Int J Adv Manuf Technol (2007) 35:15–25
DOI 10.1007/s00170-006-0704-2

ORIGINAL ARTICLE

An agent- and service-based collaborative design architecture under a dynamic integration environment

H. B. Qiu · X. Y. Shao · P. G. Li · L. Gao

Received: 24 October 2005 / Accepted: 19 June 2006 / Published online: 6 October 2006

Abstract The shortening of product realization times and the increase of product complication require multidisciplinary experts to cooperate dynamically and closely in the whole product development lifecycle, behavior that has been previously unfamiliar. To further improve the design cooperation efficiency under this environment, this paper presents a novel cooperative design architecture using revolutionary Internet, agent, and semantic Web technologies. Based on the concept of service provision, intelligent Web-agent groups act as the function backbone to organize the service-based running of the system. Multidisciplinary product development units can be dynamically formed and configured in a knowledge-integration perspective. The layer-based architecture encapsulates distributed knowledge at a higher abstract level and coordinates lifecycle activities from different domains. Semantic-rich service expression and intelligent service accessing through agent mediation make the system more flexible. Within this dynamic cooperative environment, all of the participants involved and their distributed knowledge can be integrated as a whole to achieve more efficient and more intelligent design cooperation.

Keywords Collaborative design · Web service · Semantic Web · Injection mold

H. B. Qiu (✉) · X. Y. Shao · P. G. Li · L. Gao
School of Mechanical Science and Engineering,
Huazhong University of Science and Technology,
Wuhan, Hubei, People's Republic of China
e-mail: hobqiu@yahoo.com.cn

1 Introduction

In today's highly competitive market, product development times have gradually become the most important factor for achieving product success. While at the same time as the constantly increasing requirements for product variety, cost-effective solutions and customer satisfaction have made the product development process more and more complex. Under this environment, individual companies, especially the small- and medium-sized companies, no longer need to have the breadth of knowledge and capability to understand all of the aspects along the product realization chains. Therefore, strategic alliances that share knowledge/expertise and hardware/software resources, and that could cooperate more tightly in product development processes are much needed, as are the research of architectures and technologies concerned with product development collaboration under these conditions.

Confronted with this dynamic and distributed collaboration requirement, this research uses multi-agent theory and state-of-the-art semantic Web technology to construct a novel scalable environment that can cover the whole product development lifecycle for collaborative injection mold product development. Some key issues of this environment are also discussed and a prototype system is developed to give practical validation of the theory.

Web technology and agent technology, as two of the most prominent technologies, are widely adopted to develop numerous distributed cooperative design systems. Madefast [1] was an initiative program that facilitated communication and collaboration in the cooperative design Web space. Web-related technologies were also used in

DFX [2], product data management [3], and early supplier involvement [4] for concurrent design participation. At a much higher level, some Web-based projects were accomplished to achieve distributed knowledge integration in collaborative design. Zhou et al. [5] proposed a distributed product-oriented knowledge source provider. The means for knowledge representation, acquisition, and utilization in the Internet collaborative design network were given in [6, 7]. Chin et al. [8] presented some methods for the efficient search of distributed knowledge resources. A Web-based platform-independent software framework to integrate software resources into an open design/manufacturing system was developed in [9]. Kulkarni et al. [10] used XML to facilitate the effective management of distributed resources. Furthermore, dynamic resources were also integrated in the form of loosely coupled services in collaborative design to leverage more system flexibility. Wang et al. [11] utilized service type agents to realize a Web/agent-based collaborative design environment. Web services were preliminarily used in PLM [12] and CPC [13]. Aziz et al. [14] employed the semantic Web initiative data formats RDF in a decentralized design environment, but they did not consider them from a service perspective.

On the other hand, the agents used in distributed collaborative design often took the view of autonomy and interaction. The topics mainly covered include the interaction and control of systems [15], system interoperability [16], knowledge collaboration [17], and conflict management [18]. They can better supplement the Web in an intelligent and coordinated way.

In the area of die and mold design, most papers focused on specific intelligent design support expert systems [19]. Some researches have been done on the collaborative product die and mold development. Tang et al. [20] built a Web-based environment that can support integration between die suppliers and product customers. Concurrent stamping part and die development systems were developed from the angle of knowledge sharing and tool integration [21, 22]. With regards to the injection mold, Lee et al. [23] presented a preliminary collaborative environment integrating much domain knowledge, but not in a distributed way. The information management [24] and concurrent analysis [25] of Web-based mold collaborative design were also discussed. Chung and Lee [26] used XML and CORBA to evaluate the interaction of mold design. SPEED [27] was developed to facilitate the sharing of injection-molding information between interested parties via the Internet.

As discussed above, considering the situation from the point of view of knowledge resource integration and collaboration, the previous methods and models, especially those used in mold product collaborative design, still lacked some flexibility; thus, they cannot adapt to the dynamic nature of today's design collaboration environment. Though a few researchers have used Web services in collaborative design, the attention was focused on the deployment and interoperation of services. Because of the passive role of Web services, the active coordination and intelligent integration of these services remain challenging problems. Additionally, with the lack of formal semantic service information, these integrations will be far away from real flexibility. So, the combination of agent mediation and semantic Web support in this paper provides a new possibility fo this problem which has not been considered previously by other researches. Under this architecture, knowledge resources distributed over the Web can be efficiently discovered, configured, and integrated into the system. It can better accord with the philosophy of global manufacturing collaboration.

The paper is organized as follows. Section 2 introduces our approach from a service-oriented view. Section 3 presents the agent-coordinated system architecture over the Internet. Section 4 discusses some key methods and strategies used for dynamic service management. Section 5 gives some brief implementation issues and presents a case study. The last section concludes this paper.

2 Service-oriented dynamic collaborative product development environment

2.1 Service perspective in the product development collaboration

Generally speaking, mold product development includes activities of injection part design, injection process design, mold design, and mold manufacturing. The experience from traditional sequential mold development processes shows that design iterations, especially those from different disciplines, are one of the most important causes leading to the prolonged development time, and the reason for these iterations are mostly from the lack of information or knowledge interaction among different designers or design activities. Collaboration has become the pivotal point for integrated mold product development, and the nature of this collaboration is the acquisition of services, i.e., the appropriate resources and knowledge, which should better be open, scalable, customizable, and reusable.

In this paper, we proposes a new methodology called Collaborative Injection-mold-product Development Virtual Space (CIDVS), an integrated system that enables the agile, flexible, and dynamic composition of resources, and permits their interaction in a variety of styles to match the current and changing needs of design objectives. CIDVS spans across the boundaries of collaborative product development enterprises, as well as different disciplines, and tries to completely integrate the customer, mold part supplier, plastic part designer, mold designer, injection analyzer, part former, and the mold manufacturer in order to achieve a concurrent and timely collaboration in the time and space dimensions.

With the rapid growth of reusable online resources, especially the widely deployment of Internet Web services, a global-service-based vision gives a fresh view of distributed collaboration. As an innovative paradigm, a Web service exposes the standard XML interface, communicates with the standard protocol, and can be located through a registry, so Web-service-based applications feature the characteristics of being loosely coupled, communications-oriented, and platform-independent. CIVDS is built upon Web service technology to form a collaborative environment for the active involvement of dynamic participants and effective service delivery. Additional semantic information of the service is also added to facilitate service coordination.

Design collaborations here are viewed as service-oriented. The service in this framework is a generalized concept, which can be defined as "a well-defined, self-contained modular process that provides a functional use for a user, an application program, or another service in the system." Thus, everything can be seen as a service, from general CAD data retrieval to a specific injection process analysis. Under the functional perspective, it can be grouped into different categories, including collaborative service, system service, domain service, etc. Though these services are distributed in different places and on different platforms, they can work together efficiently to achieve the overall optimum design goal, enabling enterprise-wide autonomous design participants to share information and knowledge, collaborating and coordinating their activities within the context of a design project.

2.2 Service-supported collaboration throughout the mold product development chain

The workflow of CIDVS-supported collaborative mold product development is shown in Fig. 1. The overall development process can be viewed from two aspects, i.e., horizontal product realization process and vertical decision iteration process. The horizontal view represents the macro process that advances gradually throughout all of the product development stages. It is driven by a process optimization method [28] and different collaboration support.

The vertical view contains many microprocesses accelerated by knowledge interaction from different disciplines. Every such process happens at a certain time point of

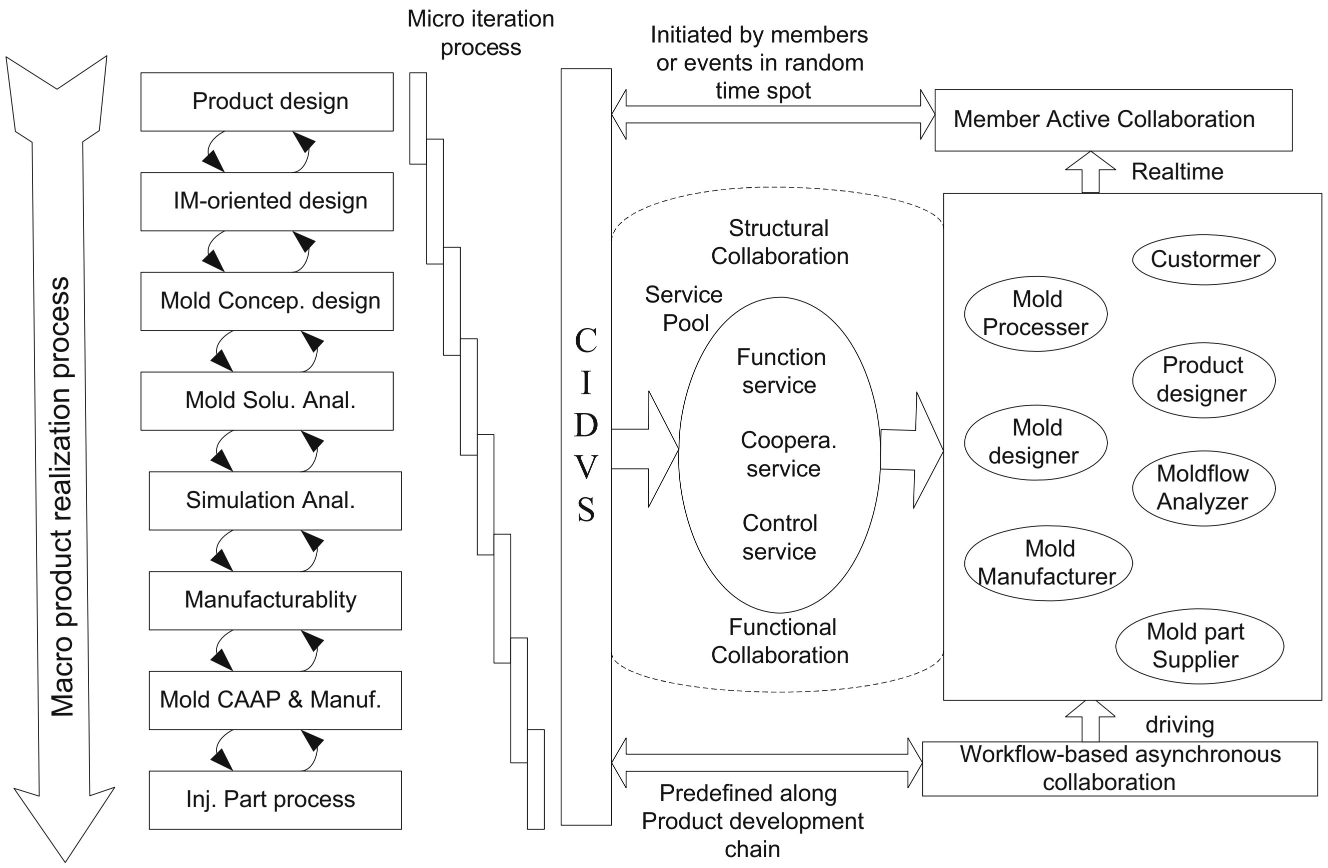


Fig. 1 Process flow and composition of the Collaborative Injection-mold-product Development Virtual Space (CIDVS) methodology

product development in the form of active and passive participance of all concerned parties and services to achieve intelligent design support. Three-dimensional cooperation and real-time feedback facilitate the synchronous collaboration. Design review and service integration act as the main means to support asynchronous collaboration, on which this paper mostly concentrates.

A service pool serves as the core for the service integration support. It is composed of various kinds of services. From the functional point of view, collaborative services and system services are needed to improve collaboration among distributed groups, endorse knowledge sharing, and assist in correct decision making. These services can be seen as ‘static’ services because they are always deployed in fixed server places as functional components, seldom changing, except when self-updating. While, on the other hand, domain services are intelligent Web-accessible tools that can provide single-discipline-related design services, such as injection process analysis, cost analysis, moldability assessment, manufacturability assessment, etc. These services can be developed by different experts and be distributed in different locations with different knowledge involved, so they are relatively more dynamic in nature. We define them as ‘dynamic’ services that need discovery, registration, configuration, and invocation. They can be integrated into CIDVS in certain project contexts to achieve semantic-rich service collaboration.

Under the CIDVS environment, the virtual service network is dynamically formed to provide service support for collaborative design project. Services communicate in a peer-to-peer and open-standard paradigm, and use the concepts of discovery and deployment to publish their capabilities or attributes, regardless of their geographic location. But they are represented only in a non-organized manner; it is more natural to provide a middle layer for better coordinating the service group behaviors. So, agent technology is used to coordinate and encapsulate these services, aiding collaborative design in a more efficient way.

3 Web-enabled multi-agent architecture of CIDVS

3.1 Agent integration and cooperation through the Internet

CIDVS can be regarded as an intelligent distributed system with the aim of accomplishing a given task through cooperation among multidisciplinary autonomous design units. Knowledge should be integrated to support the decision making of these nodes. Although Web service technology alone can provide a loosely coupled architecture and effective implementation-level means for knowledge integration, they provide no mechanisms to facilitate the problem-solving capabilities of distributed entities, which usually need coordination, collaboration, and negotiation.

The multi-agent method is naturally used. As an independent, intelligent, and socialized entity, an agent is inherently distributed and scalable; it can better adapt to the flexibility of collaborative product design. Traditionally, agents have mostly been used for supporting co-operation among designers, providing the semantic glue between traditional tools or allowing for better simulations. But in our approach, the main function of agents is to enable effective service delivery and achieve optimum service provision. Agents bring the most crucial capability to turn an entire set Web services from the existing dormant mass of information where users need to surf and browse into a dynamic set of capabilities deployed around the world and serving the users.

An agent is not designed to realize overall design automation, but instead to facilitate the concurrency of processes and the cooperation of participants, aiming at providing knowledge support for design decision making. Each agent encapsulates corresponding operations and interfaces, exhibiting different capabilities in different contexts. A predefined semantic protocol is used for the interaction between agents to achieve harmony of the entire cooperative design process.

The proposed agents can be classified into three categories: the discipline-dependent application agent, the coordinate-oriented collaboration agent, and the distributed user agent. A user agent has more units included, together with additional dynamic service management functions. A collaboration agent does not contain the user interface part, but has more complex business logic, corresponding to the encapsulation of a static service. An application agent has the simplest structure and only provides a communication mechanism above the Web encapsulation layer. These agents serve as the mediator and encapsulator for local and distributed services, further facilitating the efficient cooperation of knowledge-supported decision methods.

3.2 Hierarchical flexible architecture of CIDVS

The CIDVS architecture is highly flexible in various aspects in order to be more adaptive in virtual environments (VE) based collaborative design environments. On one hand, services make it dynamic from a physical or implementary point of view. On the other hand, agents make these dynamic services configurable logically. The overall architecture can be seen from three deployment layers: a product development unit (PDU) layer, a collaborative function layer, and a resource application layer. A detailed description is illustrated in Fig. 2.

The top layer is the PDU layer. It is composed of several PDUs, each of which contains a development user, a user

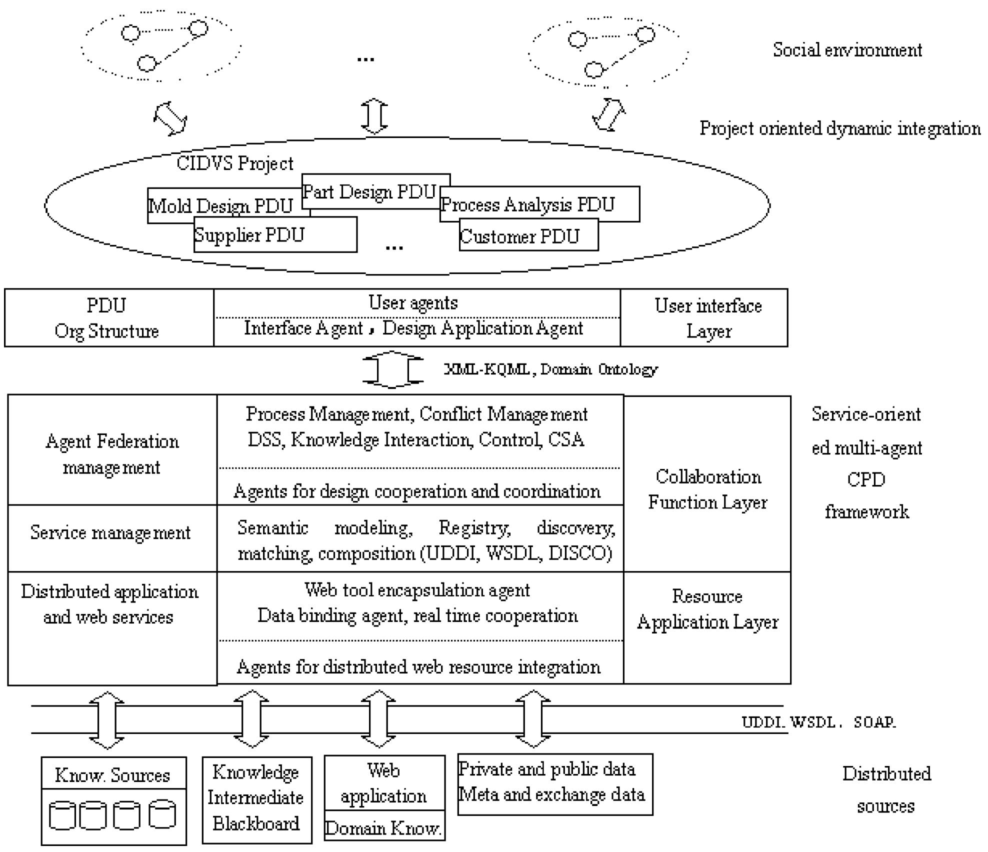

Social environment
Project oriented dynamic integration
CIDVS Project
Mold Design PDU
Part Design PDU
Process Analysis PDU
Supplier PDU
Customer PDU
PDU Org Structure
User agents
Interface Agent , Design Application Agent
User interface Layer
XML-KQML, Domain Ontology
Agent Federation management
Process Management, Conflict Management DSS, Knowledge Interaction, Control, CSA
Agents for design cooperation and coordination
Collaboration Function Layer
Service-oriented multi-agent CPD framework
Service management
Semantic modeling, Registry, discovery, matching, composition (UDDI, WSDL, DISCO)
Distributed application and web services
Web tool encapsulation agent
Data binding agent, real time cooperation
Agents for distributed web resource integration
Resource Application Layer
UDDI. WSDL. SOAP.
Know. Sources
Knowledge Intermediate Blackboard
Web application
Domain Know.
Private and public data
Meta and exchange data
Distributed sources

Fig. 2 Architecture of the agent–service cooperative design system across the Internet

agent, several corresponding interface agents, and client-side encapsulated design application agents. It is the layer directly facing design participants for interaction through an Internet-based interface. User agents construct an integrated development environment for PDUs with different roles and rights. Design application agents include application modules, such as the Solidworks modeling tool, deployed on client sides, which are encapsulated within integrated interfaces for being recalled to achieve the maximal efficiency of these traditional tools. Interface agents provide specific interfaces for users. The amounts and distribution of agents in this layer are based on PDU-oriented organization structures.

The resource application layer is situated at the bottom. It includes all of the distributed resources deployed worldwide that need to be efficiently organized by the middle layers.

It is the collaborative function layer in the middle that facilitates design collaboration among different PDUs. The main functions of the collaborative function layer can be grouped into two categories, i.e., agent federation and service management, each providing different modular collaboration support for CIDVS, respectively.

Agent federation management aims at the coordination and control of the agent group behaviors. As the communication facilitator of the overall framework, a communication service agent builds the global agent–service model and records the capabilities, names, addresses, and statuses of all of the distributed agents. A process management agent features project-oriented process management functions, including process definition, analysis, optimization, and alteration. A user management agent manages all the information of the users, roles, and authorities. A knowledge interaction agent aids multidisciplinary decision-making activities to obtain the optimized design consensus. A conflict coordination agent has the function of detecting and resolving conflicts between different PDUs. A history agent acquires the cooperative history information. Finally, a graphics agent controls the graphical presentation and operation.

Service management provides an efficient service management mechanism for the above agent layer to realize dynamic service access, especially to integrate the domain-dependent knowledge services. The underlying models and methods to support this mechanism include service ontology expression and interpretation, service registry and related resource match algorithm, service authority configuration, and service dynamic invocation. The following section will concentrate on this part to illustrate the dynamic acquisition of knowledge services in detail.

4 Agent-mediated service configuration and management

The process, resources, and even the Web services can be efficiently configured in the CIDVS environment for greater system flexibility, with the most adaptable effort put on the management of the Web services, i.e., the 'dynamic' Web service coordination within the user agent context. This flexibility is achieved through agent-based service mediating. The major role of service mediating is to guide, coordinate, and facilitate the teamwork of Web service providers, which includes acquiring appropriate services based on given task requirements and providing dynamic business support for PDUs according to predefined product development processes. It is concerned with the modeling, discovery, configuration, and invocation of Web services. The main topics will be discussed in the following sections.

4.1 Service modeling, description, and support for CIDVS

A Web service is an interface that describes a collection of operations that are network-accessible through standardized XML messaging. Its architecture is based upon the interactions between three roles: the service provider, the service registry, and the service requestor. For an application to take advantage of Web services, three behaviors must take place: the publication of service descriptions, the lookup or finding of service descriptions, and the binding or invoking of services based on the service description. Accordingly, the traditional interoperation process is based on standard procedures, such as defining WSDL (Web Services Description Language) interfaces, binding those interfaces with SOAP (Simple Object Access Protocol) messages, and publishing and discovering the services through UDDI (Universal Description, Discovery, and Integration).

However, while WSDL descriptions contain information about the operation and the parameter names in the service, they offer little/no information about the functionality of the service. Moreover, UDDI does not represent service capabilities, the tModels they use only provide a tagging mechanism, and the search performed is only done by string-matching on some fields that they have defined. There should be explicit semantics in the interaction of collaborating participants in order to achieve efficient collaboration. Web services are still insufficient in semantics as they just use XML to unify their syntax representation. So, semantic Web technology based on the resource description framework (RDF) data model is utilized together with Web services to provide semantic support in CIDVS.

The proposed service description complies with the DAML-S specification. DAML-S is an upper ontology for describing properties and capabilities of Web services using DAML+OIL. It provides an unambiguous and computer-interpretable markup language, which enables the automation of service use by agents and reasoning about service properties and capabilities [29], complementing low-level description about what a service can do. In our research, we concentrate on two different views, i.e., the profile view that describes what a service requires from and provides for the user, and the grounding view that describes how to use the service.

The profile contains a description of the service and the provider, the functional behavior of the service, several functional attributes, and also additional information. The grounding provides information for service invocation, including Web interface information, WSDL information, and instructional information. The grounding details about service access can be considered as a mapping from the general description given in the process model to a concrete implementation of the service.

The agents, especially the user agents, play an important role to mediate these semantic-rich services. The class model diagram is shown in Fig. 3. Agents act as the driving force of coordinating the Internet Web service utilization, while Web services exist as application knowledge sources.

The service also has other relationships with different kinds of agents. An application agent can encapsulate a legacy system as a Web service. A configure agent performs service configuration work, a registry agent is used for the registry, and a user agent takes the responsibility of controlling the behavior of one or more Web services. Normally, agents are built from .NET components or Java components. Some collaboration-type agents with intelligence are also developed as Web services for easy and intelligent self-upgrading.

4.2 Communication and interoperation among components of CIDVS

Communication among different components is the foundation for collaboration in the CIDVS environment. It is based on standard Web protocols. Different methods are used among agents and between agents and services. The message flow and corresponding message samples of the system are shown in Fig. 4.

The interoperation can be seen from two abstract layers. The higher layer includes the communication among agents. They are mainly based on the HTTP GET and HTTP POST methods, or through a subset of KQML (Knowledge Query Manipulation Language) to achieve simple speech act behavior. The contents of KQML are encapsulated with the XML language.

The messages used to invoke Web services form the lower communication layer. The source of these messages

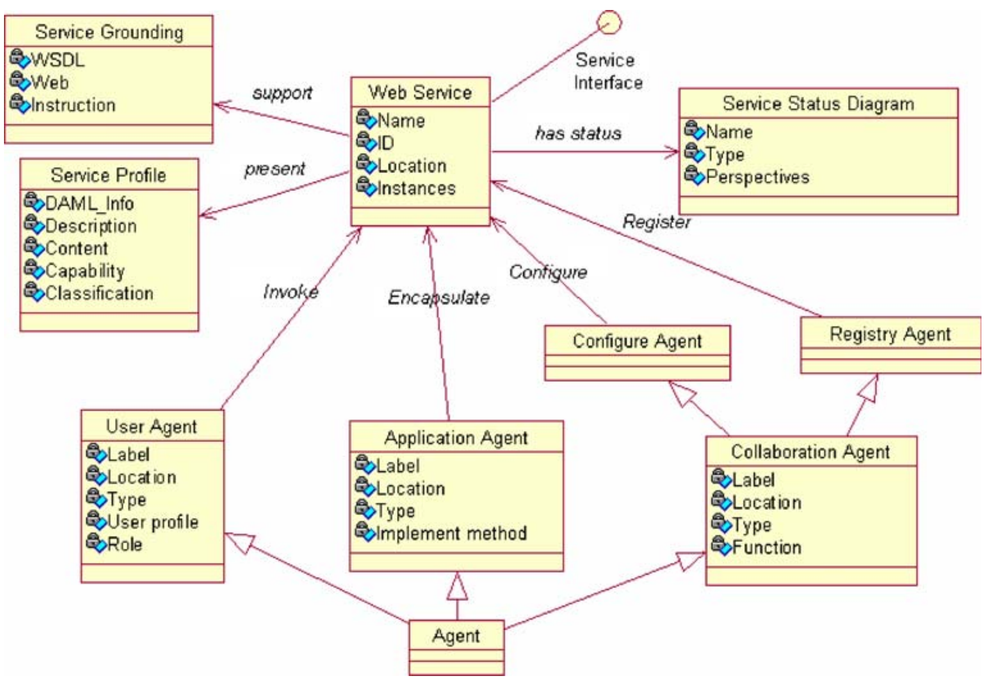
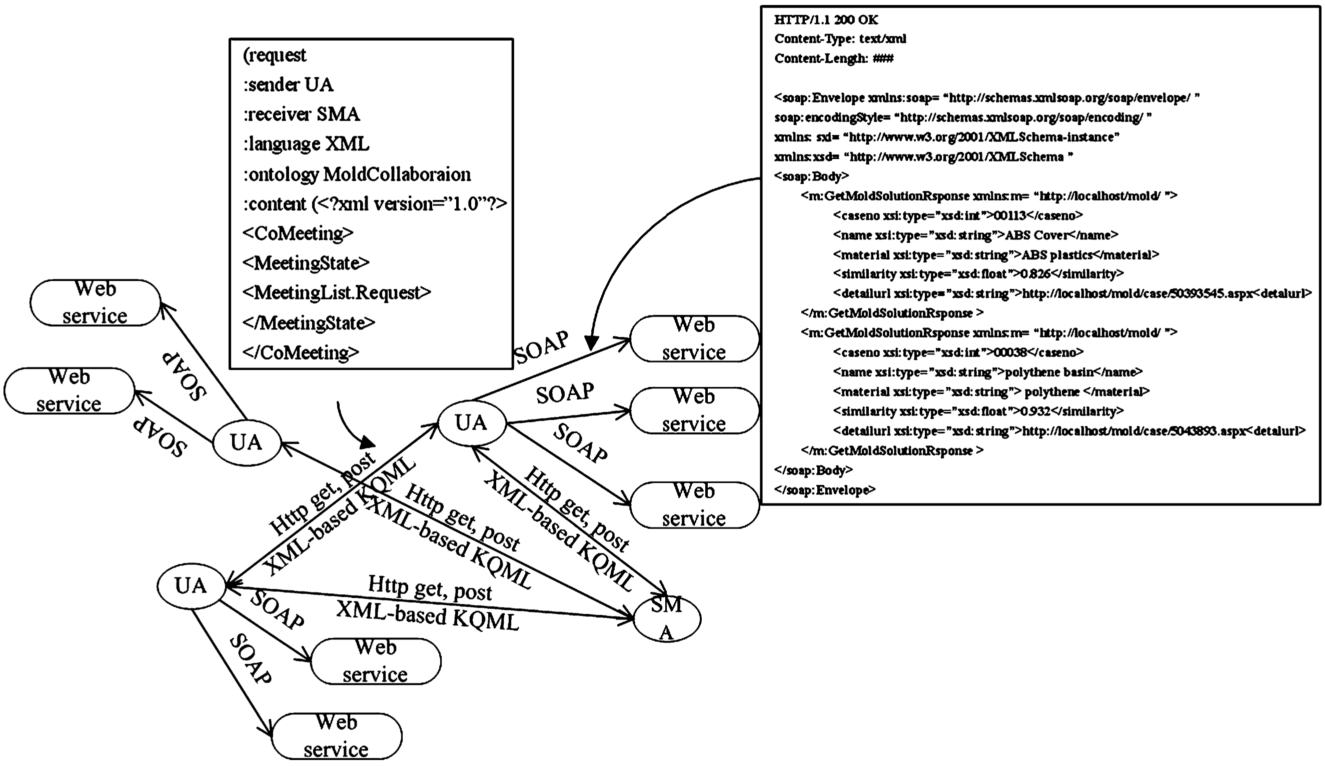


Fig. 3 Class model diagram of agents and services

may come from various service requests, including requests from user, KQML requests, and workflow-triggered requests. The user agent is responsible for constructing and transferring the requests into SOAP messages, and then forwarding them to the appropriate Web services. The messages adopted are mainly PRC/Encode-style SOAP messages used to pass operation names, parameters, and values.

4.3 Semantic-service-based publication, discovery, and matchmaking

With the change of environment, new knowledge or services may be needed to adapt to the new collaboration requirement. Since large amounts of Web services are distributed on the Internet, finding the most appropriate


```
(request
:sender UA
:receiver SMA
:language XML
:ontology MoldCollaboraion
:content (<?xml version="1.0"?>
<CoMeeting>
<MeetingState>
<MeetingList.Request>
</MeetingState>
</CoMeeting>
```

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: ###

<soap:Envelope xmlns:soap= “http://schemas.xmlsoap.org/soap/envelope/ ”
soap:encodingStyle= “http://schemas.xmlsoap.org/soap/encoding/ ”
xmlns: sxi= “http://www.w3.org/2001/XMLSchema-instance”
xmlns:xsd= “http://www.w3.org/2001/XMLSchema ”
<soap:Body>
    <m:GetMoldSolutionRsponse xmlns:m= “http://localhost/mold/ ”>
        <caseno xsi:type=”xsd:int”>00113</caseno>
        <name xsi:type=”xsd:string”>ABS Cover</name>
        <material xsi:type=”xsd:string”>ABS plastics</material>
        <similarity xsi:type=”xsd:float”>0.826</similarity>
        <detailurl xsi:type=”xsd:string”>http://localhost/mold/case/50393545.aspx<detalurl>
    </m:GetMoldSolutionRsponse >
    <m:GetMoldSolutionRsponse xmlns:m= “http://localhost/mold/ ”>
        <caseno xsi:type=”xsd:int”>00038</caseno>
        <name xsi:type=”xsd:string”>polythene basin</name>
        <material xsi:type=”xsd:string”> polythene </material>
        <similarity xsi:type=”xsd:float”>0.932</similarity>
        <detailurl xsi:type=”xsd:string”>http://localhost/mold/case/5043893.aspx<detalurl>
    </m:GetMoldSolutionRsponse >
</soap:Body>
</soap:Envelope>
```

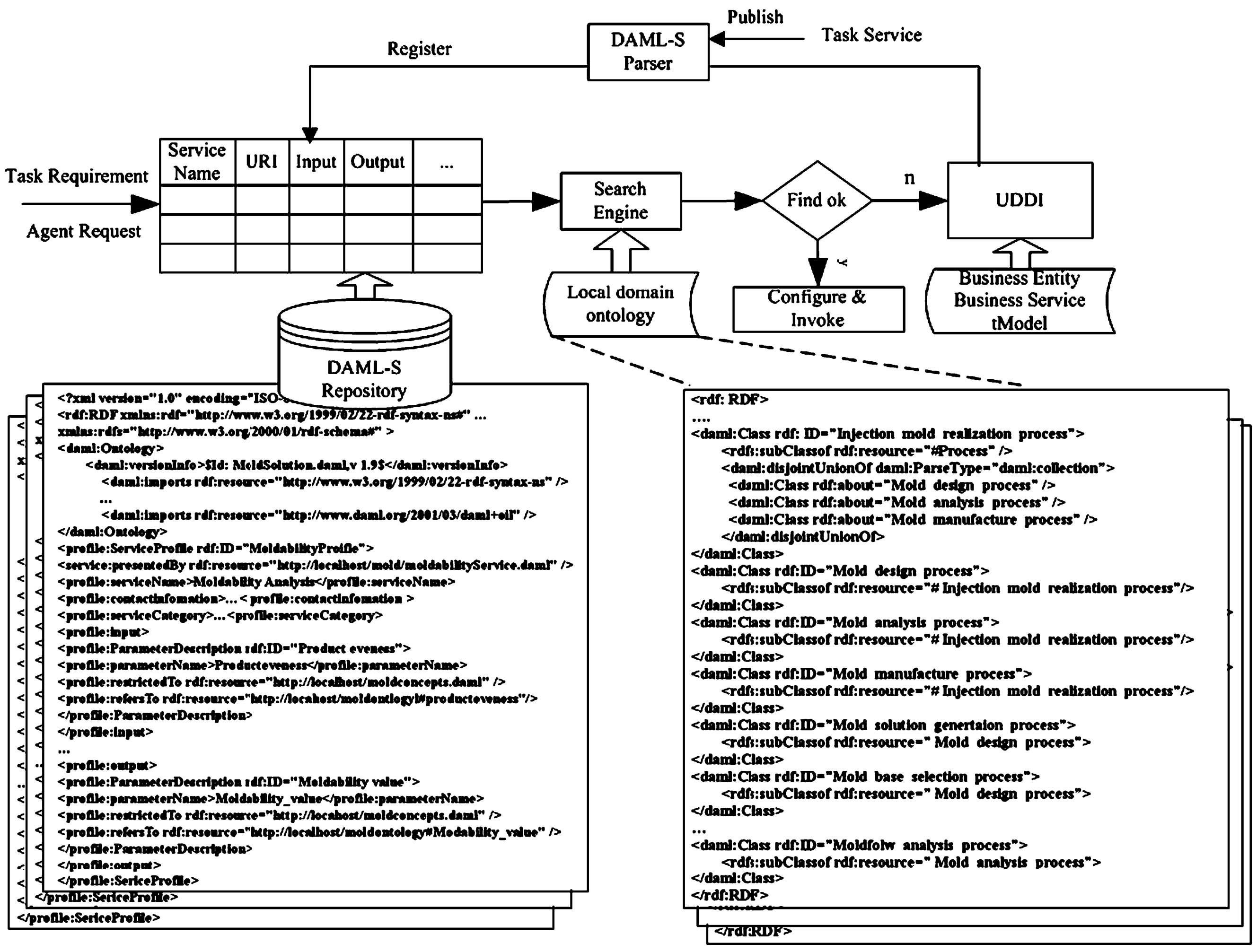
Fig. 4 Message communication and interaction flow

service is an important part of service mediating. CIDVS discovers services based on semantic information. Through capturing service request information such as properties, capabilities, execution interface, conditions, and constraints etc., Web services can be dynamically discovered and integrated into the system.

Traditional Web services discovery is based on UDDI, which does not make any use of semantic information, hence, failing to meet the problem of matchmaking between the provided capabilities of services and service requestors' needs. It does not provide facilities for service descriptions, except keyword and industrial service type categorization. Without sharing common definitions and understanding of the concepts, and without shared metadata and semantics associated with a particular Web service, an interaction between a UDDI client and a Web service cannot be performed in the correct manner. In CIDVS, a DAML-S-based local Web service registry is built to facilitate the capability-based service discovery, and the public UDDI registry acts as a complementary means for broader range service matchmaking. The detailed service discovery procedure is illustrated in Fig. 5.

The local registry stores the related profile and grounding information that were published by different service providers. A main search engine is equipped to perform matchmaking with the support of local domain ontology. Only when this search engine does not have enough knowledge on achieving successful matchmaking will it resort to external UDDI for searching in a wider range of enterprise services. After selecting an appropriate service, the related semantic information is registered back into the local registry through a DAML-S parser. Owing to this feedback mechanism, the number of locally registered Web services will grow larger and larger with the running of the system.

Different services share a common ontology in service profiles for Web service semantic matchmaking to achieve the sharing, reuse, and integration of services. The ontology is based on the uniform discovery and classification of Web services in the injection mold design domain. It provides an



```
<?xml version="1.0" encoding="ISO-
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" ...
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" >
<daml:Ontology>
    <daml:versionInfo>$Id: MoldSolution.daml,v 1.9$</daml:versionInfo>
      <daml:imports rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns" />
      ...
      <daml:imports rdf:resource="http://www.daml.org/2001/03/daml+oil" />
</daml:Ontology>
<profile:ServiceProfile rdf:ID="MoldabilityProifle">
<service:presentedBy rdf:resource="http://localhost/mold/moldabilityService.daml" />
<profile:serviceName>Moldability Analysis</profile:serviceName>
<profile:contactinfomation>...< profile:contactinfomation >
<profile:serviceCategory>...<profile:serviceCategory>
<profile:input>
<profile:ParameterDescription rdf:ID="Product eveness">
<profile:parameterName>Producteveness</profile:parameterName>
<profile:restrictedTo rdf:resource="http://localhost/moldconcepts.daml" />
<profile:refersTo rdf:resource="http://locahost/moldontlogyl#producteveness"/>
</profile:ParameterDescription>
</profile:input>
...
<profile:output>
<profile:ParameterDescription rdf:ID="Moldability value">
<profile:parameterName>Moldability_value</profile:parameterName>
<profile:restrictedTo rdf:resource="http://locahost/moldconcepts.daml" />
<profile:refersTo rdf:resource="http://localhost/moldontology#Modability_value" />
</profile:ParameterDescription>
</profile:output>
</profile:SericeProfile>
</profile:SericeProfile>
</profile:SericeProfile>
```

```
<rdf: RDF>
....
<daml:Class rdf: ID="Injection mold realization process">
    <rdfs:subClassof rdf:resource="#Process" />
    <daml:disjointUnionOf daml:ParseType="daml:collection">
     <daml:Class rdf:about="Mold design process" />
     <daml:Class rdf:about="Mold analysis process" />
     <daml:Class rdf:about="Mold manufacture process" />
    </daml:disjointUnionOf>
</daml:Class>
<daml:Class rdf:ID="Mold design process">
    <rdfs:subClassof rdf:resource="# Injection mold realization process"/>
</daml:Class>
<daml:Class rdf:ID="Mold analysis process">
    <rdfs:subClassof rdf:resource="# Injection mold realization process"/>
</daml:Class>
<daml:Class rdf:ID="Mold manufacture process">
    <rdfs:subClassof rdf:resource="# Injection mold realization process"/>
</daml:Class>
<daml:Class rdf:ID="Mold solution genertaion process">
    <rdfs:subClassof rdf:resource=" Mold design process">
</daml:Class>
<daml:Class rdf:ID="Mold base selection process">
    <rdfs:subClassof rdf:resource=" Mold design process">
</daml:Class>
...
<daml:Class rdf:ID="Moldfolw analysis process">
    <rdfs:subClassof rdf:resource=" Mold analysis process">
</daml:Class>
</rdf:RDF>
</rdf:RDF>
```

Fig. 5 Semantic Web service discovery flow

explicit conceptualization that describes the semantics of the information of the design task performing. Adopting a hierarchy structure, it contains a category domain ontology and a specific service profile ontology for each category in order to achieve efficient and precise matchmaking. The domain ontology knowledge base is the repository that stores this service ontology information. The standardized service concepts and their relationships contained within it can be used to carry out service similarity calculation and intelligent reasoning.

The input to the search engine is the discovery requirement, which can be represented in XML elements as:

Service_Requirement:=<General_classification>
<Domain_classification><General_information>
<Request_service_function><IO_info>
<Application Constraints>

It is supported by a user agent that accesses multiple ontologies using an integrated conceptual model expressed in the Web ontology language.

On the other hand, semantic-rich service descriptions are published in the local registry as service advertisements through DAML parsers. When a discovery request arises, the request and the registered advertisements are compared and a score can be obtained through a matchmaking algorithm. The service with the highest score is deemed to be the most desired service.

Currently, ServiceCategory and Service_IO information are the most often considered matchmaking factors. Four kinds of matching degrees are defined, i.e., exact match ($R \equiv A$), full match ($R \subseteq A$), part match ($A \subseteq R$), and disjoint match, each with a decreased matching level (R represents Request and A represents Advertisement). Within the same level, the match degree can be further determined by the minimal distance between concepts in the taxonomy tree. The matchmaking rationale is that the requester expects, first and foremost, that the provider achieves the output requested at the highest degree within a certain service category. Input matching is used only as a secondary score. For more accurate service retrieval, the QualityRating or other ServiceParameters can also be included in th rationale as the calculation factors.

4.4 Project-oriented service configuration

Project configuration in CIDVS is an important task that not only includes project-related process and design data, but also information about dynamic Web service distribution. The main objective is to form a dynamic workspace for each user in order to better facilitate efficient design collaboration. It follows the top-down procedure given below: project requirement acquisition→task specification definition in task manager→task configuration or PDU formation→process configuration→dynamic service configuration by user agent. Then, the product development process may begin.

Each user agent has a corresponding workspace profile to provide a personalized workspace. It contains process and product data information, as well as dynamic configuration information of the Web service stored in the blackboard as XML descriptions.

A user agent profile corresponds to certain service contexts with the aim of ensuring that the appropriate component services are involved in the collaboration according to a specific specification. The concerned Web service context includes composition information (domain info, related organization info, process info, access authority info, capability info, availability), specification descriptions, execution status information (in-progress, suspended, terminated, aborted), and invocation info. They are acquired through service discovery and direct configuration.

The invocation information comprises the location of the associated client interface or grounding description, including WSDL, that can generate the SOAP service template. Thus, the required dynamic Web service can be comprehensively invoked according to the project specification.

4.5 Dynamic binding and invocation of semantic services

The final knowledge acquirement process depends on mapping the general request to the concrete invocation after consulting the grounding of the appropriate Web services. The designer of the CIDVS environment can not only utilize the encapsulated design and analysis tools in the local host, but they can also invoke dynamic Web services to reach the design result.

The Web service invocation can be represented as WSI=<State, Web Service Set, Rule Set, Interpreter, Interaction Element>, where the 'State–Agent' mental state corresponds to the input parameters of a specific task. 'Rule Set–Rules' is obtained from the project configuration. The 'Interpreter–Reason' engine is used to interpret the rules to generate messages to access the proper Web service. The 'Interaction Element–Simple Interface' element includes the input and output parameters and the corresponding description.

Thus, two kinds of service invocation requirements exist during te product development lifecycle:

– *Static invocation*. Invoke the required service through configuration information mentioned above.
  In this case, service discovery happens during the project configuration phase according to the project specification, and then the selected service is statically configured or bounded to the given user agent. It presents to the user in the form of a client interface.

– *Dynamic invocation*. Invoke an unknown service through real-time discovery and binding.

  When a user initializes a request for a desired service during the product development phase, discovery is immediately conducted. After obtaining enough information about the appropriate service, the parameter interface can be generated and a SOAP-based message template can be sent for service access. Few researches have been concerned with this kind of invocation, which will be discussed below.

  Through service discovery, the user agent can obtain the WSDL and the instruction descriptions through parsing the corresponding grounding information. Then, with this information, a new method of dynamic invocation of. NET Web services at runtime is proposed. An instance of the *DynamicWebServiceProxy* class is created to realize this invocation. Three methods of this instance are mainly used; *GetWsdl* is utilized to acquire the WSDL document-related information; *BuildAssemblyFromWsdl* can generate the proxy class source code and a program set. And the proxy object instance is dynamically created by *CreateInstance*. Then, on the basis of this proxy instance, various information of the desired service can be dynamically acquired through using *GetMethod*, including information about its methods and parameters. Finally, the dynamic interface can be generated with the known information, and dynamic invocation can be easily achieved through this interface.

5 Prototype system implementation and a case study

A proof-of-concept software prototype has been developed for collaborative mold product development. Thin-client configuration is adopted to exert the flexibility and easiness of the Web. The system environment is built upon the .NET Framework and IIS used as the basic Web server. On the client side, Java applets, ActiveX controls, and Javascript are utilized to construct component agents. Server-side components are deployed for cooperative agents realization. A .NET Web service is used to encapsulate some basic collaborative functions and application services. The SOAP Toolkit's high-level API and WSDL Reader are used to achieve flexible service access. Some test Web services with profiles and groundings have also been developed for service discovery verification.

In the case study, we demonstrate how the mold products for a plastic cover plate part are designed. In the project configuration stage, three PDUs are formed, i.e., a part design PDU, a part analysis PDU, and a mold design PDU. Of course, more PDUs can be involved, depending on the project requirement. After the process and data configuration, services are to be configured. Through service matchmaking in the local registry, a moldability analysis service is discovered and then bound with the part analysis PDU. In the same manner, a plastic selection service is configured into the part design PDU.

Now, the mold product development process starts. The part design PDU uses a plastic selection service to select the appropriate material. After the modeling of the product by the part designer, a request is sent to the part analysis PDU for moldability analysis. This service is invoked by the part analysis PDU and corresponding suggestions are sent back for further modification on the part. This iteration can take place several times until the satisfactory result is obtained. During the development process, the mold designer wants to participate in the mold design early and efficiently, so he needs a quick mold-generation service and a request is then sent for dynamic service discovery. After an appropriate service is located, dynamic invocation is subsequently activated. The service information is obtained and a program set is built as a proxy instance. Based on them, a simple parameter interface is generated for presenting the individual service parameters and methods, each with a description illustrating its main function or value range. Then, a mold solution can be easily obtained through case-based reasoning (CBR) methods embedded within the dynamic knowledge services provided that are based on our previous work [30]. In the same way, more services can be efficiently used to aid quick knowledge collaboration. In this example, a two-plate mold solution case is retrieved, the designer only needs do some modifications on it, and other services such as gate design and process planning can also be used. Synchronous collaborations among several multidisciplinary stakeholders are carried out at certain design time points on request.

As we can see, with the standardized and active participance of multidisciplinary teams and the dynamic integration of distributed knowledge services, this system can provide efficient support for dynamically formed product development projects, further coordinating and accelerating the design process in a system and its concurrent view.

6 Conclusion

To the new problem of collaborative design under a dynamic integration environment, this paper proposes a novel framework that is dynamic configurable and highly flexible. It covers the whole product development lifecycle and integrates multidisciplinary knowledge services efficiently. Based on the idea of service-orientation, agent and semantic Web technologies are used at a more abstract level to facilitate the collaboration capability and improve the flexibility of the system. Besides the implementation of a prototype system in injection-mold design, it can be further extended to other areas of inter-enterprise collaborative design.

Acknowledgment This research is supported by the National Basic Research Program of China (grant number 2004CB719405).

References

1. Cutkosky MR, Tenenbaum JM, Glicksman J (1996) Madefast: collaborative engineering over the Internet. Commun ACM 39 (9):78–87
2. Huang GQ, Shi J, Mak KL (1997) Internet-based design for X. In: Proceedings of the 14th International Conference on Computer-Aided Production Engineering, Durham, UK, April 1997
3. Xu XW, Liu T (2003) A Web-enabled PDM system in a collaborative design environment. Robot Com-Int Manuf 19 (4):315–328
4. Huang GQ, Mak KL (2000) WeBid: a Web-based framework to support early supplier involvement in new product development. Robot Com-Int Manuf 16(2–3):169–179
5. Zhou SQ, Zhao AP, Chin K-S, Yarlagadda PKDV, Peng Z (2004) A solution for knowledge resources provider over the Internet. Int J Adv Manuf Technol 24(2):148–160
6. Zhou SQ, Chin K-S, Ling W, Xie Y (2003) Internet-based distributive knowledge integrated system for product design. Comput Ind 50(2):195–205
7. Zhou SQ, Chin K-S, Yarlagadda PKDV (2003) Internet-based intensive product design platform for product design. Knowl-Based Syst 16(1):7–15
8. Chin K-S, Zhou SQ, Krishamurthy R, Xie YB, Yarlagadda P (2002) An intelligent approach of knowledge searching within Internet-based distributive integrative environment. Eng Appl Artif Intell 15(6):607–618
9. Xiao A, Choi HJ, Kulkarni R, Allen KJ, Rosen D, Mistree F (2001) A Web-based distributed product realization environment. In: Proceedings of the ASME Design Engineering Technical Conferences, Pittsburgh, Pennsylvania, September 2001, vol 1, pp 979–991
10. Kulkarni R, Rosen D, Allen JK, Mistree F (2002) An information model for finding and integrating distributed resources for engineering design-manufacturing processes. In: Proceedings of the ASME Design Engineering Technical Conferences, Montreal, Canada, September/October 2002, vol 1, pp 445–458
11. Wang YD, Shen W, Ghenniwa H (2003) WebBlow: a Web/agent-based multidisciplinary design optimization environment. Comput Ind 52(1):17–28
12. Ming XG, Ma S, Lu WF, Ni QF (2003) Web service architecture for collaborative product lifecycle management in virtual enterprise. In: Proceedings of the 10th ISPE International Conference on Concurrent Engineering: Research and Applications, Madeira Island, Portugal, July 2003, pp 191–198
13. Chung MJ, Jung HS, Kim W, Goplannalan R, Kim H (2004) A service-oriented framework for collaborative product commerce. In: Proceedings of the 8th International Conference on Computer Supported Cooperative Work in Design (CSCWD 2004), Xiamen, China, May 2004, vol 2, pp 425–430
14. Aziz H, Gao J, Maropoulos P, Cheung WM (2005) Open standard, open source and peer-to-peer tools and methods for collaborative product development. Comput Ind 56(3):260–271
15. Yan J, Lu CY (1998) An agent-support approach to collaborative design. Annals CIRP 47(1):107–110
16. Zhao G, Deng J, Shen W (2001) CLOVER: an agent-base approach to systems interoperability in cooperative design systems. Comput Ind 45(3):261–276
17. Sun J, Zhang YF, Nee AYC (2001) A distributed multi-agent environment for product design and manufacturing planning. Int J Prod Res 39(4):625–645
18. Chira O, Chira C, Tormey D, Brennan A, Roche T (2003) A multi-agent architecture for distributed design. Lect Notes Artif Int 2744:213–224
19. Mok CK, Chin K-S, Ho JKL (2001) An interactive knowledge-based CAD system for mould design in injection moulding processes. Int J Adv Manuf Technol 17(1):27–38
20. Tang D, Eversheim W, Schuh G, Chin K-S (2004) CyberStamping: a Web-based environment for cooperative and integrated stamping product development. Int J Comput Integ M 17 (6):504–519
21. Tang D, Eversheim W, Schuh G, Chin K-S (2003) Concurrent metal stamping part and die development system. Proc I Mech Eng B—J Eng 217(6):805–825
22. Chin K-S, Tang D (2002) Web based concurrent stamping part and die development. Concurrent Eng—Res Appl 10(3):213–228
23. Lee R-S, Chen Y-M, Li C-Z (1997) Development of a concurrent mold design system: a knowledge-based approach. Robot Com-Int Manuf 10(4):287–307
24. Xie SQ, Huang H, Tu YL (2002) A WWW-based information management system for rapid and integrated mould product development. Int J Adv Manuf Technol 20(1):50–57
25. Xie SQ, Tu YL, Zhou ZD (2004) Internet-based DFX for rapid and economical tool/mould making. Int J Adv Manuf Technol 24 (11–12):821–829
26. Chung J, Lee K (2002) A framework of collaborative design environment for injection molding. Comput Ind 47(3):319–337
27. Ahmed AA, Karina R, Arturo M (2003) Internet-based collaborative design for an injection-moulding system. Concurrent Eng—Res Appl 11(4):289–299
28. Qiu H, Shao X, Li P (2004) Task flow optimization for the integrated process management of dynamic alliance collaborative product development. In: Proceedings of the 11th ISPE International Conference on Concurrent Engineering: Research and Applications, Beijing, China, July 2004, pp 871–6
29. Ankolekar A, Burstein M, Hobbs JR (2002) DAML-S: Web service description for the semantic web. In: Proceedings of the 1st International Semantic Web Conference (ISWC 2002), Sardinia, Italy, June 2002
30. Qiu HB, Li CX (2004) Conceptual design support system in a collaborative environment for injection moulding. Int J Adv Manuf Technol 24(1–2):9–15