

A Q&A Intro to ClearSpecs Usage Modeling

David Gelperin
ClearSpecs Enterprises

This article describes the components of a ClearSpecs Usage Model; explains how UML models relate; and details where such modeling can be effectively used. The reference list includes a sample model.

What are the components of a ClearSpecs Usage Model?

These models are composed of a natural language overview, a set of precise use cases, and a set of detailed definitions and facts. The domain and system-specific terms used in the overview and the use cases are defined in the detailed definitions.

What are ClearSpecs Usage Models intended to be?

These models are meant to be ultra-understandable, test-adequate, composite specifications of product requirements and usage. The requirements constrain valid usage and the usage models illustrate the behavioral requirements.

ClearSpecs usage models are meant to be comprehensive and formal enough to support automated analysis and automated test design while at the same time striking a balance between precision and readability. This balance is the “sweet spot” referred to by Leffingwell and Widrig (pp. 271-274).

What does “ultra-understandable” mean?

There are many ways (i.e., choices of notation, format, and terminology) to specify a piece of technical information. The choice of “how to say it” determines how many readers can understand the information without significant effort.

The job of any reader of technical information is to comprehend and analyze content. Any facet of a presentation style that significantly aids (1) rapid and accurate comprehension or (2) rapid and accurate determination of essential analytic outcomes supports ultra-understandability of the content.

Characteristics of presentation styles that support ultra-understandability are:

- (1) appropriate **content selection** and level of **terminology abstraction** to include and highlight relevant information while suppressing less relevant details and irrelevant content,
- (2) clear and familiar **terminology**,
- (3) clear and familiar **patterns of expression**,
- (4) analytically effective **content aggregation and arrangement** that exposes critical relationships, and
- (5) analytically effective **content orientation and humane chunking** to minimize the mental effort of comprehension and analysis.

See Tufte's description of *The Decision to Launch the Space Shuttle Challenger* for examples of

- **poor terminology abstraction** (i.e., difficult to determine degree of O-ring damage in each pre-launch test),
- **poor content aggregation** (i.e., no information display relating degree of O-ring damage and temperature existed prior to launch), and
- **poor content arrangement** (i.e., after launch, damage and temperature data displayed in test date sequence rather than temperature sequence).

[Tufte, Edward R. *Visual Explanations: Images and Quantities, Evidence and Narrative*, Graphics Press 1997 pp. 38-53]

Why is “ultra-understandable” important?

It supports ultra-reviewability of the model by a broad population of stakeholders and therefore is a major enabler of early model validation. If the model is incomplete/wrong and the derived tests are therefore incomplete/wrong (and the code is incomplete/wrong), there may be serious problems. Stakeholders can only be maximally effective information sources, if they readily understand the models.

What does “test-adequate” mean?

A test-adequate model contains sufficient information to support automated analysis and test design. ClearSpecs modeling is designed to provide sufficient information to enable both automated analysis of the consistency and (relative) completeness of the model as well as the automatic generation of test cases and scenarios. This information includes value ranges and logical or arithmetic formulas for attribute values as well as strong constraints on the input situation and required responses using pre, constant, and post conditions.

Black-box UML models are **not** test-adequate, but ClearSpecs models are.

What does “composite model” mean?

There are different types of information about behavior and usage and therefore a model of such rules must have different formats to express these different types.

For example, an ATM behavior rule may require a “sufficient balance” in order to make a withdrawal. This can be modeled as a conditional expression: e.g., if withdrawal request and balance is sufficient and (withdrawal amount not greater than allowed amount), then provide withdrawal amount. However, the rule for determining sufficient balance e.g. [(current balance – guaranteed amount - withdrawal amount) not less than zero] specifies an arithmetic computation. Finally, a rule that every customer must have at least one account is a constraint on entity relationships and any changes to those relationships.

What are the components of a complete ClearSpecs model?

These models contain:

- Application name & abstract
- Model profile info (e.g., Version, Modeler, Sources)
- User population description - including population diagram and user profiles
- Detailed definitions and facts
- Non-functional requirements – optional
- Work breakdown structure - optional
- Precise use cases

How do ClearSpecs Usage Models relate to UML models?

ClearSpecs models focus solely on specifying precise information about externally visible behavior (i.e., black-box specs), while UML models include object-oriented design elements (e.g., class methods) as well. Some elements of a ClearSpecs model are new, while others are based on, but extend some of the components in a UML model.

A ClearSpecs model should be built first and then the UML compatible elements exported into a UML system to support OO design. Application domain components such as essential class descriptions, relationships, and constraints could be imported from existing UML models.

The table below describes the structure of a ClearSpecs model and compares each element to elements in the UML.

	Constituent Models	Constituent Model Descriptions	Model Elements	Element Descriptions	UML vs. CSM
A1	Product Abstract	Text summary of product features			Old
	User Population	Detailed spec of user population and their relationships to the product and each other			
P1			Population diagram	Standard UML	Old
P2			Actor profiles	Attributes of each class of actor that support a clear understanding of actor intentions and potential product usage	New
	Application Domain	Detailed spec of essential classes in the application domain including their relationships and constraints			
D1			Entity profiles - basic attributes	Basic attributes of each essential class and their value ranges	Ext
D2			Entity profiles - derived attributes	Attributes derived from basic attributes via mathematical (e.g., logical or arithmetic) formulas and their value ranges	New
D3			ER diagram	Standard UML	Old
D4			Entity-Relationship table	Table form of ER diagrams	New
D5			Entity constraints	Constraint expressions, but not in OCL	Ext
D6			Definitions & Dependencies	Logical expressions, but not in OCL	Ext

	Reqs	Details of usage, behavior, and non-functional constraints			
R1			Non-functional Requirements		New
R2			Action Contracts		New
R3			Work breakdown structure		New
	Use Cases	Descriptions of interactions between the product and its users intended to achieve specific user goals			
C1			Precise Use Cases		Ext

Why do ClearSpecs models have so many tables?

Tables are a fundamental information presentation format. Tables support the specification of information sequence patterns (as either column headings or row headings) and then the specification of instances of these patterns (as either rows or columns of specific values).

Powerful and inexpensive automation (word processors, spreadsheets and relational databases) exists to support table creation and manipulation including the enforcement of dependencies between information items in a sequence (via cell formulas) and the reordering of pattern instances (via sorting).

All diagrammed information can be specified in tables. Tables make some information easier to see and diagrams make other information easier to see. For example, state tables make the complete state transition rules easy to see, while state diagrams make transitions easy to see. Both tables and diagrams have value.

Consider the following highly biased, completely subjective comparison of best-case outcomes for the designated information formats:

Format Attributes	Formats		
	Text	Tables	Diagrams
Scope of Applicability	5	4	3
Upward Scalability	5	4	2
Precision	5	5	3
Modification Effort	3	5	1
Reader Usability	3	5	3
Specing of Behavior Rules	3	5	1
Auto-Verifiability	0	5	3
Auto-Implementability	0	5	3
Psychological Appeal	3	1	5

If pictures are worth 1000 words, tables are worth more.

Where can ClearSpecs usage modeling be used effectively?

ClearSpecs models specify interactive processing and serial transactions, as well as real-time systems. ClearSpecs models effectively specify both IT and embedded systems. They do not support applications such as language processing, mathematical programming, graphics, nor games.

ClearSpecs modeling is applicable to all systems covered by UML modeling, plus others whose behavior is determined mostly or entirely by input data alone (i.e., not state-based). Action tables enable the clear specification of such behavior.

Are ClearSpecs usage models powerful enough to specify all forms of business rule information?

Yes. Business rules explicitly describe the (semi-)permanent **structure** as well as **operation** of an enterprise. A taxonomy of business rule information and its expression in ClearSpecs modeling is:

1. **Facts** about business elements (objects, attributes, value ranges, and relationships)

Immediate facts

Basic Terms e.g., ytd-billing is an attribute of the vendors class [in basic entity profiles or derived attribute definitions]

Relationships – unconditional (domain constants) & conditional relationships between objects or attributes e.g., customer has between one and three accounts or cancel moment later than order moment or if security type = derivative, then trader-id = unassigned [in relationship tables or entity constraint lists]

Derivable facts

Calculated values e.g., **available stock** =
on-hand stock + on-order stock - back-ordered stock
[use formula in value column of derived attribute definition]

Derived attributes e.g., a **preferred vendor** is any vendor providing more than 1M dollars in supplies [in value column of derived attribute definition or in relationship tables]

2. **Reactions** i.e., purposeful responses to an event or condition [in use case procedures or action tables]

Unconditional i.e. response to trigger event only e.g., updating a reservation when arrival date changes)

Conditional response when object attributes satisfy specified conditions e.g., return error message when input is invalid

3. Reaction Constraints “must always be” facts about reactions or “must never do” reactions [in constraint lists]

Unconditional e.g., confirmed reservations must never be deleted
or past due accounts must always be displayed in red

Conditional e.g., If no invalid transactions, throughput must be less than N
seconds, where N is the number of input transactions.

What tools are available to support development of ClearSpecs models?

The ClearSpecs Requirements Platform supports most elements of these models.. You can also use general tools such as a word processor, relational database, or spreadsheet.

References:

Business Rules

Gottesdiener, Ellen **Business Rules show Power, Promise**
Applications Development Trends March 1997
[\[www.adtmag.com/pub/mar97/softeng.htm\]](http://www.adtmag.com/pub/mar97/softeng.htm)

GUIDE **Business Rules Project – Final Report** October 1997
[\[www.essentialstrategies.com/publications/businessrules/index.htm\]](http://www.essentialstrategies.com/publications/businessrules/index.htm)

Conditions & Constraints

Gelperin, David **Specifying Consequences with Action Contracts**
[Available on www.ClearSpecs.com]

Meyer, Bertrand **Applying Design by Contract** IEEE Computer Vol 25
No 10 October 1992 pp 40-51

Meyer, Bertrand **Building bug-free O-O software: An introduction to
Design by Contract** [Available at
<http://eiffel.com/doc/manuals/technology/contract/index.html>]

Warmer, Jos and Kleppe, Anneke **The Object Constraint Language:
Precise Modeling with UML** Addison-Wesley 1999

OO Analysis

Coad, Peter and Yourdon, Edward **Object-Oriented Analysis** Yourdon Press 1991

Shlaer, Sally and Mellor, Stephan J. **Object Lifecycles: Modeling the
World in States** Yourdon Press 1991

Shlaer, Sally and Mellor, Stephan J. **Object-Oriented Systems Analysis:**

Modeling the World in Data Yourdon Press 1989

Operational Modes

Musa, John D **Operational Profiles in Software-Reliability Engineering** IEEE Software, March 1993, pp. 14-32

Musa, John **Software Reliability Engineering** McGraw-Hill 1999, Chapter 3

Requirements

Kovitz, Benjamin **Practical Software Requirements** Manning 1999

Leffingwell, Dean and Widrig, Don **Managing Software Requirements** Addison-Wesley 2000

Robertson, Suzanne and James **Mastering the Requirements Process** Addison-Wesley 1999

Sample ClearSpecs Models

Gelperin, David **Model of a Library Management System** [Available on www.ClearSpecs.com]

State Models

Harel, David and Politi, Michal **Modeling Reactive Systems with Statecharts** McGraw Hill 1998

UML

Douglass, Bruce Powel **Real-Time UML: Developing Efficient Objects for Embedded Systems** Addison-Wesley 2000

Larman, Craig **Applying UML and Patterns** Prentice Hall PTR 2002

Schmuller, Joseph **SAMS Teach Yourself UML in 24 Hours** Sams Press 1999

Use Cases

Cockburn, Alistair **Writing Effective Use Cases** Addison-Wesley 2001

Gelperin, David **Adding Bits of Precision to Usage Models** [Available on www.ClearSpecs.com]

Gelperin, David **Precise Use Cases** [Available on www.ClearSpecs.com]

Jacobson, Ivar et.al. **Object-Oriented Software Engineering** Addison-Wesley 1992, Chapter 7, pp. 153-174 [Original description]

Kulak, Daryl and Guiney, Eamonn **Use Cases: Requirements in Context** ACM Press, Addison-Wesley 2000