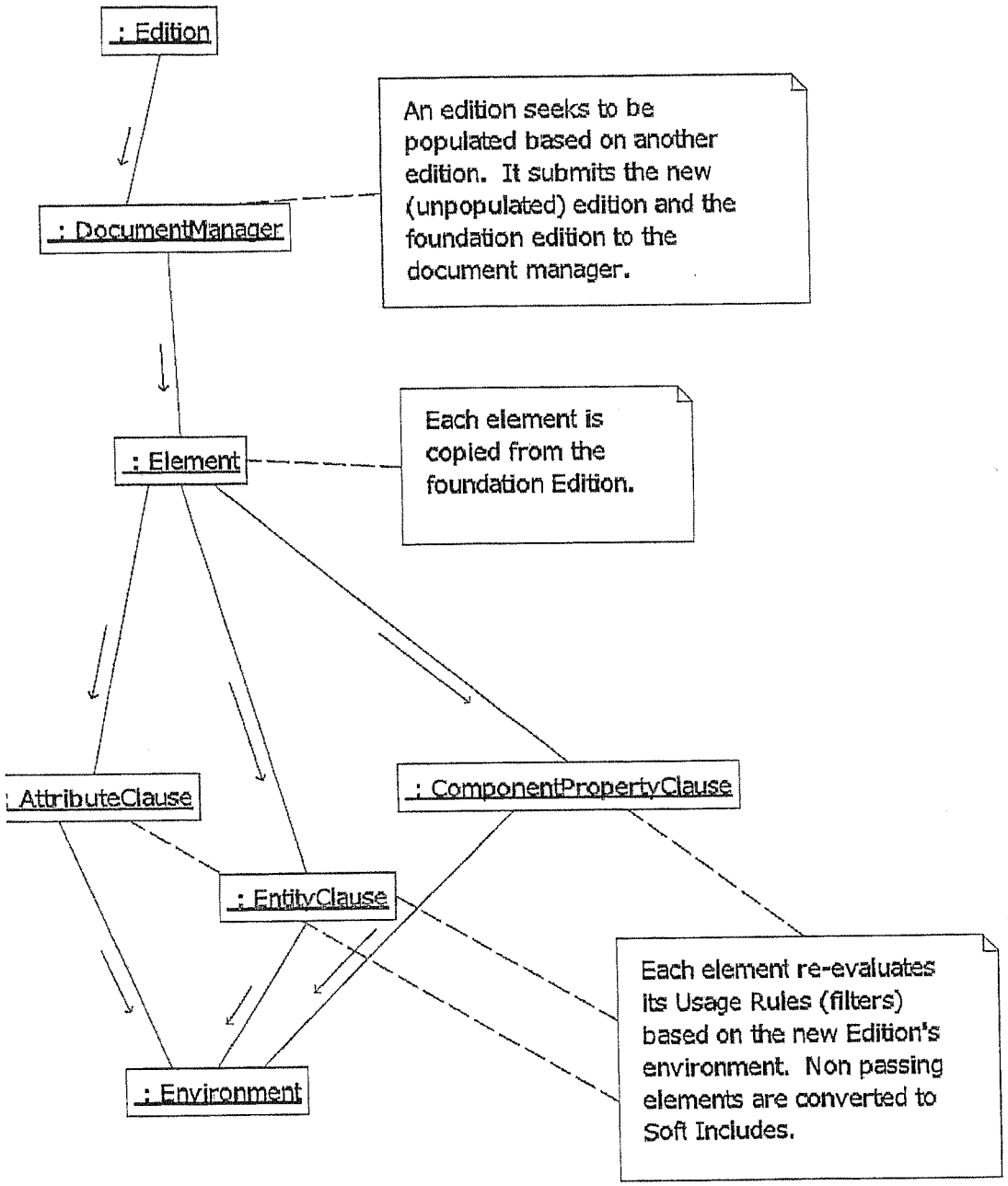

: Edition
An edition seeks to be populated based on another edition. It submits the new (unpopulated) edition and the foundation edition to the document manager.
: DocumentManager
: Element
Each element is copied from the foundation Edition.
: AttributeClause
: ComponentPropertyClause
: EntityClause
Each element re-evaluates its Usage Rules (filters) based on the new Edition's environment. Non passing elements are converted to Soft Includes.
: Environment

FIG. 13

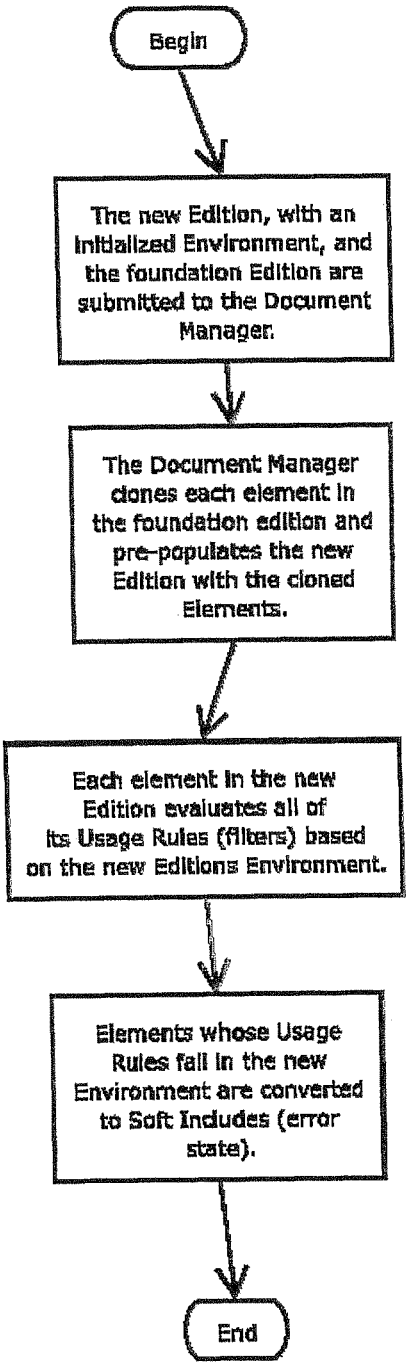

Begin
The new Edition, with an initialized Environment, and the foundation Edition are submitted to the Document Manager.
The Document Manager clones each element in the foundation edition and pre-populates the new Edition with the cloned Elements.
Each element in the new Edition evaluates all of its Usage Rules (filters) based on the new Editions Environment.
Elements whose Usage Rules fail in the new Environment are converted to Soft Includes (error state).
End

**FIG. 14**

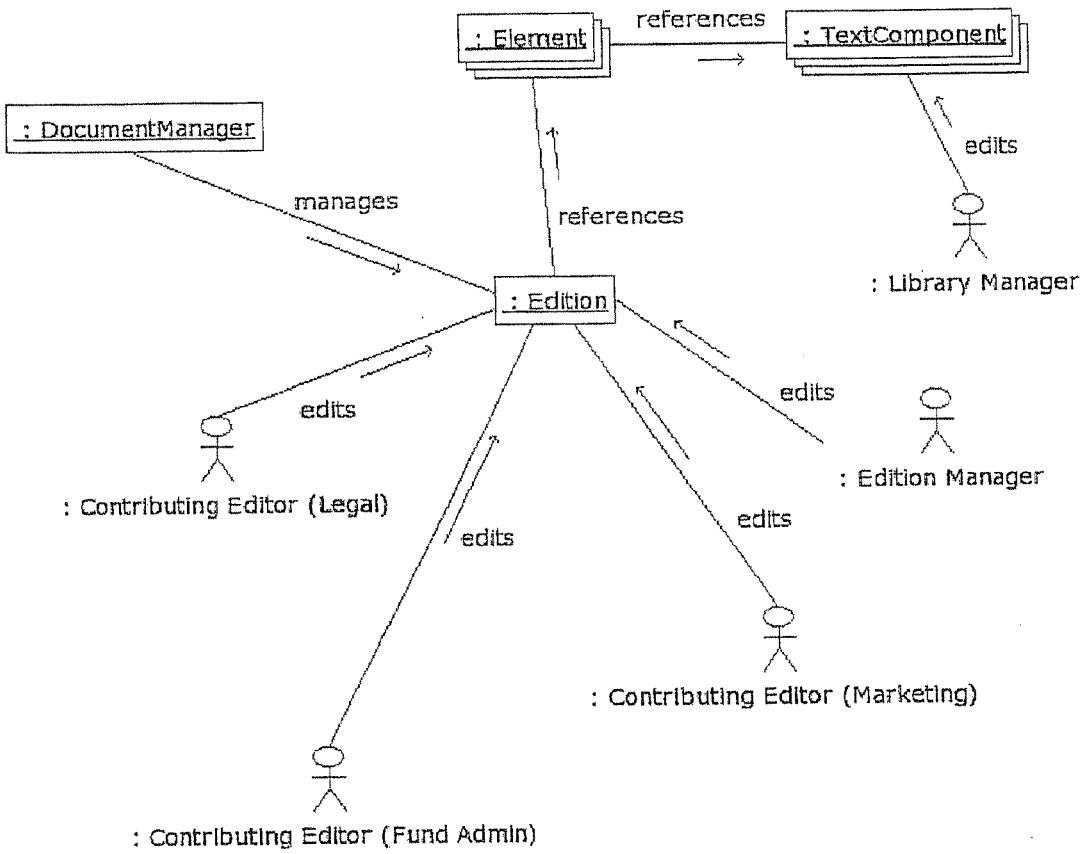
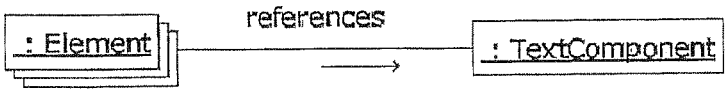

: Element
references
: TextComponent
: DocumentManager
edits
manages
references
: Library Manager
: Edition
edits
edits
: Edition Manager
: Contributing Editor (Legal)
edits
edits
: Contributing Editor (Marketing)
: Contributing Editor (Fund Admin)

FIG. 15




Many elements may have references to the same Component. Changes to Component properties or to its Content will be reflected in all of the Elements that reference that Component.

Element attributes are modified at the element level: they are unique to each individual Element.

Component properties are modified at the component level: they are shared amongst all referencing elements.

FIG. 16

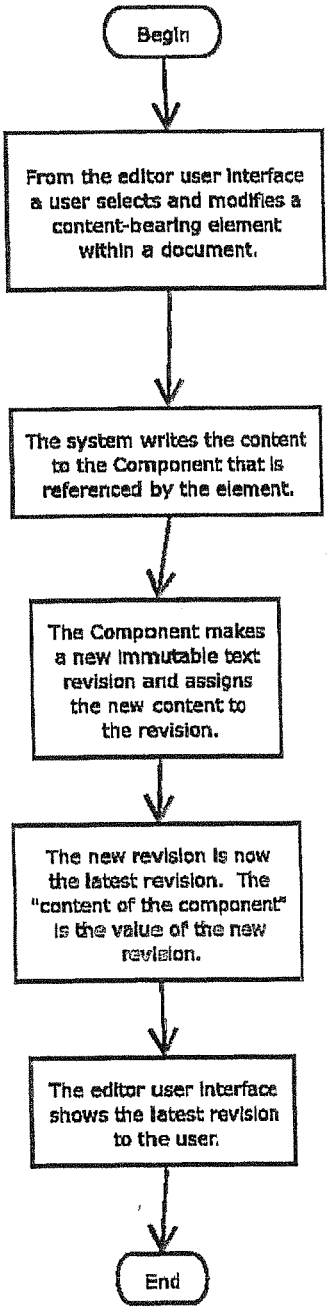

Begin
From the editor user interface a user selects and modifies a content-bearing element within a document.
The system writes the content to the Component that is referenced by the element.
The Component makes a new immutable text revision and assigns the new content to the revision.
The new revision is now the latest revision. The "content of the component" is the value of the new revision.
The editor user interface shows the latest revision to the user.
End

FIG. 17

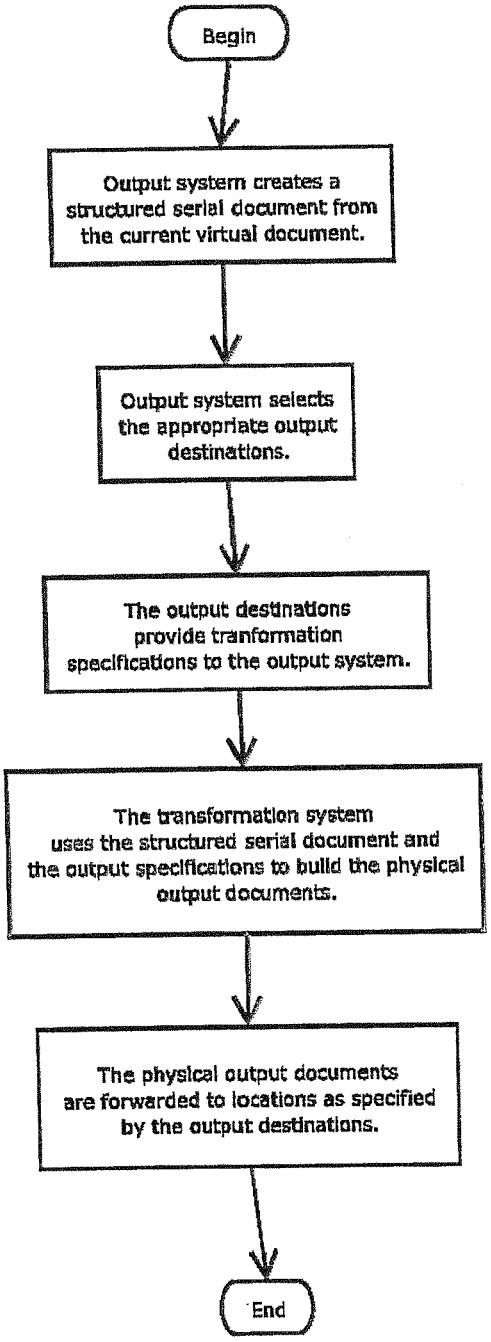

Begin
Output system creates a structured serial document from the current virtual document.
Output system selects the appropriate output destinations.
The output destinations provide tranformation specifications to the output system.
The transformation system uses the structured serial document and the output specifications to build the physical output documents.
The physical output documents are forwarded to locations as specified by the output destinations.
End

FIG. 18

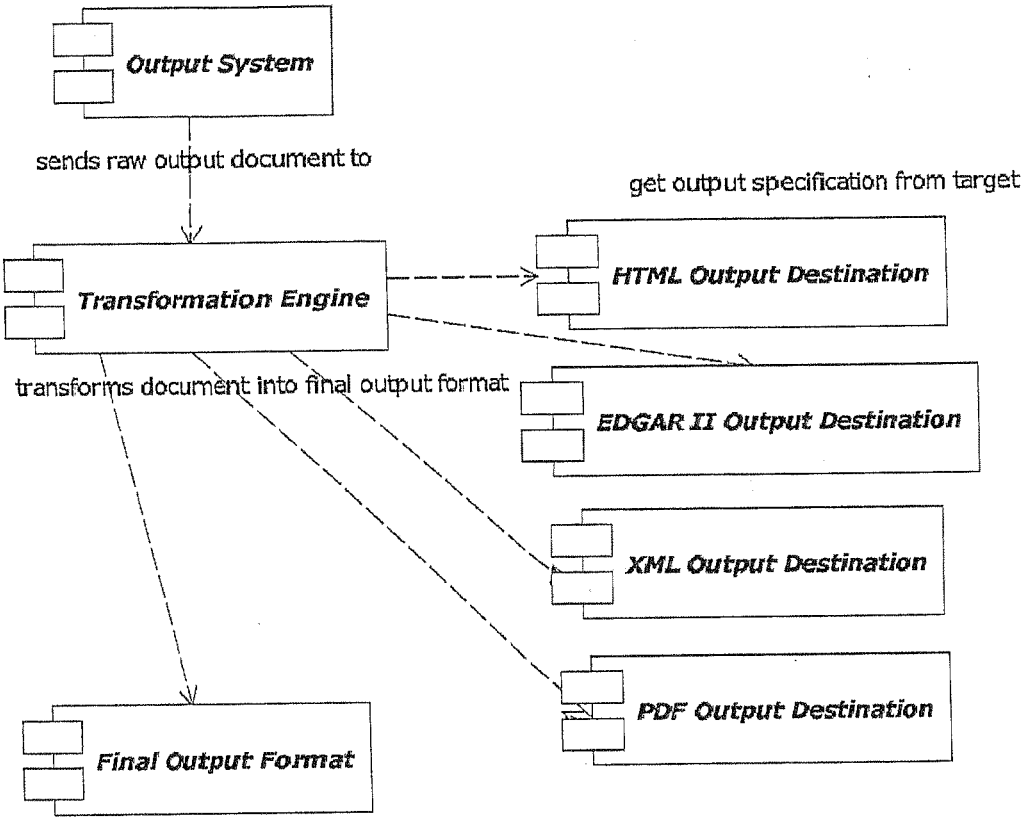

Output System
sends raw output document to
get output specification from target
Transformation Engine
HTML Output Destination
transforms document into final output format
EDGAR II Output Destination
XML Output Destination
PDF Output Destination
Final Output Format

FIG. 19

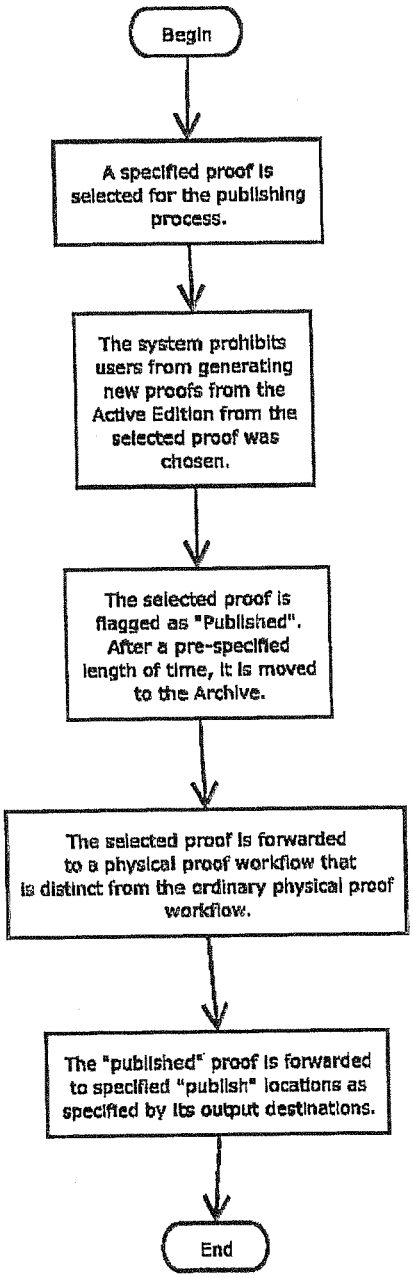

Begin
A specified proof is selected for the publishing process.
The system prohibits users from generating new proofs from the Active Edition from the selected proof was chosen.
The selected proof is flagged as "Published". After a pre-specified length of time, it is moved to the Archive.
The selected proof is forwarded to a physical proof workflow that is distinct from the ordinary physical proof workflow.
The "published" proof is forwarded to specified "publish" locations as specified by its output destinations.
End

**FIG. 20**

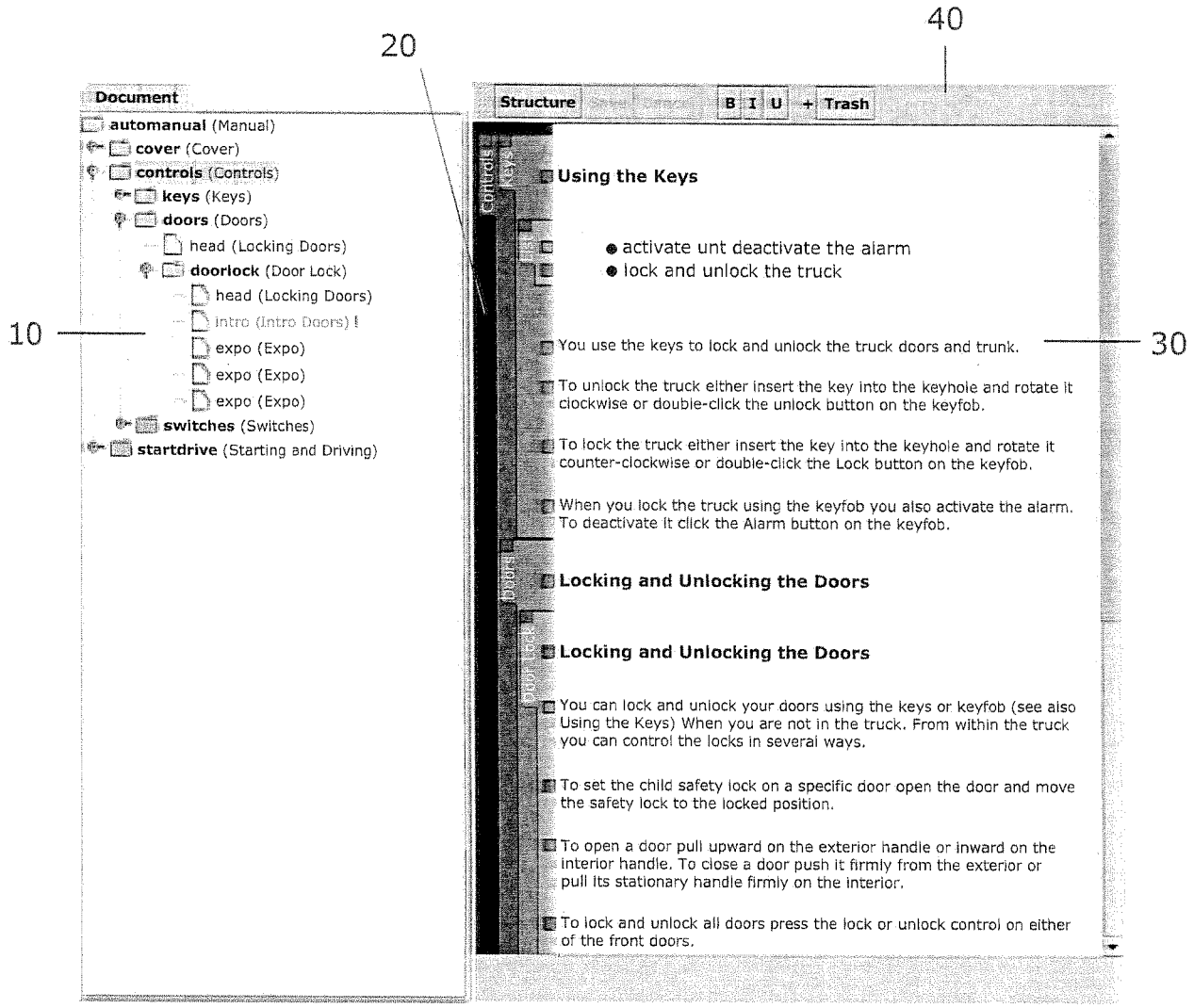

20
40
10
30
Document
automanual (Manual)
cover (Cover)
controls (Controls)
keys (Keys)
doors (Doors)
head (Locking Doors)
doorlock (Door Lock)
head (Locking Doors)
intro (Intro Doors) !
expo (Expo)
expo (Expo)
expo (Expo)
switches (Switches)
startdrive (Starting and Driving)
Structure
B I U
+ Trash
Controls
Keys
Doors
Door Lock
Using the Keys
activate unt deactivate the alarm
lock and unlock the truck
You use the keys to lock and unlock the truck doors and trunk.
To unlock the truck either insert the key into the keyhole and rotate it clockwise or double-click the unlock button on the keyfob.
To lock the truck either insert the key into the keyhole and rotate it counter-clockwise or double-click the Lock button on the keyfob.
When you lock the truck using the keyfob you also activate the alarm. To deactivate it click the Alarm button on the keyfob.
Locking and Unlocking the Doors
Locking and Unlocking the Doors
You can lock and unlock your doors using the keys or keyfob (see also Using the Keys) When you are not in the truck. From within the truck you can control the locks in several ways.
To set the child safety lock on a specific door open the door and move the safety lock to the locked position.
To open a door pull upward on the exterior handle or inward on the interior handle. To close a door push it firmly from the exterior or pull its stationary handle firmly on the interior.
To lock and unlock all doors press the lock or unlock control on either of the front doors.

FIG. 21

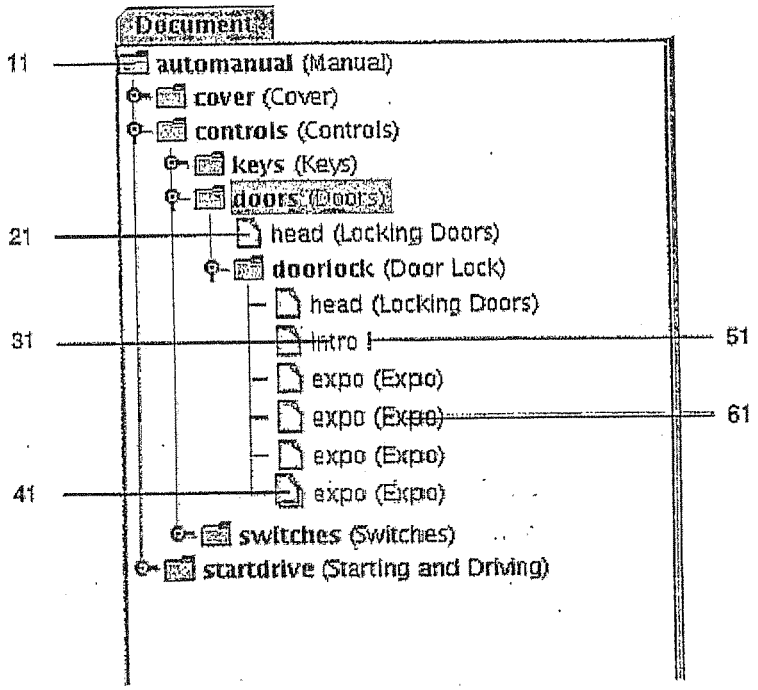
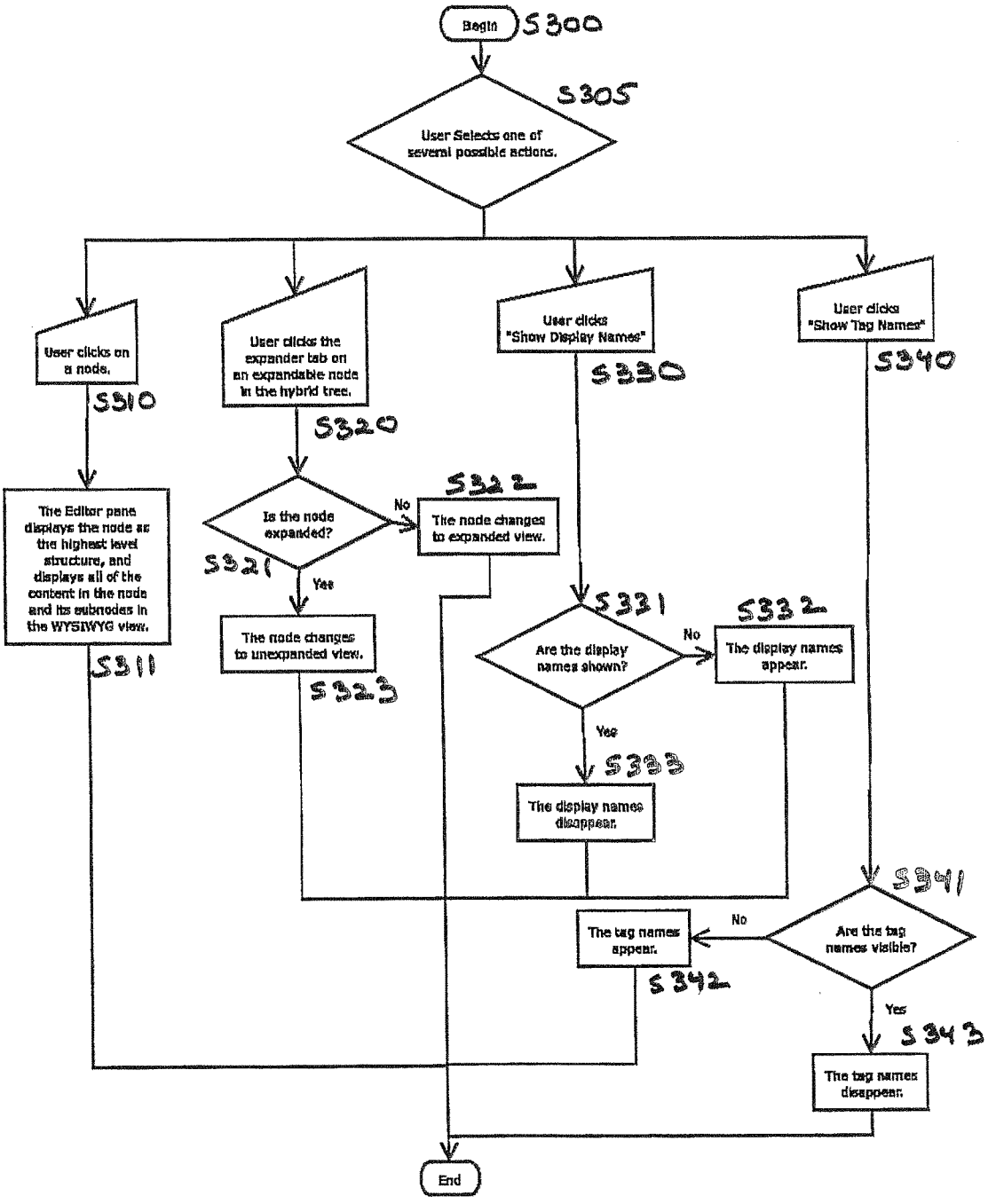

Document
11
automanual (Manual)
cover (Cover)
controls (Controls)
keys (Keys)
doors (Doors)
21
head (Locking Doors)
doorlock (Door Lock)
head (Locking Doors)
31
intro !
51
expo (Expo)
expo (Expo)
61
expo (Expo)
41
expo (Expo)
switches (Switches)
startdrive (Starting and Driving)

FIG. 22


Begin
S300
S305
User Selects one of several possible actions.
User clicks on a node.
S310
User clicks the expander tab on an expandable node in the hybrid tree.
S320
User clicks "Show Display Names"
S330
User clicks "Show Tag Names"
S340
The Editor pane displays the node as the highest level structure, and displays all of the content in the node and its subnodes in the WYSIWYG view.
S311
Is the node expanded?
S321
No
S322
The node changes to expanded view.
Yes
The node changes to unexpanded view.
S323
S331
Are the display names shown?
No
S332
The display names appear.
Yes
S333
The display names disappear.
S341
Are the tag names visible?
No
The tag names appear.
S342
Yes
S343
The tag names disappear.
End

# FIG. 23

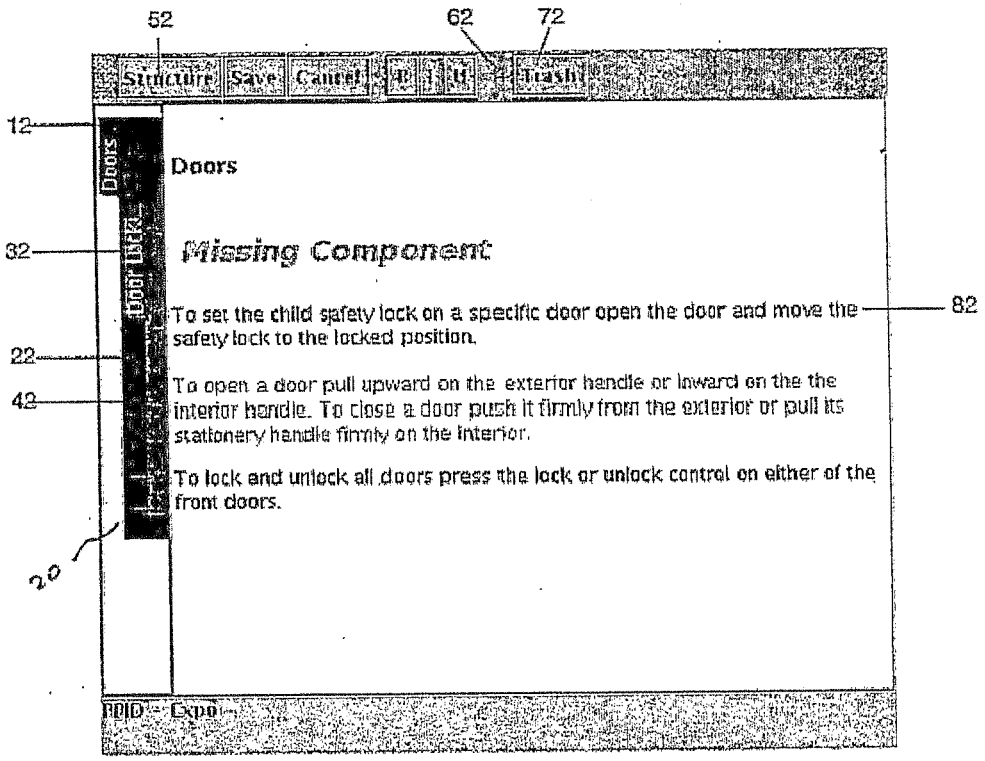

52
62
72
Structure
Save
Cancel
Trash
12
Doors
Door Locks
32
Doors
Missing Component
To set the child safety lock on a specific door open the door and move the safety lock to the locked position.
82
22
To open a door pull upward on the exterior handle or inward on the the interior handle. To close a door push it firmly from the exterior or pull its stationery handle firmly on the interior.
42
To lock and unlock all doors press the lock or unlock control on either of the front doors.
20
RDID - Expo

FIG. 24

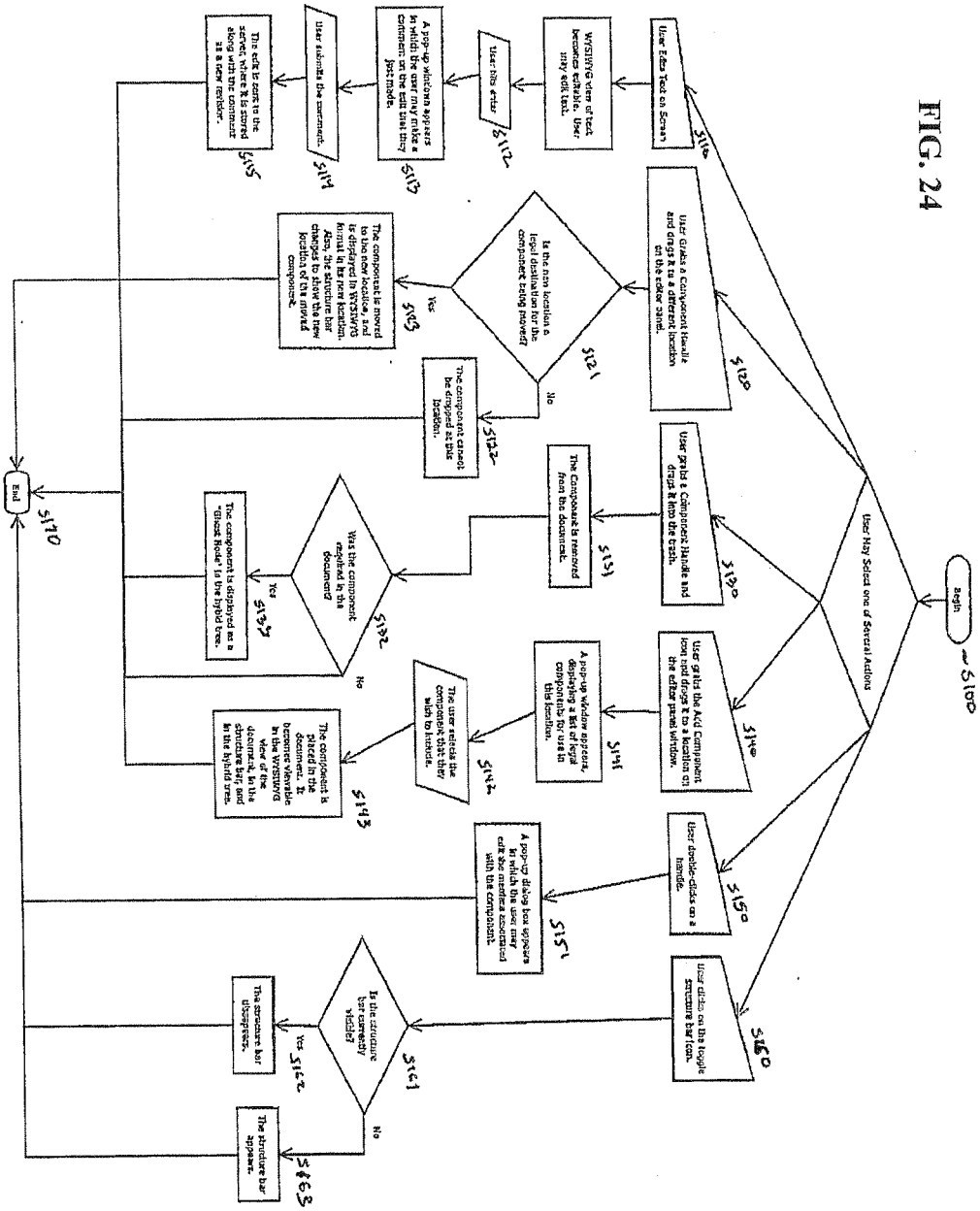
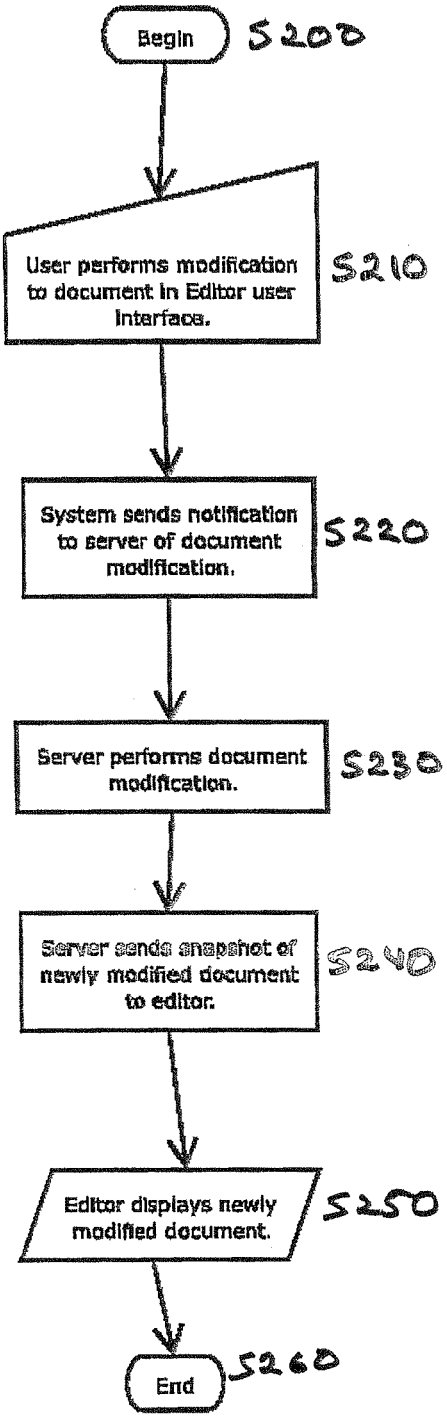
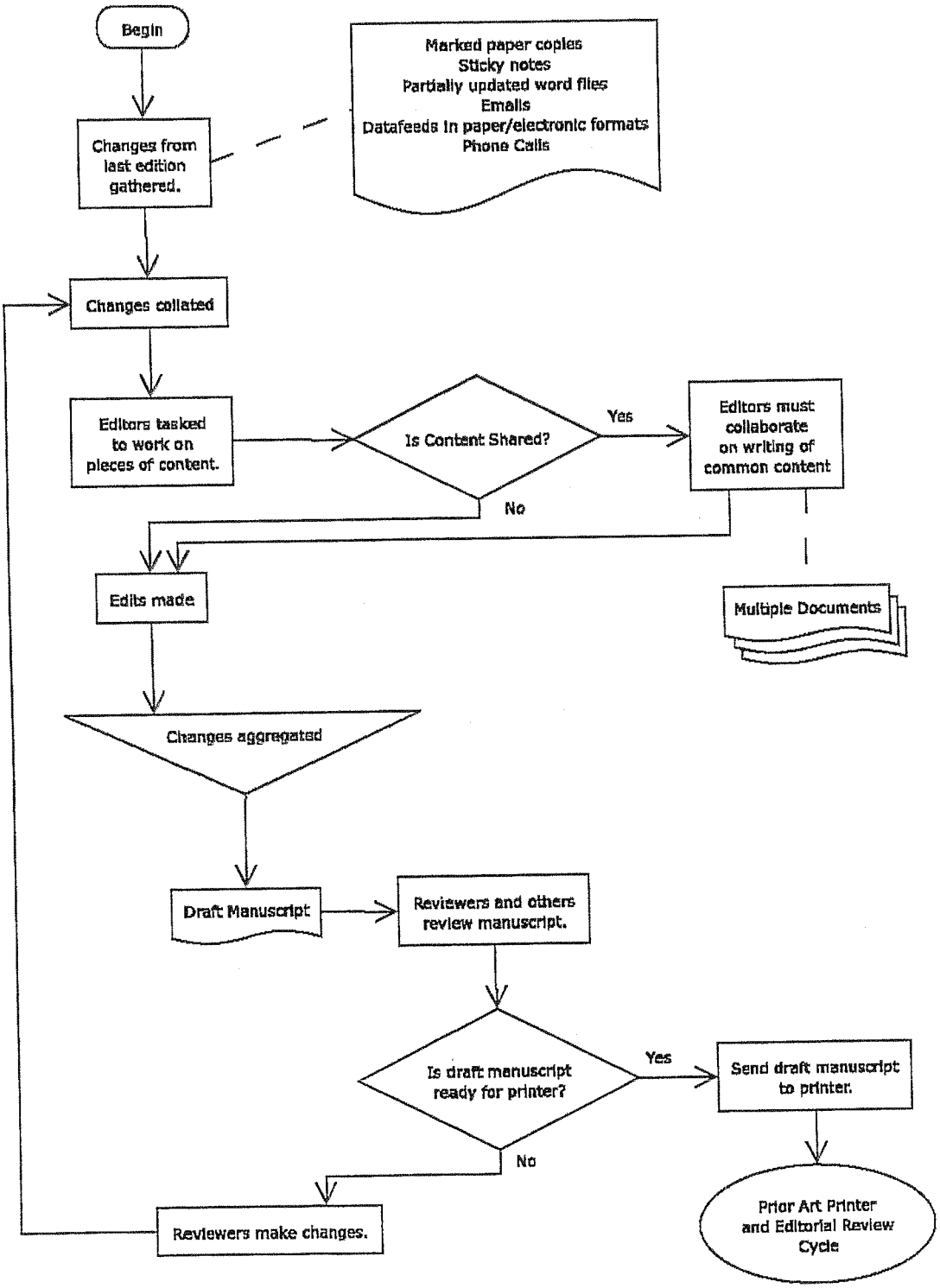

Begin
S100
User May Select one of Several Actions
User Edits Text on Screen
S110
WYSIWYG view of text becomes editable. User may edit text.
User clicks enter
S112
A pop-up window appears in which the user may make a comment on the edit that they just made.
S113
User submits the comment.
S114
The edit is sent to the server where it is stored along with the comment as a new revision.
S115
User Grabs a Component Handle and drags it to a different location on the editor panel.
S120
Is the new location a legal destination for the component being moved?
S121
Yes
The component is moved to the new location, and is displayed in its new location. Additionally, the structure bar changes to show the new location of the moved component.
S123
No
The component cannot be dropped at this location.
S122
User grabs a Component Handle and drags it into the trash.
S130
The Component is removed from the document.
S131
Was the component required in the document?
S132
Yes
The component is displayed as a "ghost Node" in the hybrid tree.
S133
No
The component is placed in the document, but becomes viewable in the WYSIWYG view of the document, in the structure bar, and in the hybrid tree.
S143
User selects the component that they wish to insert.
S142
A pop-up window appears displaying a list of legal components for use in this location.
S141
User grabs the Add Component icon and drags it to a location on the editor panel window.
S140
User double-clicks on a handle.
S150
A pop-up dialog box appears in which the user may edit the attributes associated with the component.
S151
User clicks on the Toggle structure bar icon.
S160
Is the structure bar currently visible?
S161
Yes
The structure bar disappears.
S162
No
The structure bar appears.
S163
End
S170

FIG. 25


Begin
S200
User performs modification to document in Editor user interface.
S210
System sends notification to server of document modification.
S220
Server performs document modification.
S230
Server sends snapshot of newly modified document to editor.
S240
Editor displays newly modified document.
S250
End
S260

# FIG. 26a
(PRIOR ART)



# FIG. 26b
(PRIOR ART)

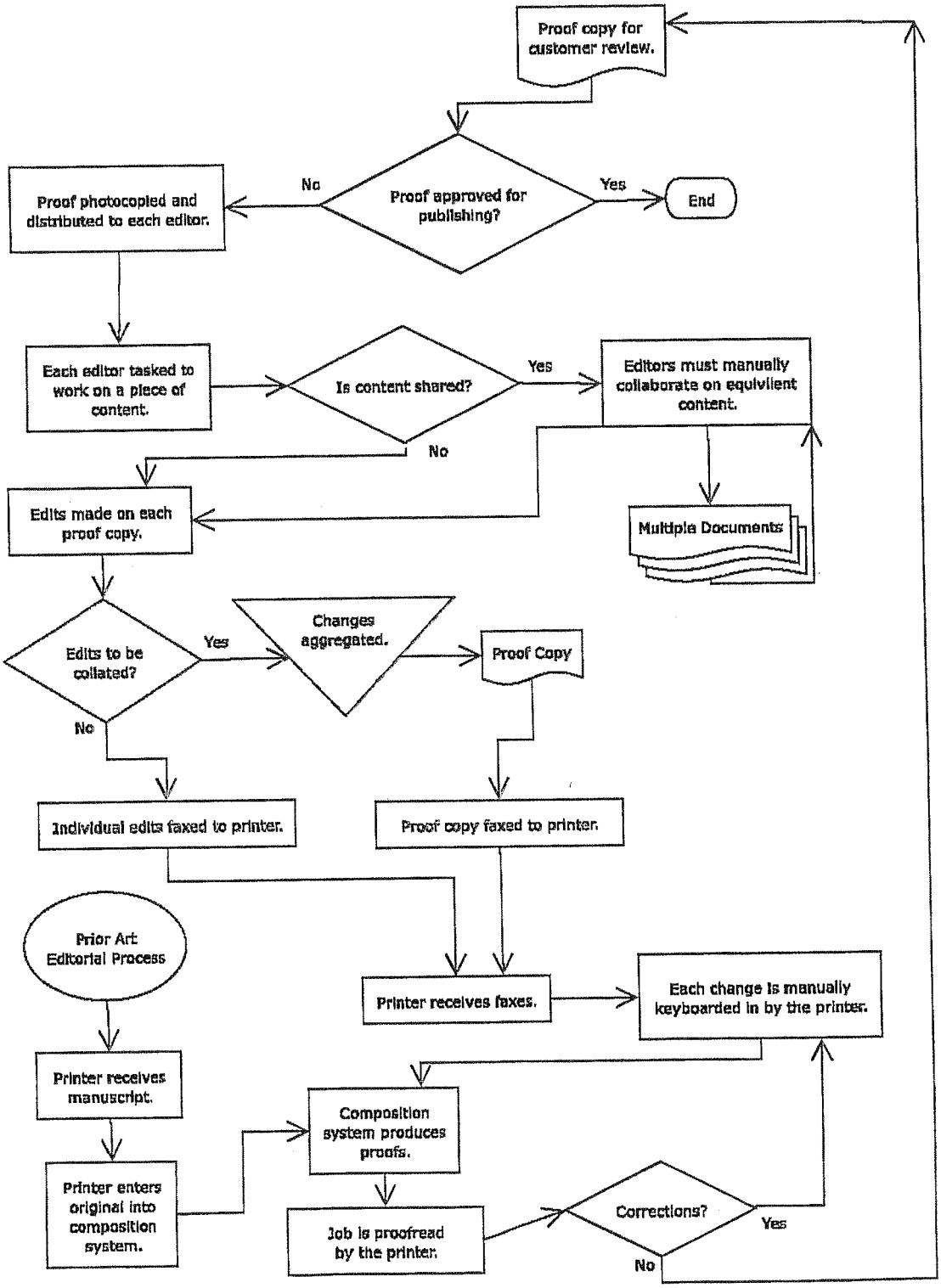
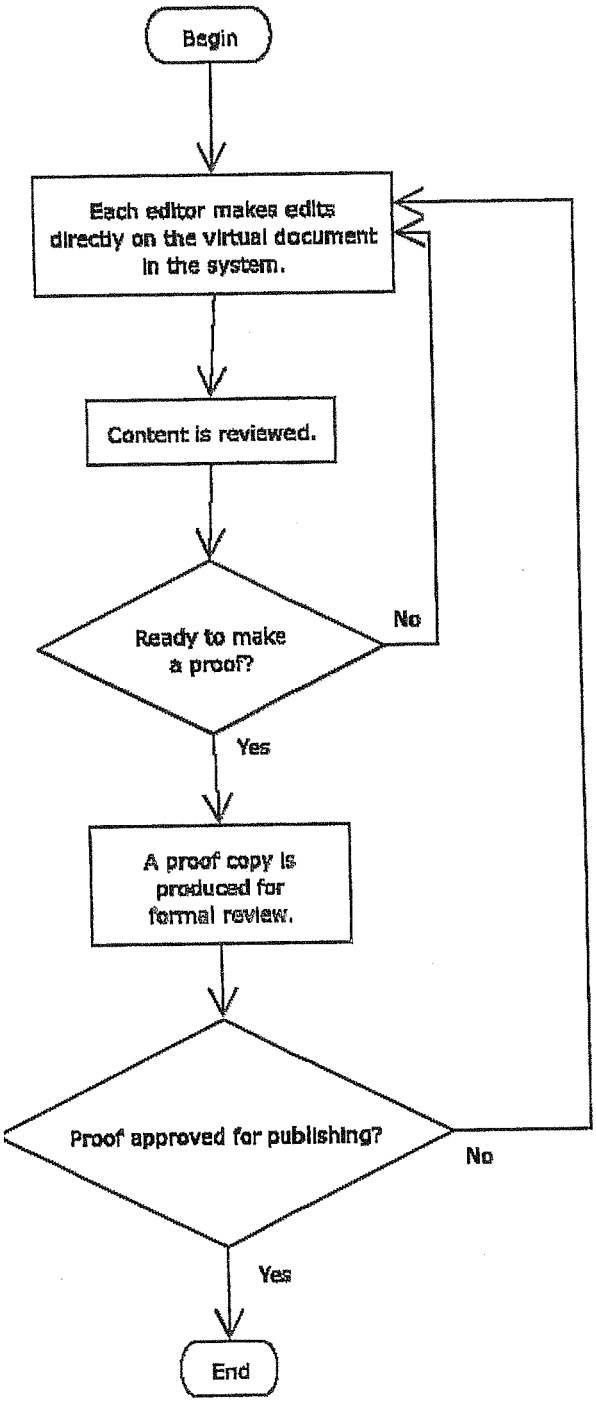
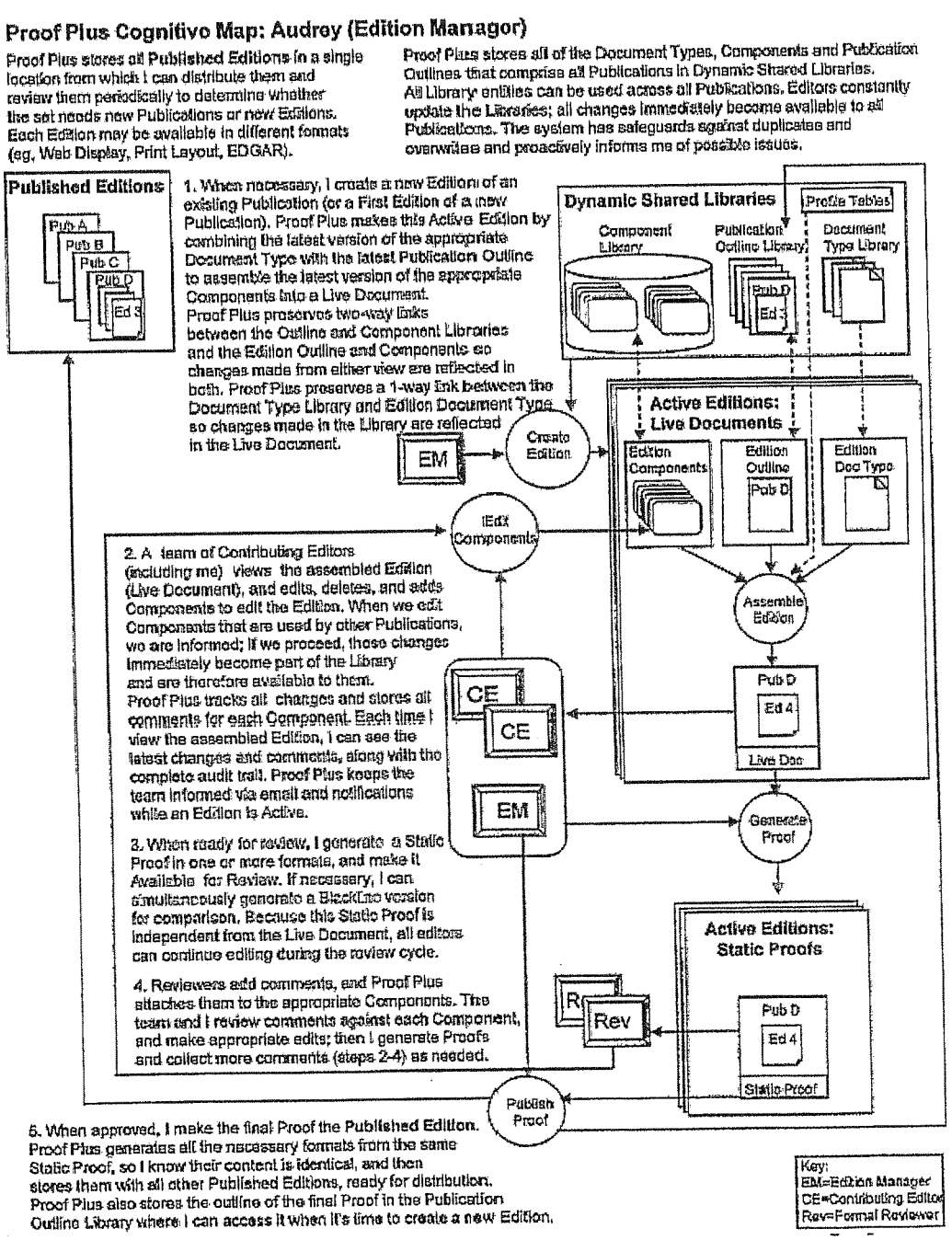


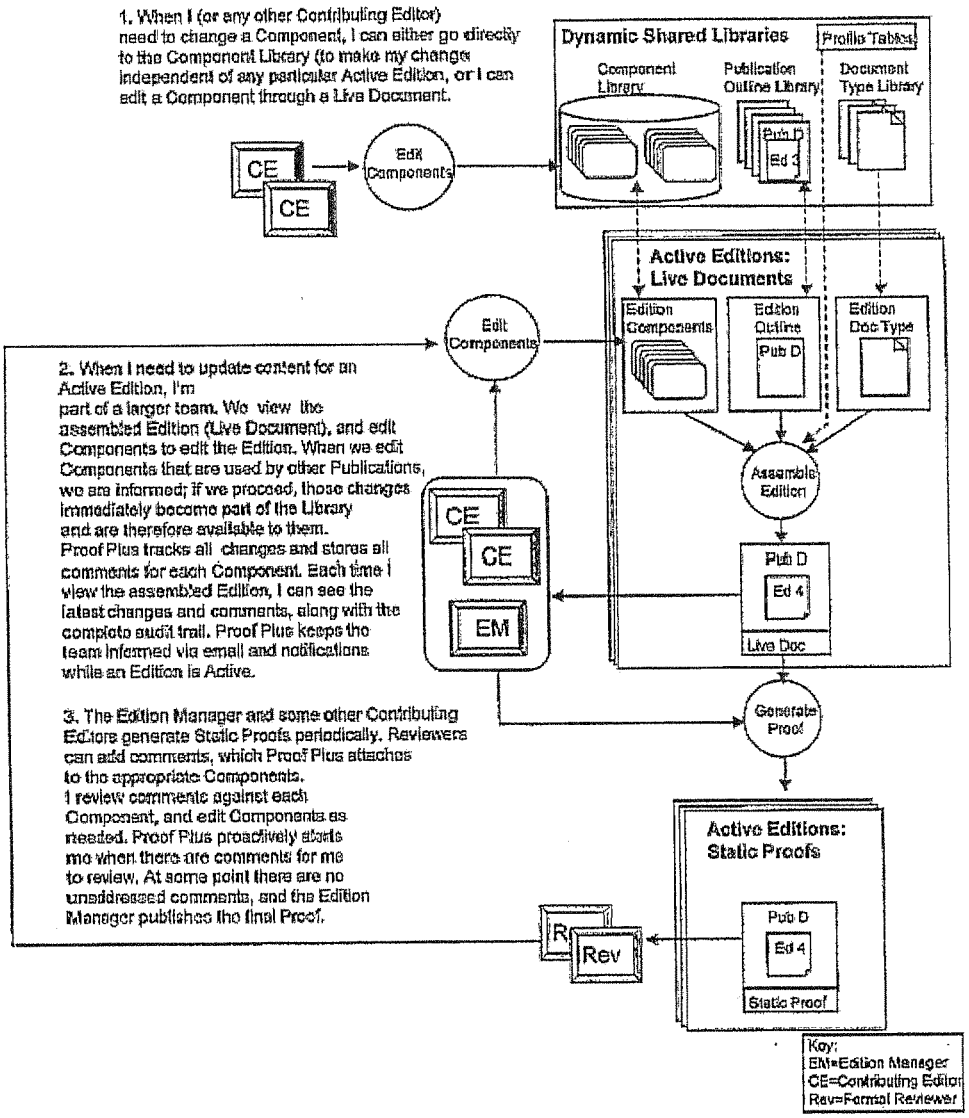
FIG. 26c


Begin
Each editor makes edits directly on the virtual document in the system.
Content is reviewed.
Ready to make a proof?
No
Yes
A proof copy is produced for formal review.
Proof approved for publishing?
No
Yes
End

# FIG. 27


Proof Plus Cognitive Map: Audrey (Edition Manager)
Proof Plus stores all Published Editions in a single location from which I can distribute them and review them periodically to determine whether the set needs new Publications or new Editions. Each Edition may be available in different formats (eg, Web Display, Print Layout, EDGAR).
Proof Plus stores all of the Document Types, Components and Publication Outlines that comprise all Publications in Dynamic Shared Libraries. All Library entities can be used across all Publications. Editors constantly update the Libraries; all changes immediately become available to all Publications. The system has safeguards against duplicates and overwrites and proactively informs me of possible issues.
Published Editions
Pub A
Pub B
Pub C
Pub D
Ed 3
1. When necessary, I create a new Edition of an existing Publication (or a First Edition of a new Publication). Proof Plus makes this Active Edition by combining the latest version of the appropriate Document Type with the latest Publication Outline to assemble the latest version of the appropriate Components into a Live Document.
Proof Plus preserves two-way links between the Outline and Component Libraries and the Edition Outline and Components so changes made from either view are reflected in both. Proof Plus preserves a 1-way link between the Document Type Library and Edition Document Type so changes made in the Library are reflected in the Live Document.
Dynamic Shared Libraries
Profile Tables
Component Library
Publication Outline Library
Document Type Library
Pub D
Ed 3
EM
Create Edition
Active Editions: Live Documents
Edition Components
Edition Outline
Pub D
Edition Doc Type
Edit Components
2. A team of Contributing Editors (including me) views the assembled Edition (Live Document), and edits, deletes, and adds Components to edit the Edition. When we edit Components that are used by other Publications, we are informed; if we proceed, those changes immediately become part of the Library and are therefore available to them.
Proof Plus tracks all changes and stores all comments for each Component. Each time I view the assembled Edition, I can see the latest changes and comments, along with the complete audit trail. Proof Plus keeps the team informed via email and notifications while an Edition is Active.
Assemble Edition
Pub D
Ed 4
Live Doc
CE
CE
EM
Generate Proof
3. When ready for review, I generate a Static Proof in one or more formats, and make it Available for Review. If necessary, I can simultaneously generate a BlackLine version for comparison. Because this Static Proof is independent from the Live Document, all editors can continue editing during the review cycle.
Active Editions: Static Proofs
4. Reviewers add comments, and Proof Plus attaches them to the appropriate Components. The team and I review comments against each Component, and make appropriate edits; then I generate Proofs and collect more comments (steps 2-4) as needed.
Rev
Pub D
Ed 4
Static Proof
Publish Proof
5. When approved, I make the final Proof the Published Edition. Proof Plus generates all the necessary formats from the same Static Proof, so I know their content is identical, and then stores them with all other Published Editions, ready for distribution. Proof Plus also stores the outline of the final Proof in the Publication Outline Library where I can access it when it's time to create a new Edition.
Key:
EM=Edition Manager
CE=Contributing Editor
Rev=Formal Reviewer

# FIG. 28


1. When I (or any other Contributing Editor) need to change a Component, I can either go directly to the Component Library (to make my change independent of any particular Active Edition, or I can edit a Component through a Live Document.
Dynamic Shared Libraries
Profile Tables
Component Library
Publication Outline Library
Document Type Library
Pub D
Ed 3
CE
CE
Edit Components
Active Editions: Live Documents
Edition Components
Edition Outline
Pub D
Edition Doc Type
Edit Components
2. When I need to update content for an Active Edition, I'm part of a larger team. We view the assembled Edition (Live Document), and edit Components to edit the Edition. When we edit Components that are used by other Publications, we are informed; if we proceed, those changes immediately become part of the Library and are therefore available to them. Proof Plus tracks all changes and stores all comments for each Component. Each time I view the assembled Edition, I can see the latest changes and comments, along with the complete audit trail. Proof Plus keeps the team informed via email and notifications while an Edition is Active.
Assemble Edition
CE
CE
EM
Pub D
Ed 4
Live Doc
3. The Edition Manager and some other Contributing Editors generate Static Proofs periodically. Reviewers can add comments, which Proof Plus attaches to the appropriate Components. I review comments against each Component, and edit Components as needed. Proof Plus proactively alerts me when there are comments for me to review. At some point there are no unaddressed comments, and the Edition Manager publishes the final Proof.
Generate Proof
Active Editions: Static Proofs
Pub D
Ed 4
Static Proof
Rev
Key:
EM=Edition Manager
CE=Contributing Editor
Rev=Formal Reviewer

# FIG. 29

Proof Plus stores all Published Editions and Static Proofs of Active Editions in a single location so I can always find the latest version, including Blackline Proofs if I have requested them, when I need it. For each Publication, Proof Plus also retains historical information that's useful for auditing.

