At the stage of problem analysis in the software development life cycle, the problem set is defined in terms of software tools and development requirements by the software development team. Computers have long been employed for problem analysis and the compilation of big data. Analytical AI systems are more comprehensive in the complexity of statistical approaches and dispose of embedded self-reliant learning algorithms that distinguish patterns based on a series of similar or recurring characteristics to enable new creative solutions AI analytics takes recourse to external data bases to get “informed” and further develop established routines. AI analytics is partly applied in at the stage of problem analysis in software engineering e.g. to predict of project success and risk an essential routine required to assess and select prospective software projects. In the design phase, the software project is clearly structured and development tasks are assigned.

The code will gain in complexity in the process of neural network processing and will not have to be understood or reviewed by human beings any more. Visual recognition, speech recognition, speech synthesis, self-driving car routines and gaming are early manifestations of self-improving and developing program codes.

Neural networks have been developed to assist software coding: Processing natural language into software code is a capability of which has been researched since the 1980ies and with increasing complexity of pattern recognition routines has advanced to class-model builders in recent years . Gathered data are transformed into contingent vectors and are used for model training to interconnect code levels systematically

**Knowledge Representation and Processing**

The system maintains multiple layers of knowledge and has the ability of self-learning and self-improvement. Especially, the system can do reasoning on incomplete information. Otherwise, the system cannot 'figure out' users' intentions and the interaction between a user and a tool will not be able to continue in some cases. Each node can be bound to a set of rules. Mixed knowledge representation is good for reducing the size of the network, and thus speeding up the learning process. In neural network, an output can always be derived from any input, even if the input is incomplete. Self learning is a natural and standard process in a neural network.

**Visual Integration of CASE Tools**

Although a CASE shell is very useful for the creation and integration of CASE tools, we believe it still is not simple and intuitive enough for CASE users. We suggest to provide CASE users with more intuitive means to describe the integration of CASE tools. A diagramming tool, similar to DFD diagramming tool, might be useful for this purpose. Pipe-filter, event-trigger and message passing models are all useful means to realize the integration of CASE tools.

**Work Flow Model**

Basically, software project development is a team based activity. A CASE tool should be able to support this feature. For this purpose, it is necessary to model the work process, and the collaboration and coordination among team members. We hope to represent all these information through a work flow model. Currently, there are many ways to model a work flow for team based collaborative software development. However, most of them are too strict to change dynamically (on the fly). We suggest using decentralized process models, such as 'ViewPoint' **(E)**, which can be described visually and is also possible to cope with deviations during process enactment.

In the maintenance phase the software company assists the customer in product application, provides regular upgrade and makes further adjustments upon client requirement.

AI instruments successfully support the maintenance and updating process of software to changing requirements in an internet environment.