# PLUG: An Agent Based Prototype Validation of CAD-Constructions

S. Baumgart[1], B. Toledo[2], K. Spors[3], M. Schimmler[1]
[1] Institute for Computer Science, Christian Albrechts Universität zu Kiel, Germany, {sba, masch} @informatik.uni-kiel.de
[2] gedas Deutschland GmbH, Germany, Begona.Toledo@gedas.de
[3] Volkswagen AG, Germany, karin.spors@volkswagen.de

## Abstract

*A new approach for an automatic consistency check in the development of prototypes in the automotive industry is presented. It is based on the observation that inconsistencies between adjacent parts of the prototype can be located at the plugs connecting these parts. This paper describes the PLUG software system whose idea is to assign intelligence to every plug in the system in order to continuously monitor the consistency of the overall design of the prototype. It has been implemented as a multi agent system where each agent is responsible for supervising a specific property of the parts adjacent to such a plug. The properties are structured according to the so called PLUG structural model which is a simple but powerful set theoretical representation of the interaction between adjacent parts. The PLUG project is a research project financed by the Volkswagen AG.*

*Keywords:* CAD, Multi Agent System, Validation and Verification, Plugs

## 1. Introduction

Today's automobile industry faces the chances and risks of globalisation [2]. More than in earlier years, the development time and the costs for prototypes influence the success of the final product in the market [1]. Therefore, the design of such a prototype must be a highly concurrent process. A natural problem in such a concurrent environment is the consistency of the design [3]. A change in one part of the design implies changes in other parts. Unfortunately, there is no support for an automatic consistency check of plugs in the commercial CAD tools used in the automotive industry [12].

This problem is addressed by the PLUG multi agent system which is described in this paper. The basic idea is to locate the possible occurrences of inconsistencies. It turns out that these locations can be identified with the contact surfaces between adjacent parts. These contact surfaces can be represented by the plug elements (bolts and nuts, welding points, gaskets, etc.) used to implement the connection between these adjacent parts. Therefore, the idea of the PLUG system is to assign "intelligence" to every plug in the design [4,13], such that the plug is able to check the consistency of its corresponding contact surface. By the term "intelligence" in this case a piece of software is meant which is able to evaluate the current state of the CAD and to point out technical inconsistencies [8].

The first step to such an assignment is the development of a generic model of the prototype under design. The PLUG structural model takes a set theoretic approach: All parts are classified as functional parts, connection parts, or connection elements according to their role in the overall design. The interaction between the different parts can then be abstracted in a graph oriented way: Every part is a vertex in an undirected labelled

graph, where the contact surfaces are represented by edges. With this model, the whole problem can be reduced to do the consistency check for every vertex for a connection element in the graph. This can be implemented very elegantly by using a multi agent system [5,11,14,15]. There is one agent for every property of the corresponding contact surface. Other agents are required for administration and for interchanging information between the "leaf agents" and the agent monitor [6].
The multi agent system provides a natural solution for the required concurrency and it is adaptive to any change in the specification due to the option to add further agents if possible. It has been implemented in Java [9,10].
The paper is organised as follows: Section 2 gives an overview over the concept of the PLUG system. The structural model is introduced in Section 3. Section 4 provides details of the implementation as a multi agent system. An Example of a gear box adaptor is given in Section 5. A discussion of the problems in the current implementation concludes the paper in Section 6.

## 2. The PLUG Concept

A typical design process consists of three phases [1, 15]. Phase one is the conception and the computer aided design of the constructional element. Phase two is a first human based validation of the constructional element and its neighbours. This validation is done at industry leaders like VW by using virtual reality [3]. The third phase is the production of a prototype and its physical and mechanical validation. This third phase is very expensive in time and money. The only way to avoid multiple iterations between all three phases is to automatically assist the designer during design and validation.

An analysis of the connections between constructional elements shows that there are common characteristics between all elements. The concept of PLUG is to abstract this common characteristic into a general structural model. The structural model generates a graph for each prototype under design. A vertex in this graph is a constructional element and an edge describes a plug between the neighbouring constructional elements and their attributes. Each characteristic is a single attribute. By comparing all attributes a full validation can be performed.
The PLUG software system implements the validation process. This process is characterised by concurrency, autonomy, reactivity, and interactivity. These are typical properties of agent based software systems.
The PLUG multi agent software uses the agent construct for each edge of the validation of the graph of the structural model. Thus, it replaces the human designer validating the construction process by hundreds of designers (agents) without further costs.

## 3. The PLUG Structural Model

The whole verification process of the PLUG multi-agent system bases on the plug structural model. Analysing the relations and functions of constructional parts, they can be classified into three classes:
The first class consists of basic assemblies like screws, bolts, nuts, and gaskets. Their function is to act as a so-called connection elements (ce). They connect other parts, like the gear box and the motor or the wheel to the steering knuckle; in the automotive industry most of these parts are also called norm parts as they are internationally standardised.

The second class consists of constructional elements connecting other parts with or without the usage of connection elements. These assemblies are called connection parts (cp). An example of a connection part is a hydraulic hose or an adaptor connecting two parts. The hose clamp in the first case would be the connection element.
The third class of constructional elements are all other assemblies. These are called parts (p). Parts are represented as vertices in the PLUG structural model, where the shape of the vertex indicates the different classes: A rectangle is a part, an ellipse is a connection part, and an Octagon is a connection element.
An example of this graph theoretic representation of a gear box adaptor is shown in Figure 1. The gear box and the car body are parts, the gear box adaptor is a connection part, and the required bolts and nuts are contact elements.

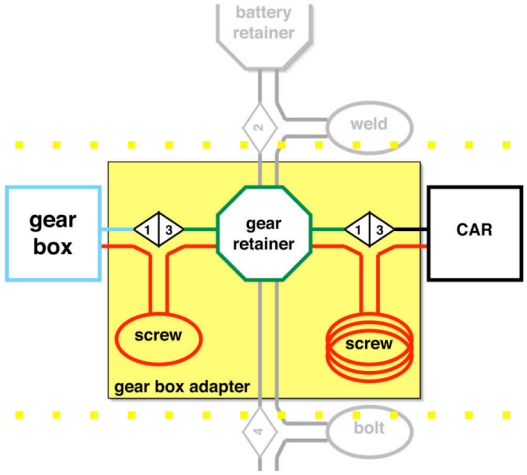


*Figure 1: Excerpt of the Graph representation of a gear box adaptor.*

The edges between these vertices represent the contact surfaces. Every contact surface is a union of attributes describing the properties of the connection. The attributes provide the elementary information for the consistency check. Attributes can be on the one hand elementary like a position in the space, a normal vector, the size of a drill, or non geometrical information as material or weight. On the other hand, they can be composed like polygons, planes (consisting of a normal vector and a position), drills (consisting of a centre point, a diameter, and a depth). For every contact surface there is a fixed number of relations between attributes or the adjacent parts.
Sometimes the attributes to be matched are not required to be exactly identical. For this purpose, the relations between attributes can have constraints like tolerances or approximation parameters.
The set theoretical interpretation of the PLUG structure model allows a simple but powerful depiction of attributes of a contact surface. Figure 2 shows an example for this. Two adjacent parts are represented as their sets of attributes. There is a subset of these attributes that have to be matched in the consistency check. This subset is exactly the intersection of the two attribute sets (the attributes consisting to both attribute sets).

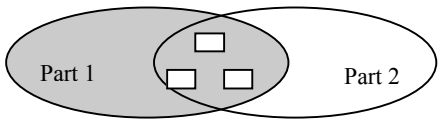


*Figure 2: Set representation of two adjacent parts*

Another key feature of the plug structural model is the hierarchy in its graph theoretical representation. A contact surface can consist of one or more sub contact plains. For example the inner surface of a drilling in a steel plate is a sub contact plain. If a screw with the right diameter is placed inside the drilling the

screw and the steel plate share multiple surfaces. Each of those surfaces is a separated sub contact surface.

## 4. Implementation as a Multi Agent System

PLUG is a ***m**ulti **a**gent **s**ystem* (MAS) for validation of constructional elements within the computer aided construction process. A multi agent system is a system composed of several autonomous, reactive programs. It is capable of reaching goals that are difficult to achieve by monolithic conventional software systems.

The agents of the PLUG multi agent system continuously inspect the constructional elements while the designers manipulate them. The software is able to detect an error in the moment it occurs in the construction process. Sets of agents forming a collision check engine verify the fitting accuracy of the assembly. Others check all members of its class and the relations within the class defined by the PLUG structural model.

PLUG works on the basis of the structural model described in Section 3. For every contact surface there are several agents checking the different attributes. The structure of PLUG reflects the idea of the structural model of the prototype under design. Specified in UML, the ***u**nified **m**odelling **l**anguage*, and implemented in Java (®*SUN*)[10], PLUG strictly follows an object oriented software engineering concept.

PLUG consists of three basic program elements. The first element is the *data core*, storing the constructional elements being parsed from the computer added design (CAD) system. Founding on the industry proved *Open-Inventor 2.0* format PLUG is not restricted to a single CAD-Software system. A graphical user interface enables the user to add or modify additional attributes like the type of the connection or the geometrical data. In contrast to a conventional agent based software system the PLUG multi agent system uses the centralised *data core* to eliminate redundancies in the description of the design and thus minimises memory usage. This is necessary because in a typical prototype constructional elements and norm parts consume a great amount of memory (> 150 MB).

The second element of the PLUG multi-agent system is a special agent named *gatekeeper*. The central function of the *gatekeeper* is to encapsulate the *data core* from the other agents. By substitution of this agent a databank system can optionally be added as a replacement for the *data core*.

The third element of the software is a collection of agents for parsing, analysing and verifying the constructional elements. The agents of the PLUG multi agent system are organised in semi-hierarchical layers. All agents act in a special runtime environment, the so-called *agent universe* like in classical agent designs. Figure 3 shows the structure of the PLUG multi-agent system. Each agent is implemented on top of a special framework. It encapsulates the inner layers of the agent and provides some basic functions to the environment.

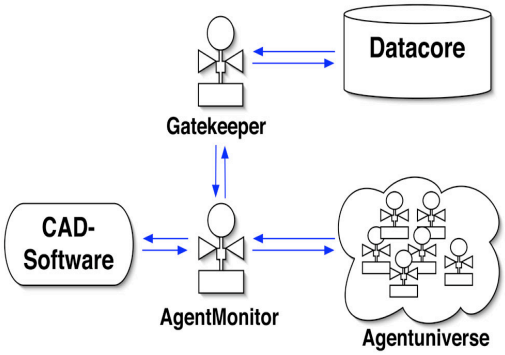


*Figure 3: Structure of the MAS*

Following BDI-model (**b**eliefs, **d**esires, **i**ntensions) each agent of the PLUG software is implemented onto a framework ensuring a common structure of all agents. For example all communication and sensory functions are encapsulated within the sensor classes. A head class holds the memory and learning abilities. A body class holds all work specific functions. The head chooses the most capable function for the special working environment.

PLUG uses a special agent named *agent monitor* to control the other agents. The *agent monitor* is able to initialise, start, stop, and even kill other agents. The insertion of the *agent monitor* as a coordinator allows the other agents to work more efficiently within their verification space.

Every core aspect of the PLUG structural model requires a specific sequence of agents to verify the attributes associated with the core aspect.

One of the most important aspects is that of the contact surface. This has already been discussed in chapter 3. The *AgentCP* verifies these contact surfaces as identified in the structural model. The *AgentSCP* verifies sub contact surfaces.

The verification of a match between two constructional elements is done on the level of contact surfaces. All CAD data are stored in the *data core.* Every shape is approximated by 3-point facets due to the Open-Inventor format. Each sub contact surface consists of one or more facets. Each facet is verified by an individual instance of the *AgentSurfaceCheck*. This set of three agents realises the whole check on the surface and facet layer. All these agents can work concurrently and multiple instances enable the PLUG multi-agent system to check complex constructional elements with more than 5.000.000 facets in less than a minute with a memory consumption of only few megabytes.

Another key aspect in the verification of assemblies is the verification and detection of drill holes and their inverse like screws or bolts. This feature recognition is realised by an *AgentDrillCheck*. With support of the *AgentPolygonCheck* and several other logical agents, checking the material and additional attributes, it is enabled to recognise functional drills from none functional drills. Functional drills are drill hole necessary for the connection of assemblies. None functional drills are drills to reduce the weight of a constructional element or to reduce the material.

To illustrate the functional and operational sequence within the PLUG multi-agent system here an exemplary sequence is discussed:

A designer has constructed a set of constructional elements. These elements are parsed in the background and imported into the *data core*. The *gatekeeper* informs the *agent monitor* of the incoming assemblies. The *agent monitor* notifies all agents necessary for the verification. The communication between the agents is realised by notification flags as usual for agents in KQML (***K****nowledge* ***Q****uery and* ***M****anipulation* ***L****anguage).*

Each notification flag stand for a special attribute of the constructional element, so that there are only agents alive and working which are necessary for the verification of the actual assembly. If an agent needs data form the *data core* it notifies the *agent monitor*. The *agent monitor* fetches the data form the core using the *gatekeeper* agent and transfers data back to the querying agent. This access is performed concurrently and asynchronously. After the completion of

the partial verification each agents notifies the *agent monitor* and quits if possible. The *agent monitor* informs the user about the result of the verification. The *AgentHumanInterface* reprocesses and purifies the information in a readable way and presents the result in three different ways in a GUI (***G**raphical **U**ser **I**nterface*). One window shows the geometrically results of the validation, a second window shows the contact-plains and a third window shows the graphical PLUG structural model (Figure 6).
In an ideal scenario all verifications of contact surfaces and drill holes happen concurrently. In some scenarios combinations of constructional elements with dependencies exist so that there is only a sequential solution during verification. The agents memorise such constellations and try to avoid them if possible. Globally, the PLUG multi agent system prefers always a concurrent verification sequence.

## 5. Example

Figure 4 shows the function of the PLUG multi-agent system for the gear box of a VW Phaeton model.

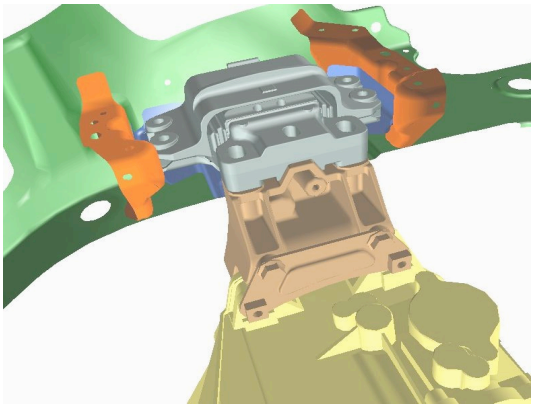

*Figure 4: Gear box adaptor for a VW Phaeton*

It corresponds to the structural model depicted in Figure 1. The gear box is marked in bright yellow and the gear box adapter is marked in bright grey. An individual agent verifies each plug within the model.

Figure 5 shows the parts in the CAD models. A part of the gear box is shown on the right side the gear box adaptor on the left side.

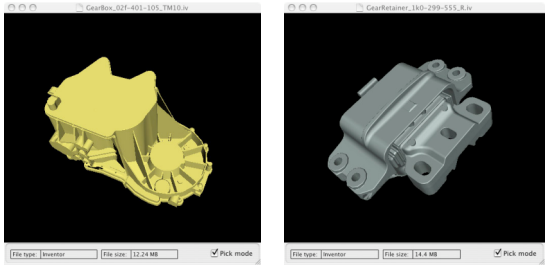

*Figure 5: Gear box and gear box adaptor for a VW Phaeton in CAD system*

The PLUG multi-agent system inspects and verifies all contact surfaces. The result is shown in Figure 6.
The runtime for this small example is less than a second for more than 250.000 polygons.

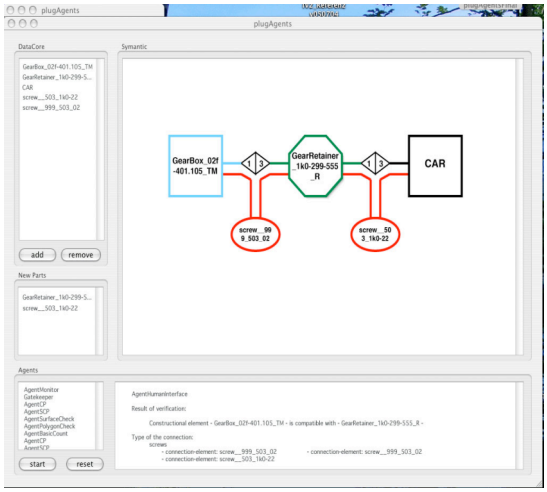


*Figure 6: GUI of PLUG system*

## 6. Conclusion

In this paper we presented the PLUG multi agent system. It implements the concept of an automated agent-based validation system for computer aided prototype development. The agent-based approach has proved as an adequate software method for solving this particular problem. The PLUG system is able to validate and verify a prototype of high complexity with reasonable time and memory requirements.
The development of the PLUG multi agent system created new questions and problem areas witch are not solved yet. The main problem is parings the CAD-Data and the feature recognition. As circles are interpolated by polygons within classic industry proved CAD-formats drills and bent or curved surfaces have to be approximated. This results in an approximation error causing wrong results in the verification process under extreme conditions. Another problem is the number of possible representations generated by exporting data into a polygon based format. Each export results in a different combination of polygons and differs between different CAD-Software systems. For plain surfaces and sharp angles these problems have been solved, for circles and bent edges approximation algorithms could be implemented solving over 90 % of the existing cases. Complex formed surfaces, like waves, are not solved satisfactorily yet.
A final solution might be a total integration within the CAD software with access to the internal vector based exposition of the assemblies losing platform and CAD-Software independency. These improvements are to be tackled in a future project. For a commercial product further improvements are required, especially for a solution of the problems discussed while parsing (transformation of the CAD-data into the internal data format).

## References


[1] Anderl, R.: Parametrics for Product Modelling. In: Hoschk, J.; Dankwort, W. (Hrsg.): Parametric Variational Design. Stuttgart: B. G. Teubner Verlag.

[2] Anderl, R.; Mengden, R.: Modelling with constaints: theoretical foundation and application. Computer-Aided Design 28 (3). 1996

[3] Bauert, F.; Weise, E.; Salem, N.: Modellierungsmethoden für Systeme zur rechnergestützen Gestaltung. Konstruktion, 42.1990

[4] Baumgart S.: Agentenbasierte Entwurfsvalidierung bei CAD-Konstruktionen. Bericht aus dem Kolloquium „Agentenbasierte Systeme in der Industrie“. Volkswagen Auto Uni und Gedas GmgH. Wolfsburg 28.11.2005

[5] Burkhard, H.D.: Software-Agenten. Hrsq. Görz, G., Rollinger und C.-R., Schneeberger, J.. Handbuch der Künstlichen Intelligenz. Oldenburg Wissenschaftsverlag GmbH 2003

[6] Conrad, T.: Agent-Oriented Software Engeneering for Internet Applications. FU-Berlin, 1995

[7] Dellschaft, K.: Softwareagenten: Theorie und Praxis: Agentenarchitekturen. Seminararbeit, Institut für

Informatik Universität Koblenz, 12 2002.

[8] Erman, L.D.; Hayes-Roth, F.; Lesser V.R.; Reddy D.R.; The Hersay-II Speech Understanding System: Integrating Kowledge to Resolve Uncertainty. Computer Surveys, 2(2), June 1980

[9] Görzig, S.: Eine generische Software-Architektur für Multiagentensysteme und ihr Einsatz am Beispiel von Fahrerassistenzsystemen. Sharker Verlag, 2003

[10] González Ordas, J.: Desarrollo de una Herramienta Case Orientada a Agentes en Java. ETS Ingenieria de Telecomunicación, Universidad de Valladolid, 1999

[11] Iglesias Fernández. C. A.; Garijo, M.: González, J. C.: A Survey of Agent-Oriented Methologies. ESPRIT Basic Reasearch Prodject MIX: Modular Integration of Connectionist and Symbolic Processing in Knowledge Based Systems, ESPRIT-9119 (1999)

[12] Toledo Muñoz, M. B.; Spors, K.; Bracht, W.; Schimmler, M.: Agenten basierte Modellierung und Analyse von Verbindungen im Produktentstehungsprozess. Zeitschrift für wissenschaftliche Fabrikplanung (ZWF) Jahrg. 100 (2005) 6, S.319-323.

[13] Toledo Muñoz M. B.: Agentenbasierte Modellierung und Analyse von Verbindungen im Produktentstehungsprozess, PhD thesis, Fakultät für Mathematik / Informatik und Maschinen der Technischen Universität Clausthal, 2006

[14] Tveit, A.: A survey of Agent-Oriented Software Engineering. Norwegian University of Science and Technology. First NTNU CSGSC, May 2001.

[15] Wagner, T. Göhner, P.; Urbano, P. G. de A.: Softwareagenten-Einführung und Überblick über eine alternative Art der Softwareentwicklung. Part III: Agentensysteme in der Automatisierungstechnik: Aufbau, Struktur und Implementierung an einem Anwendungsbeispiel. ATP – Automatisierungstechnische Praxis 45 (2003) no. 46