that the client may not even be able to verbalize his or her own needs. The client speaks with fuzzy sentences replete with tacit assumptions, and the software designers are just not able to identify his or her intentions.

Many system design or programming methods, e.g., those of Jackson (Jackson, 1975), Parnas (Parnas, 1972), Booch (Booch, 1986), Myers (Myers, 1979), Orr (Orr, 1977; Orr, 1981), etc., start from an assumed clear statement of requirements and show how to arrive at a design of a program meeting those requirements. However, none of these methods really explain how these requirements are obtained in the first place. It is clear that writing of the requirements is a major part of the problem solution, and that when this writing is done properly, many pitfalls in the path of delivering the required system may be avoided.

Large E type (Lehman, 1980) software, for which it is difficult or even impossible to obtain clear requirements, is usually developed for a client organization in which there are many people who have some view or say as to what the desired system should do. These views range from being deceptively similar to each other through being totally unrelated to each other to being totally inconsistent with each other. It is no wonder that the distillation of these views into a consistent, complete, and unambiguous statement of the requirements, albeit in natural language, is a major part of the problem of developing software which meets the client’s needs. Therefore, it is essential to have methods and tools that help in distilling these many views into coherent requirements.

Full discussions of the problems of obtaining good requirements and of the effect of the failure to obtain them may be found in a textbook by Davis (Davis, 1990) and in a paper by Krasner (Krasner, 1988).

1.2. Brief History of Requirements Engineering Methods and Tools

The early work in requirements engineering was focused on requirements specification and analysis (Davis, 1990). The tools and methods of the time permitted and assisted in the organization of the requirements, refinement of details, consistency checking, preparation of the specification, and in some cases, formalization of these requirements (Alford, 1977; Ross, 1977; Teichroew, 1977; Alford, 1978; Zave, 1982; Burstin, 1984; Alford, 1985; Borgida, 1985; Sievert, 1985; Estrin, 1986). Some of these tools also provided a means to relate design and implementation artifacts to their individual requirements to assist in the eventual validation of the realization of the requirements. The starting point for these tools and methods is a written statement of the requirements, usually a list of sentences, perhaps in a highly constrained subsets of English. However, none of these tools and methods give much help in actually obtaining the sentences in the first place and in recognizing the relevant abstractions, especially in the context of a large client organization. Methods and tools are needed to assist in getting sentences, writing them down, and finding the abstractions from which the requirements can be written.

More recently the software engineering community has been paying attention to the problem of eliciting the raw information from the clients (Lubars, 1993). Some of this work, e.g., contextual inquiry (Holtzblatt, 1993), has focused on observing the client’s