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.

Case No.: IPR2016-01014

_____

**DECLARATION OF DR. PHILIP GREENSPUN IN SUPPORT OF *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 ..... 3
I. Qualifications and Professional Experience ..... 7
II. Relevant Legal Standards ..... 13
III. Qualifications of One of Ordinary Skill in the Art ..... 14
IV. Overview of the '651 Patent ..... 15
V. Challenged Claims of the '651 Patent and Claim Construction ..... 20
VI. Overview of the Prior Art ..... 22
A. Axling's OBELICS Software - 1994 ..... 24
B. Fohn's PC/CON Software - 1995 ..... 28
C. Skovgaard's salesPLUS Software - 1995 ..... 34
VII. Grounds for Challenge ..... 36
A. 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 ..... 36
1. A Person Skilled In The Art At The Time Of The Alleged Invention Would Have Combined the Teachings of Axling and Fohn ..... 36
2. Axling and Fohn Disclose the Subject Matter of Claims 60-62, 64-68 and 70 and Render Each Claim Obvious ..... 46
B. Ground 2 – Dependent Claims 63 and 69 are Obvious in View of Axling, Fohn, Skovgaard 1995 and Baker ..... 73
1. 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 ..... 73
C. Ground 3 – Dependent Claim 72 is Obvious in View of Axling, Fohn and Skovgaard 1995 ..... 82
1. A Person Skilled In The Art At The Time Of The Alleged Invention Would Have Combined the Teachings of Axling, Fohn, and Skovgaard 1995 ..... 82
VIII. Conclusion ..... 85

## List of Exhibits

| Exhibit No. | Description | Date | Identifier |
|---|---|---|---|
| 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, A Tool For Developing Interactive Configuration Applications, 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, Configuring Computer Systems Through Constraint-Based Modeling and Interactive Constraint Satisfaction, Computers in Industry, Volume 27, Issue 1, Pages 3-21 | Sept. 1995 | Fohn |
| 1006 | 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 | Birmingham |
| 1007 | M. Stefik, Introduction to Knowledge Systems, 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 Expert System Including Arrangement for Acquiring Redesign Knowledge issued to Evangelos | Feb. 1, 1994 | Evangelos |

| Exhibit No. | Description | Date | Identifier |
|---|---|---|---|
| 1009 | H.J. Skovgaard, salesPLUS A Product Configuration Tool, *13th 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, A New Approach to Product Configuration*, Third International Conference icle*, *'95*, Concurrent Engineering &Technical Information Processing, January 30 – February 3, 1995 | 1995 | Skovgaard 1995 |
| 1011 | Patent Owner's Claim Construction Brief in *Trilogy Software, Inc. et al. v. Selectica, Inc.*, Case No. 2:04-cv-160, Dkt. #58 (E.D. Tex.) | July 1, 2005 | Patent Owner's Claim Construction Brief in *Selectica* Litigation |
| 1012 | S.C. Dewhurst, K.T. Stark, Programming in C++, 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, Efficient Implementation of Bit-Vector Operation in Common Lisp, ACM SIGPLAN Lisp Pointers 3, No. 2-4 1990 | 1990 | Baker |
| 1016 | B. Yu, H.J. Skovgaard, A Configuration Tool to Increase Product Competitiveness, 1998 | 1998 | Skovgaard 1998 |

| Exhibit No. | Description | Date | Identifier |
|---|---|---|---|
| 1017 | M. Stefik, Introduction to Knowledge Systems, Chapter 8, "Configuration," Morgan Kaufmann Publishers, Inc., San Francisco, CA (June 15, 1995) | June 15, 1995 | Stefik – Ch. 8 |
| 1018 | M.R. Wagner, Understanding the ICAD System, ICAD, Inc., 1990 | 1990 | ICAD |
| 1019 | Claim Construction Order in *Trilogy Software, Inc. v. Selectica, Inc.*, 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, A Knowledge-Based Configurator That Supports Sales, Engineering, and Manufacturing at AT&T Network Systems, *AI Magazine*, Volume 14, Number 3 (Fall 1993) | 1993 | Wright |

I, Philip Greenspun, hereby declare as follows:

1. I am making this declaration at the request of Ford Motor Company in the matter of *Inter Partes* Review of U.S. Patent No. 5,825,651 ("the '651 Patent") to Gupta *et al*.

2. I am a salaried non-owner employee of Fifth Chance Media LLC, which is being compensated for my work in this matter at a rate of $475/hour. My compensation in no way depends on the outcome of this proceeding.

3. In preparation of this declaration, I have studied the exhibits as listed in the Exhibit List shown above. Each of these exhibits is a true and accurate copy.

4. In forming the opinions expressed below, I have considered:

(a) The documents listed above as well as additional patents and documents referenced herein;

(b) The relevant legal standards, including the standard for obviousness provided in *KSR International Co. v. Teleflex, Inc*., 550 U.S. 398 (2007), and any additional legal standards set forth in the body of this declaration; and

(c) My knowledge and experience based upon my work and study in this area as described below.

## I. Qualifications and Professional Experience

5. I have provided my full background in the curriculum vitae that is attached as Exhibit 1003.

6. I earned a Ph.D. in Electrical Engineering and Computer Science from Massachusetts Institute of Technology in 1999. I also obtained a Bachelor of Science Degree in Mathematics from Massachusetts Institute of Technology in 1982 and a Master of Science Degree in Electrical Engineering and Computer Science from Massachusetts Institute of Technology in 1993.

7. My Ph.D. dissertation concerned the engineering of large online Internet communities with a Web browser front-end and a relational database management system (RDBMS) containing site content and user data.

8. I have authored five computer science textbooks in total, including Database Backed Websites (Macmillan), Software Engineering for Internet Applications, and an SQL language tutorial.

9. I have served as an independent member of various advisory and corporate boards, mostly for technology companies. For example, I joined the corporate board of an MIT materials science spin-off in late 2005 during a $550,000 seed capital phase. I stepped down when the company secured $10 million in venture capital in mid-2007.

10. I began working full-time as a computer programmer in 1978,

developing a database management system for the Pioneer Venus Orbiter at the National Aeronautics and Space Administration's Goddard Space Flight Center.

11. In the early 1980s I developed computer-aided design software for electronic systems, specifically to assist digital hardware engineers designing processors at Hewlett-Packard and Symbolics.

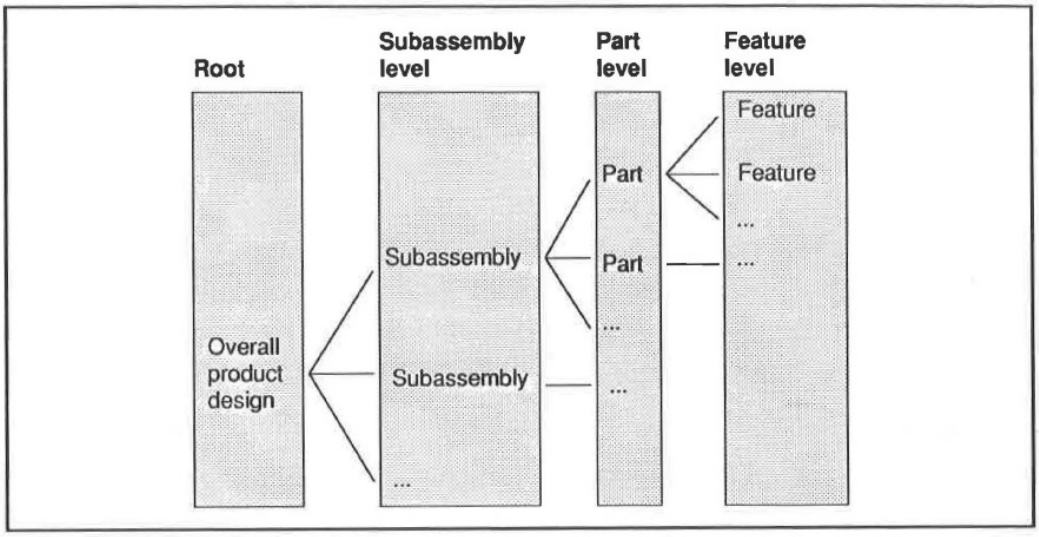
12. I co-developed a computer program for computer-aided design of mechanical systems in the mid-1980s. This was called the ICAD" System. The ICAD System enabled engineers to decompose a mechanical design into a hierarchy of subassemblies and establish configuration rules at each level of subassembly. The end-result was a system in which it was possible to go from customer specifications to a finished design without human intervention. The first applications for the ICAD System involved large structures built from steel, such as house-sized air-cooled heat-exchangers used in commercial buildings and industrial plants.

13. ICAD went public as "Concentra" in the 1990s and was acquired by Oracle Corporation in 2002. The product's mechanical design capabilities were deemphasized and its configuration capabilities were improved for use as a general-purpose sales configuration system. The product survives today as Oracle Configurator, part of the Oracle Applications suite of business software. "Understanding the ICAD System" is a 1990 marketing brochure that contains an

explanation of some of the basic capabilities. Excerpts from this brochure are reproduced below:

> A design instance of an ICAD product model is organized into a tree structure called the *product structure tree*.
>
> A tree organization naturally reflects the way that design and manufacturing engineers think about product design. It provides an unambiguous way of addressing every rule in the ICAD product model, which allows complex relationships to be set up between rules (see the section "Rule Dependencies in the ICAD Product Model", page 4-45). Tree organizations also provide a natural modularity to the product design. Complex assemblies are divided into subassemblies. Each subassembly can be designed separately, and each is further divided into components that can be designed separately.
>
> A product structure tree links the components of a product design together. The *root* of the product structure tree is a design instance of the defpart that represents the entire product structure. The root branches out into more detailed levels which are called *children* or *parts* of the root. For example, a branch might represent a subassembly of the product assembly. Each child can have additional children. The terminal components of the tree are called *leaves*; these are parts that have no children.
>
> It can be helpful to think of the different levels of the product structure tree as roughly representing a number of conceptual levels in a mechanical assembly. The root represents the entire design instance, which is divided into the design of the part. This overall design decomposes into subassemblies. Subassemblies are typically built from component parts (e.g., screws, holes, etc). Parts are often divided into their component design features. The lowest level (leaves) of the product structure tree usually represents the actual geometric structure of the mechanical part (see the section "Modeling with Simple Geometric Primitives", page 4-56).

This is shown in the following diagram. Note that product structure tree of your ICAD product model may not follow this diagram exactly. Sometimes there are many levels of subassemblies before the component part level, and in some cases component parts are not divided into design features.




*Different levels in a product structure tree.*

(Ex. 1018 [ICAD] at 4-29 – 4-31, pages 80-82.)

14. I developed my first program using a relational database management system in 1994. It was a Web interface to the Children's Hospital Oracle RDBMS, Version 6. This application enabled doctors at the hospital to view patient clinical data using any computer equipped with a Web browser.

15. In 1995, I led an effort by Hearst Corporation to set up an infrastructure for Internet applications across all of their newspaper, magazine, radio, and television properties. This infrastructure included software for managing users, shopping carts, electronic commerce, advertising, and user

tracking. One capability of the system was using private data, regarding a user's history, to determine the (publicly available) advertisements to be shown on pages viewed by that user, including pages that included order summaries and other private data.

16. Between 1995 and 1997, I significantly expanded the photo.net online community that I had started in 1993 to help people teach each other to become better photographers. I began distributing the source code behind photo.net to other programmers as a free open-source toolkit, called "ArsDigita Community System." One version of this system was an add-on to AOLserver, a Web server with an API.

17. In May 1997, Macmillan published my first textbook on Internet Application development, Database Backed Websites. A September 1998 update to this book was published as Philip and Alex's Guide to Web Publishing (hardcopy version published in April 1999).

18. In 1997, I started a company, ArsDigita, to provide support and service for the ArsDigita Community System. Between 1997 and the middle of 2000, I managed the growth of ArsDigita to 80 people, almost all programmers, and $20 million per year in annual revenue. This involved supervising dozens of software development projects, nearly all of which were Internet Applications with a Web front-end and an Oracle RDBMS back-end.

19. In 1999, I supervised the packaging up of much of our ecommerce-related code into the "ecommerce" module of the ArsDigita Community System. As the founder, CEO, and chief technical employee of the company, I personally developed functional specifications, SQL data models (Structured Query Language, or "SQL," the standard programming language for relational database management systems), and Web page flows that determined the user experience.

20. Between 2000 and the present, I have managed software development projects for philip.greenspun.com and photo.net. Both online services are implemented as relational database management applications. For photo.net, in particular, I evaluated various Web-based comparative shopping tools that would allow readers to find the best delivered prices, given a zip code, for camera equipment. In addition, I am currently developing a Facebook application that allows parents to create electronic baby books.

21. Separately from this commercial and public work, I have been involved as a part-time teacher within the MIT Department of Electrical Engineering and Computer Science, educating students in how to develop Internet Applications with an RDBMS back-end. In the spring of 1999, I taught 6.916, Software Engineering of Innovative Web Services, with Professors Hal Abelson and Michael Dertouzos. In the spring of 2002, this course was adopted into the standard MIT curriculum as 6.171. I wrote 15 chapters of a new textbook for this

class, Software Engineering for Internet Applications. This book was published on the Web at http://philip.greenspun.com/seia/ starting in 2002 and 2003 and also in hardcopy from MIT Press in 2006. I am the sole author of a supplementary textbook for the class, SQL for Web Nerds, a succinct SQL programming language tutorial available only on the Web at http://philip.greenspun.com/sql/. I use this book when I teach an intensive course in database programming at MIT, as I did most recently in January 2015.

## II. Relevant Legal Standards

22. I have been asked to provide opinions regarding the validity of claims of the ’651 Patent in light of several prior art publications.

23. It is my understanding that a claimed invention is unpatentable under 35 USC § 102 if a prior art reference teaches every element of the claim. This is sometimes referred to as “anticipation.”

24. It is my understanding that a claimed invention is unpatentable under 35 U.S.C. § 103 if the differences between the invention and the prior art are such that the subject matter as a whole would have been obvious at the time the alleged invention was made to a person having ordinary skill in the art to which the subject matter pertains. This is sometimes described as “obviousness.” I understand that an obviousness analysis takes into account the level of ordinary skill in the art, the scope and content of the prior art, and the differences between the prior art and the

claimed subject matter.

25. It is my understanding that the Supreme Court, in *KSR Int'l Co. v. Teleflex Inc*., 550 U.S. 398 (2007) and other cases, has recognized several rationales for combining references or modifying a reference to show obviousness of the claimed subject matter. Some of these rationales include the following: combining prior art elements according to known methods to yield predictable results; simple substitution of one known element for another to obtain predictable results; a predictable use of prior art elements according to their established functions; applying a known technique to a known device to yield predictable results; choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success; and some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention.

## III. Qualifications of One of Ordinary Skill in the Art

26. I have reviewed the '651 Patent, as well as the pertinent prior art documents discussed below. Based on this review and my knowledge of the configuration system field, including my work on ICAD system in the 1980's, it is my opinion that a person having ordinary skill in the art 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. The relevant field of art is product configuration software.

## IV. Overview of the '651 Patent

27. The '651 Patent is directed to using a graphical user interface to: (1) "define" a product based on its constituent parts and part relationships, and (2) "configure" a valid product using the product definition.

28. The Background of the Invention section of the '651 Patent describes certain prior art approaches to defining and configuring products. (Ex. 1001 at 1:12-60.)

29. 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.

(Ex. 1001 at 12-25.)

30. The Background also states that prior art "computer systems" were available for defining and configuring systems. (Ex. 1001 at 1:26-27.) The '651 Patent identifies two drawbacks associated with the prior art computer systems. (Ex. 1001 at 1:27-60.) First, the prior art systems required "a configuration language to define a system" which "limits the number 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.*)

31. The '651 Patent is correct that the pioneering work (discussed below) in computer-assisted configuration, e.g., from the late 1970s through the mid-1980s, relied on text-based configuration languages. As explained below, however, prior art configuration software products such as OBELICS, PC/CON and salesPLUS permitted the definition and configuration of a product using a graphical user interface, as opposed to a "configuration language." These interfaces were provided in the prior art software for the very purpose of simplifying the user's experience defining and configuring products. These prior

art systems were known within the product configuration software industry, and were published in the literature, but they were not described in the Background of the Invention or considered during examination of the '651 Patent.

32. The '651 Patent also states that the prior art computer systems suffered from the drawback of "impos[ing] a flow or ordering to the user operations" that might lead to frustration by a "novice user." (Ex. 1001 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*.)

33. As explained below, however, prior art configuration software products such as PC/CON and salesPLUS were available that, by design, did not impose a particular flow on the manner in which a product is configured. Here again, the prior art had already solved a problem identified in the '651 Patent years later. These prior art systems were not described in the Background of the Invention or considered during examination of the '651 Patent.

34. In the Summary of the Invention, the '651 Patent describes a two-part 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." (Ex. 1001 at 2:38-60.) These two systems are shown in Figure 2. As explained below, each of the prior art products

(OBELICS, PC/CON, and salesPLUS) included what the '651 Patent describes as a maintenance system and a configuration system.

35. The "maintenance system" described in the '651 Patent includes a graphical user interface to define a product, including relationships between products and parts, and relationships between the parts themselves. (Ex. 1001 at 2:38-49.) Figure 6 shows an example graphical user interface for defining a product. (Ex. 1001 at 3:65-67, 8:64-67.)

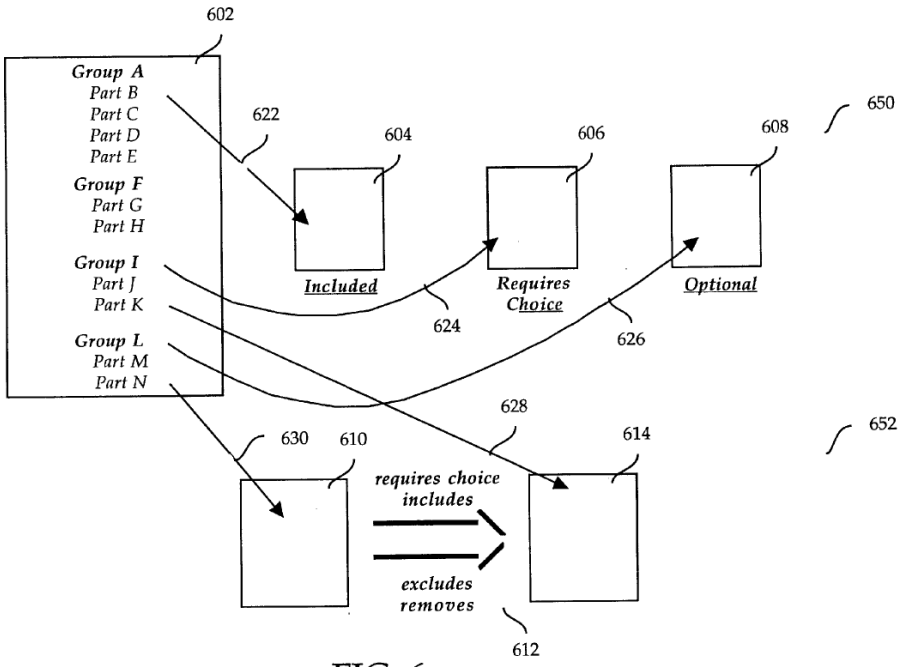



FIG. 6

(Ex. 1001, Fig. 6.)

36. 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

between parts themselves in section 652. (Ex. 1001 at 8:5-11.) 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.

37. 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." (Ex. 1001 at 3:1-9, 9:10-16.) The internal representation is used by the computer during a configuration session to validate a user's input against the product definition. (Ex. 1001 at 3:9-11, 9:30-32.) As explained below, the prior art salesPLUS product used a "compiler" for the very same purpose.

38. Figure 8B of the '651 Patent provides an example of an internal representation of relationships between products and parts defined using the GUI shown in Figure 6. In the preferred embodiment, the internal representation includes a table having a "left hand side" and a "right hand side." (Ex. 1001 at 11:1-10.) Separate tables may represent each relationship type, *e.g.*, "includes," "excludes," "removes," "requires choice," etc. (Ex. 1001 at 11:10-41.) The tables may contain bit vectors representing products and parts on the left hand side and right hand side of the relationships corresponding to the different tables. *Id.*

39. The "configuration system" described in the '651 Patent is "used to configure a system using a definition created by the maintenance system" described above. (Ex. 1001 at 2:50-51, 7:66-8:1, 11:65-12:11.) The configuration system validates user input against the product definition, and permits the user to "select and unselect parts in any order." (Ex. 1001 at 2:53-54.) Although Figure 6 is introduced as a GUI for defining a product as part of the maintenance system, the '651 Patent explains that a "configuration user" would also use a GUI similar to of Figure 6 to configure a product (based on the internal representation) after it has been defined. (Ex. 1001 at 8:64-65.)

## V. Challenged Claims of the '651 Patent and Claim Construction

40. I have been asked to review claims 60-72 of the '651 Patent.

41. I understand that in an *inter partes* review at the Patent Office, claims are to be given their broadest reasonable interpretation in light of the specification as would be read by a person of ordinary skill in the relevant art.

42. I also understand that certain terms of the challenged claims were disputed in an earlier lawsuit (not involving Petitioner) involving the '651 Patent.[1]

[1] Ex. 1011 [Patent Owner's CC Brief] is a true and accurate copy of: *Trilogy Software, Inc. v. Selectica, Inc.*, Case No. 2:04-cv-160, Docket No. 58 (E.D.Tex. 7/1/05.). Ex. 1019 [CC Order] is a true and accurate copy of: *Trilogy Software,*

I have been provided with the table below including the Patent Owner's proposed constructions, and the Court's constructions, for various terms recited in the challenged claims.

| Claim Term | Patent Owner's Proposed Construction | District Court Construction |
|---|---|---|
| "product relationship" | "A classification of the specific association that exists between a product and one or more parts." | "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." |
| "part relationship" | "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." | "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." |
| "configuration state" | "The status of the elements in the current | "The status of the elements in the current |

*Inc. v. Selectica, Inc.*, Case No. 2:04-cv-160, Claim Construction Order (E.D.Tex. 12/20/05.)

| Claim Term | Patent Owner's Proposed Construction | District Court Construction |
|---|---|---|
| | configuration." | configuration." |
| "active relationship" | "A relationship that is presently in effect." | "A relationship in which all elements on the left-hand side of the relationship are selected." |
| "notActivatable relationship" | A relationship that cannot be put into effect. | "A relationship in which the selection of certain left-hand side items results in an invalid configuration state." |

43. In applying the claims at issue to the prior art, I have given all of the claim terms their broadest reasonable interpretation in light of the specification, as would be commonly understood by those of ordinary skill in the art at the time the patent was filed. In this analysis, I have also taken the Patent Owner's proposed constructions and the Court's constructions into account where appropriate.

## VI. Overview of the Prior Art

44. More than a decade before the application for the '651 Patent was filed, the Artificial Intelligence community had developed and commercially deployed comprehensive and sophisticated product configuration applications.

45. One of the earliest such configurators, referred to as a prototype as "R1" and commercially as "XCON," was developed by Digital Equipment Corporation together with John McDermott, a Computer Science professor at

Carnegie-Mellon University, starting in 1978 and launched commercially no later than 1981. (Ex. 1020, Wright at 70; Ex. 1005 Fohn at 4 and 6.) R1/XCON used production rules to represent knowledge about configuring DEC's computer systems. (*Id*.)

46. "Pride" and "Cossack" were other configurators used commercially in the 1980s. (Ex. 1005, Fohn at 6.) These were frame-based expert systems. (*Id*.) Pride was built to support the design of paper handling systems inside copiers, and Cossack was adapted from Pride to handle the configuration of computers. (*Id*.)

47. MICON, developed in 1984 at Carnegie-Mellon University was an application written in the OPS-5 production system language for configuring microprocessors. (Ex. 1006, Birmingham at 565.)

48. Literature addressing configuration software published prior to the filing date of the '651 Patent often describe these pioneering systems. For example, these early configuration systems are surveyed in the key references (Axling, Fohn and Skovgaard) that I rely upon in my detailed analysis below.

49. The '651 Patent attempts to improve upon the usability of sophisticated product configurators that were already well known in the industry. The Background of the Invention of the '651 Patent alleges that prior art configuration software 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, '651 Patent, 1:26-38; 2:38-49.) The '651 Patent also alleges that prior art 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." (Ex. 1001, '651 Patent, 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.)

50. As explained in detail below, however, 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 claimed and described in the '651 Patent, before the application for the '651 Patent was filed. I summarize each prior art publication below.

**A. Axling's OBELICS Software - 1994[2]**

51. In 1994, in the Journal of Logic Programming, Tomas Axling and Seif Haridi referenced some of the more significant historic/academic configuration applications, including "XCON," "Cossack" and "MICON."

52. Axling described 1980s-style product configuration software as "hard

[2] Ex. 1004 [Axling] is a true and accurate copy of: T. Axling, S. Haridi, A Tool For Developing Interactive Configuration Applications, The Journal of Logic Programming, Volume 19, 658-679 (1994).

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." (Ex. 1004, Axling at 2.)

53. To address these and other drawbacks of the product configuration systems known to the industry at that time, Axling introduced a concept described as "Interactive Configuration by Selection" or "ICS." (Ex. 1004, Axling at 2.) Axling described ICS as "a less complex and more interactive approach to configuration." (Ex. 1004, Axling at 2.) Axling provided various "Design Rationales," for an ICS system, including replacing the "***unstructured collections of rules***" associated with earlier systems with "***a graphical user interface making the domain model easy to understand and work with***." (Ex. 1004, Axling at 8, emphasis added.)

54. Axling developed OBELICS, a software program for modelling, configuring, and validating a product. (Ex. 1004, Axling at 9-15.) OBELICS uses a "Domain Model Editor" to graphically define products and relationships among parts of the product. (Ex. 1004, Axling 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:

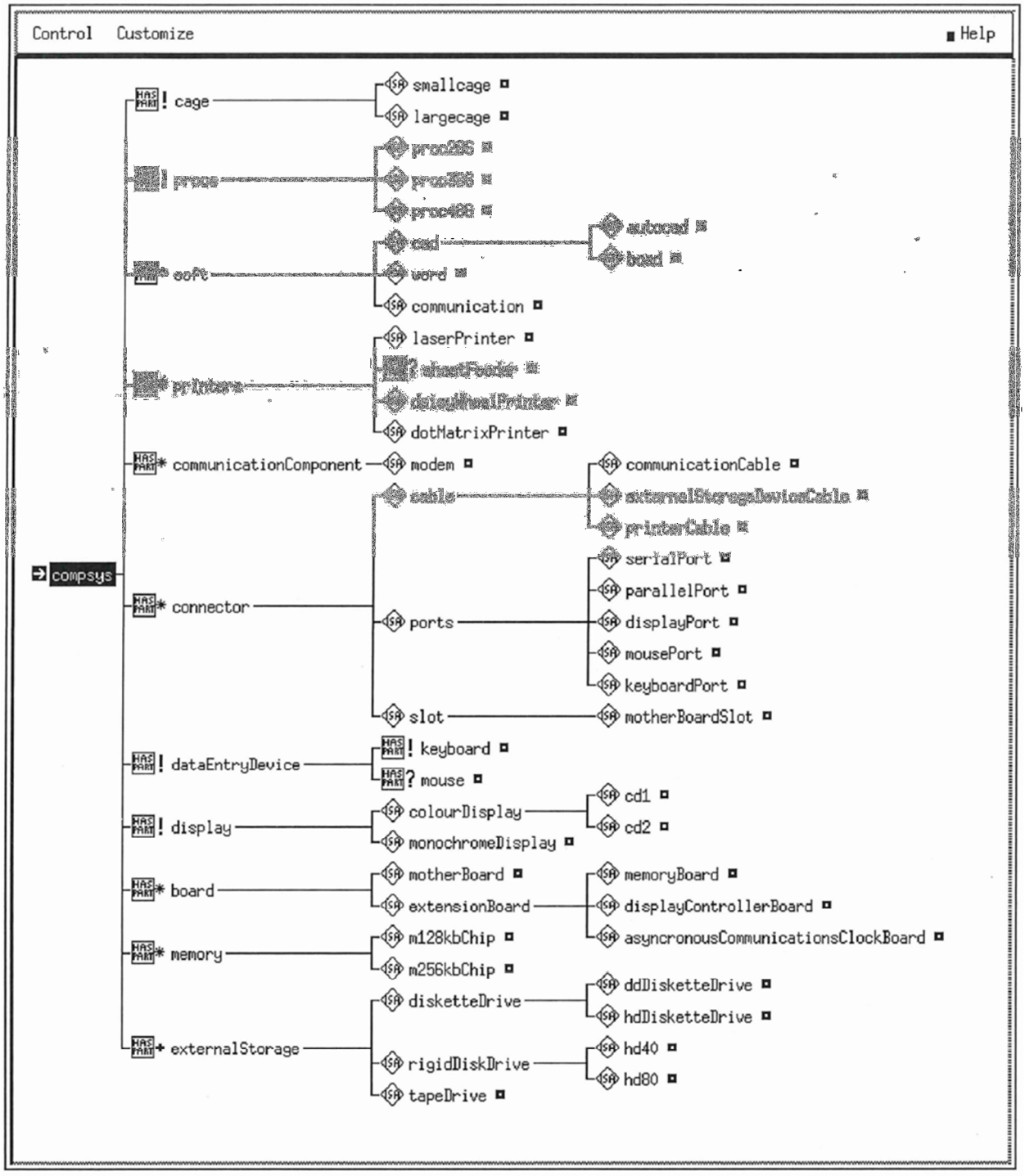



FIGURE 5.2. The Domain Model Editor

55. Relationships between the computer system "compsys" and its parts, and relationships among the parts themselves, are depicted in the graphical model

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.*

(Ex. 1004, Axling at 14.)

56. OBELICS converts the graphical representation of the product definition and part relationships created using the Domain Model Editor into SICStuc Prolog Objects. (Ex. 1004, Axling, 15-17.) SICStuc Prolog Objects are an extension of SICStus Prolog with object oriented programming. (Ex. 1004, Axling, 10.)

57. Axling also discloses an enhanced configuration experience, after the product model has been defined. Axling describes an ICS in which "the current state of the configuration" is presented during a configuration session, and the system "allow[s] the user to alter previously made configuration decisions." (Ex. 1004, Axling at 5.) The ICS system "presents all valid alternatives and also makes it easier to alter previous choices." (Ex. 1004, Axling at 6.) In OBELICS, "the case model is available to a user during a configuration session allowing the user to view the current state and alter previous decision." (Ex. 1004, Axling at 9.)

58. The "User Interface" or "GUI" described in Axling for configuring a product includes two parts: 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.)

59. Axling, and the OBELICS software that Axling developed, were not considered during examination of the '651 Patent.

**B. Fohn's PC/CON Software - 1995[3]**

60. In 1993, developers at IBM (Steffen Fohn) and North Carolina State University developed the "PC/CON" product configurator. PC/CON is an acronym for ***p***ersonal ***c***omputer ***con***figurator application.

61. Fohn published a description of the PC/CON software in 1995. Fohn's paper, like Axling's paper, discloses software for modeling a product that is later configured (based on the model) for sale to a customer.

62. Fohn discloses a "spreadsheet-like development interface that, in conjunction with a database manager, facilitates the construction of a constraint-

[3] Ex. 1005 [Fohn] is a true and accurate copy of: S.M. Fohn, J.S. Liau, 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).

based model representing a product line." (Ex. 1005, Fohn at 4.) Fohn provides an example in the context of modelling a computer system, but states that the software is "applicable to other industries where there is a wide product variety." (Ex. 1005, Fohn at 1.)

63. Fohn recognized that "one of the most important issues in computer product configuration is the maintainability and expandability of existing configuration information models." (Ex. 1005, Fohn 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***." (Ex. 1005, Fohn at 6, emphasis added.)

64. To address the limitations of earlier configuration systems, Fohn disclosed "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***." (Ex. 1005, Fohn at 6, 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***." (Ex. 1005, Fohn

at 9, emphasis added.)

65. The PC/CON application is built upon the Saturn "constraint system shell," illustrated in Figure 1, together with a relational database. (Ex. 1005, Fohn at 8.) Fohn defines a "constraint" 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.) The relational database maintains the product "model," including the "specific interrelationships (constraints) between the elements of the information model." (Ex. 1005, Fohn at 11.)

66. 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***." (Ex. 1005, Fohn 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.

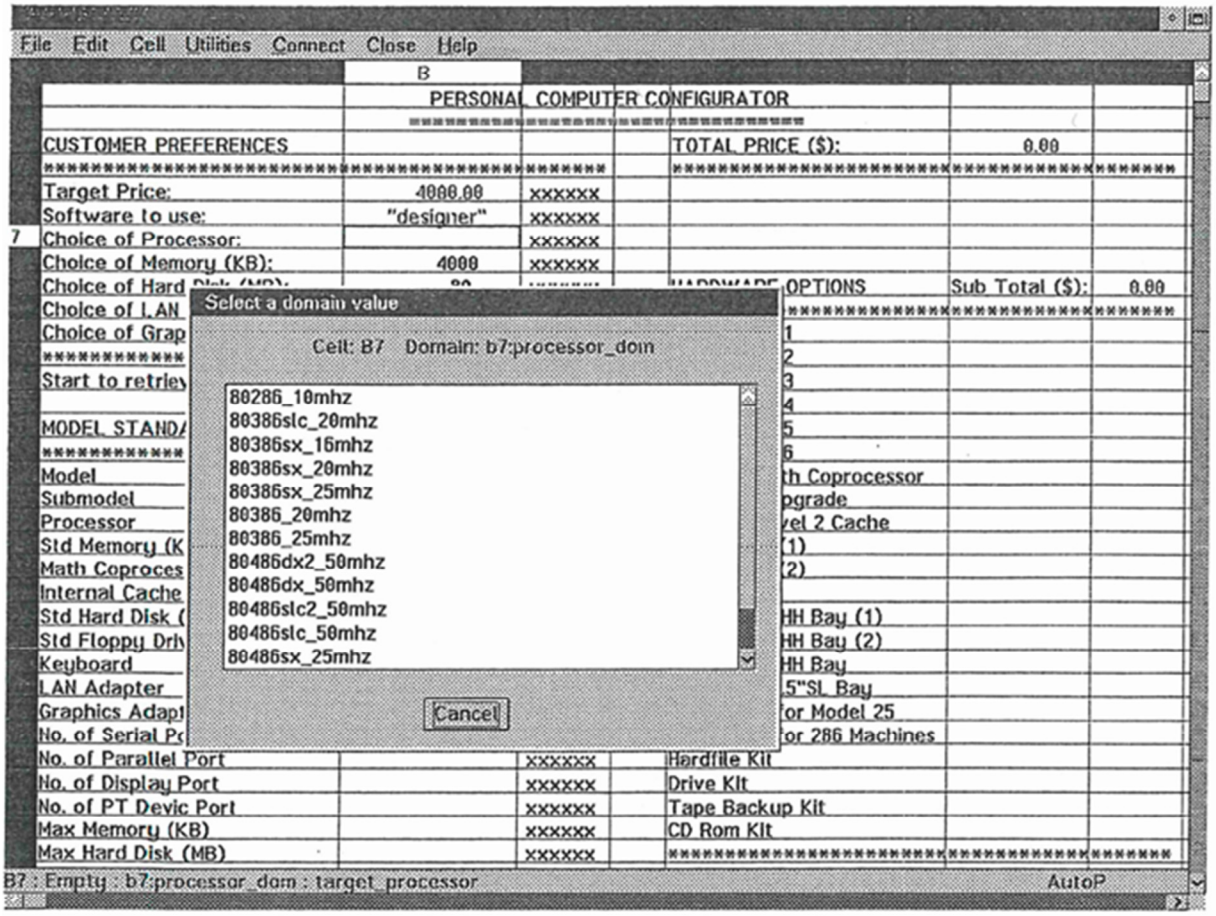



Fig. 5. Selection of a customer's preferred microprocessor.

(Ex. 1005, Fohn at 13.)

67. Fohn describes a wide variety of functions that are enabled by the constraint model implemented in the PC/CON application:

- Ensure that selected hardware requirements can support the selected software requirements.
- Check each component's pre-and co-requisites to identify their interdependencies and conflicts.
- Identify and infer missing information.

- Identify recommended and approved options.
- Based on customer preferences of hardware functionality and/ or software functionality, retrieve a basic submodel from the product database.
- Retrieve standard features of a basic submodel from the product database.
- Make all non-available options inaccessible.
- Limit all the user options to avoid error input.
- Build up a customized system configuration based on user options.
- Infer required components based on user options.
- Generate and update pricing information as a configuration evolves.
- Check the compatibility between components of the system.
- Check the compatibility between software applications and the configured system.
- Provide advice to resolve conflicts when an incompatible configuration exits.
- Ensure the customized configuration is a valid configuration.

(Ex. 1005, Fohn at 11-12.)

68. Fohn provides a detailed example of a configuration session in which user input is received, validated against the product model and the current state of

the current configuration, and the impact of user selections is propagated throughout the configuration based on applicable constraints within the model. (Ex. 1005, Fohn at 15-16.)

69. The configuration engine described in Fohn alerts a user when an invalid selection has been made that violates a defined constraint in the product model. As shown below, the software identifies the violation, and provides a recommendation to the user for resolving the conflict to arrive at a valid system configuration.

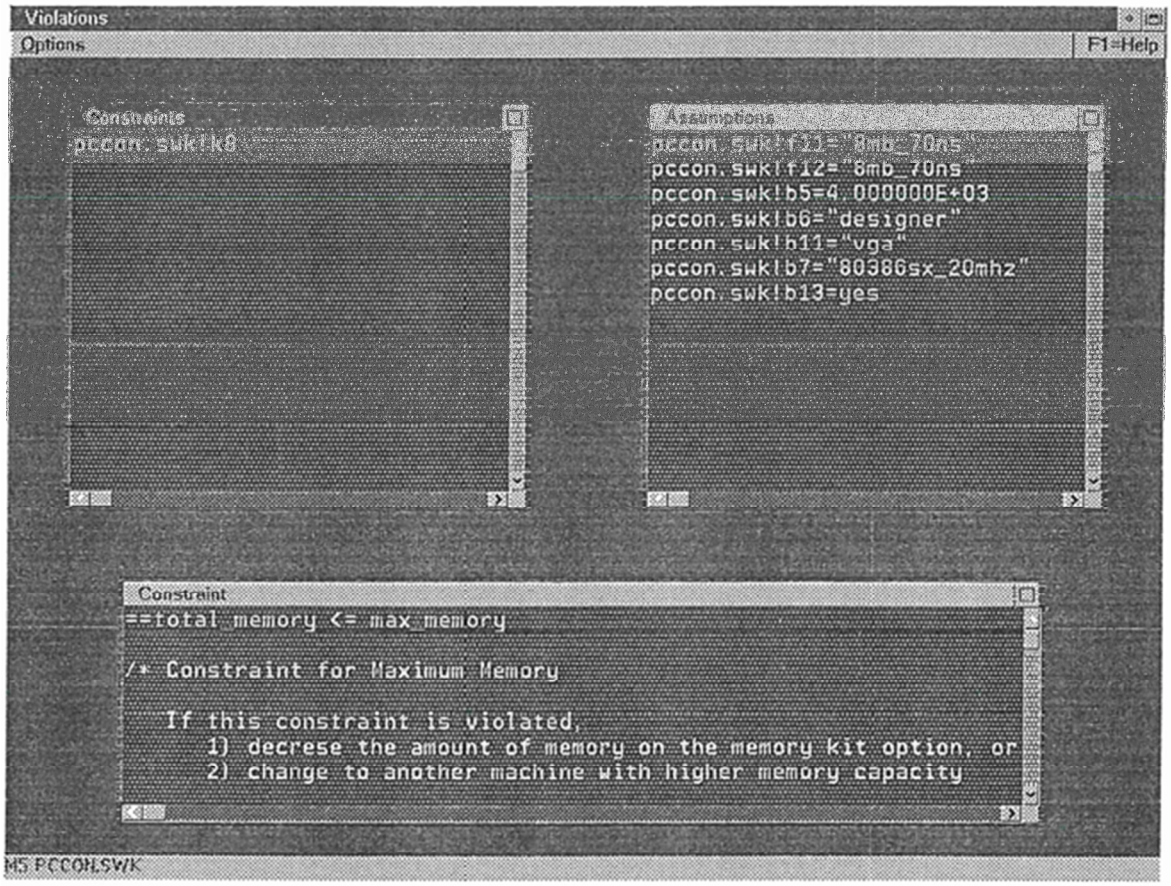



Fig. 8. Window displaying assistance to help resolve constraint violations.

(Ex. 1005, Fohn at 16.)

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

### C. Skovgaard's salesPLUS Software - 1995[4]

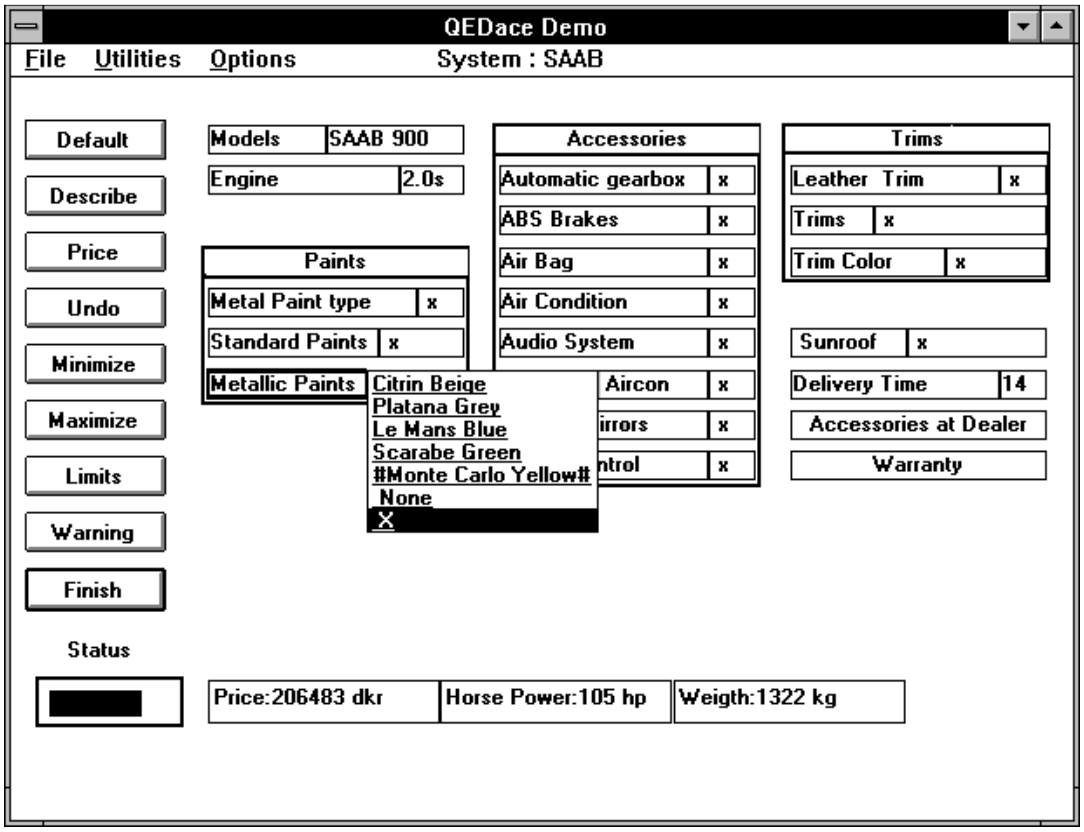
71. Skovgaard 1995 describes Beologic's "salesPLUS" product configurator, and an example in which "a car configuration is presented." (Ex. 1010, Skovgaard 1995, Abstract.)

72. Like OBELICS and PC/CON, the 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. (Ex. 1010, Skovgaard 1995 at Sections 5.2 and 6.)

73. And like Axling and Fohn, Skovgaard 1995 provides a configuration example. In this example, Skovgaard configures a 1992 Saab 900. (Ex. 1010, Skovgaard 1995 at Section 7.) Skovgaard provides an example of the

[4] Ex. 1010 [Skovgaard 1995] is a true and accurate copy of: Skovgaard, A New Approach to Product Configuration, ICLE, 3rd Int'l Conf. on Concurrent Engineering & Technical Information Processing, Feb. 3, 1995. Ex. 1009 [Skovgaard 1996] is a later version of this publication. Ex. 1009 [Skovgaard 1996] is a true and accurate copy of: Skovgaard, salesPLUS A Product Configuration Tool, 13th National Conference on Artificial Intelligence (AAAI-96), Workshop on Configuration, No AAAI FS-96-03, Portland, Oregon, 61-68, August 4-8, 1996.

configuration interface for this example, described as the "Runtime Sales Tool," reproduced below.




(Ex. 1010, Skovgaard 1995 at Section 7.3.)

74. One of the benefits of salesPLUS is what Skovgaard describes as "free order of choices." (Ex. 1010, Skovgaard 1995 at Abstract.) 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 his salesPLUS software:

> 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 at Section 5.1, emphasis added.)

75. Thus, salesPLUS described in Skovgaard 1995 expressly recognized and solved the problem described in the Background section of the '651 Patent concerning the "flow or ordering to the user operations." (Ex. 1001, '651 Patent, 1:38-48.)

## VII. Grounds for Challenge

### A. 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

#### 1. A Person Skilled In The Art At The Time Of The Alleged Invention Would Have Combined the Teachings of Axling and Fohn

76. 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.

77. While the Axling and Fohn papers each discuss all aspects of defining and configuring systems, the emphasis of the Axling and Fohn papers is somewhat different. Axling provides significant detail, illustration and examples concerning

the product definition and maintenance aspects of the software, whereas Fohn provides more detail and interface examples on the configuration and validation aspects. The two references complement one another and, taken together, comfortably render obvious the subject matter of the challenged claims.

78. A person of skill in the art at the time of the alleged invention would have had several reasons to draw from the Axling and Fohn papers in constructing a configuration application. 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. Axling states that the "graphical user interface mak[es] the domain model easy to understand and work with." (Ex. 1004, Axling at 8.)

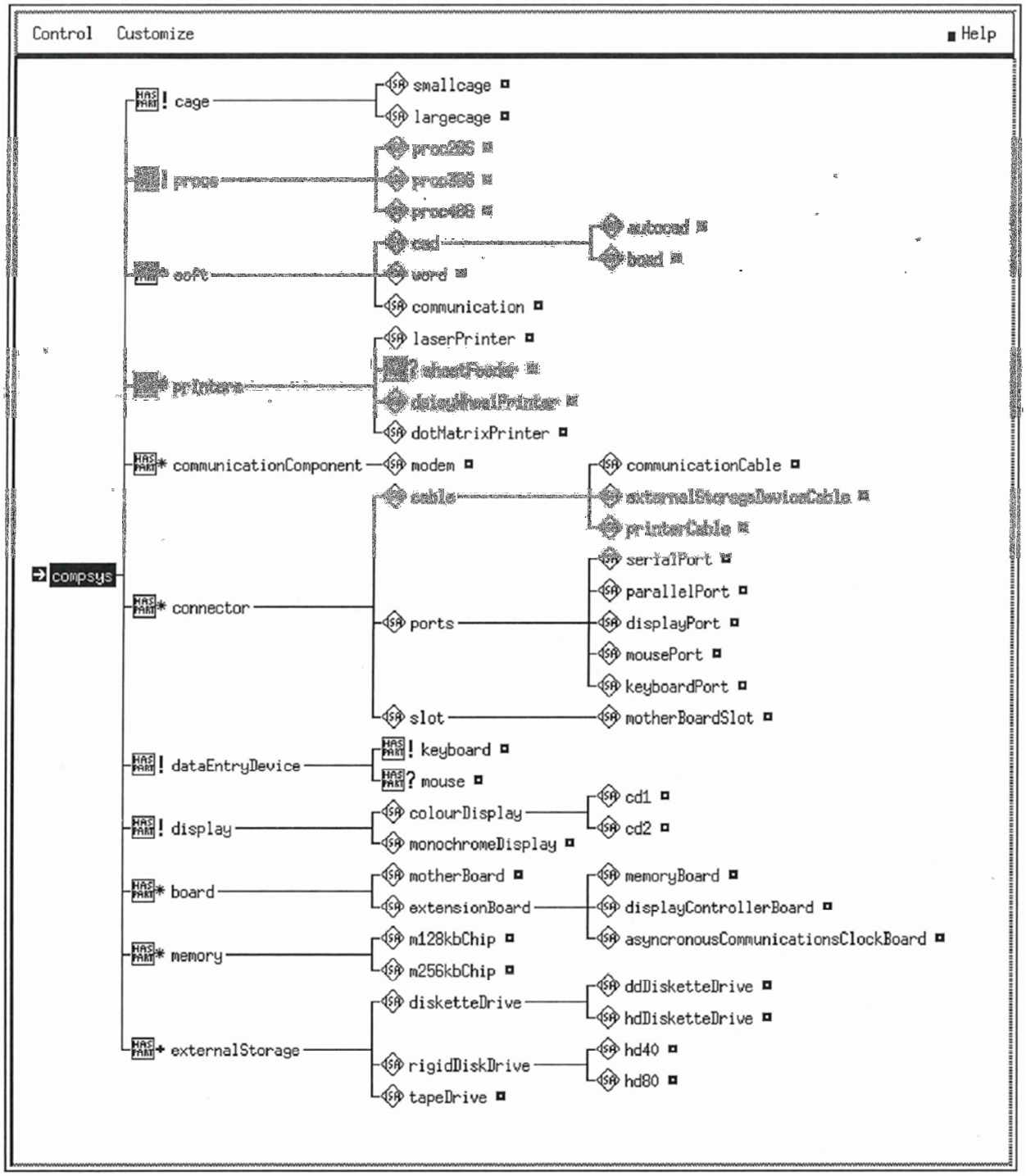



FIGURE 5.2. The Domain Model Editor

(Ex. 1004, Axling at 13.)

79. The interfaces disclosed in Fohn focus on the product configuration,

which takes place after the product model is defined. 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 relates practical efforts by International Business Machines, then and now the world's leading vendor of information technology for business. "How did IBM do it?" is a common question for a person of ordinary skill to ask.

80. In Figures 4-8, Fohn provides a series of example graphical user interfaces for interactively configuring and validating a product based on the product definition and component relationships. A representative example is provided below.

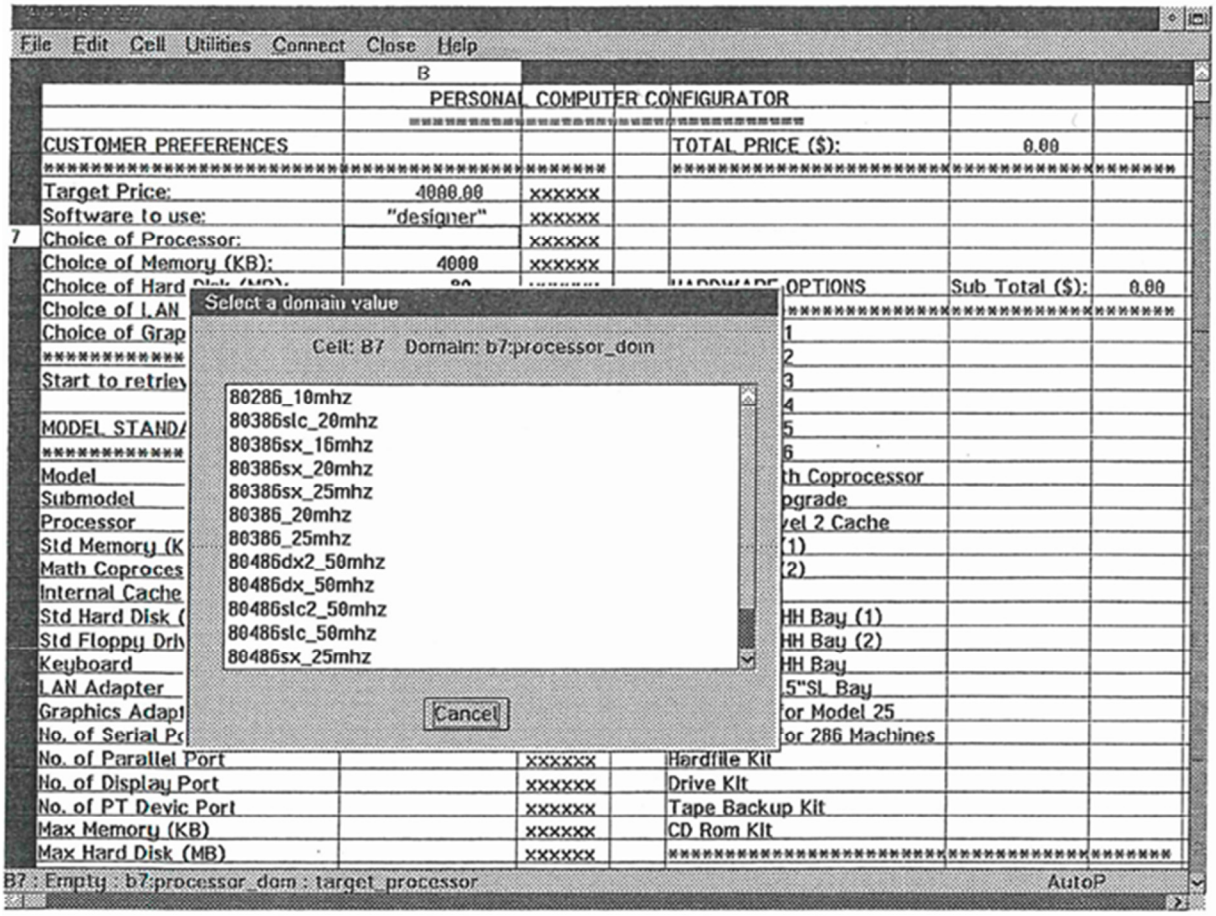



Fig. 5. Selection of a customer's preferred microprocessor.

(Ex. 1005, Fohn at 13.)

81. Axling and Fohn 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.

82. While each reference discusses all elements of a complete product definition and configuration session, the OBELICS product definition examples

provided in Axling would enhance the PC/CON system described in Fohn. Similarly, the PC/COM product configuration and validation examples provided in Fohn would enhance the OBELICS system disclosed in Axling.

83. 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. 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. 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.

84. 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. Axling recognizes the obvious objective of concluding a configuration session with a valid configuration that satisfies all applicable constraints. 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 provide an interface to intuitively take a user through the configuration validation process.

85. Fohn addresses this objective of Axling, providing a "violation window" (reproduced below) for conveniently identifying invalidity conditions in the current configuration, and proposing recommendations for resolving them. (Ex. 1005, Fohn at 15-16.)

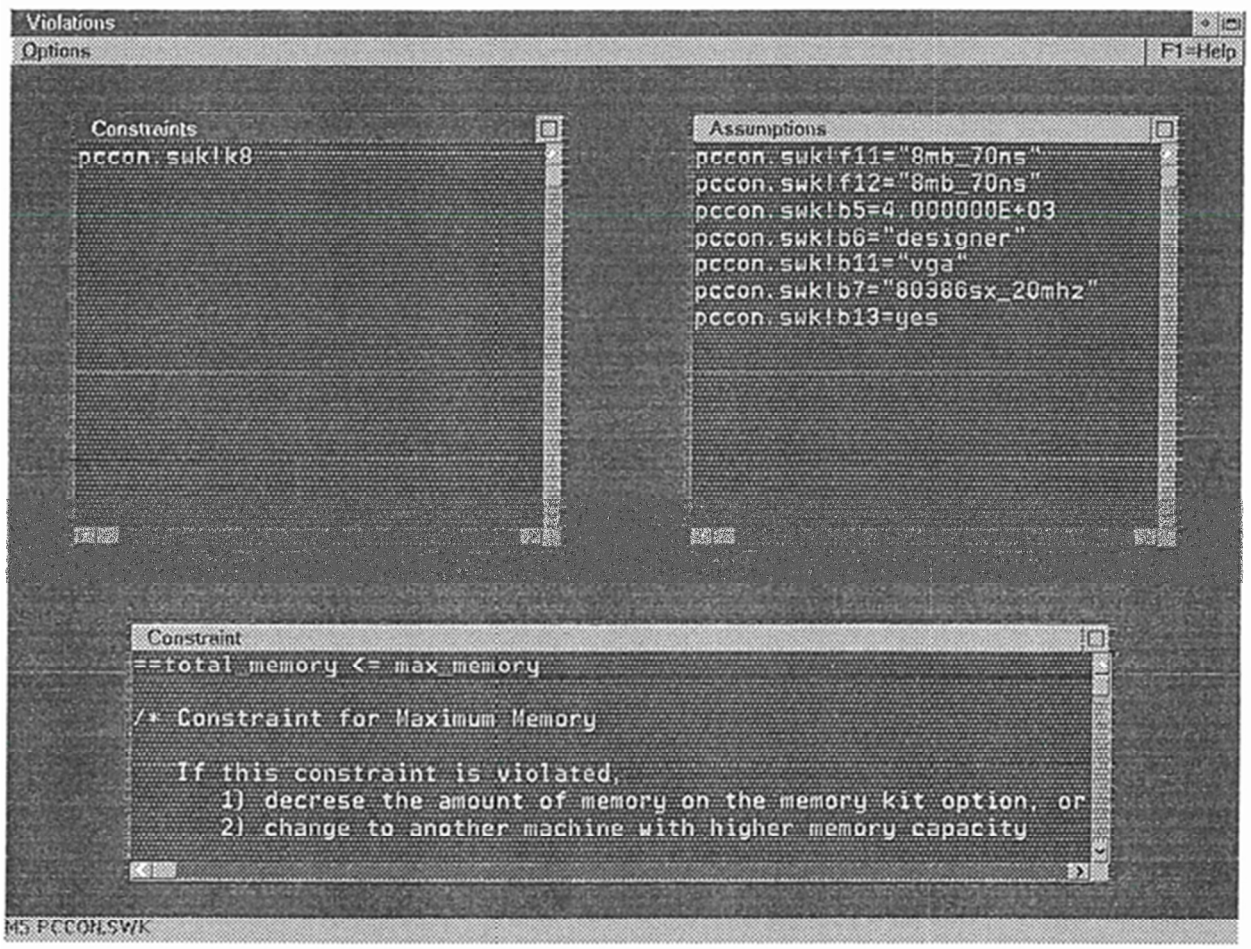



Fig. 8. Window displaying assistance to help resolve constraint violations.

(Ex. 1005, Fohn at 16.)

86. 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.

87. Fohn (and Patent Owner Trilogy) recognized the known market impact of selling 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.)

88. 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.)

89. 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.

90. 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. The configuration and validation interfaces and functionality described in Fohn could have readily been introduced to the code base for OBELICS disclosed in Axing to yield a highly-predictable product modeling and configuration utility.

91. 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. 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.)

92. 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. 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. In this manner, Fohn provides 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.

93. I am also of the opinion that it would have been obvious to try Fohn's approach to configuration and validity checking with the OBELICS software disclosed in Axling. 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.

94. 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. Below, I explain specifically where each limitation of the challenged claims is found in Axling and/or Fohn.

### 2. Axling and Fohn Disclose the Subject Matter of Claims 60-62, 64-68 and 70 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;***

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

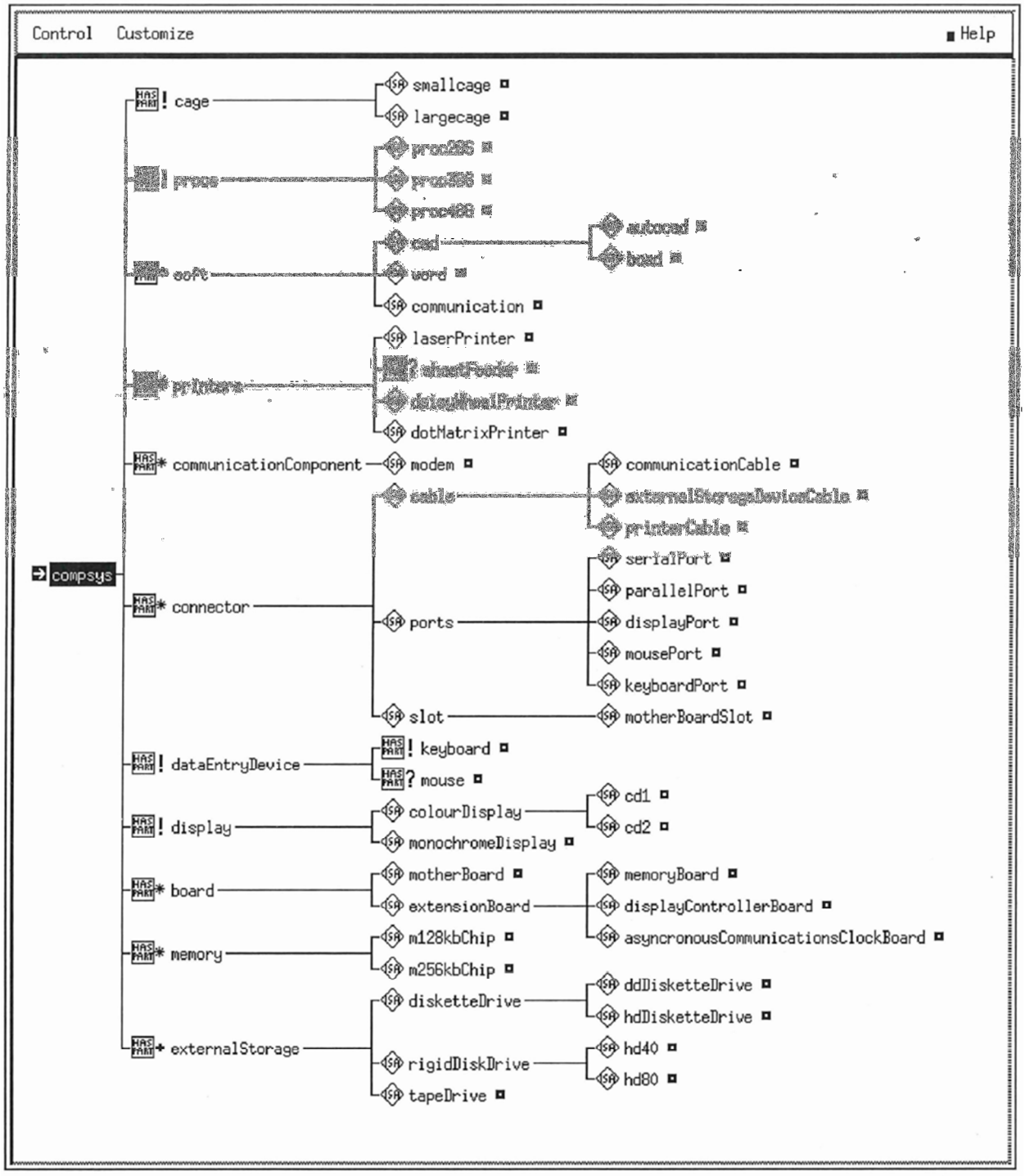



FIGURE 5.2. The Domain Model Editor

(Ex. 1004, Axling at 13.)

96. 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 +. 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.)

97. 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.)

98. Axling satisfies both constructions. 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. The relationship itself is depicted, using the symbols !, *, ? and + described above.

99. Axling also discloses generating "part relationships" using the Domain Model Editor shown in Figure 5.2. 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. For example, the part "printer" may optionally include the part "sheetFeeder." This relationship is depicted graphically in Figure 5.2 with the symbol "?".

100. 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.)

101. Axling satisfies both constructions. 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.). The relationship itself is depicted in the middle, using the symbols ! and ? described above.

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

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

103. Axling introduces the concept of "Interactive Configuration by Selection" or "ICS." (Ex. 1004, Axling at 2.) Axling describes ICS as "a less complex and more interactive approach to configuration." (Ex. 1004, Axling at 2.) Specifically, Axling describes an ICS that "allow[s] the user to alter previously made configuration decisions." (Ex. 1004, Axling at 5.) The ICS system "presents all valid alternatives and also makes it easier to alter previous choices." (Ex. 1004, Axling at 6.) 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.)

104. 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.) 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. 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.)

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

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

106. 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.) 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.

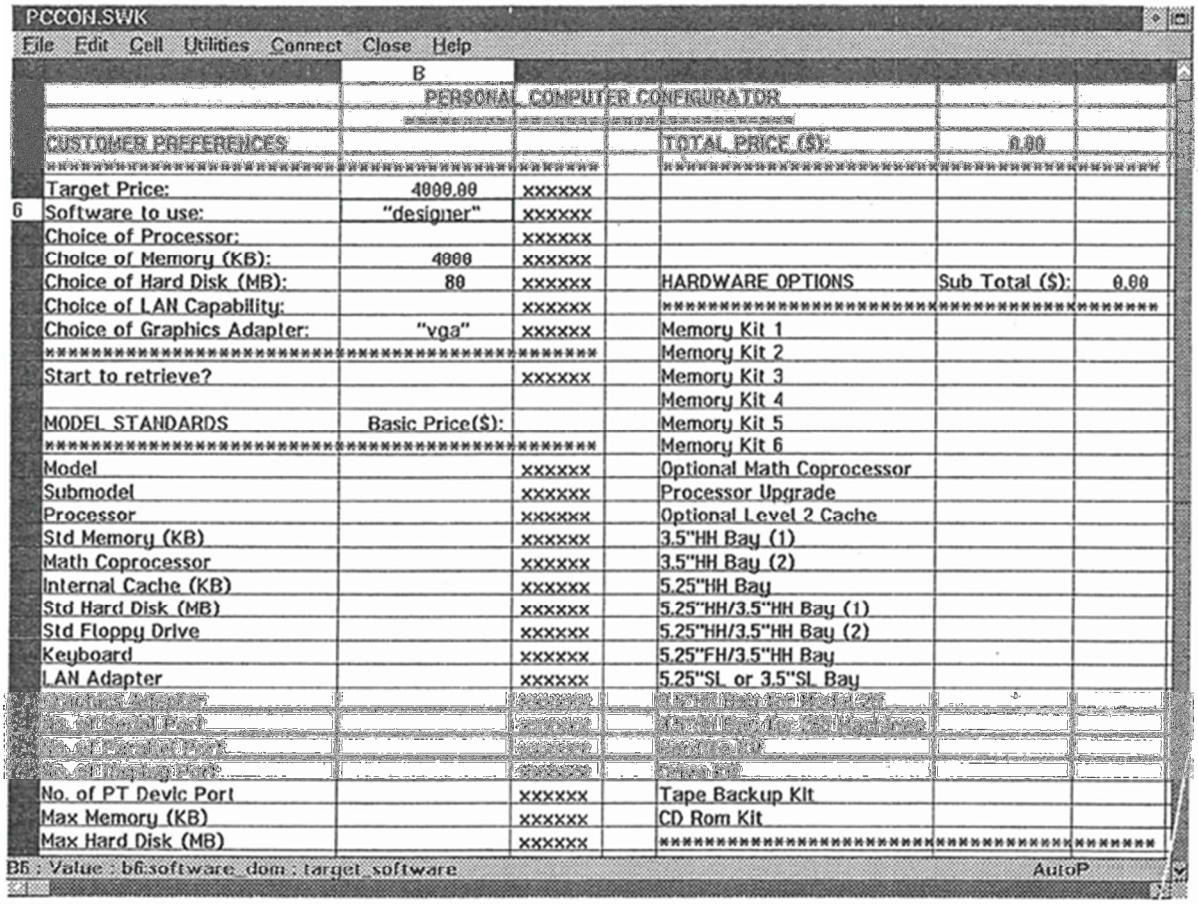
> **2. Try to satisfy the instance's preconditions** If any of the preconditions fails then ***the instance cannot be included in the construction, so the evaluation of the instance fails***.

(Ex. 1004, Axling at 17-18, emphasis added.)

107. 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.) 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.)

108. 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.)

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


PCCON.SWK

File Edit Cell Utilities Connect Close Help

| | B | | | | | |
|---|---|---|---|---|---|---|
| | PERSONAL COMPUTER CONFIGURATOR | | | | | |
| | CUSTOMER PREFERENCES | | | TOTAL PRICE ($): | 0.00 | |
| | Target Price: | 4000.00 | xxxxxx | | | |
| 6 | Software to use: | "designer" | xxxxxx | | | |
| | Choice of Processor: | | xxxxxx | | | |
| | Choice of Memory (KB): | 4000 | xxxxxx | | | |
| | Choice of Hard Disk (MB): | 80 | xxxxxx | HARDWARE OPTIONS | Sub Total ($): | 0.00 |
| | Choice of LAN Capability: | | xxxxxx | | | |
| | Choice of Graphics Adapter: | "vga" | xxxxxx | Memory Kit 1 | | |
| | | | | Memory Kit 2 | | |
| | Start to retrieve? | | xxxxxx | Memory Kit 3 | | |
| | | | | Memory Kit 4 | | |
| | MODEL STANDARDS | Basic Price($): | | Memory Kit 5 | | |
| | | | | Memory Kit 6 | | |
| | Model | | xxxxxx | Optional Math Coprocessor | | |
| | Submodel | | xxxxxx | Processor Upgrade | | |
| | Processor | | xxxxxx | Optional Level 2 Cache | | |
| | Std Memory (KB) | | xxxxxx | 3.5"HH Bay (1) | | |
| | Math Coprocessor | | xxxxxx | 3.5"HH Bay (2) | | |
| | Internal Cache (KB) | | xxxxxx | 5.25"HH Bay | | |
| | Std Hard Disk (MB) | | xxxxxx | 5.25"HH/3.5"HH Bay (1) | | |
| | Std Floppy Drive | | xxxxxx | 5.25"HH/3.5"HH Bay (2) | | |
| | Keyboard | | xxxxxx | 5.25"FH/3.5"HH Bay | | |
| | LAN Adapter | | xxxxxx | 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 | | | |

B6 : Value : b6.software_dom : target_software AutoP

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

(Ex. 1005, Fohn at 12.)

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

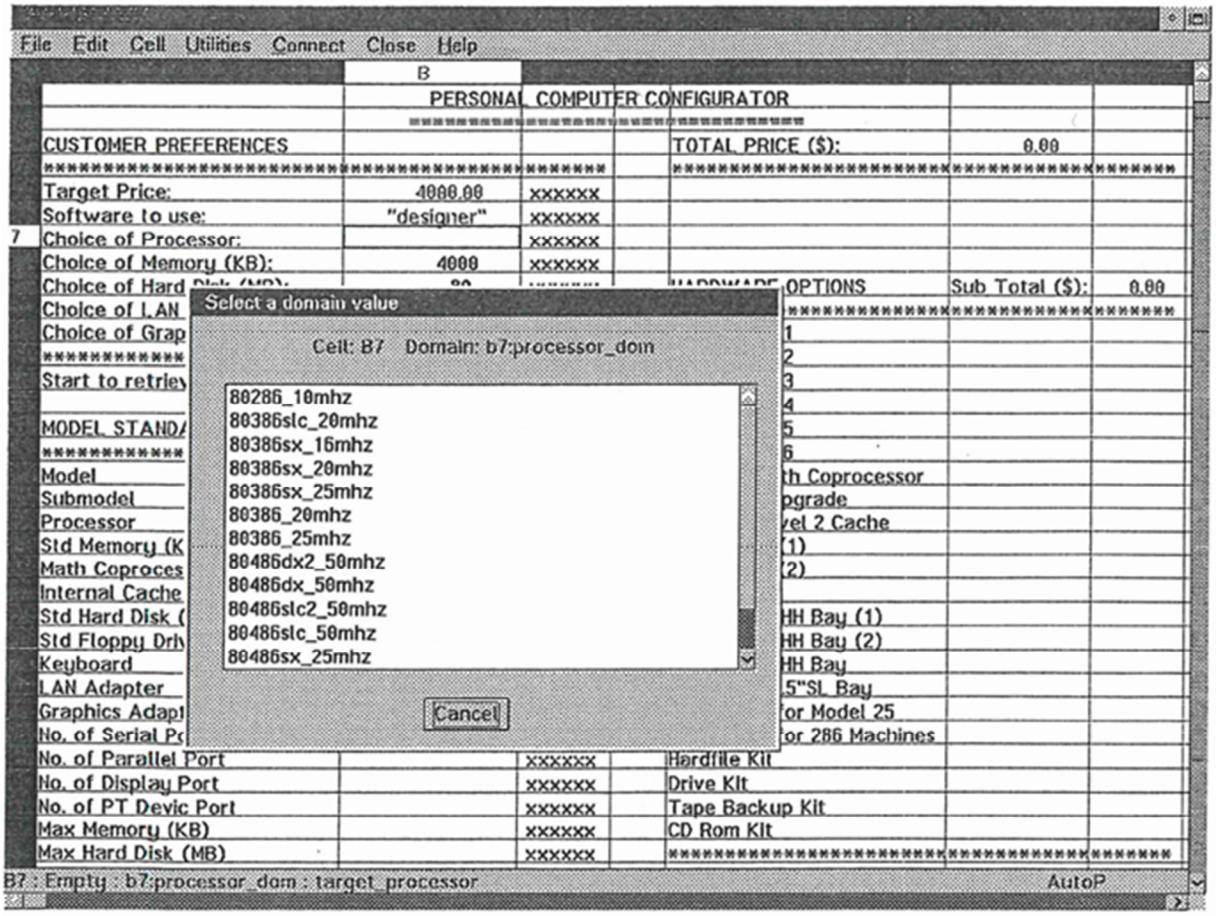



Fig. 5. Selection of a customer's preferred microprocessor.

(Ex. 1005, Fohn at 13.)

111. 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.) 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.) This scenario creates a "violation" condition. (Ex. 1005, Fohn at 16.)

112. 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.)

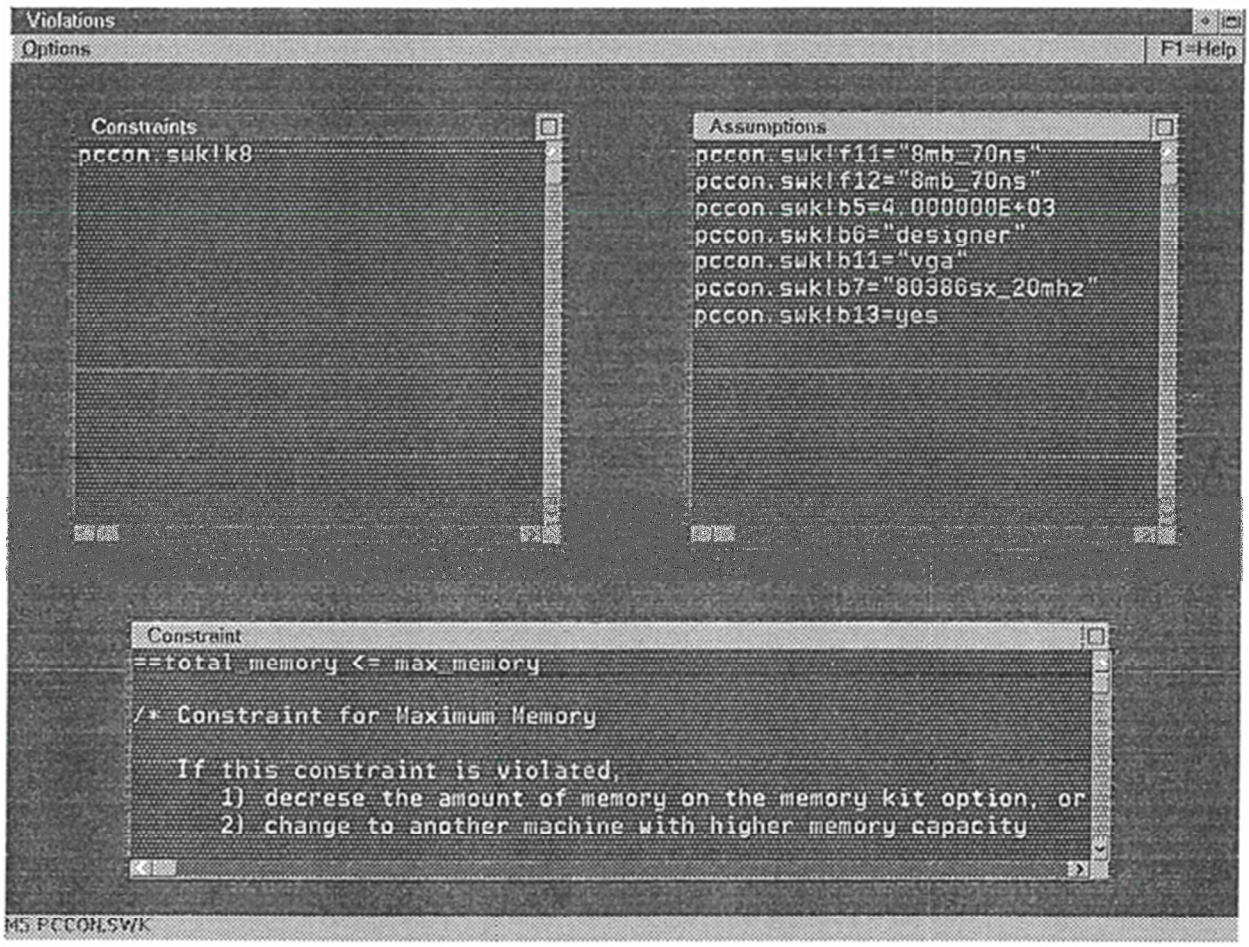



Fig. 8. Window displaying assistance to help resolve constraint violations.

(Ex. 1005, Fohn at 16.)

113. This validation of user input is based on the product definition, relationships between the product and parts and the current configuration state.. 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.) 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 (1333X the memory of a 1951 UNIVAC mainframe, which weight 16,000 lbs.; 1/8000$^{th}$ the maximum memory of a modern desktop computer). 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.)

114. 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." Indeed, the constraint in the model that was violated is provided in the violation window shown in Figure 8.

> … ***[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.***

115. 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 user selections in the current configuration state. (Ex. 1004, Axling at 17.) Based on that model, Axling "presents all valid alternatives and also makes it easier to alter previous choices." (Ex. 1004, Axling at 6.)

116. 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.) 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.) 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*.)

117. 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. 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. These options are

based on the product definition, the product and part relationships defined in the model, and the current configuration state. 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.) 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.

> … ***[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;***

118. 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.

119. 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.)

120. 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.

121. 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.) These constraints correspond to Memory Kit #1 (8MB) and Memory Kit #2 (8MB).

122. Fohn also discloses modifying the configuration to satisfy the active relationship when it is determined that a modification is needed. The "violation window" shown in Figure 8 reveals a determination that modification is needed. The window provides recommended steps for modifying the configuration, *i.e.*, "decrease the amount of memory on the memory kit option." (Ex. 1005, Fohn at 16, Fig. 8.) Fohn then explains that "[b]y deleing 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.)

123. 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.

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

124. Patent Owner previously asserted in the *Selectica* litigation that a "not Activateable relationship" is "a relationship that cannot be put into effect."

125. Fohn discloses determining at least some relationships that would result in an invalid configuration, and making those relationships "not applicable" or "na."

126. 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.) 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.)

127. Thus, Fohn discloses each limitation of dependent claim 61.

> … ***[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.***

128. Claim 62 introduces the limitation "the left-hand side of said relationship," yet neither claim 60 nor claim 61 recite a "left-hand side." This limitation thus lacks antecedent basis in claim 60 or 61.

129. 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.)

130. The '651 specification describes "Relationship Evaluation" for determining whether a particular relationship is "active" or "inactive." (Ex. 1001, '651 Patent at 12:35-43.) 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*.)

131. 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.) 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.)

> 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.)

132. 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.

133. 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.

134. In addition, it was well known at the time of the invention to represent knowledge and rules in the context of expert and configuration systems using a left-hand / right-hand expression. For example, in MICON: A Knowledge Based Single Board Computer Designer,[5] Birmingham describes MICON, a ***Mi***cro-processor ***Con***figurator 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 software. (Ex. 1004, Axling at 6.) 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.)

[5] Ex. 1006 [Birmingham] is a true and accurate copy of: 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).

135. In Introduction to Knowledge Systems,[6] 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.) 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.) "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.) The left-hand side of a "production rule" may include a "design constraint." (*Id.*, 5:9-10.) The right-hand side may include an "action list" which operates "to remove the element from, or add the element to, the design representation." (*Id.*, 5:27-30.)

136. 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.

[6] Ex. 1007 [Stefik] is a true and accurate copy of: Mark Stefik, Introduction to Knowledge Systems, Morgan Kaufmann Publishers, Inc., San Francisco, CA (June 15, 1995).

… ***[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.***

137. As explained above with respect to claim 61, Fohn discloses an "includes" relationship, and modifying the configuration to satisfy the relationship. Fohn also discloses including in the configuration elements identified in a right-hand side of the "includes" relationship. 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. (Ex. 1005, Fohn at 14-15, Fig. 6.) In this example, the selection of the processor (80386sx_20mxz) is the left-hand side, and the standard values populated in Fig. 6 as a result of value propagation constitute the right-hand side of the includes relationship.

138. 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." 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. 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.

139. 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.

> … ***[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.***

140. 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.) 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.) 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.

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

> … ***[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.***

142. 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.) 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.) In this manner, Fohn discloses removing elements from the configuration.

143. 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. 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.) 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.

144. Thus, Fohn discloses a removes relationship, and modifying the configuration to remove components as a result of that relationship.

> … ***[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.***

145. 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.)

146. 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. After the sales representative makes a *selection*, the system performs further data retrieval as well as value propagation." (Ex. 1005, Fohn at 14-15.)

147. 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. 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. This determination causes the pop-up menu described in Fohn for selecting the submodel.

148. 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.) 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.

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

149. 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.) 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.)

150. 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. First, there are only two possible options for implementing in a given relationship, either the left-hand side or the right-hand side. Implementing the relationship on one side, as opposed to the other, is not inventive. Given that there are only two alternatives, it would be obvious to try both of them. 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. 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. 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. As explained in the obviousness section above, that was a well-known problem and there was a substantial market force to solve it. A person of ordinary skill in the art at the time of the invention would have considered this a matter of common sense. 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.

> … ***[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.***

151. Fohn discloses the step of validating whether relationships are satisfied without regard to elements that are marked notActivateable. 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.) For

example, selection of the chosen processor makes Memory Kits 3-6 and several disk bays not applicable to the current configuration. Fohn states that "[a]fter the sales representative makes a selection, the system performs further data retrieval as well as value propagation. Propagation is automatic and results in a fully configured standard offering as shown in Fig. 6." (Ex. 1005, Fohn at 14-15.)

152. Thus, Fohn discloses determining whether the relationships other than those marked notActivateable in said set of relationships can be satisfied.

> … ***[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.***

153. 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.) Axling states that an ICS system "makes it easy to alter previous choices." (Ex. 1004, Axling at 6.) 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.) The Domain Model Editor "makes it easy to add and modify components." (Ex. 1004, Axling at 12.) Axling thus permits input without regard

for the order in which elements are selected for the definition as claimed.

154. Thus, claim 71 is obvious in view of Axling.

### B. Ground 2 – Dependent Claims 63 and 69 are Obvious in View of Axling, Fohn, Skovgaard 1995 and Baker

#### 1. 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

155. 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. Fohn states that a database is implemented to record "any interrelationships among elements of a model." (Ex. 1005, Fohn at 8.) These "interrelationships," which Fohn describes as "constraints," may be represented as "a database table or any data structure." (*Id*.)

156. 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.

157. 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.) This was the conventional architecture of the prior art configuration systems.

158. 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.) 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.)

159. 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." As Skovgaard 1995 states, this "State of the Art" engine promotes "efficient execution" of the constraints in a configurator. 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.

160. 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. Baker,[7] 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)

161. 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

[7] Ex. 1015 [Baker] is a true and accurate copy of: Baker, Henry G., Efficient Implementation of Bit-Vector Operations in Common Lisp, ACM SIGPLAN Lisp Pointers, Volume III Issue 2-4, April-June 1990, pp. 8-22.

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. 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. 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.

> … ***[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.***

162. 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." Neither claim 62 nor claims 60 and 61 from which it depends recite "bits corresponding to the elements." This limitation thus lacks antecedent basis.

163. Axling and Fohn do not disclose representing elements and relationships as bit vectors, and comparing bit vectors, as recited in claim 63. 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.)

164. In Skovgaard's 1995 paper,[8] 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.

> 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.)

165. 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

[8] Ex. 1015 [Skovgaard 1995] is a true and accurate copy of: Skovgaard, H.J., A New Approach to Product Configuration, ICLE, 3rd Int'l Conf. on Concurrent Engineering & Technical Information Processing, Feb. 3, 1995.

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.

166. My interpretation of Skovgaard 1995 is confirmed in Skovgaard 1998[9] which describes salesPLUS used commercially in May 1996 by Wittenborg A/S. (Ex. 1016, Skovgaard 1998 at 41.) Skovgaard 1998 explains that the "Kernel," which "holds an internal representation of the model," includes an "inference engine" that performs "Boolean constraint handling." (Ex. 1016, Skovgaard 1998 at 36-37.) Skovgaard 1998 explains that "salesPLUS compiles the product model with constraints to create several truth tables" in "binary-array form." (*Id*. at 37.) The paper also states that logical operations may be performed on the binary arrays, such as AND, OR and XOR. (*Id*.)

167. 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.

[9] Ex. 1016 [Skovgaard 1998] is a true and accurate copy of: Skovgaard, HJ, Yu, Bei, A Configuration Tool to Increase Product Competitiveness, IEEE Intelligent Systems 1094-71767/98 (1998).

168. 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. Baker,[10] 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.)

169. In the context of claim 63, bit vectors are implemented to represent "sets and relations" – a purpose for using bit vectors disclosed in Baker.

170. Baker also discloses a "Chart of analogous operations for integers, bit-arrays and bit sequences." (Ex. 1015, Baker at 2.11.) The chart includes an

[10] Ex. 1015 [Baker] is a true and accurate copy of: Baker, Henry G., Efficient Implementation of Bit-Vector Operations in Common Lisp, ACM SIGPLAN Lisp Pointers, Volume III Issue 2-4, April-June 1990, pp. 8-22.

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.

171. 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.

172. For example, a 1991 C++ library[11] 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.)

173. 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 context of a bit set class to compare the contents of two bit vectors. (Ex. 1014, Texas Instruments at 8-7.) Conventional bitwise operators are a staple of C++

[11] Ex. 1014 [Texas Instruments] is a true and accurate copy of: Texas Instruments C++ Object-Oriented Library User's Manual, 1991.

programming.[12] Note that the capability of storing and computing with bit sets proved sufficiently conventional that the disparate implementations in libraries such as that from Texas Instruments were harmonized and folded into the C++ Standard Library in 1996 to create the "bitset" class.

174. 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. 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. 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 (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. 1015, Baker at 2.8.)

[12] Ex. 1012 [Dewhurst] is a true and accurate copy of: 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.***

175. 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.) Each limitation of this claim falls within the scope of "sets and relations."

176. 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.)

177. 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.

### C. Ground 3 – Dependent Claim 72 is Obvious in View of Axling, Fohn and Skovgaard 1995

#### 1. A Person Skilled In The Art At The Time Of The Alleged Invention Would Have Combined the Teachings of Axling, Fohn, and Skovgaard 1995

178. 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.

179. 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.

> … ***[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.***

180. The Saturn-based PC/CON application disclosed in Fohn does not impose an order on the configuration process. 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.)

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

182. 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.

183. Similar to Fohn, Skovgaard 1995 describes a key advantage of the salesPLUS product configurator as "free order of choices." (Ex. 1010, Skovgaard 1995 – Abstract.) 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.

> 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.)

184. 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.

185. Thus, claim 72 is obvious in view of Axling, Fohn and Skovgaard 1995.

## VIII. Conclusion

186. In my opinion, all the elements of the challenged claim limitations are disclosed by the references discussed above and that the claims are unpatentable in view of these prior art references.

187. Specifically, as detailed in the analysis provided above, it is my opinion that 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. It is also my opinion that claims 63 and 69 are obvious in view of Axling, Fohn, Skovgaard 1995 and Baker. Finally, it is my opinion that claim 72 is obvious in view of Axling, Fohn, and Skovgaard 1995.

188. I reserve the right to supplement my opinions to address any information obtained, or positions taken, based on any new information that comes to light throughout this proceeding.

I declare under penalty of perjury that the foregoing is true and accurate to the best of my ability.

Executed on: 5/9/2016 _____

Dr. Philip Greenspun