

UNITED STATES PATENT AND TRADEMARK OFFICE

---

BEFORE THE PATENT TRIAL AND APPEAL BOARD

---

FORD MOTOR COMPANY  
Petitioner,

v.

VERSATA SOFTWARE, INC.  
Patent Owner.

---

U.S. Patent No. 5,825,651 to Gupta et al.

IPR Case No.: IPR2016-01014

---

**PETITION FOR *INTER PARTES* REVIEW  
UNDER 35 U.S.C. § 311 *ET SEQ.* AND 37 C.F.R. § 42.100 *ET SEQ.*  
(CLAIMS 60-72 OF U.S. PATENT NO. 5,825,651)**

## **Table of Contents**

List of Exhibits.....	iii
Mandatory Notices Under 37 C.F.R. § 42.8 .....	vi
Real Party-In-Interest – 37 C.F.R. § 42.8(b)(1) .....	vi
Related Matters – 37 C.F.R. § 42.8(b)(2).....	vi
Lead and Back-Up Counsel Under 37 C.F.R. § 42.8(b)(3).....	vii
Service Information Under 37 C.F.R. § 42.8(b)(4) .....	vii
I.    Standing Requirements Under 37 C.F.R. § 42.104 .....	1
A.    Grounds for Standing – 37 C.F.R. § 42.104(a) .....	1
B.    Challenged Claims – 37 C.F.R. § 42.104(b)(1) .....	1
C.    Grounds of Challenge – 37 C.F.R. § 42.104(b)(2) .....	1
II.   Introduction.....	3
III.  Overview of the '651 Patent.....	3
IV.   Person having ordinary skill in the art (PHOSITA) .....	7
V.    State of the art .....	7
A.    Axling: “OBELICS” - 1994 .....	8
B.    Fohn: “PC/CON” - 1995 .....	11
C.    Skovgaard: “salesPLUS” - 1995 .....	14
VI.   Claim Construction – 37 C.F.R. § 42.104(b)(3).....	16
VII.  Unpatentability Grounds.....	16
Ground 1.  Claims 60-62, 64-68 and 70-71 are obvious in view of Axling, Fohn and the general knowledge of a person of ordinary skill in the art.....	17
A.    A person skilled in the art at the time of the alleged invention would have combined the teachings of Axling and Fohn.....	17
B.    Axling and Fohn disclose the subject matter of claims 60-62, 64-68 and 70-71 and render each claim obvious .....	24
Ground 2.  Dependent Claims 63 and 69 are obvious in view of Axling, Fohn, Skovgaard 1995 and Baker .....	52
A.    A person skilled in the art at the time of the alleged invention would have combined the teachings of Axling, Fohn, Skovgaard 1995, and Baker .....	52

Ground 3. Dependent Claim 72 is obvious in view of Axling, Fohn and Skovgaard .....	61
A. A person skilled in the art at the time of the alleged invention would have combined the teachings of Axling, Fohn, and Skovgaard 1995 .....	61
VIII. Conclusion .....	63
IX. Fee Statement.....	63
Certificate of Service .....	65
Certificate of Compliance Pursuant to 37 C.F.R. § 42.24 .....	66

**List of Exhibits**

<b>Exhibit No.</b>	<b>Description</b>	<b>Date</b>	<b>Identifier</b>
1001	U.S. Patent No. 5,825,651	Oct. 20, 1998	'651 Patent
1002	Declaration of Dr. Philip Greenspun	n/a	Greenspun
1003	Curriculum Vitae of Dr. Philip Greenspun	n/a	Greenspun CV
1004	T. Axling, S. Haridi, <u>A Tool For Developing Interactive Configuration Applications</u> , The Journal of Logic Programming, Volume 19, 658-679 (1994)	1994	Axling
1005	S.M. Fohn, J.S. Liau, A.R. Greef, R.E. Young, P.J. O'Grady, <u>Configuring Computer Systems Through Constraint-Based Modeling and Interactive Constraint Satisfaction</u> , Computers in Industry, Volume 27, Issue 1, Pages 3-21	Sept. 1995	Fohn
1006	W.P. Birmingham, D.P. Siewiorek, MICON: <u>A Knowledge Based Single Board Computer Designer</u> , Proceedings of 21st Design Automation Conference, IEEE CS Press, Pages 565-571	1984	Birmingham
1007	M. Stefik, <u>Introduction to Knowledge Systems</u> , Chapter 1, "Symbol Systems," Morgan Kaufmann Publishers, Inc., San Francisco, CA (June 15, 1995)	June 15, 1995	Stefik – Ch. 1
1008	U.S. Patent No. 5,283,857 titled <u>Expert System Including Arrangement for Acquiring Redesign Knowledge</u> issued to Evangelos	Feb. 1, 1994	Evangelos

Exhibit No.	Description	Date	Identifier
1009	H.J. Skovgaard, <u>salesPLUS A Product Configuration Tool</u> , 13 <sup>th</sup> National Conference on Artificial Intelligence (AAAI-96), Workshop on Configuration, No AAAI FS-96-03, Portland, Oregon, 61-68, August 4-8, 1996	Aug. 1996	Skovgaard 1996
1010	H.J. Skovgaard, <u>A New Approach to Product Configuration</u> , Third International Conference, '95, Concurrent Engineering & Technical Information Processing, January 30 – February 3, 1995	1995	Skovgaard 1995
1011	Patent Owner's Claim Construction Brief in <i>Trilogy Software, Inc. et al. v. Selectica, Inc.</i> , Case No. 2:04-cv-160, Dkt. #58 (E.D. Tex.)	July 1, 2005	Patent Owner's Claim Construction Brief in <i>Selectica</i> Litigation
1012	S.C. Dewhurst, K.T. Stark, <u>Programming in C++</u> , Prentice Hall Software Series	1989	Dewhurst
1013	U.S. Patent No. 5,825,651 File History	n/a	'651 Patent File History
1014	Texas Instruments C++ Object-Oriented Library User's Manual, 1991	1991	Texas Instruments
1015	H.G. Baker, <u>Efficient Implementation of Bit-Vector Operation in Common Lisp</u> , ACM SIGPLAN Lisp Pointers 3, No. 2-4 1990	1990	Baker
1016	B. Yu, H.J. Skovgaard, <u>A Configuration Tool to Increase Product Competitiveness</u> , 1998	1998	Skovgaard 1998

Exhibit No.	Description	Date	Identifier
1017	M. Stefik, <u>Introduction to Knowledge Systems</u> , Chapter 8, “Configuration,” Morgan Kaufmann Publishers, Inc., San Francisco, CA (June 15, 1995)	June 15, 1995	Stefik – Ch. 8
1018	M.R. Wagner, <u>Understanding the ICAD System</u> , ICAD, Inc., 1990	1990	ICAD
1019	Claim Construction Order in <i>Trilogy Software, Inc. v. Selectica, Inc.</i> , Case No. 2:04-cv-160, 405 F.Supp.2d 731 (E.D.Tex. 12/20/05)	Dec. 20, 2005	CC Order
1020	J.R. Wright, E.S. Weixelbaum, G.T. Vesonder, K.E. Brown, S.R. Palmer, J.I. Berman, and H.H. Moore, <u>A Knowledge-Based Configurator That Supports Sales, Engineering, and Manufacturing at AT&amp;T Network Systems</u> , <i>AI Magazine</i> , Volume 14, Number 3 (Fall 1993)	1993	Wright

**Mandatory Notices Under 37 C.F.R. § 42.8**

**Real Party-In-Interest – 37 C.F.R. § 42.8(b)(1)**

Petitioner certifies that Ford Motor Company (“Ford”) is the real party-in-interest.

**Related Matters – 37 C.F.R. § 42.8(b)(2)**

Petitioner identifies the following related judicial matter: *Ford Motor Company v. Versata Software, Inc. F/K/A Trilogy Software, Inc., Trilogy Development Group, Inc., and Trilogy, Inc.*, Case No. 2:15-cv-10628-MFL-EAS. U.S. Patent No. 5,825,651 (“the ’651 Patent”) is being asserted by Versata in this proceeding, along with two other patents from the same family (U.S. Patent Nos. 6,405,308 and 6,675,294) and five additional patents. Versata’s counterclaim against Ford for infringement of the ’651 Patent was filed and served on October 28, 2015. In connection with this litigation, Versata Software, Inc. has stated that it holds all right, title, and interest in and to the ’651 Patent.<sup>1</sup>

The ’651 Patent was also asserted in *Versata Development Group, Inc. F/K/A Trilogy Development Group, Inc., Versata Software, Inc., F/K/A Trilogy Development Group, Inc., and Trilogy, Inc. v. Ford Motor Company*, Case No. 4:15-cv-00316-RC-CMC. This case was dismissed without prejudice on

---

<sup>1</sup> The most recent assignment recorded with the U.S. Patent and Trademark Office states that the assignee of the ’651 Patent is Versata Development Group, Inc.

December 3, 2015.

Petitioner has not filed any concurrent petitions concerning the '651 Patent. Petitioner is concurrently filing the following petitions relating to other patents from the same family as the '651 Patent: IPR2016-01017 (concerning U.S. Patent No. 6,405,308) and IPR2016-01016 (concerning U.S. Patent No. 6,675,294). Petitioner has also filed the following petitions against other Versata patents at issue in Case No. 2:15-cv-10628: IPR2016-01012 and IPR2016-01013 (concerning U.S. Patent No. 7,882,057); IPR2016-01015 (concerning U.S. Patent No. 7,200,582); and IPR2016-01019 (concerning U.S. Patent No. 7,464,064). This Petition is not redundant to any previously or concurrently filed petitions.

**Lead and Back-Up Counsel Under 37 C.F.R. § 42.8(b)(3)**

Petitioner appoints John S. LeRoy (Reg. No. 48,158) of Brooks Kushman P.C. as lead counsel, and appoints Thomas A. Lewry (Reg. No. 30,770), Frank A. Angileri (Reg. No. 36,733), John P. Rondini (Reg. No. 64,949), Christopher C. Smith (Reg. No. 59,669) and Jonathan D. Nikkila (Reg. No. 74,694) of Brooks Kushman P.C. as back-up counsel. An appropriate Power of Attorney is filed concurrently herewith.

**Service Information Under 37 C.F.R. § 42.8(b)(4)**

Service of any documents to lead and back-up counsel can be made via hand-delivery to Brooks Kushman P.C., 1000 Town Center, Twenty-Second Floor,



Southfield, Michigan 48075. Petitioner consents to service by email at  
FPGP0126IPR1@brookskushman.com.

**I. Standing Requirements Under 37 C.F.R. § 42.104**

**A. Grounds for Standing – 37 C.F.R. § 42.104(a)**

Petitioner certifies that the '651 Patent is available for *inter partes* review and that the Petitioner is not barred or estopped from requesting *inter partes* review challenging the patent claims on the grounds identified in this Petition.

**B. Challenged Claims – 37 C.F.R. § 42.104(b)(1)**

Petitioner requests *inter partes* review of claims 60-72 of the '651 Patent and requests that the Patent Trial and Appeal Board (“PTAB”) cancel those claims as unpatentable.

**C. Grounds of Challenge – 37 C.F.R. § 42.104(b)(2)**

Petitioner relies on the following prior art for the grounds of unpatentability:

(i) **Axling** – T. Axling, S. Haridi, A Tool For Developing Interactive Configuration Applications, The Journal of Logic Programming, Volume 19, 658-679 (1994) (hereinafter “Axling”). Axling was published in the Journal of Logic Programming publication in 1994 and qualifies as prior art under 35 U.S.C. § 102(a) and (b). (Ex. 1004 [Axling].)

(ii) **Fohn** – S.M. Fohn, J.S. Liao, A.R. Greef, R.E. Young, P.J. O'Grady, Configuring Computer Systems Through Constraint-Based Modeling and Interactive Constraint Satisfaction, Computers in Industry, Volume 27, Issue 1, Pages 3-21 (September 1995) (hereinafter “Fohn”). Fohn was published in

Computers in Industry publication in September 1995 and qualifies as prior art under 35 U.S.C. § 102(a) and/or (b). (Ex. 1005 [Fohn].)

(iii) **Skovgaard 1995** – H.J. Skovgaard, A New Approach to Product Configuration, *Third International Conference icle '95*, Concurrent Engineering & Technical Information Processing (January 30 – February 3, 1995) (hereinafter “Skovgaard 1995”). Skovgaard 1995 qualifies as prior art under 35 U.S.C. § 102(a) and (b). (Ex. 1010 [Skovgaard 1995].)

(iv) **Baker** – H.G. Baker, Efficient Implementation of Bit-Vector Operation in Common Lisp, ACM SIGPLAN Lisp Pointers 3, No. 2-4 1990. Baker qualifies as prior art under 35 U.S.C. § 102(a) and (b). (Ex. 1015 [Baker].)

The grounds of unpatentability presented in this petition are as follows:

GROUND	BASIS	REFERENCES	CLAIMS CHALLENGED
1	§ 103	Axling, Fohn and the general knowledge of a PHOSITA	60-62, 64-68, and 70-71
2	§ 103	Axling, Fohn, Skovgaard 1995, and Baker	63 and 69
3	§ 103	Axling, Fohn, and Skovgaard 1995	72

The unpatentability grounds set forth in this Petition are confirmed and

supported by the declaration of Dr. Philip Greenspun. (“Greenspun,” attached as Ex. 1002.)

## **II. Introduction**

The Petitioner Ford Motor Company (“Ford”) respectfully requests *inter partes* review of claims 60-72 of U.S. Patent No. 5,825,651 (“the ’651 Patent,” attached as Ex. 1001) in accordance with 35 U.S.C. §§ 311–319 and 37 C.F.R. § 42.100 *et seq.*

## **III. Overview of the ’651 Patent**

The ’651 Patent is titled “Method and Apparatus for Maintaining and Configuring Systems” and is directed to software for interactively defining and configuring a system, and for validating the configuration. (Ex. 1001, ’651 Patent Abstract.)

The Background of the Invention section of the ’651 Patent describes certain prior art approaches to defining and configuring products. (*Id.* at 1:12-60.) The Background explains that salespeople, such as an “automobile salesperson,” traditionally define, configure, and validate vehicle configurations before they are sold to a customer:

A system is comprised of components. Before a system can be built the components of the system must be identified. To configure a system, a user must select the parts to include in the system. Typically, one who is knowledgeable about a system and its

components defines the system. Thus, for example, an automobile salesperson assists an automobile buyer in determining the type and features of the automobile. The salesperson understands the features and options that are available to create a valid configuration. Some features and options cannot be combined. The selection of some features caused other features to be unavailable, etc. It would otherwise be difficult for the buyer to identify all of the features and options available on the automobile that can be combined to create a valid configuration.

(*Id.* at 1:12-25.)

The Background also states that prior art “computer systems” were available for defining and configuring systems. (*Id.* at 1:26-27.) The Background does not identify the prior art configuration systems, and the Patent Owner did not disclose any prior art configuration systems to Examiner Kreiss during examination.<sup>2</sup>

The ‘651 Patent identifies two alleged drawbacks associated with the prior art configuration software. (*Id.* at 1:27-60.) First, the prior art systems allegedly required “a configuration language to define a system” which “limits the number

---

<sup>2</sup> With one exception, each of the prior art references listed on the face of the ‘651 Patent was located by Examiner Kreiss. (Ex. 1013, ‘651 Patent File History at 162.) The Patent Owner submitted a single prior art reference (U.S. 4,796,194) directed to a factory operations control processes—not product configuration software. (*Id.* at 163-164.) The ‘194 Patent does not refer to “configuration.”

of users who are able to use the configuration system.” (*Id.*) The patent states that “the level of sophistication needed to communicate with the configuration system (through a configuration language) results in less sophisticated users being unable to use the system.” (*Id.*)

The ’651 Patent also alleges that the prior art configurators “impose a flow or ordering to the user operations” that might lead to frustration by a “novice user.” (*Id.* at 1:38-48.) The patent explains that this limitation of the prior art systems may cause a “novice user” to “become frustrated or confused and abort the configuration process.” (*Id.*)

As explained in detail in Petitioner’s substantive invalidity analysis below, prior art literature describing configuration software that was not considered during examination of the ’651 Patent had already recognized, and solved, the alleged problems in the manner described and claimed in the ’651 Patent.

In the Summary of the Invention, the ’651 Patent describes a conventional two-part configuration system having a “maintenance system” for “defining” a product, and a “configuration system” that is used to configure a product “using a definition created by the maintenance system.” (*Id.* at 2:40-62, Fig. 2.)

The “maintenance system” described in the ’651 Patent includes a graphical user interface (Fig. 6 shown below) to define a product, including relationships between products and parts, and relationships between the parts themselves. (*Id.* at

2:38-49.)

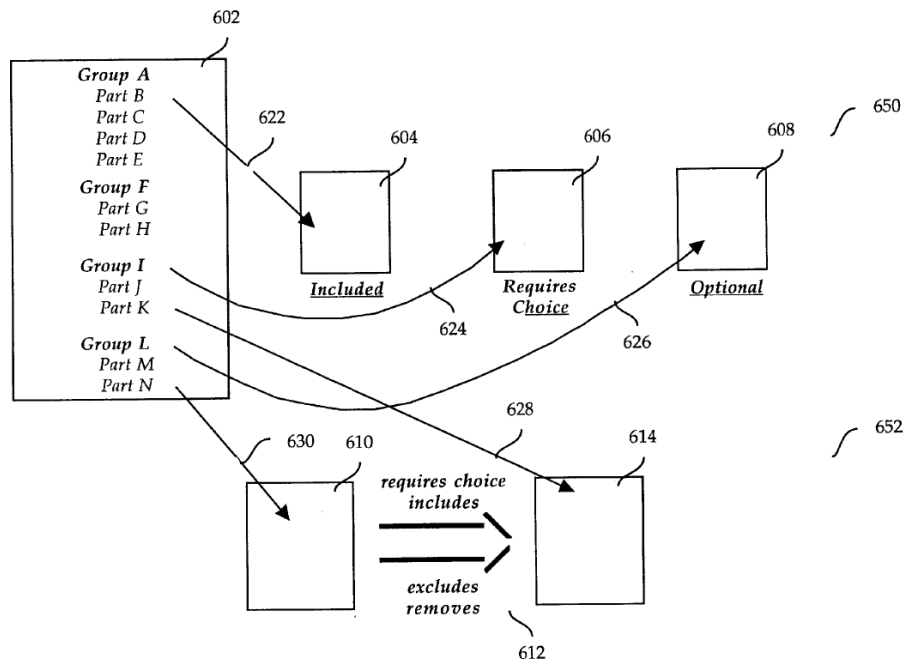


FIG. 6

In this example, the product is defined by dragging-and-dropping parts from pane 602 into panes 604-614 to create relationships (*e.g.*, “includes,” “excludes,” “requires choice”) between a product and parts in section 650, and to create relationships between parts themselves in section 652. (*Id.* at 8:5-7.) Although Figure 6 is introduced as an interface for *defining* a product model within the maintenance system, a “configuration user” may use a GUI similar to Figure 6 to *configure* a product after it has been defined. (*Id.* at 8:64-65.)

An example relationship is as follows: “Part A *includes* Part B and Part C.” In this example, “Part A” is the left-hand side of the relationship, and “Part B and Part C” is the right-hand side of the relationship. The relationship itself is an

“includes” relationship. (Ex. 1002, Greenspun ¶36.)

The '651 Patent describes the product definition created with the GUI of Figure 6 as an “external representation” that is translated by a “compiler” into an “internal representation.” (*Id.* at 3:1-9, 9:10-16.) The computer uses the internal representation during a configuration session to validate a user’s input against the product definition. (*Id.* at 3:9-11, 9:30-32.)

#### **IV. Person having ordinary skill in the art (PHOSITA)**

A PHOSITA would have either: (1) a bachelor’s degree in computer science, electrical engineering, computer engineering, or similar technical field; or (2) equivalent experience in the design or implementation of configuration systems. (Ex. 1002 [Greenspun] ¶26; *see also* ¶¶5-21.) The relevant field of art is product configuration software. (*Id.*)

#### **V. State of the art**

More than a decade before the application for the '651 Patent was filed, the Artificial Intelligence community had developed sophisticated product configuration systems which generally enable a user to (1) assemble a product model defining interrelationships between components of the product, and (2) configure a product (*e.g.*, for sale to a customer) based on the product model. (Ex. 1002, Greenspun ¶44.) These early configuration systems include Digital Electronics Corporation’s “XCON” system for configuring computers (1982), and



the MICON system developed at Carnegie-Mellon University (1984) for configuring microprocessors, the Cossack system, and others. (Ex. 1002, Greenspun ¶¶45-48.)

As explained above, the '651 Patent seeks to improve upon the usability of earlier product configurators known in the industry. (Ex. 1002, Greenspun ¶49.) The Background of the Invention of the '651 Patent alleges that the earlier systems lacked the '651 Patent's "graphical user interface" permitting a maintenance user to define a product model without requiring the knowledge and use of a "configuration language." (Ex. 1001, at 1:26-38; 2:38-49.) The '651 Patent also alleges that earlier configuration systems "impose a flow or ordering to the user operations" during a configuration session, which might cause a user to "become frustrated or confused." (*Id.*, 1:38-49.) To address this alleged deficiency in prior art systems, the '651 Patent permitted a user to "select and unselect parts in any order." (Ex. 1001, '651 Patent, 2:50-55.)

Yet, at least three configuration systems were described in prior art literature as both recognizing the drawbacks described in the '651 Patent, and solving them, in the manner described and claimed in the '651 Patent. Each prior art publication and configuration system is summarized below.

**A. Axling: "OBELICS" - 1994**

In 1994, Tomas Axling, Seif Haridi and others at the Swedish Institute of

Computer Science developed the “OBELICS” interactive product configurator. (See Ex. 1002, Greenspun ¶¶51-59.) The Axling paper, like the ’651 Patent, described the early product configuration systems as “hard to maintain” because “new products are constantly designed and old products are modified.” (Ex. 1004, Axling at 2.) Axling recognized that “there exists a need for methods and tools that speed up the development of configuration systems and make the systems more maintainable.” (*Id.*)

To address these drawbacks, Axling introduced a concept described as “Interactive Configuration by Selection” or “ICS.” (*Id.*) Axling described ICS as “a less complex and more interactive approach to configuration.” (*Id.*) Axling introduces an ICS having “*a graphical user interface making the domain model easy to understand and work with.*” (*Id.* at 8, emphasis added.)

Axling developed the “OBELICS” ICS product configurator. (*Id.* at 9-15.) OBELICS uses a “Domain Model Editor” to graphically define products and relationships among parts of the product. (*Id.* at 9, 12-15.) Figure 5.2, reproduced below, depicts the OBELICS graphical Domain Model Editor for defining an example product – a computer system (“compsys”) in this case:

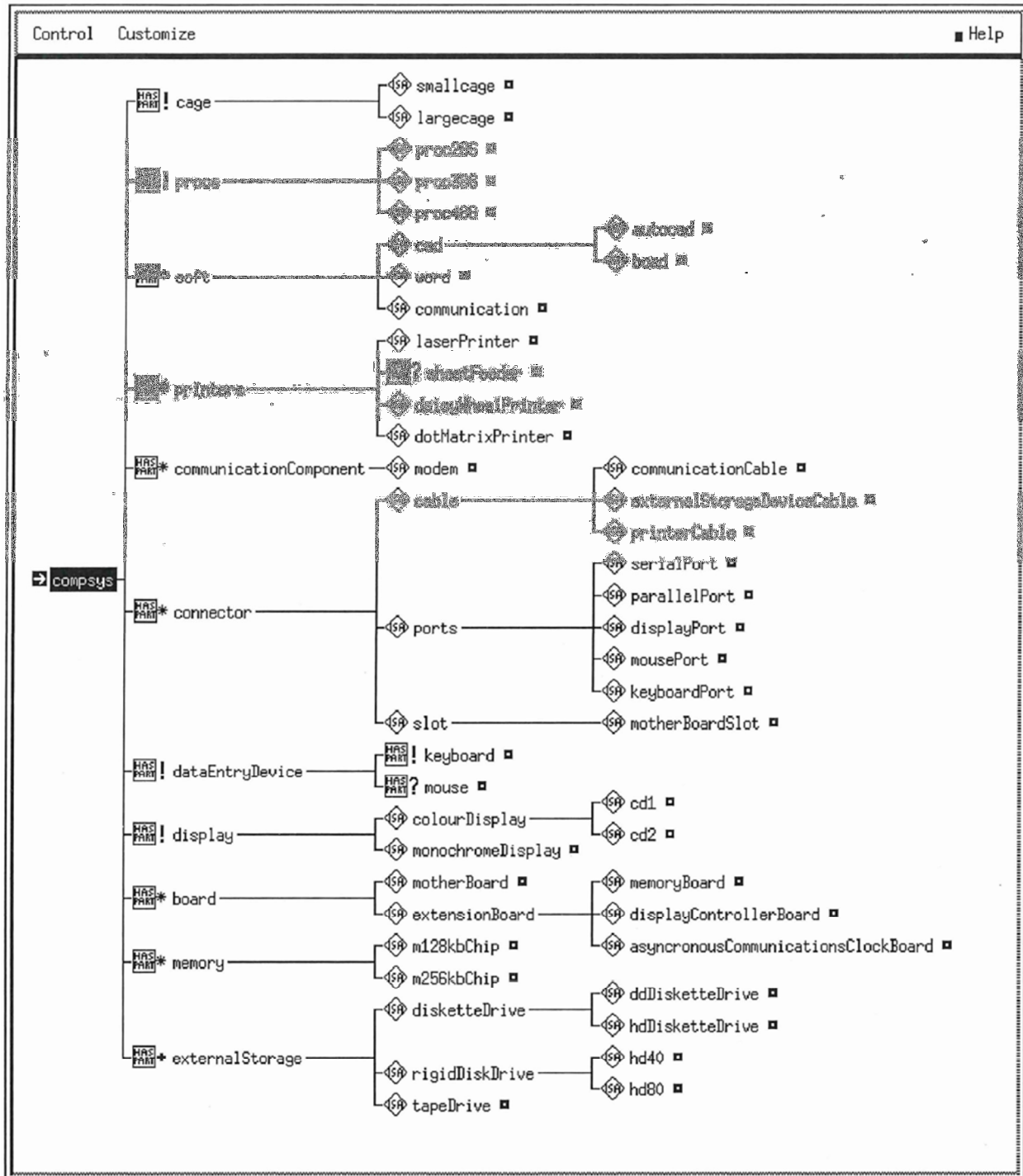


FIGURE 5.2. The Domain Model Editor

Relationships between the “compsys” product and its available parts, and relationships between the parts themselves, are depicted in the Domain Model

Editor above with symbols corresponding to “required,” “optional,” “optional iteration,” and “required iteration” relationship classes:

- ! : Exactly one is required to make the construction complete. We label these subcomponents *required component classes*.
- ? : One may be included. We label these subcomponents *optional component classes*.
- \* : Zero or more may be included. We label these subcomponents *optional iteration component classes*.
- + : One or more must be included. We label these subcomponents *required iteration component classes*.

(*Id.* at 14.)

Independent of the Domain Model Editor for graphically defining relationships among components within the product model, Axling describes a “User Interface” or “GUI” for configuring a product based on the model. (*Id.* at 19-20.) The configuration interface “present[s] alternatives in an understandable way to the user” and enables a user “at all times during a configuration session to view and edit the current state of the configuration.” (*Id.*)

The Axling paper, and the OBELICS software that Axling and his team developed, were not considered during examination of the ’651 Patent.

#### **B. Fohn: “PC/CON” - 1995**

In 1993, developers at IBM (Steffen Fohn) and North Carolina State University developed the “PC/CON” Personal Computer CONfigurator application. (*See* Ex. 1002, Greenspun ¶¶60-70.) Details describing PC/CON were published in 1995. (Ex. 1005.)

Fohn's paper, like Axling's paper, discloses software for both creating a modeling a product, and configuring a product for sale to a customer based on the model. Fohn discloses a "spreadsheet-like development interface that, in conjunction with a database manager, facilitates the construction of a constraint-based model representing a product line." (Ex. 1005, Fohn at 4.)

Fohn recognized that "one of the most important issues in computer product configuration is the maintainability and expandability of existing configuration information models." (*Id.* at 7.) Fohn recognized (1) that "it is better to train personnel from the field in information modeling and the use a software package than to train Artificial Intelligence (AI) specialists in the field" and (2) that "***an environment is demanded that will enable rapid model building by product and manufacturing engineers, and sales representatives.***" (*Id.* at 6, emphasis added.) Fohn described "the creation of a model that can represent the complex interrelationships among its elements ***while retaining an organizational simplicity such that people with no specific training in a computer-related field can perform development, maintenance, and expansion.***" (*Id.*, emphasis added.) Fohn describes an approach in which "developers are supported by a familiar spreadsheet-like interface that enables more emphasis to be placed on modeling ***and not on programming nor on the difficulties of logic theory.***" (*Id.* at 9, emphasis added.)

To configure a product based on the model created with PC/CON, Fohn discloses “*a computer tool that allows product configuration systems to be built by non-specialists and exercised by personnel with diverse skills.*” (Id. at 6, emphasis added.) Fohn provides several examples of the user interface used to configure a product based on the underlying model. A representative interface is Figure 5, reproduced below.

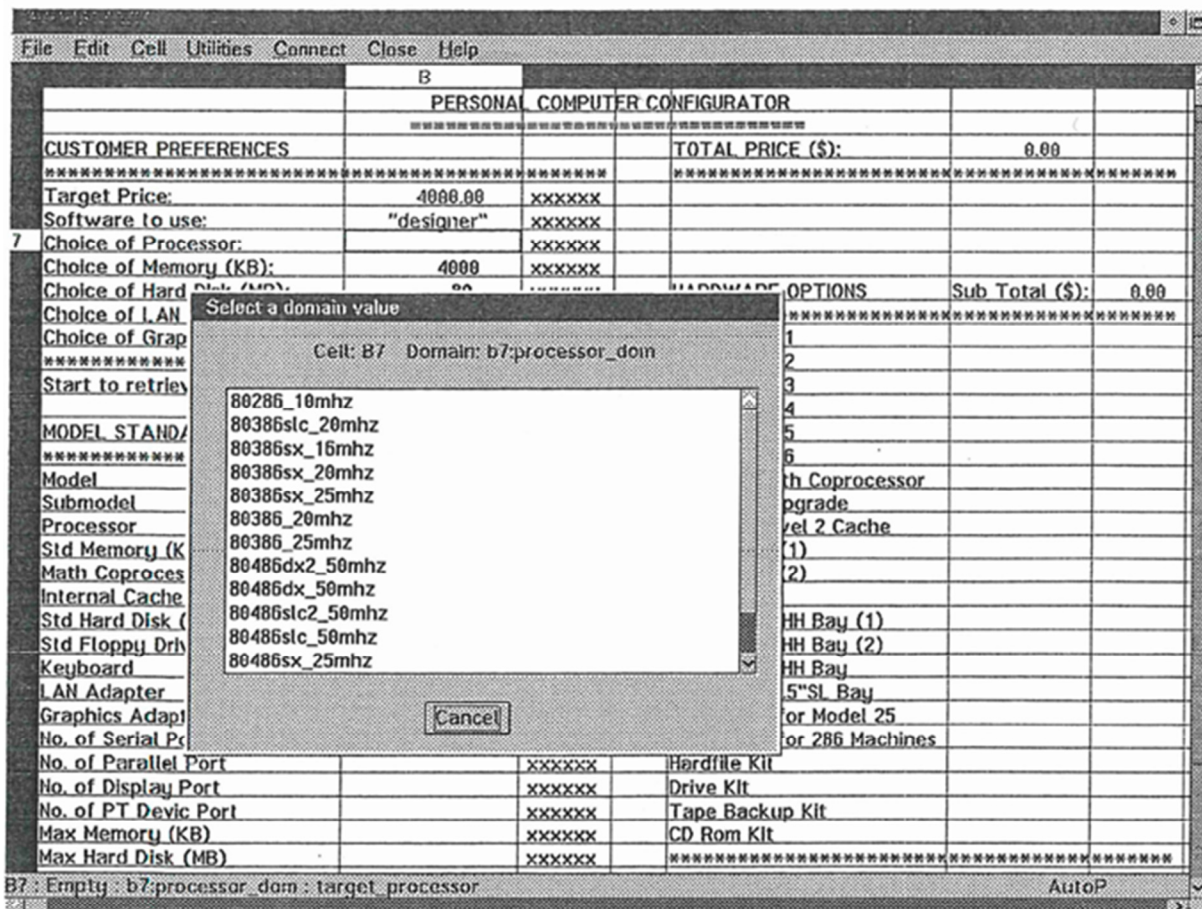


Fig. 5. Selection of a customer's preferred microprocessor.

(Id. at 13.)

Fohn provides an example of a configuration session in which user input is

received to configure a computer system. The user's input is validated against the product model throughout the configuration session to ensure that the session concludes with a valid computer configuration. (*Id.* at 15-16.)

Neither the Fohn paper nor the PC/CON application were considered during examination of the '651 Patent.

**C. Skovgaard: "salesPLUS" - 1995**

Skovgaard 1995 describes Beologic's "salesPLUS" product configurator, and an example in which "a car configuration is presented." (Ex. 1010, Skovgaard 1995, Abstract; *see* Ex. 1002, Greenspun ¶¶71-75.)

Like OBELICS and PC/CON, Beologic's salesPLUS software included a "maintenance" module in which constraints were input to define a product model, and a "configuration" module in which a sales person configured a product based on the model. (*Id.* at Sections 5.2 and 6.) Skovgaard provides an automotive example in which a 1992 Saab 900 is graphically configured using salesPLUS. (*Id.* at Section 7.) The configuration interface for salesPLUS is reproduced below.

The screenshot shows the 'QEDace Demo' window for a 'SAAB' system. The interface is organized into several sections:

- Left Sidebar:** Contains buttons for 'Default', 'Describe', 'Price', 'Undo', 'Minimize', 'Maximize', 'Limits', 'Warning', and 'Finish'. Below these is a 'Status' section with a black rectangular indicator.
- Main Configuration Area:**
  - Models:** A dropdown menu showing 'SAAB 900'.
  - Engine:** A dropdown menu showing '2.0s'.
  - Paints:**
    - Metal Paint type:** A dropdown menu showing 'x'.
    - Standard Paints:** A dropdown menu showing 'x'.
    - Metallic Paints:** A dropdown menu with options: 'Citrin Beige', 'Platana Grey', 'Le Mans Blue', 'Scarabe Green', 'Monte Carlo Yellow', 'None', and 'X' (selected).
  - Accessories:** A list of options with checkboxes: 'Automatic gearbox' (x), 'ABS Brakes' (x), 'Air Bag' (x), 'Air Condition' (x), 'Audio System' (x), 'Aircon' (x), 'Mirrors' (x), and 'Control' (x).
  - Trims:** A list of options with checkboxes: 'Leather Trim' (x), 'Trims' (x), and 'Trim Color' (x).
  - Other Options:** 'Sunroof' (x), 'Delivery Time' (14), 'Accessories at Dealer', and 'Warranty'.
- Status Bar:** Located at the bottom, it displays 'Price: 206483 dkr', 'Horse Power: 105 hp', and 'Weight: 1322 kg'.

(Id. at Section 7.3.)

Skovgaard describes one of the key drawbacks of prior configuration software: “the user is forced to make his choices in a predefined order.” (Id. at Section 5.1.) The salesPLUS configurator solves this problem by offering users “free order of choices.” (Id. at Abstract.) “*salesPLUS gives full freedom in the order of choices.*” (Id., Section 5.1, emphasis added.)

Skovgaard 1995, and the salesPLUS software described in that paper, were not considered during examination of the ’651 Patent. Collectively, Axling, Fohn and Skovgaard had already recognized, and solved, the problems identified in the ’651 Patent.



## **VI. Claim Construction – 37 C.F.R. § 42.104(b)(3)**

“A claim in an unexpired patent . . . shall be given its broadest reasonable construction in light of the specification of the patent in which it appears.” 37 C.F.R. § 42.100(b). For purposes of this petition, Petitioner does not believe any terms in the challenged claims require construction beyond their plain and ordinary meaning under the broadest reasonable interpretation standard required for this proceeding.

Petitioner respectfully requests the opportunity to present additional claim construction positions in the event that the broadest reasonable interpretation standard required for this Petition is modified and further attention to claim construction is required in this proceeding.

## **VII. Unpatentability Grounds**

Petitioner asserts that the Challenged Claims are invalid under 35 U.S.C. §103. For the reasons explained in detail below, Petitioner has established a reasonable likelihood of prevailing on the following grounds of unpatentability. 35 U.S.C. § 314(a); 37 C.F.R. § 42.104(b)(4).

**Ground 1. Claims 60-62, 64-68 and 70-71 are obvious in view of  
Axling, Fohn and the general knowledge of a person of  
ordinary skill in the art**

**A. A person skilled in the art at the time of the alleged  
invention would have combined the teachings of  
Axling and Fohn**

As summarized above, the Axling and Fohn papers each disclose a software application for (1) “defining” or “maintaining” a product model in terms of its possible components and component “relationships,” and (2) “configuring” a valid product for sale to a customer based on the model. (Ex. 1002, Greenspun ¶76.) Each of the papers seek to improve upon the original product configurators described in the literature, such as XCON, MICON, Cossack and others. (Ex. 1004, Axling at 6; Ex. 1005, Fohn at 6.)

A person of skill in the art at the time of the alleged invention would have had several reasons to draw from both the Axling and Fohn papers in constructing a configuration application. (Ex. 1002, Greenspun ¶¶76-93.) First, both papers address a common problem: the high complexity and limited usability of the original configurators. (*Id.* at ¶¶77-80.) Axling describes the original configurators as “time consuming and hard to maintain,” and introduces the OBELICS configurator which “present[s] a less complex and more interactive approach to configuration.” (Ex. 1004, Axling at 1-2.) Fohn similarly describes

the original configurators as “burdened by their size and complexity” and introduces the PC/CON configurator which “has the features and capabilities necessary to support sales representatives in personal computer product configuration.” (Ex. 1005, Fohn at 7 and 11.)

Together, Axling and Fohn provided a skilled artisan at the time of the invention with a comprehensive set of intuitive user interfaces and techniques to simplify the product definition and configuration experience associated with the original configurators. (Ex. 1002, Greenspun ¶¶81-83.) As a matter of common sense, a person of ordinary skill in the art at the time of the invention would have sought to implement a configurator that was both comprehensive in terms of its functionality, and user friendly to appeal to a broad audience of potential users. (*Id.*)

A person of ordinary skill in the art would have drawn from the teachings of both Fohn and Axling, because while each discusses all aspects of defining and configuring systems, the emphasis of the Axling and Fohn papers is somewhat different. (Ex. 1002, Greenspun ¶77.) Axling provides significant detail, illustration and examples concerning the product definition and maintenance aspects of the configurator, whereas Fohn provides more detail and interface examples on the end-user configuration and validation aspects. (*Id.*) The two references complement one another. (*Id.*)

Axling provides an intuitive “Domain Model Editor” (Figure 5.2) that enables a maintainer to graphically define a product in terms of its components and component relationships. (Ex. 1004, Axling at 13; Ex. 1002, Greenspun ¶78.) Axling states that the “graphical user interface mak[es] the domain model easy to understand and work with.” (*Id.* at 8.)

The interfaces disclosed in Fohn focus on the product configuration, which would take place after a product model is defined using the Domain Model Editor in Axling. In Figures 4-8, Fohn provides a series of example user interfaces for interactively configuring and validating a product based on the product definition and component relationships. (Ex. 1002, Greenspun ¶¶79-80.)

Axling and Fohn thus complement one another in the sense that Axling provides additional detail, examples and interfaces associated with product and relationship definition, whereas Fohn provides additional detail, examples and interfaces associated with the configuration session and validation of user selections. (Ex. 1002, Greenspun ¶81.) A person of skill in the art at the time of the alleged invention would have drawn upon the most innovative and intuitive aspects of Axling and Fohn to create a comprehensive and easy-to-use product configuration application. (*Id.* at ¶¶82-83.) For example, a person of skill in the art would have recognized the obvious advantages of coupling the Domain Model Editor interface described in Figure 5.2 of Axling for product modeling and

definition with the configuration interfaces described in Figures 4-8 of Fohn for product configuration and validation. (*Id.*) Combined, these teachings depict a well-rounded and intuitive product configuration application that can be used by individuals lacking a detailed knowledge of a configuration or programming language. (*Id.* at ¶83.)

Another reason for combining aspects of the PC/CON system described in Fohn with the OBELICS system described in Axling is the improvement achieved by the enhanced validation process described in Fohn. (*Id.* at ¶84.) Axling recognizes the obvious objective of concluding a configuration session with a valid configuration that satisfies all applicable constraints. (*Id.*) For example, Axling states that “if all constraints, preconditions, and dependencies are satisfied in a component hierarchy instance, it represents a valid configuration.” (Ex. 1004, Axling at 17.) Axling introduces a “Problem Solving Method” for achieving this, but does not illustrate an interface to intuitively take a user through the configuration validation process.

Fohn addresses this objective of Axling, providing a “violation window” (Fig. 8) for conveniently identifying invalidity conditions in the current configuration, and proposing recommendations for resolving them. (Ex. 1005, Fohn at 15-16; Ex. 1002, Greenspun ¶85.)

A person of skill in the art at the time of the alleged invention would have

recognized the obvious advantage of having a graphical user interface to identify conflicts and assist a user to resolve them during a product configuration session. (*Id.* at ¶86.)

Fohn (and Patent Owner Trilogy) recognized the obvious negative market impact of selling customers products that have been incorrectly configured:

Trilogy estimates that in the personal computer (PC) industry, between 30 and 85 percent of all orders are *incorrectly configured*. The result is an enormous expense for a company, especially when realizing that *infeasible product configurations* made at the point of sale may only be discovered much later in manufacturing or, worse yet, after delivery to the customer. The consequences of late discovery include having to change a customer's order (thereby delaying manufacturing and increasing the lead-time), forcing a customer to return a shipment of faulty orders, and the loss of sales. Furthermore, products that are incorrectly configured are most likely incorrectly quoted. As a result, a manufacturer may be required to give away free components which were omitted from the sales quote yet are needed to supply the customer with the promised functionality. *The costs involved, both monetary and the loss of customer good will, can be very high.*

(Ex. 1005, Fohn at 3-4, emphasis added.)

Similarly, as recognized in the Background of the Invention section of the '651 Patent, configuration users who become "frustrated or confused" may "abort" the configuration process altogether. (Ex. 1001 at 1:43-49.)

Recognizing these known market forces, and as a matter of common sense, a person of skill in the art at the time of the alleged invention had good reason to select the best interface features and configuration validation functionality among the handful of product configuration applications available at the time, such as OBELICS, PC/CON, and salesPLUS. (Ex. 1002, Greenspun ¶¶89.)

Combining the input validation and “violation window” concepts disclosed in Fohn with the configuration system disclosed in Axling could have been accomplished at the time of the invention according to traditional computer programming methods to yield very predictable results. (*Id.* at ¶¶90.) The configuration and validation interfaces and functionality described in Fohn could have readily been introduced to the code base for OBELICS disclosed in Axling to yield a highly-predictable product modeling and configuration utility. (*Id.*)

Configuration validation and resolution is a known technique disclosed in Fohn that will help to improve similar configuration software, such as that disclosed in Axling. (*Id.* at ¶¶91.) The improvement arises from minimizing the configuration of invalid or unbuildable combinations of parts, which can be costly to resolve and likely to frustrate customers as disclosed in Fohn. (Ex. 1005, Fohn at 3 – Introduction; Ex. 1002, Greenspun ¶¶91.)

Fohn recognized that efforts have been made to address this market need by performing “configuration checks” at the point of sale, or before the start of

manufacture. (Ex. 1005, Fohn at 4.) Fohn sought to improve on the prior approach in the manner described above, namely by validating user input during the configuration session itself. (Ex. 1002, Greenspun ¶92.) The existing and market forces recognized in Fohn would have encouraged the skilled artisan to introduce the concept of user input validation at the configuration stage into software that did not otherwise do so. (*Id.*) In this manner, Fohn provides a good reason to combine the validity checking techniques disclosed in Fohn's PC/CON application to other configuration software, such as the OBELICS software disclosed in Axling. (*Id.*)

It would have been obvious to try Fohn's approach to configuration and validity checking with the OBELICS software disclosed in Axling. (Ex. 1002, Greenspun ¶93.) Validating user input, improving the efficiency of the configuration process, and ultimately ensuring the configuration of a valid, buildable product were obvious and common sense objectives of configuration software at the time of the alleged invention. (*Id.*)

For the above reasons, it would have been obvious to a person of skill in the art at the time of the alleged invention to combine the teachings of Axling and Fohn. (*Id.* at ¶¶76-94.)



**B. Axling and Fohn disclose the subject matter of claims  
60-62, 64-68 and 70-71 and render each claim obvious**

*[60.0] A method of configuring a system comprising the steps of:*

*[60.1] generating a definition for said system, said definition  
containing one or more elements and being conveyed graphically  
using a set of product relationships;*

*[60.2] generating a set of part relationships between said one or  
more elements, said set of part relationships being conveyed  
graphically using a set of part relationships;*

In Figure 5.2, Axling discloses a Domain Model Editor for generating a definition for a system, *i.e.*, “compsys.”

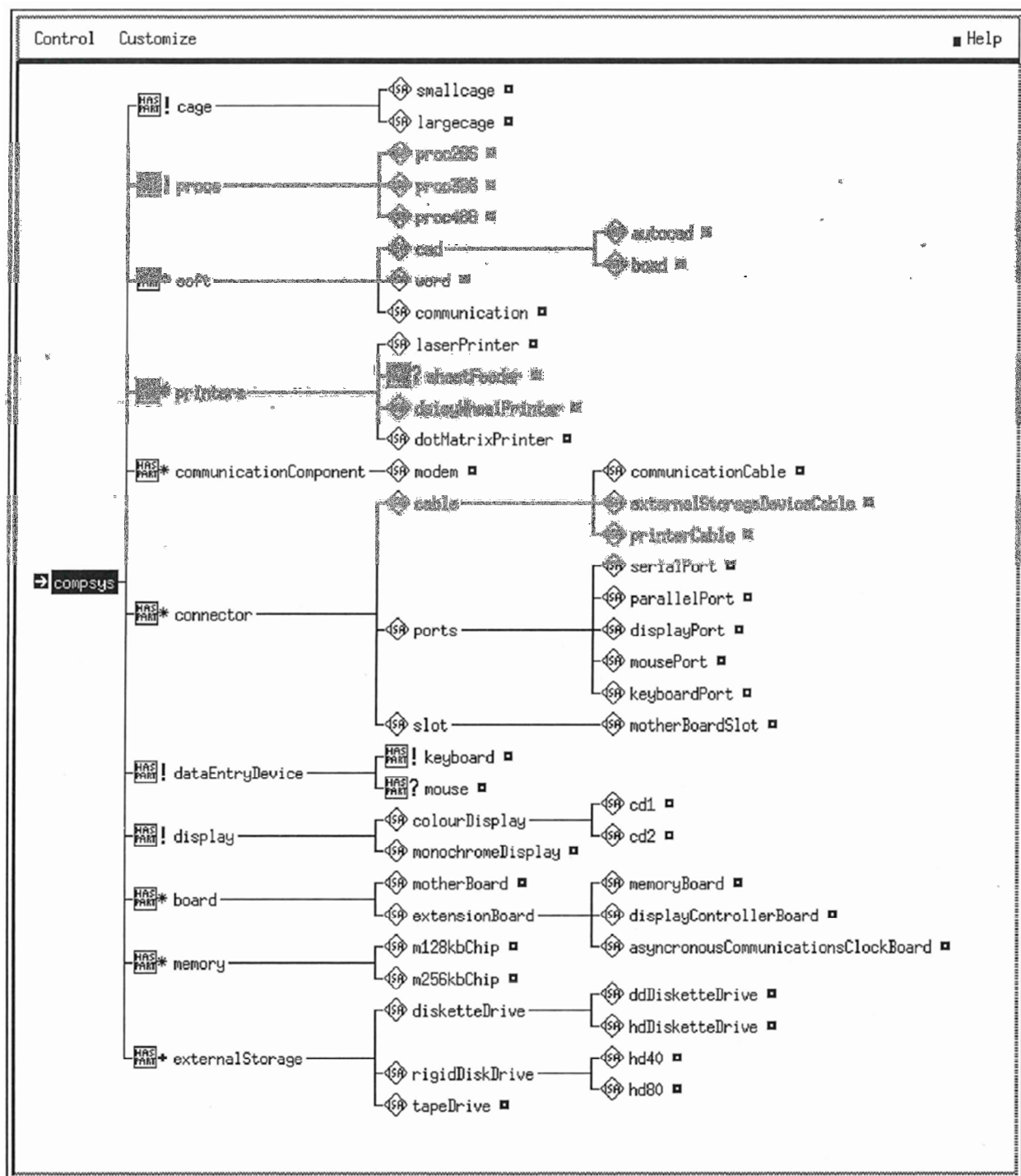


FIGURE 5.2. The Domain Model Editor

(Ex. 1004, Axling at 13; Ex. 1002, Greenspun ¶95.)

The definition includes a plurality of elements, such as “cage,” “procs”

(processor), “soft” (software), “printers,” and other elements that are related to the compsys product. As shown in Figure 5.2, these elements are conveyed graphically in the Domain Model Editor using a set of product relationships, depicted with the symbols ?, \*, !, and +. (Ex. 1002, Greenspun ¶¶96.) Axling describes these relationships as follows:

- ! : Exactly one is required to make the construction complete. We label these subcomponents *required component classes*.
- ? : One may be included. We label these subcomponents *optional component classes*.
- \* : Zero or more may be included. We label these subcomponents *optional iteration component classes*.
- + : One or more must be included. We label these subcomponents *required iteration component classes*.

(Ex. 1004, Axling at 15.)

As indicated above, the Patent Owner contended, in the 2005 *Selectica* litigation, that the term “product relationship” should be construed to mean “[a] classification of the specific association that exists between a product and one or more parts.” (Ex. 1011, Patent Owner’s Claim Construction Brief at 20 of 48.) The Eastern District of Texas later construed the term “product relationship” in claim 60 as “an association between a product and one or more parts, the association having a left-hand side and a right-hand side. The product represents the left-hand side of the relationship, and the set of elements represents the right-hand side of the relationship.” (Ex. 1019, CC Order at 11.)

Axling satisfies both constructions. (Ex. 1002, Greenspun ¶¶98.) As shown

above, the Domain Model Editor in Figure 5.2 discloses an association between the product “compsys” on the left, and its constituent parts on the right: “cage,” “procs,” “soft,” “printers,” etc. (*Id.*) The relationship itself is depicted, using the symbols !, \*, ? and + described above. (*Id.*)

Axling also discloses generating “part relationships” using the Domain Model Editor shown in Figure 5.2. (Ex. 1002, Greenspun ¶¶99.) Similar to the relationships between the “compsys” system and its component parts described above, the Domain Model Editor permits a user to define and graphically display relationships between the parts themselves. (*Id.*) For example, the part “printer” may optionally include the part “sheetFeeder.” (*Id.*) This relationship is depicted graphically in Figure 5.2 with the symbol “?”. (*Id.*)

The Patent Owner contended in the 2005 *Selectica* litigation that the term “part relationship” should be construed to mean “[a] classification of a specific association that exists between a first set of one or more parts and a second set of one or more parts or groups.” (Ex. 1011, Patent Owner’s Claim Construction Brief at 23 of 48.) The Eastern District of Texas later construed the term “part relationship” to mean “an association that exists between a first set of parts and a second set of parts, the association having a left-hand side and a right-hand side. The first set of parts represents the left-hand side of the relationship and the second set of parts represents the right-hand side of the relationship.” (Ex. 1019, CC

Order at 11.)

Axling satisfies both constructions. (Ex. 1002, Greenspun ¶101.) As shown above, the Domain Model Editor in Figure 5.2 discloses associations between a first set of parts on the left (e.g., “printers,” “dataentrydevice,” etc.), and a second set of parts on the right (“sheetFeeder,” “keyboard,” “mouse,” etc.). (*Id.*) The relationship itself is depicted in the middle, using the symbols ! and ? described above. (*Id.*)

***[60.3] receiving input from a configuration user;***

Axling and Fohn each disclose receiving input from a configuration user.

Axling introduces a concept he describes as “Interactive Configuration by Selection” or “ICS.” (Ex. 1004, Axling at 2; Ex. 1002, Greenspun ¶103.) Axling describes ICS as “a less complex and more interactive approach to configuration.” (*Id.*) Specifically, Axling describes an ICS that “allow[s] the user to alter previously made configuration decisions.” (Ex. 1004, Axling at 5; Ex. 1002, Greenspun ¶103.) The ICS system “presents all valid alternatives and also makes it easier to alter previous choices.” (Ex. 1004, Axling at 6; Ex. 1002, Greenspun ¶103.) Axling describes a “User Interface” or “GUI” including: one part for “presenting alternatives in an understandable way to the user,” and another part for enabling a user “at all times during a configuration session to view and edit the current state of the configuration.” (Ex. 1004, Axling at 19-20; Ex. 1002,

Greenspun ¶103.)

Similarly, PC/CON, disclosed in Fohn, is “a computer tool that allows product configuration systems to be built by non-specialists and exercised by personnel with diverse skills.” (Ex. 1005, Fohn at 6; Ex. 1002, Greenspun ¶104.) As explained above, Fohn provides an example interface in Figure 4, and pop-up window in Figure 5, where a sales representative inputs customer preference and option selections during a configuration session. (Ex. 1002, Greenspun ¶104.) This “example PC/CON session” during which user input is received is described and illustrated in detail at in connection with Figures 4-8. (Ex. 1005, Fohn at 12-15; Ex. 1002, Greenspun ¶104.)

***[60.4] validating said input based on said definition, said set of product relationships, said set of part relationships, and a current configuration state;***

Axling and Fohn each disclose validating user input based on the product and part relationships, and a current configuration state.

Axling discloses a “Problem Solving Method” which includes iteratively “evaluating” instances of a component hierarchy to validate those instances against the defined model. (Ex. 1004, Axling at 17; Ex. 1002, Greenspun ¶106.) Axling states that “if all constraints, preconditions, and dependencies are satisfied in a component hierarchy instance, it represents a valid configuration.” (*Id.*) Axling discloses a multi-step process for validating an instance:

**1. Set user-specifiable attribute values** The user is prompted to set the instance's user-specifiable attribute values.

**2. Try to satisfy the instance's preconditions** If any of the preconditions fails then *the instance cannot be included in the construction, so the evaluation of the instance fails.*

(Ex. 1004, Axling at 17-18, emphasis added; Ex. 1002, Greenspun ¶106.)

Fohn states that an objective of PC/CON is to “[e]nsure the customized configuration is a valid configuration.” (Ex. 1005, Fohn at 11-12; Ex. 1002, Greenspun ¶107.) The product model disclosed in Fohn is defined in terms of “constraints” which Fohn describes as “any interrelationship among elements of a model; for example, a constraint can be a rule, equation, relation, rows in a database table or any data structure.” (Ex. 1005, Fohn at 8; Ex. 1002, Greenspun ¶107.)

Fohn states that PC/CON performs “[o]rder [v]alidation” to “[d]etermine that the system ordered by a customer can indeed be manufactured.” (Ex. 1005, Fohn at 13; Ex. 1002, Greenspun ¶108.)

Fohn provides a detailed “example PC-CON session” in which a user (e.g. salesperson) provides input including “customer preferences,” “model standards,” and “hardware options.” (Ex. 1005, Fohn at 12-13; Ex. 1002, Greenspun ¶109.)

Figure 4 shows a graphical user interface for providing this information.

PCCON.SWK

File Edit Cell Utilities Connect Close Help

PERSONAL COMPUTER CONFIGURATOR			
CUSTOMER PREFERENCES		TOTAL PRICE (\$): 0.00	
Target Price:	4000.00	xxxxxx	
Software to use:	"designer"	xxxxxx	
Choice of Processor:		xxxxxx	
Choice of Memory (KB):	4000	xxxxxx	
Choice of Hard Disk (MB):	80	xxxxxx	HARDWARE OPTIONS Sub Total (\$): 0.00
Choice of LAN Capability:		xxxxxx	
Choice of Graphics Adapter:	"vga"	xxxxxx	
Start to retrieve?		xxxxxx	
MODEL STANDARDS Basic Price(\$):			
Model		xxxxxx	Memory Kit 1
Submodel		xxxxxx	Memory Kit 2
Processor		xxxxxx	Memory Kit 3
Std Memory (KB)		xxxxxx	Memory Kit 4
Math Coprocessor		xxxxxx	Memory Kit 5
Internal Cache (KB)		xxxxxx	Memory Kit 6
Std Hard Disk (MB)		xxxxxx	Optional Math Coprocessor
Std Floppy Drive		xxxxxx	Processor Upgrade
Keyboard		xxxxxx	Optional Level 2 Cache
LAN Adapter		xxxxxx	3.5"HH Bay (1)
			3.5"HH Bay (2)
			5.25"HH Bay
			5.25"HH/3.5"HH Bay (1)
			5.25"HH/3.5"HH Bay (2)
			5.25"FH/3.5"HH Bay
			5.25"SL or 3.5"SL Bay
No. of PT Devic Port		xxxxxx	Tape Backup Kit
Max Memory (KB)		xxxxxx	CD Rom Kit
Max Hard Disk (MB)		xxxxxx	

B5: Value : bf:software: dom: target: software AutoP

Fig. 4. Configuration's attribute region in the PC/CON constraint sheet.

(Ex. 1005, Fohn at 12; Ex. 1002, Greenspun ¶109.)

Information may be input (e.g., "Target Price"), or selected from a drop-down menu of available options (e.g., "Choice of Processor"). (Ex. 1002, Greenspun ¶110.) Figure 5 shows the drop-down menu providing the user with processor options from which to choose.



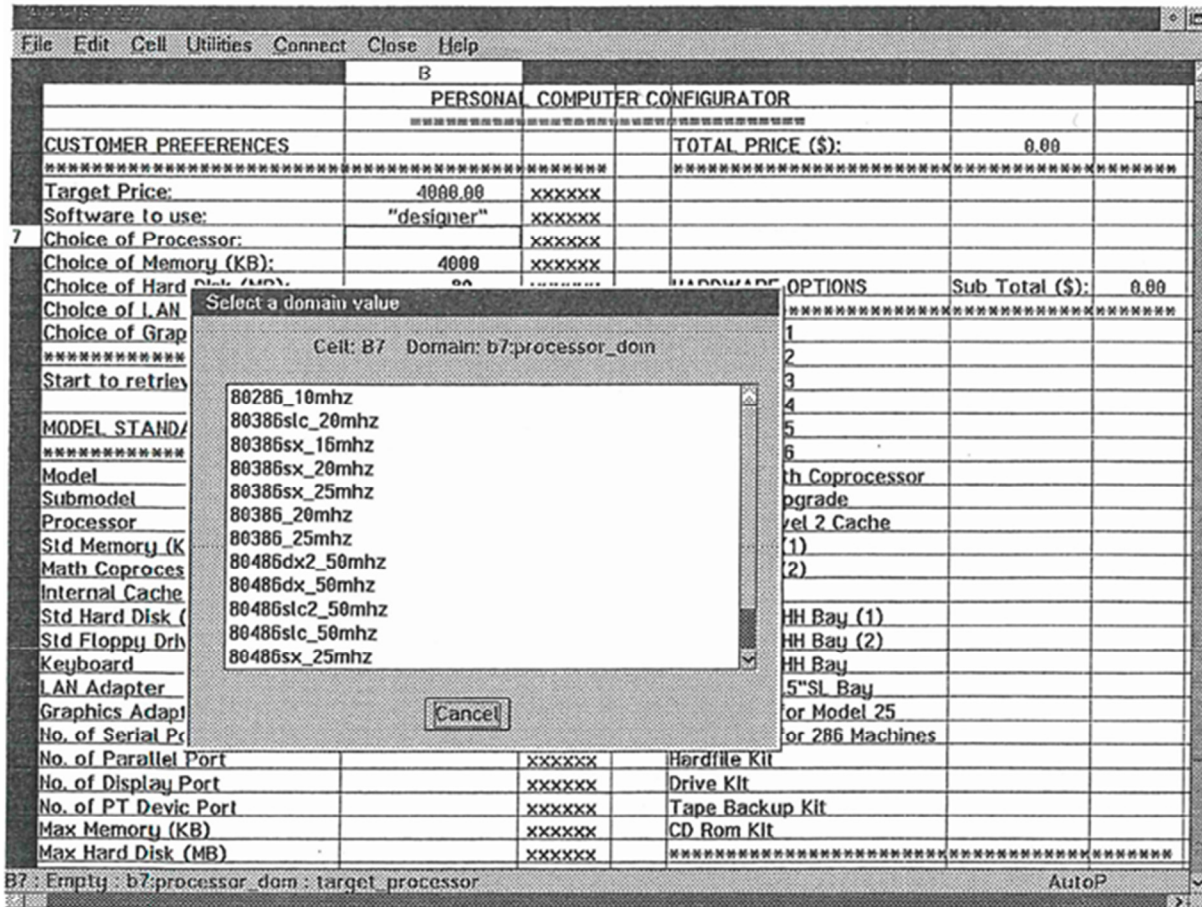


Fig. 5. Selection of a customer's preferred microprocessor.

(Ex. 1005, Fohn at 13; Ex. 1002, Greenspun ¶110.)

Fohn provides an example in which user input is validated against the product model maintained in the relational database, and the state of the current configuration. (Ex. 1005, Fohn at 15-16; Ex. 1002, Greenspun ¶111.) In the example, a user selects 20MB of memory, whereas the maximum allowable memory for the previously-selected processor is 16MB. (Ex. 1005, Fohn at 15-16; Ex. 1002, Greenspun ¶111.) This scenario creates a "violation" condition. (Ex. 1005, Fohn at 16; Ex. 1002, Greenspun ¶111.)

Figure 8 below shows a “violation window” that identifies the constraint that was violated, the input that caused the violation, and a “description of how to resolve the violation.” (Ex. 1005, Fohn at 16; Ex. 1002, Greenspun ¶112.)

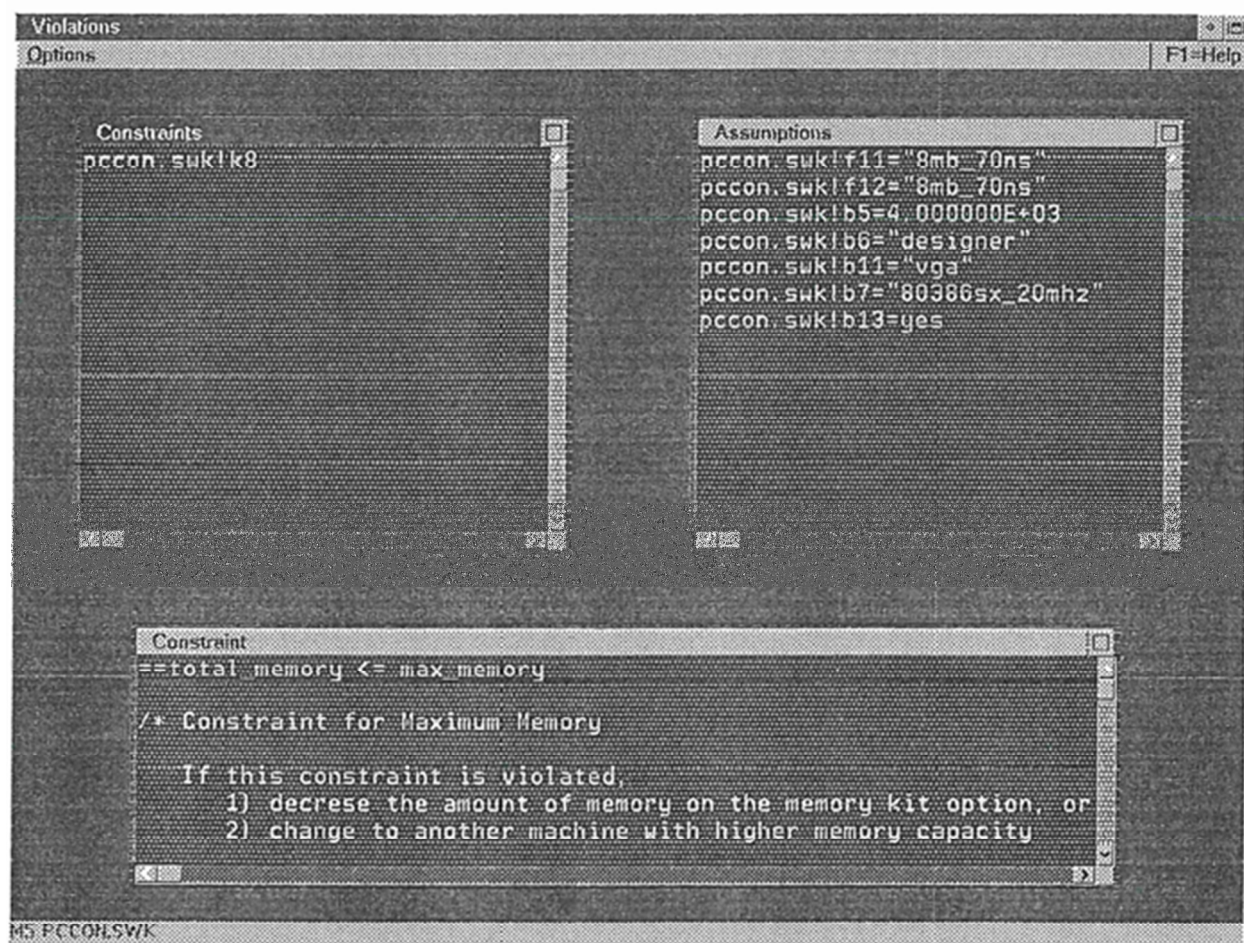


Fig. 8. Window displaying assistance to help resolve constraint violations.

(Ex. 1005, Fohn at 16; Ex. 1002, Greenspun ¶112.)

This validation of user input is based on the product definition, relationships between the product and parts and the current configuration state. (Ex. 1002, Greenspun ¶113.) In this example, the violation was created because the product definition specified an included part relationship, *i.e.*, the computer system

included 4MB of memory. (Ex. 1005, Fohn at 15-16; Ex. 1002, Greenspun ¶113.) Part-to-part relationships were defined in the model such that multiple memory kits are available (e.g., “Memory Kit 1” and “Memory Kit 2”) for the selected processor (e.g., “80386sx\_20mhz”), having a maximum allowable memory of 16MB. (Ex. 1002, Greenspun ¶113.) These relationships between the processor and memory kits defined in the model are propagated during the configuration session based on the processor selection. (Ex. 1005, Fohn at 14-15; Ex. 1002, Greenspun ¶113.)

The validation of the user selections is based on the current configuration state. In the provided example, the user selected two 8MB memory kits, which when combined with the 4MB already included in the system, results in 20 MB of memory – more than the maximum allowable memory thus creating the “violation.” (Ex. 1002, Greenspun ¶114.) Indeed, the constraint in the model that was violated is provided in the violation window shown in Figure 8. (Ex. 1002, Greenspun ¶114.)

***[60.5] identifying a set of valid configuration options using said definition, said set of product relationships, said set of part relationships and said current configuration state.***

Both Axling and Fohn disclose providing valid configuration options to a user based on the underlying product model, including the product and part relationships, and the current configuration state. As explained above with respect

to the “validating” step, Axling discloses a “Problem Solving Method” which includes iteratively “evaluating” instances of a component hierarchy to validate those instances against the defined model and other users selections in the current configuration state. (Ex. 1004, Axling at 17; Ex. 1002, Greenspun ¶115.) Based on that model, Axling “presents all valid alternatives and also makes it easier to alter previous choices.” (Ex. 1004, Axling at 6; Ex. 1002, Greenspun ¶115.)

Similarly, in Fohn, the PC/CON application identifies a set of valid configuration options using the system definition, product and part relationships, and the current configuration state. In the configuration session example, Fohn discloses value propagation following the user’s selection of the chosen processor (*i.e.*, 80386sx\_20mhz). (Ex. 1005, Fohn, at 14; Ex. 1002, Greenspun ¶116.) This value propagation updates the current configuration state to reflect the selection. Next, Fohn describes presenting the user with “submodels which meet the customer’s preferences.” (Ex. 1005, Fohn at 14; Ex. 1002, Greenspun ¶116.) Fohn states that the submodels are then displayed in a window similar to that shown in Figure 5 for the sales representative to select. (*Id.*)

In this example, presentation of the submodels to the user following (and based on) the user’s previous processor selection constitutes identifying a set of valid configuration options using the definition, the set of product relationships, the set of part relationships and the current configuration state. (Ex. 1002, Greenspun

¶117.) In addition, Figure 8 of Fohn depicts a “violation window” having “a set of valid configuration options,” namely decreasing the amount of memory on the memory kit option, or changing to another machine with a higher memory capacity. (Ex. 1002, Greenspun ¶117.) These options are based on the product definition, the product and part relationships defined in the model, and the current configuration state. (*Id.*) As shown in Figure 8, PC/CON reports that the selected memory must be “decrease[d],” or a machine with a “higher memory capacity” must be selected. (Ex. 1005, Fohn at Fig. 8; Ex. 1002, Greenspun ¶117.) This is because the validation, and the recommended solution to the violation, are based on the underlying system definition, relationships, and the current configuration state. (Ex. 1002, Greenspun ¶117.)

***[61.0] The method of claim 60 wherein said step of identifying further comprises the steps of:***

***[61.1] determining whether a relationship in said set of relationships is active;***

***[61.2] modifying said configuration to satisfy said active relationship when it is determined that a modification is needed;***

Patent Owner previously asserted in the 2005 *Selectica* litigation that an “active” relationship is “a relationship that is presently in effect.” (Ex. 1011, Patent Owner’s Claim Construction Brief at 32 of 48.) The Eastern District of Texas later construed the term as “a relationship in which all elements on the left-



hand side of the relationship are selected.” (Ex. 1019, CC Order at 13.) Fohn satisfies both constructions. (Ex. 1002, Greenspun ¶118.)

The ’651 specification describes the “left hand side” as a part or group of parts to the left of a specified relationship (*e.g.*, includes, excludes, requires choice, etc.), and the “right-hand side” is the part or group of parts to the right of a relationship. (Ex. 1001, ’651 Patent, 8:43-53; Ex. 1002, Greenspun ¶119.)

An example relationship is as follows: “Part A *includes* Part B and Part C.” In this example, “Part A” is the left-hand side of the relationship, and “Part B and Part C” is the right-hand side of the relationship. (Ex. 1002, Greenspun ¶120.) The relationship itself is an “includes” relationship. (*Id.*)

In the configuration example provided in Fohn, the PC/CON software determines that the relationship between the selected processor “80386sx\_20mhz” (LHS) and available memory (RHS) (16MB) is active, or presently in effect, because the software informs the user that constraints associated with the user’s selections have been violated during validation: “[c]onsequently, two constraints located in cells J8 and K8 are now violated and their respective cell boxes would change from ‘#T’ to ‘#F’.” (Ex. 1005, Fohn at 16; Ex. 1002, Greenspun ¶121.) These constraints correspond to Memory Kit #1 (8MB) and Memory Kit #2 (8MB). (Ex. 1002, Greenspun ¶121.)

Fohn also discloses modifying the configuration to satisfy the active

relationship when it is determined that a modification is needed. (Ex. 1002, Greenspun ¶122.) The “violation window” shown in Figure 8 reveals a determination that modification is needed. (*Id.*) The window provides recommended steps for modifying the configuration, *i.e.*, “decrease the amount of memory on the memory kit option.” (Fohn at 16, Fig. 8; Ex. 1002, Greenspun ¶122.) Fohn then explains that “[b]y deleting the value for “Memory Kit 2,” the system will re-evaluate the configuration and conclude that the configuration is valid.” (Ex. 1005, Fohn at 16; Ex. 1002, Greenspun ¶122.)

Therefore, the relationship between the processor and memory is determined to be active in Fohn, and the configuration is modified (removing Memory Kit #2) to satisfy the active relationship between the processor and memory. (Ex. 1002, Greenspun ¶123.)

***[61.3] making said relationship not Activateable when it is determined that said relationship is not active and the activation of said relationship would render the configuration invalid.***

Patent Owner previously asserted in the *Selectica* litigation that a “not Activateable relationship” is “a relationship that cannot be put into effect.” (Ex. 1011, Patent Owner’s Claim Construction Brief at 33 of 48.) Fohn discloses determining at least some relationships that would result in an invalid configuration, and making those relationships “not applicable” or “na.” (Ex. 1002, Greenspun ¶125.)

Specifically, Fohn discloses “value propagation” in which user selections during the configuration session impact the scope of available options for future selections by the user. (Ex. 1005, Fohn at 14-15; Ex. 1002, Greenspun ¶126.) In the example configuration shown in Figure 6, Fohn discloses that certain “not applicable” relationships are not put into effect because they would result in an invalid configuration:

If we continue with the example and focus on the Hardware Options section of Fig. 6, all of the Bay Options and some of the other options are filled with the value *na*. *na* means that an option is "not applicable" and is used to indicate *those options that have been removed from further consideration because they are not compatible with options already selected*.

(Ex. 1005, Fohn at 15, emphasis added; Ex. 1002, Greenspun ¶126.)

Thus, Fohn discloses each limitation of dependent claim 61. (Ex. 1002, Greenspun ¶127.)

***[62.0] The method of claim 61 wherein said step of determining whether a relationship is active further comprises the step of determining whether the elements specified in the left-hand side of said relationship are present in said configuration.***

Claim 62 introduces the limitation “the left-hand side of said relationship,” yet neither claim 60 nor claim 61 recite a “left-hand side.” (Ex. 1002, Greenspun ¶128.) This limitation thus lacks antecedent basis in claim 60 or 61.



Claim 62 does not specify the nature of the claimed “left hand side of the relationship.” As explained above, the ’651 specification describes the “left hand side” as a part or group of parts to the left of a specified relationship (*e.g.*, includes, excludes, requires choice, etc.), and the “right-hand side” is the part or group of parts to the right of a relationship. (’651 Patent, 8:43-53; Ex. 1002, Greenspun ¶129.)

The ’651 specification describes “Relationship Evaluation” for determining whether a particular relationship is “active” or “inactive.” (Ex. 1001, ’651 Patent at 12:35-44; Ex. 1002, Greenspun ¶130.) The specification states that “relationships associated with items contained in the product definition are evaluated when user input is received.” (*Id.*) “A relationship is “active” when all the items on the left-hand side of the relationship are selected.” (*Id.*)

Fohn discloses a table in Figure 6 in which user selections during the configuration session are propagated throughout the table, defining the scope of available options for future selections by the user. (Ex. 1005, Fohn at 13-15; Ex. 1002, Greenspun ¶131.) Fohn discloses that “[a] user can assign certain values to variables solely to view the propagation and the effect that the variable’s value will have on a configuration.” (Ex. 1005, Fohn at 13; Ex. 1002, Greenspun ¶131.)

Continuing with the example, after choosing 80386sx 20mhz, the sales representative initiates value propagation by selecting yes for the variable “Start to retrieve.” Once value propagation is initiated, the

system accesses the database to find the submodels which meet the customer's preferences.

(Ex. 1005, Fohn at 14.)

Figure 6 illustrates the “[r]esulting propagation after entering the customer’s preferences.” The purpose of value propagation described in Fohn is to determine which relationships are active when the “left-hand side” of the relationship is present in the configuration. (Ex. 1002, Greenspun ¶132.)

In this fashion, Fohn discloses determining whether elements specified in the left-hand side of a relationship are present in a configuration as recited in claim 62. (Ex. 1002, Greenspun ¶133.)

In addition, it was well known at the time of the invention to represent knowledge relationships and rules in the context of expert and configuration systems using a left-hand / right-hand expression. (Ex. 1002, Greenspun ¶134.) For example, in MICON: A Knowledge Based Single Board Computer Designer,<sup>3</sup> Birmingham describes MICON, a *Micro*-processor *Configurator* application written in the OPS-5 production system language. (Ex. 1006, Birmingham at 1.) MICON is addressed in Axling’s survey of existing product configuration

---

<sup>3</sup> Ex. 1006 [Birmingham] is W.P. Birmingham, D.P. Siewiorek, MICON: A Knowledge Based Single Board Computer Designer, Proceedings of 21st Design Automation Conference, IEEE CS Press, Pages 565-571 (1984).

software. (Ex. 1004, Axling at 6; Ex. 1002, Greenspun ¶134.) Birmingham states:

Knowledge is explicitly represented in production systems as a set of rules or productions. The terminology used to refer to these productions is *situation-action* pairs. The left hand side (LHS) of a production, corresponding to the IF part, defines some situation or pattern to match. The right hand side (RHS), corresponding to the THEN part, describes some action that is to take place.

(Ex. 1006, Birmingham at 569, italics in original.)

In Introduction to Knowledge Systems,<sup>4</sup> Stefik similarly explains, in the context of “if-then” rules, that “the if-part is often called the **left side** and the then-part is called the **right side**.” (Ex. 1007, Stefik at 115, emphasis in original; Ex. 1002, Greenspun ¶135.) Similarly, U.S. Patent No. 5,283,857 to Evangelos discloses an “expert system” having a variety of different rule types. (Ex. 1008, Evangelos at 5:1-35; Ex. 1002, Greenspun ¶135.) “Production rules,” for example, include a “left hand side, which identifies one or more conditions, and a right hand side, which has an action list.” (*Id.*, 5:5-8; Ex. 1002, Greenspun ¶135.) The left-hand side of a “production rule” may include a “design constraint.” (*Id.*, 5:9-10; Ex. 1002, Greenspun ¶135.) The right-hand side may include an “action list” which operates “to remove the element from, or add the element to, the design

---

<sup>4</sup> Ex. 1007 [Stefik] is Mark Stefik, Introduction to Knowledge Systems, Morgan Kaufmann Publishers, Inc., San Francisco, CA (June 15, 1995).

representation.” (*Id.*, 5:27-30; Ex. 1002, Greenspun ¶135.)

These publications further confirm that it was well known and obvious at the time of the alleged invention to determine whether a relationship, rule or condition applies by determining whether the conditions on the “left-hand side” are satisfied. (Ex. 1002, Greenspun ¶136.)

***[64.0] The method of claim 61 wherein said relationship is an includes relationship, said step of modifying further comprises the step of including in said configuration elements identified in a right-hand side of said relationship.***

As explained above with respect to claim 61, Fohn discloses an “includes” relationship, and modifying the configuration to satisfy the relationship. (Ex. 1002, Greenspun ¶137.) Fohn also discloses including in the configuration elements identified in a right-hand side of the “includes” relationship. (*Id.*) Specifically, when the user selects the processor as shown in Figure 5, the “value propagation” process causes “included” features to populate as a result of that selection, such as standard memory, standard floppy drive, and a maximum memory limit of 16000 KB – the limit that caused the configuration to be modified as explained above with respect to claim 61. (Fohn at 14-15, Fig. 6; Ex. 1002, Greenspun ¶137.) In this example, the selection of the processor (80386sx\_20mxz) is the left-hand side of the includes relationship, and the standard values populated in Fig. 6 as a result of value propagation constitute the

right-hand side of the includes relationship. (Ex. 1002, Greenspun ¶137.)

A modification to the configuration that constitutes including an element identified in a right-hand side of a relationship may also include adding the “keyboard.” (Ex. 1002, Greenspun ¶138.) Figure 6 shows that a keyboard is part of the “model standard” associated with the user-selected 80386sx\_20mxz processor. In other words, the keyboard is a right-hand side element of an “includes” relationship with the left-hand side processor element. (*Id.*) Because the keyboard selection has not been made, however, the configuration user must modify the current configuration state to add the necessary keyboard and complete the right-hand side of the relationship. (*Id.*)

Thus, Fohn discloses an “includes” relationship, and modifying the configuration to include elements identified in a right-hand side of a relationship as recited in claim 64. (Ex. 1002, Greenspun ¶139.)

***[65.0] The method of claim 61 wherein said relationship is an excludes relationship, said step of modifying further comprises the step of marking said elements identified in the right-hand side of said relationship as excluded and notSelectable.***

As explained above with respect to claim 61, Fohn discloses “value propagation” in which user selections during the configuration session impact the scope of available options for future selections by the user. (Ex. 1005, Fohn at 14-15; Ex. 1002, Greenspun ¶140.) In the example configuration shown in Figure 6,

Fohn discloses that certain “not applicable” relationships resulting from the user’s processor selection are not put into effect because they would result in an invalid configuration: “*na* means that an option is ‘not applicable’ and is used to indicate *those options that have been removed from further consideration because they are not compatible with options already selected.*” (Ex. 1005, Fohn at 15, emphasis added; Ex. 1002, Greenspun ¶140.) With the “na” designation, these elements on the right-hand side of the excludes relationship (*e.g.*, Memory Kit 3-6, 3.5" HH Bay, etc.) are marked as excluded and notSelectable (“na”) in the configuration as shown in Figure 6. (Ex. 1002, Greenspun ¶140.)

Thus, Fohn discloses an excludes relationship, and marking components on the right-hand side as notSelectable as a result of that relationship. (Ex. 1002, Greenspun ¶141.)

***[66.0] The method of claim 61 wherein said relationship is a removes relationship, said step of modifying further comprises the step of removing said elements identified in the right-hand side of said relationship from said configuration.***

As explained above with respect to claim 61, Fohn discloses “value propagation” in which user selections during the configuration session impact the scope of available options for future selections by the user. (Ex. 1005, Fohn at 14-15; Ex. 1002, Greenspun ¶142.) In the example configuration shown in Figure 6, Fohn discloses that certain “not applicable” relationships are not put into effect

because they would result in an invalid configuration: “*na* means that an option is “not applicable” and is used to indicate *those options that have been removed from further consideration because they are not compatible with options already selected.*” (Ex. 1005, Fohn at 15, emphasis added; Ex. 1002, Greenspun ¶142.) In this manner, Fohn discloses removing elements from the configuration. (Ex. 1002, Greenspun ¶142.)

The current configuration may also be modified, such that other elements on the right-hand side of a removes relationship are removed, when a user iteratively changes aspects of the configuration to observe the impact of value propagation. (Ex. 1002, Greenspun ¶143.) For example, Fohn states “a user can assign certain values to variables solely to view the propagation and the effect that the variable’s value will have on a configuration.” (Ex. 1005, Fohn at 13, emphasis added; Ex. 1002, Greenspun ¶143.) This iterative value propagation in a configuration model will naturally cause *different* model standards and hardware options to bear the “na” designation resulting from their removal, thus removing incompatible items from the previous configuration. (Ex. 1002, Greenspun ¶143.)

Thus, Fohn discloses a removes relationship, and modifying the configuration to remove components as a result of that relationship. (Ex. 1002, Greenspun ¶144.)

***[67.0] The method of claim 61 wherein said relationship is a requires choice relationship, said step of modifying further comprises the steps of:***

***[67.1] determining whether a plurality of elements identified in a right-hand side of said relationship are present in said configuration such that said relationship is satisfied;***

***[67.2] modifying said configuration to satisfy said requires choice relationship when it is determined that only one path exists to satisfy said relationship and said relationship is not already satisfied.***

In Figure 5, Fohn discloses a “Choice of Processor” which, upon selection, causes the display of a pop-up window to enable a configuration user to make the required choice. (Ex. 1005, Fohn at 13-14; Ex. 1002, Greenspun ¶145.)

Fohn explains that when a user selects a processor from the pop-up window (80386sx\_20mhz), value propagation occurs and “submodels” are displayed in a window for the sales representative to *select* in a manner similar to that just discussed. (Ex. 1002, Greenspun ¶146.) After the sales representative makes a *selection*, the system performs further data retrieval as well as value propagation.” (Ex. 1005, Fohn at 14-15; Ex. 1002, Greenspun ¶146.)

In this example, the selected processor (80386sx\_20mhz) is the left-hand side of a “requires choice” relationship, and the “submodels” that are subsequently displayed for user selection are the right-hand side of the relationship. (Ex. 1002, Greenspun ¶147.) Fohn thus determines whether a plurality of elements identified



in a right-hand side of said relationship are present in said configuration such that said relationship is satisfied. (*Id.*) This determination causes the pop-up menu described in Fohn for selecting the submodel. (*Id.*)

Upon receiving input selecting one of the submodels on the right-hand side of the relationship, the configuration is modified through value propagation to satisfy the relationship, and that modification “results in a fully configured standard offering as shown in Fig. 6.” (Ex. 1005, Fohn at 15; Ex. 1002, Greenspun ¶148.) In this manner, Fohn discloses modifying the configuration to satisfy the requires choice relationship because only one path exists to satisfy said relationship (selecting a submodel) and that relationship was not already satisfied. (Ex. 1002, Greenspun ¶148.)

***[68.0] The method of claim 61 wherein said step of making said relationship notActivatable further comprises the step of marking one or more elements in said left-hand side of said relationship as notSelectable.***

As explained above with respect to claim 61, Fohn discloses “value propagation” in which user selections during the configuration session impact the scope of available options for future selections by the user. (Ex. 1005, Fohn at 14-15; Ex. 1002, Greenspun ¶149.) In the example configuration shown in Figure 6, Fohn discloses that certain “not applicable” relationships are not put into effect because they would result in an invalid configuration: “*na* means that an option is

"not applicable" and is used to indicate *those options that have been removed from further consideration because they are not compatible with options already selected.*" (Ex. 1005, Fohn at 15, emphasis added; Ex. 1002, Greenspun ¶149.)

Providing a "not Selectable" designation on the right-hand side of a relationship, or the left-hand side, would have been considered obvious to a person of skill in the art at the time of the alleged invention. (Ex. 1002, Greenspun ¶150.) First, there are only two possible options for implementing in a given relationship, either the left-hand side or the right-hand side. (*Id.*) Implementing the relationship on one side, as opposed to the other, is not inventive. (*Id.*) Given that there are only two alternatives, it would be obvious to try both of them. (*Id.*) Second, the left hand side and the right hand side of a relationship in a *product* model are often *physically* interrelated in the sense that some components are physically combinable in the product, and others are not. (*Id.*) Thus, a person of ordinary skill in the art would have good reason to provide the "not Selectable" designation on the right-hand side of a relationship, or the left-hand side. (*Id.*) This ensures that whether the person configuring a product modifies an element on the left-hand side or the right-hand side of a relationship involving incompatible parts, the other side in the configuration is excluded to avoid physical incompatibility in the resulting product. (*Id.*) As explained in the obviousness section above, that was a well-known problem and there was a substantial market force to solve it. (*Id.*) A

person of ordinary skill in the art at the time of the invention would have considered this a matter of common sense. (*Id.*) Finally, placing the “na” or “not Selectable” designation on the right-hand side of a relationship, or the left-hand side is highly predictable, as these are low-level constraints having a low level of granularity in configuring a product. (*Id.*)

***[70.0] The method of claim 60 wherein said step of validating further comprises the step of determining whether the relationships other than those marked notActivateable in said set of relationships can be satisfied.***

Fohn discloses the step of validating whether relationships are satisfied without regard to elements that are marked notActivateable. (Ex. 1002, Greenspun ¶151.) In Figure 6, several Hardware Options are designated “na” or “not applicable” based on previous user selections and value propagation. (Ex. 1005, Fohn at 15; Ex. 1002, Greenspun ¶151.) For example, selection of the chosen processor makes Memory Kits 3-6 and several disk bays not applicable to the current configuration. (Ex. 1002, Greenspun ¶151.) Fohn states that “[a]fter the sales representative makes a selection, the system performs further data retrieval as well as value propagation. (*Id.*) Propagation is automatic and results in a fully configured standard offering as shown in Fig. 6.” (Fohn at 14-15; Ex. 1002, Greenspun ¶151.)

Thus, Fohn discloses determining whether the relationships other than those

marked notActivatable in said set of relationships can be satisfied. (Ex. 1002, Greenspun ¶152.)

***[71.0] The method of claim 60 wherein said step of generating a definition selects said one or more elements for inclusion in said definition, said input is made without regard for the order in which said one or more elements are selected for said definition.***

Axling discloses a tool for “Interactive Configuration by Selection” or ICS. Axling seeks to maximize flexibility in the product definition process, stating “an ICS system should, at any time during a configuration, be able to present the current state of the configuration and allow the user to alter previously made configuration decisions.” (Ex. 1004, Axling at 5; Ex. 1002, Greenspun ¶153.) Axling states that an ICS system “makes it easy to alter previous choices.” (Ex. 1004, Axling at 6; Ex. 1002, Greenspun ¶153.) With respect to the OBELICS application, Axling discloses a “Domain Model Editor” that allows the user to “alter previous decisions” in the configuration. (Ex. 1004, Axling at 9; Ex. 1002, Greenspun ¶153.) The Domain Model Editor “makes it easy to add and modify components.” (Ex. 1004, Axling at 12; Ex. 1002, Greenspun ¶153.) Axling thus permits input without regard for the order in which elements are selected for the definition as claimed. (Ex. 1002, Greenspun ¶153.) Thus, claim 71 is obvious in view of Axling. (*Id.* at ¶154.)

**Ground 2. Dependent Claims 63 and 69 are obvious in view of**  
**Axling, Fohn, Skovgaard 1995 and Baker**

**A. A person skilled in the art at the time of the alleged invention would have combined the teachings of Axling, Fohn, Skovgaard 1995, and Baker**

As summarized above, the Axling and Fohn papers each disclose a software application for (1) “defining” or “maintaining” a product model in terms of its possible components and component “relationships,” and (2) “configuring” a valid product for sale to a customer based on the model. (Ex. 1002, Greenspun ¶155.) Fohn states that a database is implemented to record “any interrelationships among elements of a model.” (Ex. 1005, Fohn at 8; Ex. 1002, Greenspun ¶155.) These “interrelationships,” which Fohn describes as “constraints,” may be represented as “a database table or any data structure.” (*Id.*)

Fohn requires a database to maintain interrelationships among elements, but does not provide much detail about the manner in which the described “interrelationships” are represented in the database itself. (Ex. 1002, Greenspun ¶156.)

Skovgaard 1995 describes Beologic’s “salesPLUS” product configurator that, like the OBELICS and PC/CON configurators described in the Axling and Fohn papers, included a “maintenance” module in which constraints were input to define a product model, and a “configuration” module in which a sales person

configured a product based on the model. (Ex. 1010, Skovgaard 1995 at Sections 5.2 and 6; Ex. 1002, Greenspun ¶157.) This was the conventional architecture of the prior art configuration systems. (Ex. 1002, Greenspun ¶157.)

Skovgaard 1995 also describes an “inference engine” for managing the “constraint representation” of the product model itself. (Ex. 1010, Skovgaard 1995 at Section 4.1; Ex. 1002, Greenspun ¶158.) Skovgaard 1995 describes the inference engine as “a State of the Art constraint solver”:

It handles both Boolean algebra and finite domain arithmetic constraints based upon interval calculus and the patented Beologic *array inference technology*.

\* \* \*

The inference engine uses a *compiled version of the constraints* for efficient execution.

(Ex. 1010, Skovgaard 1995 at Section 4.1 and 7.1, emphasis added; Ex. 1002, Greenspun ¶158.)

A person of ordinary skill in the art at the time of the alleged invention reading Axling and Fohn would have taken into account Skovgaard’s “State of the Art” approach to encoding constraints in a product model as a “compiled” version of an “array.” (Ex. 1002, Greenspun ¶159.) As Skovgaard 1995 states, this “State of the Art” engine promotes “efficient execution” of the constraints in a configurator. (Ex. 1002, Greenspun ¶159.) This is an express and common sense

reason a person of ordinary skill in the art at the time of the alleged invention would have combined the teachings of Axling, Fohn and Skovgaard. (Ex. 1002, Greenspun ¶159.)

In addition, it was well known at the time of the alleged invention to encode information into bits in a “bit vector,” and perform bit-level comparisons to compare the encoded information. (Ex. 1002, Greenspun ¶160.) Baker, for example, states “bit vectors are extremely useful for computing everything from the Sieve of Eratosthenes for finding prime numbers, *to the representation of sets and relations*, to the implementation of natural language parsers . . .” (Ex. 1015, Baker at 2.8, emphasis added; Ex. 1002, Greenspun ¶160.) Baker states:

The manipulation of bit-vectors and bit-arrays is an important implementation technique of symbolic processing, because *bit-vectors allow for the efficient implementation of very complex and relatively unstructured relations*. Bit-vectors and bit-arrays therefore offer *an efficient method for representing and manipulating certain complex relationships . . . .*

(Ex. 1015, Baker at 2.8, emphasis added.)

A person of ordinary skill in the art at the time of the alleged invention would have naturally drawn upon the “efficient method for representing and manipulating certain complex relationships” disclosed in Baker to represent and process the element “sets and relationships” in the compiled constraint “array”

representing a product model as disclosed in Skovgaard 1995. (Ex. 1002, Greenspun ¶161.) Bit vectors and bit-arrays were well-known techniques at the time of the alleged invention for representing information at a very low-level of granularity. (*Id.*) As such, the use of bit vectors and bit arrays would have been a common sense and reliable option for a skilled artisan to represent product and part relationships in a product configurator. (*Id.*)

***[63.0] The method of claim 62 wherein said step of determining whether the elements specified in the left-hand side of said relationship are present in said configuration further comprises the step of performing a bit-level comparison of the bits corresponding to the elements in said left-hand side of the relationship with a "selected" bit vector.***

Dependent claim 63 depends from dependent claim 62 addressed above. Dependent claim 63 adds the limitation “performing a bit-level comparison of the bits corresponding to the elements in said left-hand side of the relationship with a ‘selected’ bit vector.” (Ex. 1002, Greenspun ¶162.) Neither claim 62 nor claims 60 and 61 from which it depends recite “bits corresponding to the elements.” This limitation thus lacks antecedent basis.

Axling and Fohn do not disclose representing elements and relationships as bit vectors, and comparing bit vectors, as recited in claim 63. (Ex. 1002, Greenspun ¶163.) As explained above, however, Baker states that “bit vectors are



extremely useful for computing everything from the Sieve of Eratosthenes for finding prime numbers, to *the representation of sets and relations*, to the implementation of natural language parsers . . .” (Ex. 1015, Baker at 2.8; Ex. 1002, Greenspun ¶163.)

In Skovgaard’s 1995 paper, he describes a “constraint solver” in salesPLUS located logically between the maintenance module for defining a product model in terms of constraints, and a configuration module for configuring a product based on the model. (Ex. 1002, Greenspun ¶164.)

The inference engine in salesPLUS is a State of the Art constraint solver. It handles both *boolean algebra* and finite domain arithmetic constraints based upon interval calculus and the patented Beologic *array inference technology*.

(Ex. 1010, Skovgaard 1995, Section 4.1, emphasis added.)

Skovgaard 1995 also states that “the inference engine uses a *compiled* version of the constraints for efficient execution.” (Ex. 1010, Skovgaard 1995 at Section 7.3, emphasis added.) These statements indicate to a person of skill in the art that the constraints defining the product model are represented internally in salesPLUS as bit arrays, and are processed as such during the configuration session. (Ex. 1002, Greenspun ¶165.)

The fact that the prior art salesPLUS software used bit-arrays and conventional logical operations to represent and process binary representations of

constraints within a product model confirms my opinion that a person of ordinary skill in the art at the time of the alleged invention would have considered it obvious to do so. (Ex. 1002, Greenspun ¶167.)

Nonetheless, it was well known at the time of the alleged invention to encode information into bits in a “bit vector,” and perform bit-level comparisons to compare the encoded information. (Ex. 1002, Greenspun ¶168.) Baker, for example, states “bit vectors are extremely useful for computing everything from the Sieve of Eratosthenes for finding prime numbers, *to the representation of sets and relations*, to the implementation of natural language parsers . . .” (Ex. 1015, Baker at 2.8, emphasis added.) Baker states:

The manipulation of bit-vectors and bit-arrays is an important implementation technique of symbolic processing, because *bit-vectors allow for the efficient implementation of very complex and relatively unstructured relations*. Bit-vectors and bit-arrays therefore offer *an efficient method for representing and manipulating certain complex relationships . . . .*

(Ex. 1015, Baker at 2.8, emphasis added)

In the context of claim 63, bit vectors are implemented to represent “sets and relations” – a purpose for using bit vectors disclosed in Baker. (Ex. 1002, Greenspun ¶169.)

Baker also discloses a “Chart of analogous operations for integers, bit-arrays

and bit sequences.” (Ex. 1015, Baker at 2.11; Ex. 1002, Greenspun ¶170.) The chart includes an operation called “Compare bit-vectors,” confirming that the comparison of bit vectors representing sets of information and relationships was well known to those of skill in the art at the time of the alleged invention. (Ex. 1002, Greenspun ¶170.)

It was also well-known at the time of the alleged invention that C++ could be used to implement “bit vectors” representing information, and performing “bit-level comparison” between bit vectors was well known at the time of the alleged invention. (Ex. 1002, Greenspun ¶171.)

For example, the C++ Library<sup>5</sup> includes the Bit\_Set Class, which is a generic class that “implements efficient bit sets” including “bit vectors”:

**Bit\_Set Class**

**8.5** The **Bit\_Set** class is publicly derived from the **Generic** class and implements efficient bit sets. These bits are stored in an arbitrary-length vector of bytes (unsigned **char**) large enough to represent the specified number of elements. A bit set is indexed by integer values: Zero represents the first bit, one the second, two the third, and so on with each integer value actually indicating the zero-relative bit position in the bit vector. Elements can be integers, enum values, constant symbols from the enumeration package, or any other type of object or expression that results in an integral value. All operations involving bit shifting are performed in byte increments, giving the most efficient operation on common hardware architectures.

(Ex. 1014, Texas Instruments at 8-6; Ex. 1002, Greenspun ¶172.)

It would have been obvious to a person of ordinary skill in the art to use common C++ operators (e.g., +, &, | and ^) to perform bitwise operations in the

---

<sup>5</sup> Ex. 1014 [Texas Instruments] is Texas Instruments C++ Object-Oriented Library User’s Manual, 1991.

context of the <bitstring> and /or <bitset> classes to compare the contents of two bit vectors. (Ex. 1002, Greenspun ¶173.) As disclosed in Dewhurst,<sup>6</sup> such bitwise operators are a staple of C++ programming. (*Id.*)

Based on the prior art disclosures described above, it would have been obvious to a person of skill in the art at the time of the alleged invention to utilize bit vectors to represent the left-hand and right-hand side of the relationships described in Axling and Fohn, and to perform bit-level comparisons to determine if selected bit vectors correspond to one side, or the other, of a particular relationship. (Ex. 1002, Greenspun ¶174.) This would have been one of a handful of ways known to those skilled in the art to represent the relationships and the rule elements, and as such would have been obvious to try. (Ex. 1002, Greenspun ¶174.) The obviousness of this approach is especially apparent in view of (1) Skovgaard's utilization of constraints compiled into an "array," and Boolean algebra "for efficient execution" of constraints (Ex. 1010, Skovgaard 1995 at 4.1 and 7.3), and (2) Baker's recognition that bit vectors are particularly well-suited for representing "sets and relations" (Ex. 1012, Baker at 2.8). (Ex. 1002, Greenspun ¶174.)

---

<sup>6</sup> Ex. 1012 [Dewhurst] is Dewhurst, Stephen C., Stark, Kathy T., Programming in C++, Prentice Hall Software Series, pp. 14-15 1989.

***[69.0] The method of claim 60 wherein said definition, each of said set of relationships, and said current configuration state are expressed as bit vectors having bits that correspond to elements.***

As explained above with respect to claim 63, Baker discloses that bit vectors are particularly well-suited for representing “sets and relations.” (Baker at 2.8; Ex. 1002, Greenspun ¶175.) Each limitation of this claim falls within the scope of “sets and relations.” (Ex. 1002, Greenspun ¶175.)

As also explained above, Skovgaard 1995 described compiling constraints into an “array” and performing Boolean algebra on them in the context of the product model definition and the product configuration supported by salesPLUS. (Ex. 1010, Skovgaard 1995 at 4.1 and 7.3; Ex. 1002, Greenspun ¶176.)

Based on these teachings, and the other prior art disclosures described above in connection with claim 63, it would have been considered obvious at the time of the alleged invention to express the product definition, relationships and configuration state described in Axling, Fohn and Skovgaard 1995 as bit vectors having bits as elements. (Ex. 1002, Greenspun ¶177.)

**Ground 3. Dependent Claim 72 is obvious in view of Axling, Fohn and Skovgaard**

**A. A person skilled in the art at the time of the alleged invention would have combined the teachings of Axling, Fohn, and Skovgaard 1995**

As set forth above, a person of ordinary skill in the art would have been motivated to combine the teachings of Axling, Fohn, and Skovgaard 1995 for all of the reasons set forth above with respect to Grounds for Challenge 1 and 2.

As explained in greater detail below, Skovgaard 1995 itself provides a common sense and practical reason for combining the teachings of that paper with Axling and Fohn. (Ex. 1002, Greenspun ¶179.)

***[72.0] The method of claim 60 wherein said configuration state identifies a set of selected elements and a set of selectable elements for said system, said input capable of selecting any member of said set of selectable elements and being made despite the order in which members of said set of selected elements were selected.***

The Saturn-based PC/CON application disclosed in Fohn does not impose an order on the configuration process. (Ex. 1002, Greenspun ¶180.) On the contrary, Fohn states:

Saturn allows users the flexibility to ***start anywhere in the problem space***. This means that a user can start by assigning values to any of the decision variables at any time. ***Value propagation ensures that the configuration will be automatically adjusted to support the***

*starting point chosen by a user.* A user can assign certain values to variables solely to view the propagation and the effect that the variable's value will have on a configuration.

(Ex. 1005, Fohn at 13, emphasis added; Ex. 1002, Greenspun ¶180.)

In Figure 6, selected and selectable elements are identified, and the user can select any one of them during the configuration process. (Ex. 1002, Greenspun ¶181.) Fohn discloses that changes can be made, and, “continuing in this way, more options can be added.” (Ex. 1005, Fohn at 16; Ex. 1002, Greenspun ¶181.)

Fohn thus discloses a configuration state including selected and selectable components, and selecting any member of the selectable components without regard to the order in which the selected components were selected as recited in claim 72. (Ex. 1002, Greenspun ¶182.)

Similar to Fohn, Skovgaard 1995 describes a key advantage of the salesPLUS product configurator as “free order of choices.” (Ex. 1010, Skovgaard 1995 – Abstract; Ex. 1002, Greenspun ¶183.) In the following statement, Skovgaard describes the drawbacks of “decision trees” used in prior configuration tools, and the benefits of “full freedom in the order of choices” provided by salesPLUS. (Ex. 1002, Greenspun ¶183.)

For decision trees the user is forced to make his choices in a predefined order. This reduces the problem by avoiding unordered information flow. This yields an application that is very constraining

for the user. Imagine that a customer wants a red car. The system will prompt for model, engine,.. etc. And when it comes to the colour it might not be possible to select red at all.

***salesPLUS gives full freedom in the order of choices.***

(Ex. 1010, Skovgaard 1995 – Section 5.1, emphasis added; Ex. 1002, Greenspun ¶183.)

This statement reveals the common sense reasons why a person of skill in the art at the time of the alleged invention would want to permit user input making element selections during a configuration session in Axling, Fohn and Skovgaard 1995 without regard for the order in which elements are selected as recited in claim 72. (Ex. 1002, Greenspun ¶184.)

Thus, claim 72 is obvious in view of Axling, Fohn and Skovgaard 1995. (Ex. 1002, Greenspun ¶185.)

## **VIII. Conclusion**

There is a reasonable likelihood that Petitioner will prevail on claims 60-72 of the '651 Patent. *Inter partes* review should be instituted for each of these identified claims.

## **IX. Fee Statement**

The filing fees associated with this Petition are being charged to Deposit Account 06-1510. The Board is authorized to charge any additional fees or credit any refunds pertaining to this Petition to Deposit Account 06-1510.



Respectfully submitted,

Dated: May 9, 2016

/John S. LeRoy/  
John S. LeRoy (Reg. No. 48,158)  
Thomas A. Lewry (Reg. No. 30,770)  
Frank A. Angileri (Reg. No. 36,733)  
John P. Rondini (Reg. No. 64,949)  
Christopher C. Smith (Reg. No. 59,669)  
Jonathan D. Nikkila (Reg. No. 74,694)  
**Brooks Kushman P.C.**  
1000 Town Center, 22<sup>nd</sup> Floor  
Southfield, MI 48075  
(248) 358-4400

*Attorneys for Petitioner*

**Certificate of Service**

The undersigned hereby certifies that the foregoing **PETITION FOR INTER PARTES REVIEW UNDER 35 U.S.C. § 311 ET SEQ. AND 37 C.F.R. § 42.100 ET SEQ. (CLAIMS 60-72 OF U.S. PATENT NO. 5,825,651)**, including all exhibits and supporting evidence, was served in its entirety on **May 9, 2016**, upon the following parties by overnight courier:

<b>Terrile, Cannatti, Chambers &amp; Holland, LLP P.O. Box 203518 Austin, TX 78720 512-797-6965</b>	<b>Terrile, Cannatti, Chambers &amp; Holland, LLP 11675 Jollyville Rd., #100 Austin, TX 78759 512-338-9100</b>
---	--

The undersigned also certifies that on the same date a courtesy copy was served in its entirety on the following parties via electronic service:

**Mr. Steven Mitby  
Ahmad Zavitsanos & Anaipakos  
1221 McKinney St.  
3460 One Houston Center  
Houston, TX 77010-2009  
713-655-1101  
[smitby@azalaw.com](mailto:smitby@azalaw.com)**

Respectfully submitted,

/John S. LeRoy/  
John S. LeRoy (Reg. No. 48,158)  
Thomas A. Lewry (Reg. No. 30,770)  
Frank A. Angileri (Reg. No. 36,733)  
John P. Rondini (Reg. No. 64,949)  
Christopher C. Smith (Reg. No. 59,669)  
Jonathan D. Nikkila (Reg. No. 74,694)  
**Brooks Kushman P.C.**  
1000 Town Center, 22<sup>nd</sup> Floor  
Southfield, MI 48075  
(248) 358-4400

*Attorneys for Petitioner*

**Certificate of Compliance Pursuant to 37 C.F.R. § 42.24**

This paper complies with the type-volume limitation of 37 C.F.R. § 42.24. The paper contains 12,193 words, excluding the parts of the paper exempted by §42.24(a).

This paper also complies with the typeface requirements of 37 C.F.R. § 42.6(a)(ii) and the type style requirements of § 42.6(a)(iii)&(iv).

Respectfully submitted,

Dated: May 9, 2016

/John S. LeRoy/  
John S. LeRoy (Reg. No. 48,158)  
Thomas A. Lewry (Reg. No. 30,770)  
Frank A. Angileri (Reg. No. 36,733)  
John P. Rondini (Reg. No. 64,949)  
Christopher C. Smith (Reg. No. 59,669)  
Jonathan D. Nikkila (Reg. No. 74,694)  
**Brooks Kushman P.C.**  
1000 Town Center, 22<sup>nd</sup> Floor  
Southfield, MI 48075  
(248) 358-4400

*Attorneys for Petitioner*