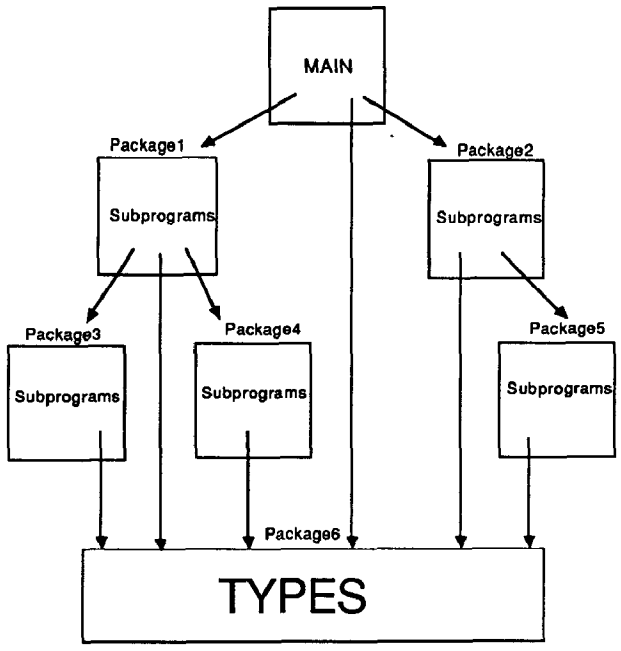
LESSONS LEARNED BY USING DIFFERENT METHODOLOGIES ON FIVE Ada PROJECTS

Mary E. Biddle
GTE Government Systems
Rockville, Maryland

Mary E. Biddle has recently joined GTE Government Systems as a Member of the Technical Staff working in SW Support to Systems Engineering where she is supporting the Ada project, MINSTREL. Before coming to work at GTE, she was employed at Magnavox Electric Systems in Fort Wayne, Indiana where she worked on several Ada projects. Previous to this experience, she was employed at Harris Corp. in Melbourne, Florida where she also worked on several Ada projects.

PROJECT 1

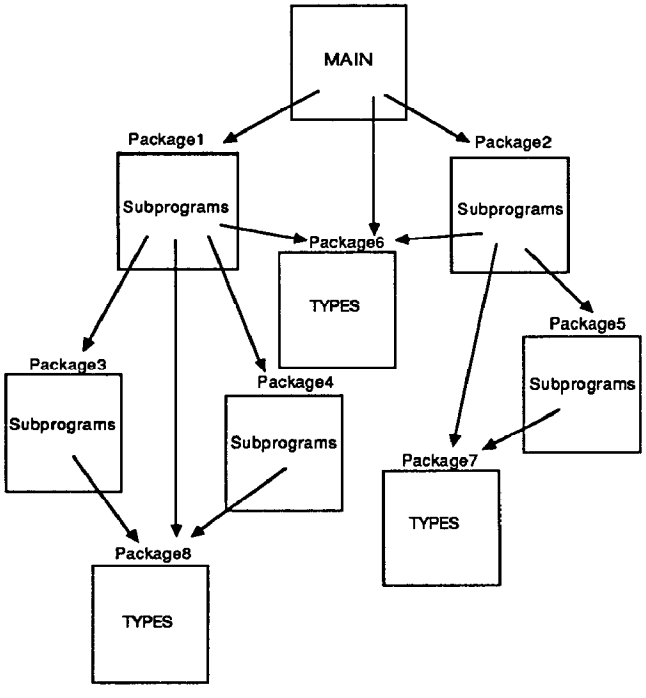
- No Methodology
- Packages formed around Functional Capabilities
- Types grouped into One Large Types Package



PROJECT 1

PROJECT 2

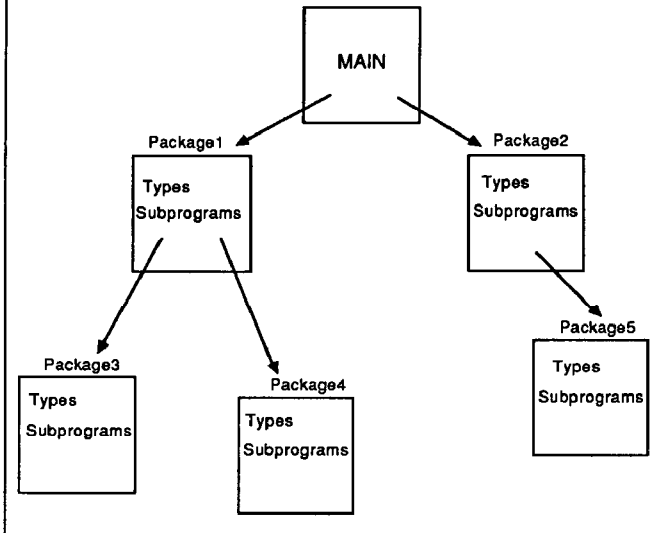
- SA/SD
- Packages formed around Functional Capabilities
- Types grouped into several small Types Packages



PROJECT 2

PROJECT 3

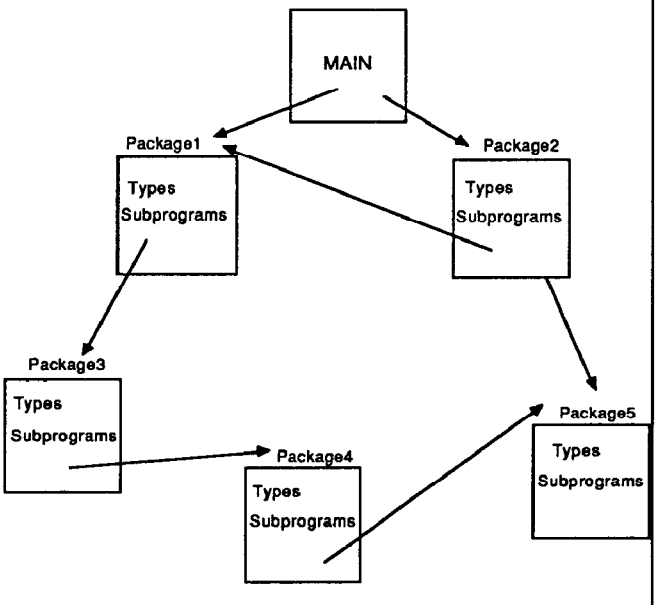
- SA/SD
- Packages formed around Functional Capabilities
- Types grouped into the same Packages as the Subprograms



PROJECT 3

PROJECT 4

- SA/OOD
- Packages formed around Objects
- OOD started after the system was broken into Functional Groups



PROJECT 4

PROJECT 5

- SA/OOD
- Packages formed around Objects
- OOD started after System Analysis

was one instance in which 15 different packages had to be examined in order to debug one line of code. It was very frustrating and not very efficient.

o Types were duplicated quite often. It was much easier to create a new type than to trace through the system to look for the existing type. Not only was a new type created, but in many cases a whole new package would be created.

o Subprograms were duplicated at an alarming rate. The system was much too complex to spend the time looking for the subprograms needed. Instead new subprograms would be created.

4. Project 3

Background. The third project was an EW program resulting in 50,000 SLOC. The target was the Motorola 68020. The host and development environment was the VAX/VMS 8550 with the Verdix Ada Development System.

Methodology. The developers on the third project were introduced to Object Oriented Development (OOD), but were reluctant to use it. They were new to Ada and had difficulty grasping the concepts of OOD. Due to their past experience, the staff chose functional decomposition according to DeMarco as the methodology for the project. Data flow diagrams were used for analysis and structure charts were used for design. The importance of package design was stressed but no common methods were incorporated.

Topology. Due to the introduction of OOD, the types and subprograms were not separated into different packages. The packages were designed around functionality but each package included the types as well as the subprograms. The complexity of the "with"ing structure was far less complex than the two previous projects. (See figure 3.)

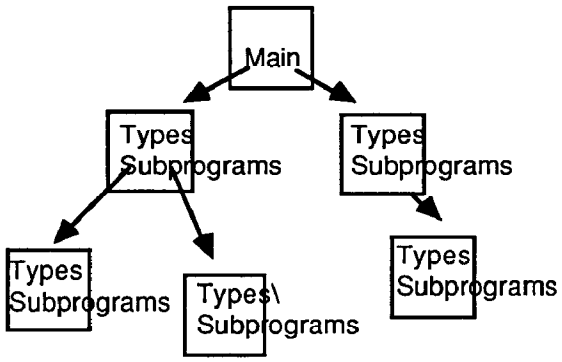


Figure 3. Package Structure for Project 3.

Experiences and Lessons Learned.

o Recompiling was reduced to a minimum. Since packages were formed around functional capabilities, when more than one functional capability needed the same type it would be duplicated in several packages.

o System test and integeration went more quickly with fewer problems than any program of this kind had in the past.

o Types were duplicated. Types that were needed by several logical functionalities were duplicated in several packages.

o Subprograms were duplicated. Since the types were duplicated, some of the basic utilities needed by the types were also duplicated.

o Types collided. Since types were duplicated there were occasions when the two types needed to be the same type. Explicit type conversion could be used for simple types. When compound types were duplicated the process became very lengthy and complicated.

o Circular dependency was also a problem. The use of functional capabilities to design packages often resulted in circular dependencies that required a rework of the design that distorted the original concept.

o During system test and integration it was difficult to trace a problem to the source. The types package was so large that it was difficult to find the appropriate data representation being used by the source. When several data types were used to form a new data type, the problem was compounded. Many hours were spent tracing data. This only increased the problem by making the package much larger.

o Types were often duplicated. When a new function was added which used a type that the developer did not know existed, the developer added a new definition. It was much easier and faster for the developer to add a new type than to look through the types package to see if the definition existed.

o Subprograms were often duplicated. When the developers duplicated a type, they would tend to duplicate the subprogram the type needed as well.

3. Project 2

Background. The second project was an automatic test equipment program resulting in approximately 200,000 SLOC. The DEC compiler was used on the VAX/VMS 8600.

Methodology. After examining the mistakes of the first project, the program developers decided to use a more stringent methodology, Yourdon's functional decomposition methodology. The developers used data flow diagrams to express the analysis phase and structure charts to help determine the design. No formal techniques were incorporated to develop the package design, although it was decided the large types package would not be used.

Topology. The packages were formed based on functionality. Types were added as an afterthought. The grouping of the types into packages was based on need. In one case two CPCIs needed to communicate, a types package was formed. In another case a package needed certain types to communicate with its subordinate packages, so a types package was formed. The structure be gan to look like a spider web. (See figure 2.)

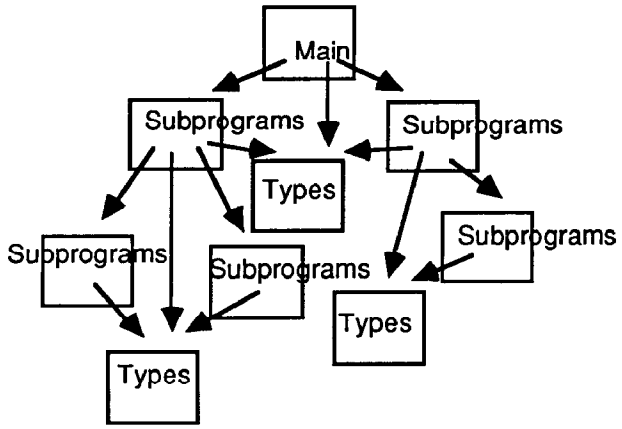


Figure 2. Package Structure for Project 2.

Experiences and Lessons Learned.

o Whenever data changed it was not unusual to recompile large portions of the system. There was much less compiling than with the one common types package, but many packages were coupled through the sharing of the types package.

o During system test and integration it was very difficult to trace a problem. The complexity of the system was increased by the "with"ing structure to the point where it was almost unmanageable. When looking at a procedure in a package it was not unusual to require five or six types packages. When that procedure referenced several procedures in different packages it became too complex to sort out. There