Reconstructing the Blade Technology Domain with Grounded Theory

André Zwanziger

Institute for Technical and Business Information Systems,
Department of Computer Science,
Otto-von-Guericke Universität, Magdeburg, Germany
andre.zwanziger@ovgu.de
http://www.ovgu.de/

Abstract. Domain models are often reconstructed from different resources and contain domain elements, their properties and relationships. Ideally, the domain creation process is made transparent, so that each decision in the creation process can be traced back to the original source. One way to reach this goal is to use qualitative content analysis methods, e.g. the Grounded Theory. In this paper, the Grounded Theory is applied to the domain of IT infrastructures especially the blade technology and shows the necessary steps to extract a domain model from arbitrary documents.

Keywords: Grounded Theory, IT infrastructure, blade technology, domain engineering.

1 Introduction

One central question in domain engineering focusses on how a domain model can be extracted from a given field of application and/or its context. Starting from various definitions of the term "domain" e.g. [2,11,6,12] it can be stated that many authors assume that most domains already exist. The domains are often implicitly given or hidden in various documents (e.g. specifications, requirement documents, textbooks, source code, and/or standards) and very well known and understood by domain experts. The challenge for domain language creators therefore is to externalize this knowledge from given sources and to build a widely accepted model with an appropriate abstract and concrete syntax.

One way to externalize this knowledge for a domain model (or product family) are commonality and variability analysis [11]. Both methods require existing products and/or product families, which are compared and result in a list of domain elements as well as a list of variation points. In [11] three different kinds of commonality and variability analysis are proposed: (1) The application requirement matrix based analysis contrasts a list of requirements with different products and marks requirements as mandatory or optional for a given product. If all products have the same mandatory requirement, the requirement belongs to a domain. (2) The priority based analysis involves different stakeholders in

C. Salinesi and O. Pastor (Eds.): CAiSE 2011 Workshops, LNBIP 83, pp. 187–196, 2011.

the domain analysis process. The stakeholders are asked to prioritise predefined requirements for different products. Basic and high-rated requirements constitute the basis of the domain model. (3) The Check-list based analysis sets out from a given list of requirements that must be considered for the domain model. Another existing method is "abstracting away" characteristics from domain elements that potentially vary in the quality of the elements [5]. Finally, intuitive ways to elaborate a domain model, being made by one expert or agreed by a group of experts can be mentioned as possible method.

While intuitive methods are mainly the subject of implicit rules commonality and variability analysis and the abstraction mechanism follow more explicit rules. However, the basic question still remains, how the initial resources for the analysis are created. Commonality and variability analysis need a list of requirements or (in other analysis settings) features, use cases or components. The same issue occurs in the case of abstracting model elements, because a list of "to be abstracted" elements is required as a basis of the analysis.

Extracting those initial analysis items from arbitrary sources is usually examplary for social science methods, e.g. Grounded Theory [4] or qualitative content analysis [8]. Commencing with a predefined research question, different sources are selected, gradually analysed, interpreted, and result in a model that explains a social phenomenon. Applied to the field of domain analysis the research question is how a domain is structured (which elements exist, which properties do the elements consist of, which relationships exist between the elements). A possible result of the analysis can be a structural model, such as an UML class diagram, an Entity Relationship diagram, or a Feature diagram etc. that describes the domain.

In this paper, the Grounded Theory is applied to the field of IT infrastructures and is used to reconstruct a vendor neutral domain model for the blade technology. Section 2 introduces the overall application field of IT infrastructures and is narrowed to the blade technology. In section 3 the Grounded Theory is explained and applied to the domain of blade technology. Further, in section 4 the domain model is derived from the analysed data. Finally, the paper closes with a conclusion and outlook in section 5.

2 IT Infrastructures

In the context of this research, IT infrastructures are defined as all hardware and software entities, as well as facilities that are needed to run end user applications [10]. Hardware comprises computing and storage systems (e.g. blades, storage area networks), network technology (e.g. switches, cables), peripheral equipment (e.g. keyboards, printers), and additional equipment that is required to run the hardware (e.g. racks, uninterruptible power supplies). Software encompasses system software (e.g. operating systems) and basic software (e.g. databases, middleware systems). Additionally, IT infrastructures include rooms and buildings that are specially equipped to run hardware (e.g. data centers with heating, ventilation, and air conditioning (HVAC), redundant power supplies).

In most cases, IT infrastructures are operated in a managed system environment to ensure a demanded service level complying with a financial budget. Managed system environments comprise organisational aspects (product lifecycle phases, typical processes within the phases and responsible persons) as well as supporting technical applications (e.g. configuration management systems, monitoring tools). Changes in a managed system environment involve different stakeholders from different domains and needs to be coordinated in various directions. Here, models can help for the following purposes: (1) models enhance the *communication* between stakeholders through standardized language and graphical symbols. (2) Models *document* decisions made in the software introduction process and act as reference for future changes of the IT infrastructure. (3) Models can be *validated* to ensure the correctness of the required structure at an early stage of the deployment process. (4) Models can be used to *analyse* the behaviour in case of an incident or error, and (5) models can be used to *generate* artifacts that are used at runtime (e.g. configuration files) or to define a target state of the IT infrastructure in a managed system environment (e.g. configuration management system).

Even though these modeling purposes emerge, a gap between general purpose modeling languages (e.g. UML [9], Archimate [7]) and detailed class models for the management of IT infrastructures (e.g. Common Information Model [3]) can be identified. UML and Archimate define abstract modeling elements such as *node* and *device* on which *artifacts* can be deployed. Nodes and devices can be connected with a special dependency relationship called *Communication Path*. By default, these modeling elements do not include any additional, characterizing attributes. On the other hand, the Common Information Model (CIM) contains vendor independent, concrete type definitions for several (low-level) IT infrastructure elements with detailed attributes and operations (such as *CIM_Processor* or *CIM_PCIController*). Low-level refers to the fact, that most of these elements are configurable building blocks in a higher-level (fully-functional) computing system. Higher-level structures such as rack-mountable computing servers, switches or storage systems are not defined in CIM. In both cases, modelling IT infrastructures lacks concrete model elements, that are very well known by IT infrastructure domain experts (e.g. system administrators). A (set of) domain specific language(s) for IT infrastructures could help to close this gap.

Due to the vast amount of IT infrastructure components, the focus for a qualitative analysis in this paper is set on the blade technology. Blade technology is widely used in data centers as compact computing units. It consists of highly configurable elements and forms a well defined (sub-) domain of its own inside IT infrastructures. Moreover, there are many vendors for the blade technology, but each vendor has their own naming for similar products. Even though, blade products are incompatible between vendors in most cases, there are apparent similarities among products from different vendors and therefore a suitable field for domain engineering.

3 Qualitative Content Analysis with Grounded Theory

The goal of the Grounded theory is to develop a theory, that is based on systematically analysed data [4]. It has its origin in social science and is used to discover theories that explain social phenomena. Starting from a research question, different documents (e.g. transcribed interviews) are analyzed line by line and categories, their properties and relationships are elaborated [13]. During the analysis process, all documents are compared continuously and coded in three stages: (1) The goal of the *open coding* is to find categories and their properties in the documents. (2) The *axial coding* stage reveals relationships between categories and (3) the *selective coding* stage is to identify the core category and systematically relates it to other categories to find "hidden" relationships. During the coding stages, all documents are marked with text notes that refer to categories, properties, and/or relationships and thereby ensure a traceable decision making process.

Even though, the theoretic models in social science differ from the expected results of the domain engineering process, all coding stages can be applied with slight changes to the domain analysis process. Depending on the intended meta-model of the domain model, categories from the open coding can be seen as domain elements, such as classes, entities or features. Attributes and their types can be related to the intended domain elements and form the internal structure of a domain element. Relations between domain elements can be gained in the axial coding stage. Here, relations defined in the meta-model determine a starting point for the analysis. The third coding stage differs most from the original approach, due to the fact, that a domain model typically contains many elements on the same level of abstraction. Here, the relation of domain elements to other domains is the more interesting part for domain engineering, which leads to an integration of the new domain into existing domains.

In the context of this paper, the *research question* is how the blade technology can be structured in a vendor independent domain model. To further narrow the focus of the research question, the following analysis questions have been formulated in two categories: (1) *Structural Questions:* (a) What kind of single marketable/configurable components exist for the blade technology? (b) Which properties and their characteristics are named for each item/in relation to other items? (c) How are the components related to each other (dependency, aggregation, composition, specialization)? (d) Which cardinality (min/max) do the items have in relation to each other? (2) *Physical Characteristics:* (a) Which measures are used to describe the physical characteristics of each item? (b) Which measures are used to describe the environment of each item in operating and non-operating mode?

The next step is to obtain an appropriate *set of data* that forms the basis of the analysis. Here, Gartner's magic quadrant about blade server technology was the starting point for the document collection [1]. All listed vendors in the study have an international market presence and a sales volume of at least

$5 million in 2010. All vendors have been categorized into one sector of the quadrant, market leaders, niche players, visionary players. The fourth sector, challengers, remained empty in the study. To create a widely applicable domain model, the two market leaders HP and IBM, a visionary player (Cisco) and a niche player (Oracle) have been chosen for this analysis. In the next step, each vendor's product family for blade technology was investigated and specification documents have been retrieved from the public websites. Even though, most vendors offer more than one product relevant for this analysis, the vendor's product specification documents barely differ from each other. Therefore, it was sufficient to select one product and its specification from each vendor. In the following, the coding stages for two domain elements (BladeChassis and Blade) are described.

In the *open coding* stage of the analysis, all specification documents for BladeChassis and Blades have been compared and marked with categories, which can be aligned with the two parts of the analysis questions. Structural elements for BladeChassis contained information about (A) Blades, (B) Interconnects, (C) Fans, and (D) Power Supplies. Further information given in the specifications have been classified into the category (E) Other. Structural elements for Blades have been annotated with a lower case letter and comprise information about (a) Processors, (b) Memory, (c) Expansion Cards, (d) Storage Volumes, (e) Network Adapters, and (f) Other data. The physical characteristics for both domain elements comprise the following categories: (1) Dimension (height, depth, width), (2) Temperature, (3) Humidity, (4) Altitude, (5) Weight, and (6) Other. The category identifiers and the corresponding specification entries for each product are listed in Tables 1 and 2 for BladeChassis and Blades.

Parallel to the open coding stage, the *axial coding* has been performed to the specification documents and revealed the dependencies between domain elements. On the one hand, the structural information in the tables automatically relates different domain elements to each other. On the other hand, the relationship's cardinality has been coded in Table 1 in squared brackets to give a more precise understanding of the relationship. If no lower or upper cardinality could be determined from the specifications, a dash is printed as value.

During the axial coding phase two basic architectural differences have been discovered. HP, IBM and Cisco provide Blades, which contain slots for Expansion Cards. Expansion cards are mostly used to equip Blade servers with additional network adapters (such as Ethernet and Fibre Channel). In Oracle's blade technology, these expansion cards are not built into the Blades, but into the BladeChassis.

In the *selective coding* stage, all identified elements were compared with classes of the Common Information Model (CIM) to integrate the blade technology domain into an existing framework. The categories (C) Fan, (D) Power Supply, (a) Processor, (b) Memory, (d) Storage Volume, and (e) Network Adapter already exist in CIM and provide nearly all properties found in the specification documents. Therefore, these classes are subject to reuse in the domain model.

Table 1. Categories and assigned specification entries for the domain element Blade Chassis

Vendor / Product	Structural Information [min/max cardinality]	Physical Characteristics [Unit]
Cisco UCS 5100 Series Blade Server[1]	(A) Blade Server slots [-/8] (E) Fabric Extender slots [-/2] (B) Fabric Interconnects [-/4] (D) Four hot-swappable, redundant, power supplies, with IEC-320 C20 connections (C) hot-swappable fans [-/8] (E) Backplane with 1.2 Tb of aggregate throughput [1/1]	(1) Height / Width / Depth [in inch/cm; Height also in Rack Units] (2) Temperature (operating/non-operating) [in F/C] (3) Humidity (operating/non-operating) [in %] (4) Altitude (operating/non-operating) [in ft/m]
HP Blade System c3000 Enclosure[2]	(A) Device Bays – full-height [-/4] – half-height [-/8] – mixed-configuration (true, false) (B) Interconnect Bays [-/4] (C) redundant fans (D) power connectors – AC: c13/c14 [-/6] – DC: D-SUB Power 3W3 [-/-]	(6) rackable (1) Height / Width / Depth [in inch/mm]
IBM BladeCenter E[3]	(A) Blade bays [-/14] (B) Switch modules [-/4] (D) Power supply modules [-/4] hot-swap and redundant with load-balancing and failover capabilities (C) hot-swap and redundant blowers [2/-] (E) DVD multi-burner [0/1] (E) I/O-Ports: Keyboard, video, mouse, Ethernet, USB	(1) Height [in Rack Units] (6) Form factor (Rack-mount)
Oracle Sun Blade 6000 Series[4]	(A) server modules [-/10] (B) Network Express Modules [-/2] (D) AC power supply [-/4] – hot-swappable, load-sharing, load-balancing (C) fans: rear, front	(1) Height / Depth / Width [in mm/inch; Height also in U] (2) Temperature (operating / non-operating) [in C/F] (2) Optimum ambient [in C/F] (3) Relative Humidity (operating / non-operating) [in %] (4) Altitude (operating / non-operating) [in m/ft] (6) Sine-Vibration (operating / non-operating) [in G] (6) Shock [in G] (6) acoustic noise [in B] (5) Weight (fully loaded, empty) [in kg/lbs]

[1] http://www.cisco.com/en/US/prod/collateral/ps10265/ps10279/data_sheet_c78-526830.pdf
[2] http://h20195.www2.hp.com/v2/GetPDF.aspx/4AA0-5978ENW.pdf
[3] ftp://public.dhe.ibm.com/common/ssi/ecm/en/bld03018usen/BLD03018USEN.PDF
[4] http://www.oracle.com/us/products/servers-storage/servers/blades/033613.pdf
(Last access of websites on March, 7th 2011.)

Table 2. Categories and assigned specification entries for the domain element Blade

	Cisco UCS B250 M2[5]	HP ProLiant 620c G7[6]	IBM BladeCenter HX5[7]	Oracle Sun Blade X6275 M2[8]
Structural Information	(a) Processor: – Type – Amount (min/max) (b) Memory: – Amount of Memory Slots – Type of Memory Slots (c) Mezzanine – Amount (max) – Supported Cards (d) Hard disk drives (Storage) – Amount – Type – hot swappable – size of hdd (in inch)	(a) Processor: – Number of Processors – Maximum number of cores – Supported Processors – Cache / processor [in MB] – Processor speed [in GHz] (b) Memory – Memory Type – Standard Memory – Maximum Memory – Memory Slots (e) Network and I/O – Network Adapter (integrated) – Amount of I/O Expansion Slots – Standard/Maximum I/O bandwidth – Networking and I/O options (d) Storage – Storage type – Amount – Storage controller – Storage options	(a) Processor: – Amount (std/max) – Amount of Cores – Processor speed [in GHz] (a) Cache per processor [in MB] (b) Memory – Amount of DIMM slots – DIMM Type – DIMM speed [in MHz] (c) Expension Slots – Amount of slots – Supported Types (d) Disk Bays (Storage) – Amount of devices – hot-swappable (true/false) – max storage (e) Type of network interface (f) Raid support	(a) Processor – Amount – Type (a) Cache per Processor / Core [in kB or MB] (b) Memory – DIMM Type – DIMM Speed – Amount of DIMM slots – Max Amount of RAM [in GB] (e) Network interfaces on board (d) Storage – Type – Connection Speed (f) Graphics controller (c) Midplane I/O – Extension I/O cards Type (f) Front Panel I/O (VGA, Serial Console, USB ports)
Physical Characteristics [Unit]	(2) Temperature (operating / non-operating) [in F/C] (3) Humidity (operating / non-operating) [in %] (4) Altitude (operating / non-operating) [in ft/m]	(1) Form factor (Full-height, single-wide)	(1) Form factor (single/double-wide)	(1) Height / Width / Depth [in mm/inch] (5) Max Weight [in kg/lbs]

[5] http://www.cisco.com/en/US/prod/collateral/ps10265/ps10280/ps10915/data_sheet_c78-588109.pdf
[6] http://h20341.www2.hp.com/enterprise/downloads/4AA3-0959ENW.PDF
[7] ftp://public.dhe.ibm.com/common/ssi/ecm/en/bld03032usen/BLD03032USEN.PDF
[8] http://www.oracle.com/us/products/servers-storage/servers/blades/sun-blade-x6275-m2-server-ds-182643.pdf
(Last access of websites on March, 7th 2011.)

4 Deriving a Domain Model

Generally, the determined categories of the analysis serve as a basis to directly derive domain model elements. In this paper, the domain model is represented as UML class diagram, even though other representations (such as Entity Relationship or Feature diagrams) are also applicable.

The internal structure of the blade technology domain is shown in Figure 1. The class `BladeChassis` contains zero to many `Blade`s and `Interconnect`s and holds two aggregation relationships to the classes `CIM_PowerSupply` and `CIM_Fan`. The classes PowerSupply and Fan are imported from the Common Information Model, which is indicated by the `CIM_` prefix in the class name. The class `CIM_Fan` is specialized in an own class `Fan`, which holds the two additional attributes `redundantConfiguration` and `hotSwappable`, that could be identified during the open coding stage of the analysis.

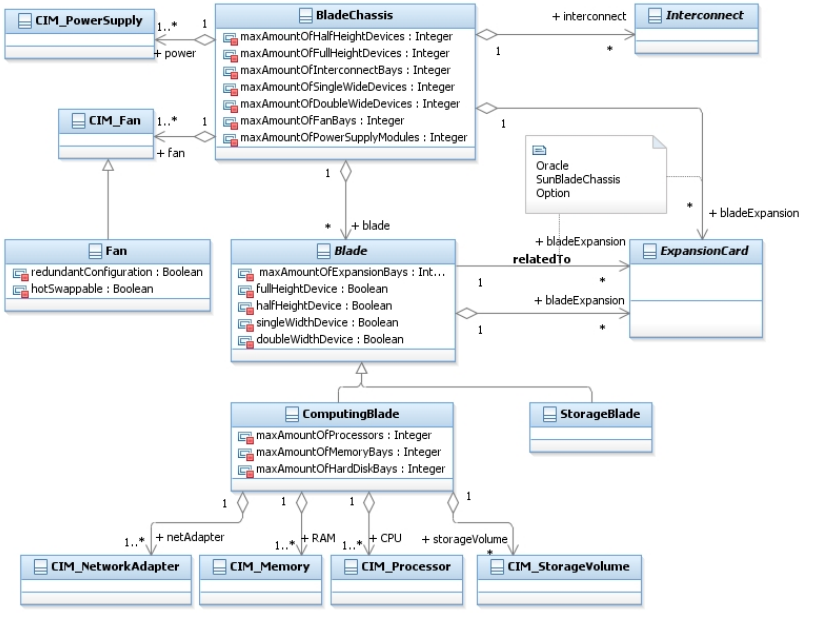

Fig. 1. Reconstructed Domain Model for the Blade Center Technology

During the product family investigation, the two blade types `ComputingBlade` and `StorageBlade` could be identified and are specialized in the domain model from the abstract class `Blade`. The class `ComputingBlade` is composed of the CIM classes `CIM_NetworkAdapter`, `CIM_Memory`, and `CIM_Processor`, whereby each relationship has a one to many cardinality. `ComputingBlade`s can have storage volumes, but can also be operated in a disk-less mode with network boot option. Therefore, only a zero to many aggregation relationship is modeled

to the class `CIM_StorageVolume`. Due to space limitations `StorageBlade`s can also be subject for future analysis and are not documented in detail here.

Both classes `BladeChassis` and `Blade` have an aggregation relationship to the abstract class `ExpansionCard`. During the axial coding stage, an architectural difference concerning the expansion cards has been elaborated, so that an expansion card can be modeled (a) as part of a blade and (b) as part of the blade chassis. In the latter case, an additional dependency relation between the classes `Blade` and `ExpansionCard` is used to assign an expansion card to a blade.

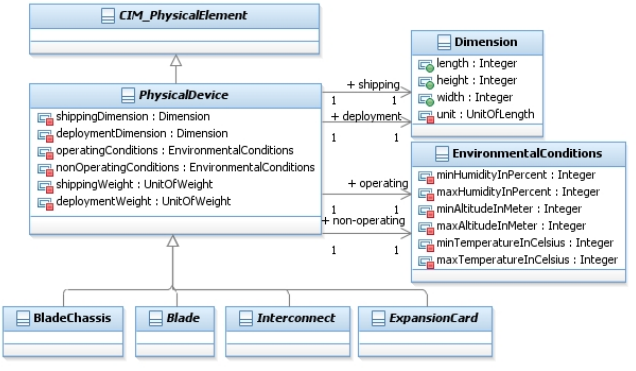

Fig. 2. Physical characteristics and type hierarchy of domain elements

In Figure 2 the domain elements `BladeChassis`, `Blade`, `Interconnect`, and `ExpansionCard` are shown in a second class diagram revealing the type hierarchy and the physical characteristics of the domain. All elements are derived from the abstract domain class `PhysicalDevice`, which itself is a specialization of `CIM_PhysicalElement` from the CIM framework. Even though all categories still can be seen as own classes, the model in Figure 2 comprises most physical characteristics into one class `EnvironmentalConditions`. The class `PhysicalDevice` holds two dependency relationships to the class `EnvironmentalConditions` that specify the conditions in operating and non-operating mode. The class `Dimension` is also referred with two dependencies from `PhysicalDevice` and is used to model the physical space of each object during shipping and deployment.

5 Conclusion and Outlook

The Grounded Theory approach enables domain engineers to extract domain models from arbitrary resources in a structured way. It extends the set of existing approaches for domain engineering and can be applied especially if handbooks, specifications or expert interviews are the main resources to create or reconstruct a domain model. Advanced concepts of the method (such as theoretic sampling

or sensitivity analysis) have not been discussed in this article, but help to refine the domain model after a first version has been created.

For domain engineering, the content analysis methods (coding stages) of the Grounded Theory have been slightly adapted. The open coding stage, still remains to detect categories from arbitrary resources. All detected categories can be seen as structural elements such as UML classes. The axial coding stage was also left unchanged in its intention and aims on finding relationships between the structural elements. The selective coding was redefined to integrate existing domain models and is intended to find software reuse options in this case.

Finally, the approach is not limited to IT infrastructures and can be used for various domains. Nonetheless, the domain of IT infrastructure is a promising subject for future domain analysis. The blade technology domain is only a small piece of a bunch of IT infrastructure elements and further domain models can support many processes during deployment, operation and change.

References

1. Butler, A., Weiss, G.J.: Magic Quadrant for Blade Servers. Gartner RAS Core Research Note G00207244. Gartner (2011)
2. Clements, P., Northrop, L.: Software Product Lines: Practices and Patterns. Addison-Wesley, Pearson Education, Boston (2002)
3. Distributed Management Task Force: Common Information Model Specification. Version 2.28.0 (2010), `http://dmtf.org/standards/cim/cim_schema_v2280` (March 7, 2011)
4. Glaser, B.G., Strauss, A.C.: Discovery of Grounded Theory. Strategies for Qualitative Research. Aldine Transaction (1967)
5. Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Spencer Patterson, A.: Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report No. CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA (1990)
6. Kelly, S., Tolvanen, J.-P.: Domain Specific Modeling. John Wiley & Sons, Inc., Hoboken (2008)
7. Lankhorst, M.: Enterprise Architecture at Work: Modelling, Communication, and Analysis. Springer, Heidelberg (2005)
8. Mayring, P.: Qualitative Inhaltsanalyse: Grundlagen und Techniken. 10., neu ausgestattete Auflage; Beltz Verlag (2008)
9. Object Management Group: Unified Modeling Language: Superstructure. Version 2.1.1 (non-change bar) (2007)
10. Patig, S.: IT-Infrastruktur. In: Kurbel, K., Becker, J., Gronau, N., Sinz, E., Suhl, L. (eds.) Lexikon der Wirtschaftsinformatik. Oldenbourg, München (2009)
11. Pohl, K., Böckle, G., van der Linden, F.: Software Product Line Engineering: Foundations, Principles, and Techniques. Springer, Heidelberg (2005)
12. Reinhartz-Berger, I., Sturm, A.: Utilizing domain models for application design and validation. In: Information and Software Technology, vol. 51, pp. 1275–1289. Elsevier B.V., Amsterdam (2009)
13. Strauss, A.C., Corbin, J.M.: Basics of Qualitative Research: Grounded Theory Procedures and Techniques, 2nd edn. Sage Publications, Inc., Thousand Oaks (1990)