1987 DECUS
EUROPE SYMPOSIUM
(Rome, Italy, September 7-11, 1987)

2057

DEVELOPMENT MANAGEMENT SYSTEM
DEVELOPMENT MANAGEMENT SYSTEM

Ben Livson

Ultraware Ltd.

P.O. Box 367, Kiryat Ono 55102, Israel. +972-3-341366.

ABSTRACT

Initial work during the last ten years for defining and prototyping software Development Management Systems has prepared the ground for presenting the concepts for the year 2000 DMS. The current DMS features are highlighted.

DMS PRODUCT EVOLUTION

Survey and development of software tools was started by the author in 1978 as employee of the Israel Aircraft Industries IAI. The first survey phase included working as a software quality assurance engineer in a large avionics project until 1982. The conclusion was that the software engineering methods have not matured to permit a 'big bang' DMS development project. Instead, an incremental evolutionary DMS development approach was selected. DMS tool development started by computerizing the modest, but highly successful TRW Unit Development Folder concept. Specification of the IAI UDF was completed in 1982-1983, the first prototypes entered use in IAI during 1984 and have been in trial use to support avionics and dynamic simulation software development. See [1] for documentation about the UDF tool. Limitations of the UDF tool were lack of configuration management services and lack of support for project management. The problem of configuration management is critical, because IAI both develops and maintains computer embedded avionic systems for the Israel Air Force in an open shop development environment. There is no clear division between development and production as is customary with most commercial data processing organizations. Bad synchronization between software development and the often delayed preparation of the folders was perceived as a major problem for the UDF tool. The other major project management problem was how to cross milestones and deliverable items with the actual status of the files system. An off-line auxiliary UDF tool could not provide answers to these problems. IAI UDF experience led to the development of the Ultraware Ltd. Development Management System.
PRESENT DMS FEATURES SNAPSHOT

DMS is a software environment driver that integrates meta-data and execution system. DMS supports software systems development, configuration management and project management for the complete life-cycle. DMS features AI type semantic analysis of the software process; solves the open shop SCM problem; traces development to project planning; maximizes management visibility; automates version control and build; supports SQA and IVV&T standards; traces test cases to requirements; computerizes the TRW unit development folder concept; helps human communications, and data sharing; controls access for work safety; provides callable interface library for customized environments.

PRESENT DMS APPROACH, CONCEPTS AND PHILOSOPHY

The DMS is a software environment driver. The DMS approach is to support software engineering in its natural environment. It is an integrated meta-information handler and execution system. Its objective is to promote data sharing and human communications in a distributed software development and support process.

The DMS is tailored to the files management of your computer system. Its meta-data supports software project organization into the natural hierarchy of directories and files. The DMS system is an opposite approach to a number of container type software tools such as the Softool CCC (tm). The advantage of the DMS as compared to the container type tools is that no export or import operations are needed to communicate between the container and the natural file system. The container tools cause a considerable overhead both in terms of wasted work time and computer resources, and compel users to learn additional command syntax. The separate hierarchy of the container tools is a likely source of confusion and a considerable burden to users. Perhaps the most severe drawback of container type tools is that low level configuration management operations are done manually off-line from the main software development process. The unique advantage of the DMS system is that all low level configuration management operations are automatically executed in parallel to the software engineering process. In particular, version control is automatically synchronized to the software engineering process and is not a separate delayed process.
A key issue of the DMS system is to minimize meta-information requested from users. The minimum requested from users is to state the names of directories and files. This information is used by the DMS to automatically execute low level configuration management commands and to provide users an optimized access to files management. This minimal information is in any case compulsory and the DMS captures this information from users. Users are relieved from restating directory and file names each time a file is being accessed. In short, the number of keyboard strokes is minimized by the DMS system. Basically, all additional meta-information such as milestone data is voluntary and aimed at providing greater management control and visibility.

The DMS system is an open access system. It does not force any specific software engineering method. It does offer comprehensive life-cycle support but it does not force users to follow any specific life-cycle model such as the waterfall model or rapid prototyping. However, considerable semantic analysis is done by the background housekeeping functions about the reasonableness of the software engineering process. Meta-data is tied to the files management system to detect anomalies. The aim is to guide DMS users to devote considerable effort in the conceptual phase of software development. The DMS system does not address itself to issues such as which method to use in requirements definition. It is only a software engineering environment driver. Each organization should use its own methods and tools via the DMS system.

Future directions of the DMS system are not aimed at providing support to any specific software engineering methodology. The experience with these methodologies has been that they cause more harm than good. The aim is to provide general purpose services such as autoregression test support, document trace tools to support verification, and templates.

A major effort will be to port the DMS tool to a number of major operating systems. Software development in a heterogeneous operating system environment is the next DMS goal. Currently, DMS is implemented for DEC VAX/VMS, Microsoft DOS for IBM PC/PS, and the first ATT Bell Laboratories UNIX V compatible implementations are nearing completion. See [2] - [3] or contact the author for information about the current status of the DMS implementations.
'Future Generation Development Management System Requirements' was initially displayed at the Tools Presentation Track, 9th ICSE, Monterey, 30.3 - 2.4.87. This is the first paper form presentation.

Future DMS requirements shall be based on the subset of SOLVABLE software engineering problems. The general theme of formalizing and automating the software process is not in the author’s opinion a solvable problem. The software process may be formalized and automated only in application domains that are completely understood and stable.

There is no need for a DMS in such application domains. Commercial DP systems that can be automatically generated by application generators are examples. Future application generator will have natural language interface and built-in configuration management services. Thus, future application generators will have similar functions to the DMS, but the basic concepts are different.

Future DMS will not be based on any of the new paradigms for software engineering. Both formal and semi-formal software engineering methods are likely to continue to do more harm than good until the year 2000. In most application domains even formalized executable requirements are not enough. The problem of capturing requirements still remains the classic problem of developer and customer understanding each other. The automation of the implementation phase is gradual but certain. The "final solution", if ever attained, must come from the AI research of cognitive reasoning. We do not believe that an AI solution that supports the conceptual phase will be available before the 2020 - 2050 period. New tools supporting conceptual phase may be integrated to DMS 2000 open ended architecture. The focus, however, shall NOT be on external tools integrated to DMS 2000.
DMS 2000 ARCHITECTURE: THE PRIMARY SUBSYSTEMS

- Standard Relational Information Architecture
- Natural Language Interface
- Execution Engine
- AI Component Analyzing the Software Process

Standard Relational Information Architecture Supports:

- Automatic creation of test cases from requirements
- Traceability
- Interfaces control definition
- Incremental development
- Reusability
- Configuration management
- Contract management
- Management visibility
- Distributed project environment

Natural Language Interface to DMS Information Architecture shall hide database aspects of DMS 2000. In particular, no language or syntax for data retrieval or entry is necessary. Natural language interface is crucial for the DMS to succeed in a hostile environment characteristic many (most?) projects.

DMS Execution Engine Requirements

The DMS shall be a meta-data driven execution system in the natural working environment. The DMS engine shall automatically link the relational information architecture into the execution system. The engine shall capture execution data and enter it to the information architecture, thus minimizing manual data entry requirements. The engine shall do parallel processing of all low level configuration management operations without user intervention.

DMS AI Component for Analyzing the Software Process

The DMS shall have a software engineering rules base and an AI component capable of verifying compliance to the rules by matching the rules and the relational meta-data. The rules shall cover adherence both to life-cycle and to contractual constraints.
DMS 2000 shall NOT be based on the 'Value Added Approach'

The current trend of integrating tool fragment .. utilities into packages is useful only for the implementation phase that approaches the 'completely understood and stable domain' stage. Examples of successful implementation phase packages are: UNIX, VAX Set, Toolpack .. APSE ... The value added approach of incrementally building the DMS 2000 does not work, because the concept is based on the DMS engine, also called 'software environment driver', relational database architecture, natural language front end, and background AI component.

Standard Computer Vendor Solutions For DMS Implementation

DMS relational information architecture shall be based on computer vendor standard. Also, the natural language interface shall be supplied by computer vendor. The unique components for the year 2000 DMS shall the DMS engine and the DMS AI component.

Stage of Current DMS Implementation versus DMS 2000

Information Architecture -- Yes, but weak
Natural Language Interface -- None
DMS Execution Engine -- Powerful
AI Background Component -- Yes, but weak

DMS PROJECT EDUSET EDUCATIONAL SOFTWARE ENGINEERING TOOL

The Ultraware Ltd. EduSET project develops an educational software engineering tool with DMS as its kernel. Submissions from other software tools will be considered. Contact Ultraware Ltd. for further information about EduSET.

REFERENCES

