11

Interflow systems for manufacturing: concepts and a construction

J. Goossenaerts and D. Bjørner
United Nations University, International Institute of Software Technology
P.O. Box 3058, Macau

Abstract

A conceptual framework for modelling, simulation and control in manufacturing industry is proposed. The framework builds on three organizing principles: a *decomposition principle*, the *life cycle principle*, and the *interflow principle*. The links of these principles with the provision of information and command infrastructure services for manufacturing enterprises are indicated. A terminology is proposed for the construction of interflow models of manufacturing enterprises and industry and a generic plant interflow model is constructed. Issues such as the particularization of models and the interfacing of enterprise models to manufacturing phenomena are also touched upon.

Keywords

Enterprise models, industry models, model execution services, formal methods

1 INTRODUCTION

A *manufacturing industry information and command infrastructure* (abbreviated: MI^{2}CI) is an information processing and activity control and monitoring system which provides information and control services in support of the operations and projects of manufacturing enterprises. Sophisticated such services may enable industries in moving in a relatively short period of time from an early stage of development towards a lean/agile supply-based sustainable system. An industrial system is lean when it is capable of achieving results without using superfluous resources (e.g., equipment, workers, investments, stock), it is agile when it is capable of deciding quickly and intelligently and accordingly adapting operations quickly and easily. An industry is called supply-based when a large number of the parts in products are procured from (a large number of) suppliers. An industry is sustainable when products are designed, produced, distributed and disposed with minimal (or none) environmental and occupational health damages, and with minimal use of resources (materials and energy) (Alting and Jorgensen, 1993).

Information and command infrastructure services that rely on model execution services and interflow with manufacturing phenomena, are expected to become the enabling technology for lean operations and agile projects of extended enterprises. Without an integrated problem domain understanding, and without inter-operable services built on such, non-productive concerns

P. Ladet et al. (eds.), *Integrated Manufacturing Systems Engineering*

threathen to inflate costs and complexity due to the increasing number of specialized enterprises and other actors involved in engineering projects and manufacturing operations. A rigourous understanding of the interactive and synchronized dynamics which is possible between computational flows on networked computers and the life cycles of products, plants, enterprises, extended enterprises and markets, is instrumental for the definition and implementation of information and command infrastructure services and for integrated manufacturing systems engineering.

Overview and relationships to other work
Adequate tools and services in support of integrated manufacturing systems engineering should build on a proper conceptual understanding of manufacturing industry. Such conceptual understanding should be cast in a formal framework which affords construction, computation and formal reasoning.

The *decomposition principle* and *life cycle principle* proposed in section 2 complement the concepts of *generic*, *partial* and *particular* model consolidated in ENV 40 003 (1989). The decomposition principle has been elicited from the discussion about the relationship between an enterprise model (catering for operations) and the full life history of an enterprise (in which also improvements and innovations must be considered) (Williams, 1993); and that about the relationship between enterprise models and environment models (Goossenaerts, 1993). Decomposition is a device for classifying the wide range of concepts that must be catered for in enterprise and industry models. It reflects a manufacturing domain understanding which draws on those of CIM-OSA (Vernadat, 1993), ATLAS (ATLAS, 1993) and MS^{2}O (Matthiesen, 1994), all EU ESPRIT Projects. The *life cycle principle* reflects the concern with life cycle engineering in manufacturing. Indeed whether an industrial system is sustainable or not depends on the extent to which the life cycles of plants, products and materials affect the environment, during their creation, their operation or use, and after their disposal.

Section 3 proposes the *interflow principle* and links it to commonly used terms in enterprise modelling. The principle draws our attention to the interface between enterprise model driven computational flows and shop floor and engineering office activities. An interflow system is a system in which computational flow is kept in step with activities and changes in an application domain. The generic plant interflow model constructed in section 4 synthesizes the generic properties of plant systems into a single coherent system, and it articulates the distinction between computational and physical activities. The selection of the primitive templates for the bottom-up construction was made on the basis of work by Scheer (1989) and Atsumi (1993). The construction focuses exclusively on operations, as separated from innovations and improvements. A RSL specification (The RAISE Language Group, 1992) of the construction can be found as an annex to the report by Goossenaerts & Bjørner (1994).

2 INFORMATION & COMMAND SERVICES FOR MANUFACTURING INDUSTRY

Information and command infrastructure services for manufacturing industry have to be conceived in reference to full *plant life cycles* and full *product life cycles* seen in the context of extended enterprises embedded in business environments. Below the concepts of product and plant life cycle are defined and a decomposition of manufacturing industry is proposed. The latter is referred to in a summary of the information and command infrastructure services for manufacturing industry.

Plant and Product Life Cycles in the Business Environment
The *life cycle of a product* is concerned with: (a) Its *creation* in a (virtual) plant. (b) Its *usage*

by consumers on the market. Usage of the product may include maintenance, repair, upgrading, etc. (c) Its *decomposition* – preferably with recycling or reuse of its composing materials – in a (virtual) plant.

The *life cycle of a plant* is concerned with: (a) *Projects* which phase *a plant and its products* into *a market*: the gradual introduction of a new plant and its products into a market (and environment). (b) *Operations*: a plant will respond to *orders* issued by customers, by exploding the orders into *plant programme steps*, by scheduling and carrying out the (production) steps, and by delivering the goods ordered. *Projects* during operations may concern: how to phase in&out *products*, *equipment* and *human resources* into a plant, and how to improve *plant operations*. (c) *Projects* which phase out *a plant and its products*: the gradual withdrawal from use of an old plant and its old products.

The Orthogonal Systems of Manufacturing Industry

The focus on (collaborative) labour in the product and plant life cycles and the requirement of orthogonality for the systems to describe these suggest: First, to make, in an enterprise, a distinction between "repetitive" *operations*, to be catered for in a *plant model*; and "one-of-a-kind" *improvements* and *innovations* to be catered for in *projects* carried out by *teams*. Secondly, when extending the viewpoint beyond the boundary of the enterprise to let *extended-enterprise operations* be carried out by so-called *virtual plants*, and to let *extended-enterprise projects* be carried out by *virtual teams*. A virtual enterprise (or extended enterprise (Browne *et al.*, 1994)) comprises a virtual team and a virtual plant. And thirdly, to consider *markets* as the contexts within which enterprises are embedded, and to use the term *industry* to denote this context when considering also the projects and virtual projects executed in the market (e.g. industrial policy projects).

The following domains (Table 1) are proposed to organize the objects, processes and issues in industry (see also Goossenaerts & Bjørner (1994)): (a) A *team* phases in and out a plant or a product in response to *goals* formulated by entrepreneurs or other agents of change. A team refines and enriches the goals into *project programmes* for changing or making a (virtual) plant, it schedules and carries out these programmes, finally, it delivers the new or improved plant, capable of producing goods, with the performance expressed in the goals. (b) A *plant* responds to *orders* issued by customers, by exploding the orders into *plant programme steps*, by scheduling and carrying out the (production) steps, and by delivering the goods ordered. (c) A *virtual plant* comprises several plants which synchronize their operations as if they were carried out by a single plant. (d) A *virtual team* comprises several teams which synchronize their work and decisions as if they would carry them out as a team within a single enterprise. (e) A *market* forms the context for the operations of plants and virtual plants. Market regulations (e.g. such as in Company Law and Tax Law) affect the operations of plants and virtual plants. (f) An *industry* forms the context for the projects of (virtual) teams. Regulations (e.g. such as in Environmental and Technical Regulations and international standards) affect projects.

The desirable conditions of operations (leanness) and projects (agility) must be realized in different contexts: in the – extended – enterprise and in the industry.

Projects are further classified. *Entrepreneurial Projects* deal with changes of enterprise level programmes during the enterprise life cycle (e.g. the introduction of new products, business process reengineering, investment appraisal). *Engineering Projects* are concerned with product life cycle and production processes. Usually they require innovation. *Quality Improvement Projects* are concerned with product and production process improvements. *Industrial Policy Projects* pertain to the industry. Virtual plants result from the successful implementation of projects by virtual teams. These require inter-organisational engineering – in the extended enterprise – and

a (standardized) technology infrastructure.

Table 1 Domains for organizing objects, processes and issues in industry

activities / *context*	*operations* / "repetitive"	*projects* / "one-of-a-kind"
enterprise	plant / production	team / innovation
extended enterprise	virtual plant	virtual team
environment	market / trade	industry / (policy) innovation
desirable condition	leanness	agility

Information and Command Infrastructure Services
The provision of information and command infrastructure services requires (computer network supported) enterprise model execution services. See CEN/TC310/WG1 (1994a) for a statement of requirements for such services and CEN/TC310/WG1 (1994b) for an evaluation of some related initiatives. As to design services, relevant requirements can be sourced from the literature on computer aided design and project support environments.

Subsystems of an information and command infrastructure for enterprises and industry are classified in accordance with the domain decomposition in Table 1 (IMES stands for Interflow Model Execution Services and ISDACS stands for Interflow System Design and Construction Services). An enterprise information and command infrastructure comprises:
Plant IMES: a plant component which supports the repetitive operations of the enterprise (cf. enterprise wide information systems built around enterprise-wide data models (Scheer, 1989)).
Plant ISDACS: a team component which supports the innovations and improvements which the enterprise needs to compete in the market. Application packages in a Plant ISDACS may include an *entrepreneuring project coach*, an *engineering project coach* and a *quality project coach.*

An industry information and command infrastructure comprises:
Market IMES: a market component which supports the repetitive operations at the market. Application packages in a Market IMES may include *material flow monitors*, *employment monitors*, *(value added) tax collection systems*, *trade and transportation systems*, *an intellectual property monitor*, etc.
Market ISDACS: an industry (team) component which supports the one-of-a-kind (industrial policy) innovations and improvements which the industry needs to sustain its competitiveness in the global market. Application packages in a Market ISDACS may include an *industrial policy project coach.*

3 INTERFLOW MODELS FOR MANUFACTURING INDUSTRY

Considered are the interplay of phenomena flows and computational flows; assumptions and a basic terminology for interflow models; the use of enterprise (reference) models as tools for organizing and integrating information about enterprise processes and industry, with attention for generic models and how particularizations can be accommodated.

The Interplay of Phenomena Flows and Computational Flows
The division of labour – and the productivity gains and accelerated innovation enabled by it – has made manufacturing industry increasingly dependent on computational flows to support operations and projects. A manufacturing system has become a pair of a phenomena flow – on the shop floor and in the engineering office – and a computational flow. The former can

be described by mathematical models, the latter is defined over the mathematical model and implemented by some computing system. Both flows are linked by multiple interface channels which enable synchronized dynamic behaviour.

A better understanding of the interplay of computational flows and business/manufacturing processes is useful for a number of reasons. Costs and performance in manufacturing inherently depend on physical world phenomena. Consider for instance the case of a "hand over" – the passing on of a job to a fellow worker – in business processes (see Hammer & Champy (1993) for elaborate illustrations). Criteria as to whether or not a hand over (and a specialized work) is required are contingent on phenomena flow conditions and resources (both for material transformation and computation). But resulting decisions should be consolidated in an enterprise model, to scrutinize them as to their consistency with other decisions, to propagate changes due to new decisions, to support the automatic derivation of programmes for implementing decisions, etc. By focussing on the division of activities between phenomena flow and computational flow and by modelling the enterprise, one also can swiftly compare business and manufacturing processes which are more or less automated and support decisions about whether or not to model objects and activities in the enterprise (for instance prior to automation or reengineering) or in the business environment.

Assumptions and Terminology: Interflow Models

Mathematical models are built for a selection of objects, concepts, and changes in these, as designated and measured in an application domain. The utilization of mathematical models for modelling an application domain rests on two assumptions:

The **two worlds assumption**: We deal with two worlds: one created by means of mathematical symbols, and another one, the *physical system* (application domain) catering for the physical space & time, with the physical objects having life cycles in it.

The **measurement assumption**: There exists methods (*measurements* or *observations*) of assigning mathematical symbols to particular states or conditions of phenomena in the physical system. It is possible to model properties of phenomena by means of mathematical constructs.

These assumptions are of fundamental importance in physical science as succinctly expressed by Hertz (1894). They are also referred to in the General Design Theory (Yoshikawa, 1981).

Some terms for phenomena flows and computational flows: (1) An *application domain* is a part of the world to which we restrict our attention with the purpose to build a mathematical model, for the derivation of properties, for simulation or interflow. (2) *Phenomena Flow* over an application domain is the amalgam of observable changes of physical objects in the course of time and in a designated environment (space). An *object* can be a building, a part, a tool, a worker, a work sheet, etc., and *change* may result from work by man or machine (in which case we use the term *activity*), from the laws of physics, chemistry or biology, etc. (3) The *designation* of an application domain is by means of a *space-time-resource filter* which selects objects and changes and their properties (concepts for which to execute measurements, changes and the objects involved, the precision of measurements, the intervals between measurements) for consideration in the model. (4) A *template* is a formally defined collection of data which is used to reproduce a same thing (typically a representation of an object or an activity) many times. It comprises two orthogonal kinds of data. *Attributes* contain data about the *intrinsic* properties of the template's instances. *Linkages* contain data related to the *extrinsic* or *accidental* aspects of the template's instances. (4.a) Illustration. In manufacturing industry, geometric properties and material of a part are among its intrinsic properties. Extrinsic aspects of a part include the parent parts in which a part can be assembled, the activities in which it can be transformed, as well as the positions (on the shop floor) where a part can be found during the manufacturing operations.

Another example: an extrinsic property of an instance of an activity template, is the part on which it is invoked. (4.b) Intrinsic and extrinsic properties of templates are particularized in accordance with a *space-time-resource filter* which determines a *space-time interval* and a space-time-concept granularity. (4.c) In the constructions in section 4, the particularization attribute "τ" indicates an abstraction of the descriptions in templates as regards their intrinsic properties (granularity, formality, and focus (completeness)). When particularizing a particular model from a generic model, it is recommended when introducing, enriching or refining the templates of the particular model, to develop granularity, formality, and focus simultaneously. I.e. it is recommended that for instance geometric attributes (assumed under τ in the generic system), are simultaneously taken into consideration for all templates for which they are relevant in the particular system. (5) A mathematical model *models* ($\boxed{\models}$) an application domain if and only if it supports the derivation of (observable) properties of the corresponding phenomena. The mathematical model is called a *model* of the domain. (5.a) Illustration: The schematic of a house supports the derivation of properties (e.g. dimensions of rooms) of the house: schematic $\boxed{\models}$ house (one or more tangible buildings). (6) A computational flow over a mathematical model *simulates* ($\boxed{\wr\models}$) a phenomena flow in an application domain if and only if successive states of the computational flow describe successive states of affairs of the phenomena flow. Measurement methods are used to validate or deny "description". Successive states of the computational flow are mutually linked by transitions (symbol manipulation, computation). There is no interaction nor synchronization between the computational flow and the phenomena flow. The computational flow is called the *simulation model* of the phenomena flow.

The utilization of computational flows for interflowing a phenomena flow rests on one further assumption. The **interface assumption**: it is possible to establish *interface channels* which link particular objects and changes in a phenomena flow to particular instances in a computational flow. Some terms for interfaces: (7) An *interface* is the area between a phenomena flow and a computational flow , it consists of a number of *interface channels.* (8) An *interface channel* is a link between a phenomena flow – some objects and changes in it – and instances in a computational flow. Over a channel *tasks* and *signals* are transmitted. They are the elementary units of information exchange and synchronization. Tasks and signals which the computational flow receives are called *inputs*, those which it sends are called *outputs.* (9) A computational flow *interflows* ($\boxed{\aleph\models}$) a phenomena flow if and only if its behaviour simulates the phenomena flow and if it synchronizes with it along one or more interface channels. The computational flow and the phenomena flow proceed *synchronously* and *inter-actively* with tasks and signals flowing (simultaneously) along *interface channels.* The computational flow over the mathematical model is called the *interflow model* of the phenomena flow in the application domain. (10) An *interflow system* comprises the joint behaviour of the computational flow and the phenomena flow.

Enterprise and Industry Models; From Generic to Particular

The use of enterprise models and reference models as tools for organizing and integrating information about enterprise processes is well established. See for instance ENV 40 003 (1990), Scheer (1989), CIMOSA (1993), Spur *et al.* (1994). Goossenaerts & Bjørner (1994) introduce the related concept of industry model.

Enterprise Modelling

The distinction in the enterprise between repetitive operations and one-of-a-kind projects allows one to split an enterprise model into two components: a plant model and a team model. These models are related to each other and to phenomena flow. Projects, supported by the team

model in execution, transform the plant model and the physical plant. Operations, controlled and supported by the plant model in execution, transform material inputs and produce the products.

Enterprise modelling is concerned with the construction of generic, partial or particular (as defined in ENV 40 003 (1990)) plant models and team models and their integration.

The Construction of a Generic Plant Interflow Model.
A construction of a generic plant interflow model is summarized in Table 2. The first three steps deal with statical structural properties of plant systems. The fourth step allows one to define computational flows over *instances* of the templates (cf. types) in a plant structure. Computational flows involving these instances can simulate plant operations. The fifth step deals with the interfaces between the computational flows and plant operations. Some remarks to justify the focus of the five steps in the construction: (**1**) Materials and the work to transform them are described *statically* by means of *part templates* describing units of material and *work templates* describing units of change to material. A *Part-Work Structure* integrates the information in a bill of materials and in process charts and describes the parts and works required for making products. (**2**) Cells and the orders they send and receive incorporate the division of labour and the coordination of results. *Cell Templates* describe cell(s), a cell is a unit capable of sustaining activities/work. Cf. resource model. *Order Templates* describe orders. An order is sent by one cell to another to request the delivery or transformation of a part. A *Cell-Order Structure* integrates the properties described in organization charts and in information flow charts. It is concerned with the division of labour and coordination of results. (**3**) A *Plant Structure* results from joining a part-work structure and a cell-order structure. It integrates the statical properties of a plant. (**4**) *Plant System.* Instances of cell templates and their responses to order instances can simulate the operations at a shop floor. The term *pulse* denotes the response by a cell instance to an order instance it receives. Cf. the order handling process. The term *flow* denotes the integration of pulses. A *plant system* controlled by a model execution system and interfaced to a suitable discrete event generator, can simulate shop floor operations. (**5**) *Plant Interflow Model. Channel templates* describing interface channels are added to the plant system. Interface channels are used to interface a computational flow to phenomena flow on a shop floor. They support synchronization through the input and output of particular tasks and signals.

Table 2 The bottom-up construction of a Plant Interflow Model

1	*Part-Work Structure*	(*Part Template* $\models$ part(s), *Work Template* $\models$ work(s))
2	*Cell-Order Structure*	(*Cell Template* $\models$ cell(s), *Order Template* $\models$ order(s))
3	*Plant Structure* $\models$ plant(s)	
4	*Plant System* $\wr\models$ plant operations (*Pulse* $\wr\models$ response to order instance, *Flow* $\wr\models$ operations)	
5	*Plant Interflow Model* $\aleph\models$ plant operations (*Channel Template* $\models$ interface channel) *Plant Interflow System, sync(hronize) cycle*	

Team Interflow Models
A team working in an (virtual) enterprise aims for improvements or innovations in a (virtual) plant interflow system. In pursuit of a *goal* – the sequence of activities towards its accomplishment has not been secured yet – a team will first – during the design&drafting phase – transform or particularize a plant interflow model (templates, plant structure and plant system consolidate the deliverables during this phase) and next – during the construction phase – construct or

upgrade the plant interflow system, by transforming the physical plant and its computational flow. In the course of a project, the composition of the team is adapted according to changing project activities. In contrast, the cell structure of a (traditional) plant remains static during operations.

Team interflow models in execution support team activities. A generic team interflow model forms a basis for the implementation of an *Integrated Project Support Environment.*

Industry Modelling

The contemporary market is governed by a large and changing collection of regulations and constraints for the operations of plants, producers and consumers. Coping with the regulations and their changes requires high costs. More integrated and harmonized industry models and the corresponding market-wide information and command infrastructures, and proper model-driven interfaces between their enterprise infrastructure and the industry infrastructure are desirable.

Industry Modelling is concerned with the construction of generic, partial or particular market interflow models or industry interflow models. Such models enhance the understanding, coordination and harmonization of rules and other market information.

A Generic Market Interflow Model

The construction of a generic market (interflow) model parallels the construction of a generic plant (interflow) model. A significant difference is that the cell-order structure is replaced by a person-contract structure. Cells (in plants) communicate with each other in a master-slave relation, client-to-server, whereas legal persons (persons, companies and public bodies) in a market communicate (also) as equals, peer-to-peer. *Contracts* commit two or more legal persons to carry out, during a certain period, a number of exchanges of products or services (including labour, money, etc.). They may express terms of synchronization, obligations, etc.; e.g. to incorporate a company a suitable "legal person template" must be selected, this will determine reporting obligations, and – to some extent – the contracts the company can enter into. The legal personality and contracts may also be correlated with the products/services which the enterprise can provide on the market (e.g. banks must be registered in a special commission).

Templates of all legal persons and contracts that exist in a market are joined in a *person-contract structure.* This structure includes person templates for the public bodies that – because of statutory regulations – are involved in the registration and monitoring of (certain) contracts and exchanges. It also includes contract templates such as for incorporation contracts, labour contracts, etc. (Goossenaerts & Bjørner, 1994).

Industry Interflow Models

Industry-wide projects such as industrial policy and legislative projects aim at transformations of a market, industry or sector as a whole. Such transformations may be planned (or designed) in terms of particular market interflow models, prior to their implementation.

Our approach suggests to consolidate the deliverables of industry-wide projects as templates of products, exchanges, persons and contracts, (in) a market structure, (in) a market system, (in) a particular market interflow model, and eventually, as (in) a market interflow system.

Models and their Execution Services

Particular (virtual) plant models, (virtual) team models, market models and industry models will drive the information and command infrastructure services for manufacturing enterprises and industry.

Generic models, and the particularization operators for them, should ensure that particular

models can easily be developed, and that they can be executed on commercial computer and communications hardware, with interfaces to shop floor and engineering office resources.

4 CONSTRUCTION OF A GENERIC PLANT INTERFLOW MODEL

A factory is the application domain. The space-time-resource filter selects: Space: the shop floor and the engineering offices of the factory; Time: the hours of operation of the factory; Resources: materials and parts, machines, workers and related concepts. The statical model of the factory should cater for bill of materials, process charts and resource models. The dynamics of a simulation model should be related to order handling (incoming orders require creation of new variables and transitions), work in progress records, production plans, work schedules, etc.

The construction which is summarized in Table 2 is explained.

4.1 Part-Work Structure

Terminology

1. A *Part* is one of the things that make up another thing. The concept of part includes: *end product*: a part which receives no further processing within the firm; *assembly*: a part which is constructed from other parts and used in the construction of further parts; and *component (material)*: a part that is not produced in the firm, but bought-in, including raw materials.
2. *(i) Work* is any activity that transforms a part. The concept of work includes: *transporting*: moving a part without changing its properties; *assembling*: constructing a part from other parts; *transforming*:adding or removing features to or from a part; *purchasing* and *selling*.

Notations

1. Π is the universe of part templates; Γ is the universe of work templates; $\subset$ or $\subseteq$ denote sub-set relationship; $\in$ denotes element relationship; ϵ denotes instance relationship; Γ^S denotes the powerset of Γ; Γ^ϵ ($\mathbf{G}^\epsilon$) denotes the set of instances of elements of Γ ($\mathbf{G}$); $\emptyset$ denotes the empty set $\{\ \}$; $\mathbf{P}, \mathbf{Q}, \mathbf{R}, \ldots \subseteq \Pi$ and $\mathbf{G}, \mathbf{H}, \mathbf{I}, \ldots \subseteq \Gamma$; $\mathbf{p}$,$\mathbf{q}$,$\mathbf{r}$, $\ldots \in \Pi$ and $\mathbf{g}, \mathbf{h}, \mathbf{i}, \ldots \in \Gamma$; $p_1, p_2, p_3, \ldots \epsilon\ \mathbf{p}$ ($p_1 \boxed{\models}$ a particular physical part, $\mathbf{p} \boxed{\models}$ a class of physical parts) and $g_1, g_2, g_3, \ldots \epsilon\ \mathbf{g}$.
2. The following functions are referred to in the definitions of Part and Work Templates:

(i) creation_works: κ: $\Pi \rightarrow \Gamma^S : \mathbf{p} \mapsto \kappa(\mathbf{p})$	(works that create a **p**-instance)
(ii) affecting_works: α: $\Pi \rightarrow \Gamma^S : \mathbf{p} \mapsto \alpha(\mathbf{p})$	(works that affect a **p**-instance)
(iii) usage_works: υ: $\Pi \rightarrow \Gamma^S : \mathbf{p} \mapsto \upsilon(\mathbf{p})$	(works that use a **p**-instance)
(iv) is_component: **p** is a *component iff* $\kappa(\mathbf{p}) = \emptyset$	(no work to create a **p** instance)
(v) is_product: **p** is a *product iff* $\upsilon(\mathbf{p}) = \emptyset$	(no work uses a **p** instance)
(vi) input_parts: ι: $\Gamma \rightarrow \Pi^S : \mathbf{g} \mapsto \iota(\mathbf{g})$	
(vii) affected_parts: β: $\Gamma \rightarrow \Pi^S : \mathbf{g} \mapsto \beta(\mathbf{g})$	
(viii) output_parts: ω: $\Gamma \rightarrow \Pi^S$: $\mathbf{g} \mapsto \omega(\mathbf{g})$	
(ix) is_consumer: **g** is a *consumer iff* $\omega(\mathbf{g}) = \emptyset$	(a **g** activity has no output)
(x) is_producer: **g** is a *producer iff* $\iota(\mathbf{g}) = \emptyset$	(a **g** activity has no input)

Definitions

1. A *Part Template* describes (all relevant) attributes of a single part (for production, design, etc.). It is 4-tuple: $(\tau, \kappa : \{\mathbf{g}_1, \ldots, \mathbf{g}_m\}, \alpha : \{\mathbf{g}_{m+1}, \ldots, \mathbf{g}_{m+n}\}, \upsilon : \{\mathbf{g}_{m+n+1}, \ldots, \mathbf{g}_{m+n+o}\})$.
2. A *Work Template* contains (all relevant) attributes of a single work.

It is 4-tuple: $(\tau, \iota : \{\mathbf{p}_1, \ldots, \mathbf{p}_k\}, \beta : \{\mathbf{p}_{k+1}, \ldots, \mathbf{p}_{k+l}\}, \omega : \{\mathbf{p}_{k+l+1}, \ldots, \mathbf{p}_{k+l+m}\})$.

3. Let $\mathbf{P} \subseteq \Pi$ and $\mathbf{G} \subseteq \Gamma$. The pair $(\mathbf{P}, \mathbf{G})$ is a *Part-Work Structure iff (i)* for all $\mathbf{p} \in \mathbf{P}$: $\kappa(\mathbf{p}) \subset \mathbf{G}$, $\alpha(\mathbf{p}) \subset \mathbf{G}$ and $\upsilon(\mathbf{p}) \subset \mathbf{G}$; *(ii)* for all $\mathbf{g} \in \mathbf{G}$:$\iota(\mathbf{g}) \subset \mathbf{P}$, $\beta(\mathbf{g}) \subset \mathbf{P}$ and $\omega(\mathbf{g}) \subset \mathbf{P}$.
4. On a Part-Work Structure $(\mathbf{P}, \mathbf{G})$ we define: *(i)* The parents and childs of a part (work): $\mathbf{p}, \mathbf{q} \in \mathbf{P}$. $\mathbf{p}$ is *parent part* of $\mathbf{q}$ and $\mathbf{q}$ is *child part* of $\mathbf{p}$ iff there is $\mathbf{g} \in \mathbf{G}$ such that $\mathbf{q} \in \iota(\mathbf{g})$ and $\mathbf{p} \in \omega(\mathbf{g})$. *(ii) Parent/Child Generations* (generations 1, 2, ...). *(iii) Ancestry* (of a part): union of all parent generations (parts in which the part can be used). *(iv) Usage Trace*: a sequence of work templates connecting a part to a parent. *(v) Usage Bundle*: the union of all usage traces. *(vi) Posterity* (of a part) : union of all child generations (parts used to create the part). *(vii) Creation Trace*: a sequence of work templates connecting a child to a part. *(viii) Creation Bundle*: the union of all creation traces.
5. Given a Part-Work Structure $(\mathbf{P}, \mathbf{G})$ then we can construct a *place/transition tree* for each *product* $(\mathbf{p})$ in $\mathbf{P}$: take a *place* for each part in the *posterity* of $\mathbf{p}$, and a *transition* for each work in it. A transition fires (and marks its parent) iff all its childs are marked.

4.2 Cell-Order Structure

Given a part-work structure, then the important activities in phenomena flow are *delivery, transformation and absorption of parts* (instances of part templates) and *execution of works* (instances of work templates). A cell is a system capable of sustaining the delivery. transformation and absorption of parts and/or the execution of works (transforming information or material in response to orders). Cells can be classified according to the works they can sustain. These then determine the parts (the inputs of the works) that the cell has to acquire and the parts that the cell can deliver or distribute (output parts). For the latter parts – and in a system of autonomous cells – the cell can receive orders to which it responds by sending orders for child parts – to other cells –, and after delivery of the child parts, by executing the appropriate works.

Terminology
1. A *cell* is a system capable of sustaining: *(i)* the execution of a number of activities. *(ii)* the receipt of the input parts for its activities; *(iii)* the sending of orders for input parts which it needs for its activities; *(iv)* the receipt of orders for the parts it can produce; and *(v)* the delivery to the order sender of the parts which it creates (in response to orders).
2. An *order* is a document that is sent by one cell (the sender) to another cell (the receiver). An order is sent to ask the receiver to deliver the ordered part to the sender of the order.

Remarks
1. The definition of cell and order requires reference both to the mathematical system (a cell's capabilities relate to the division of labour and the subsequent coordination of results) and to phenomena flow. In a sense we can say that an order thanks its existence to the division of labour.
2. Atsumi (1993) gives an elaborate introduction of the *order cell* concept and its link to the division of labour.
3. Cells can be classified according to the works they sustain. These then determine the parts (the inputs of the works) that the cell acquires and the parts that the cell delivers or distributes (output parts). For the latter parts – and in a system of autonomous cells – the cell can receive orders to which it responds by sending orders for child parts – to other cells –, and after delivery of the childs, by executing the appropriate works.

Notations

1. Σ is the universe of cell templates; Ω is the universe of order templates; $\mathbf{C}, \ldots \subseteq \Sigma$; $\mathbf{O}, \ldots \subseteq \Omega$; $\mathbf{c}, \ldots \in \Sigma$; $\mathbf{o}, \ldots \in \Omega$; $c_1, c_2, \ldots \in \mathbf{c}$; $o_1, o_2, \ldots \in \mathbf{o}$.
2. Some functions (relations) on Σ, Π and Γ:
   - *(i) capability* : $cap : \Sigma \rightarrow \Gamma^S : \mathbf{c} \mapsto cap(\mathbf{c})$
   - *(ii) inputs_to_cell_works* : $\iota^c : \Sigma \rightarrow \Pi^S : \mathbf{c} \mapsto \iota^c(\mathbf{c}) \stackrel{\text{def}}{=} \bigcup_{\mathbf{g} \in cap(\mathbf{c})} \iota(\mathbf{g})$
   - *(iii) affected_by_cell_works* : $\beta^c : \Sigma \rightarrow \Pi^S : \mathbf{c} \mapsto \beta^c(\mathbf{c}) \stackrel{\text{def}}{=} \bigcup_{\mathbf{g} \in cap(\mathbf{c})} \beta(\mathbf{g})$
   - *(iv) outputs_of_cell_works* : $\omega^c : \Sigma \rightarrow \Pi^S : \mathbf{c} \mapsto \omega^c(\mathbf{c}) \stackrel{\text{def}}{=} \bigcup_{\mathbf{g} \in cap(\mathbf{c})} \omega(\mathbf{g})$
   - *(v) materiality*: $mat : \Sigma \rightarrow \Pi^S : \mathbf{c} \mapsto mat(\mathbf{c}) \stackrel{\text{def}}{=} \omega^c(\mathbf{c}) \cup \iota^c(\mathbf{c}) \cup \beta^c(\mathbf{c})$
   - *(vi) part-work structure of the cell* **c**: $pw : \Sigma \rightarrow \Pi^S \times \Gamma^S : \mathbf{c} \mapsto pw(\mathbf{c}) \stackrel{\text{def}}{=} (mat(\mathbf{c}), cap(\mathbf{c}))$
3. Some functions and relations link Π, Σ and Ω:
   - *(i) receiver*: $\rho : \Omega \rightarrow \Sigma : \mathbf{o} \mapsto \rho(\mathbf{o})$
   - *(ii) sender*: $\sigma : \Omega \rightarrow \Sigma : \mathbf{o} \mapsto \sigma(\mathbf{o})$
   - *(iii) part_ordered*: $\pi : \Omega \rightarrow \Pi : \mathbf{o} \mapsto \pi(\mathbf{o})$
4. Some additional short hands:
   - *(iv)* **a***cquired (or absorbed) parts of cell* **c** : $ap(\mathbf{c}) \stackrel{\text{def}}{=} \{\mathbf{p} | \exists \mathbf{o} : \pi(\mathbf{o}) = \mathbf{p} \wedge \sigma(\mathbf{o}) = \mathbf{c}\}$
   - *(v)* **d***elivered parts of cell* **c** : $dp(\mathbf{c}) \stackrel{\text{def}}{=} \{\mathbf{p} | \exists \mathbf{o} : \pi(\mathbf{o}) = \mathbf{p} \wedge \rho(\mathbf{o}) = \mathbf{c}\}$
   - *(vi)* **t***ransformed parts of cell* **c** : $tp \stackrel{\text{def}}{=} mat(\mathbf{c}) \setminus (ap(\mathbf{c}) \cup dp(\mathbf{c}))$

Definitions

1. Given a part-work structure (**P**, **G**). A *Cell Template* describes (all relevant) attributes of a cell. These include templates for the parts that are acquired, transformed, and delivered by the cell and templates for the works the cell can sustain. It is a tuple: $(\tau, cap : \{\mathbf{g}_1, \ldots, \mathbf{g}_s\}, dp : \{\mathbf{p}_1, \ldots, \mathbf{p}_k\}, ap : \{\mathbf{p}_{k+1}, \ldots, \mathbf{p}_{k+l}\}, tp : \{\mathbf{p}_{k+l+1}, \ldots, \mathbf{p}_{k+l+m}\})$
2. Given a set of parts **P** and a set of cells **C**. An *order template* describes (all relevant) attributes of an order. These are: a part (for which production activities must be planned), the sender (cell) of the order, and the receiver (cell) of the order. It is a tuple: $(\tau, \pi : \mathbf{p}, \sigma : \mathbf{c}, \rho : \mathbf{d})$.
3. A *Cell-Order Structure* is a tuple (**C**, **O**) such that: *(i)* $\mathbf{C} \subseteq \Sigma$; *(ii)* $\mathbf{O} \subseteq \Omega$; *(iii)* for all $\mathbf{o} \in \mathbf{O}$: $\{\sigma(\mathbf{o}), \rho(\mathbf{o})\} \subset \mathbf{C}$; *(iv)* for all $\mathbf{c} \in \mathbf{C}$: there exists $\mathbf{o} \in \mathbf{O}$ such that $\mathbf{c} = \sigma(\mathbf{o}) \wedge \mathbf{c} = \rho(\mathbf{o})$.

In a cell-order structure all order templates concern cell templates of the set of cell templates, and all cell templates are involved in one or more order templates.

Remarks

1. Interaction among cells is by means of orders which request for the delivery of a part described by a part template, element of **P**. We restrict our treatment to orders as they are issued and handled in the computational flow of the cells, disregarding their effects in material flow.
2. It is only when considering the interflow and dynamic behaviour of a plant system that one must account for the full meaning of an order, i.e., that it is a request to create a (result) part *res* such that $\mathbf{p} \models res$ and that it can only be fulfilled when a sufficient number of all required input parts (as described by $\iota(\mathbf{g})$) are present) . Order fulfilment requires – in computational flow – the execution of the *creation bundle for the part* (as contained in the Part-Work structure (order explosion)). In phenomena flow it requires time, space, material, capability.

4.3 Plant Structure

To execute the work and produce/absorb the parts described in a part-work structure, one

needs a set of cells such that all works in the part-work structure are sustained. These and other requirements are stated (statically) in a *plant structure* in which a part-work structure and a cell-order structure are joined.

Definition

1. A *Plant Structure* is a tuple (**C**, **O**, **P**, **G**) such that:
   (i) (**C**,**O**) is a cell-order structure ;
   (ii) (**P**,**G**) is a part-work structure.
   (iii) The cells in **C** can sustain all works in **G**: $\bigcup_{\mathbf{c}\in\mathbf{C}} cap(\mathbf{c}) = \mathbf{G}$
   (iv) The cells in **C** can deliver and absorb, or affect all parts in **P**: $\bigcup_{\mathbf{c}\in\mathbf{C}} mat(\mathbf{c}) = \mathbf{P}$
   (v) **c** ($\in$ **C**) can receive **o** ($\in$ **O**) if and only if it can distribute $\pi(\mathbf{o})$, $\pi(\mathbf{o}) \in dp(\mathbf{c})$:
   $\rho(\mathbf{o}) = \mathbf{c}\ iff\ \pi(\mathbf{o}) \in dp(\mathbf{c})$
   (vi) **c** ($\in$ **C**) can send **o** ($\in$ **O**) if and only if it can absorb $\pi(\mathbf{o})$: $\sigma(\mathbf{o}) = \mathbf{c}\ iff\ \pi(\mathbf{o}) \in ap(\mathbf{c})$

4.4 Plant Systems over a Plant Structure

A large number of conditions to be satisfied by the dynamic behaviour of manufacturing systems are statically – with reference to template structures only – expressed in a plant structure. Plant systems, or computational flows over a plant structure, are defined irrespective of whether the system is interfaced to phenomena flow or not. The dynamic behaviour of a plant system – capable of simulating shop floor operations – is defined by the procedures (or programmes) which it – its cell instances – executes in response to order instances. Computational flows over a plant structure will instantiate, transform and delete instances of templates much in the same way as a computational algorithm works with dynamic variables of data types (e.g. integers, reals, booleans) and (invokes) functions and procedures transforming their values.

In programming, the notions of *encapsulation* and abstract data types allows one to separate internal and external structure and behaviour of computational objects. In analogy one can separate internal structure and behaviour of a plant system on the one hand and the behaviour of the plant system as embedded in a market system on the other hand.

In the computational flow over a given plant structure we propose a distinction between the *specification* of a plant system (in a market) and its implementation. The specification determines in response to which order instances the plant system will produce and absorb part instances, and to which cell instances orders should be addressed. The *implementation* of a plant system determines how cell *instances* will explode order instances and how they will schedule the work instances. Combining the concept of plant structure and the definition of an *abstract data type* in CLU data abstractions (Guttag & Liskov, 1986) one can define:

Definitions

1. A *plant system specification* is a tuple PSS:
   $(\mathbf{PS}, \{\mathbf{c}^{sales}, \mathbf{c}^{purchase}\}, \mathbf{O}^{sales}, \mathbf{O}^{purchase}, \mathbf{P}^{sales}, \mathbf{P}^{purchase}, PS, O\_h\_p)$ with:
   (i) **PS** is a plant structure (**C**, **O**, **P**, **G**) such that: $\mathbf{P}^{sales} \subseteq \mathbf{P}$ and $\mathbf{P}^{purchase} \subseteq \mathbf{P}$;
   (ii) $\mathbf{c}^{sales}$ and $\mathbf{c}^{purchase}$ are cell templates;
   (iii) $\mathbf{O}^{sales}$ is a set of order templates such that: *(i)* for each $\mathbf{o} \in \mathbf{O}^{sales}$:
   $\rho(\mathbf{o}) = \mathbf{c}^{sales}$; and (ii) for each $\mathbf{p} \in \mathbf{P}^{sales}$ there is $\mathbf{o} \in \mathbf{O}^{sales}$ with $\pi(\mathbf{o}) = \mathbf{p}$;
   (iv) $\mathbf{O}^{purchase}$ is a set of order templates such that: *(ix)* for each $\mathbf{o} \in \mathbf{O}^{purchase}$:
   $\sigma(\mathbf{o}) = \mathbf{c}^{purchase}$; and (ii) for each $\mathbf{p} \in \mathbf{P}^{purchase}$ there is $\mathbf{o} \in \mathbf{O}^{purchase}$ with $\pi(\mathbf{o}) = \mathbf{p}$;
   (v) PS is an unbounded mathematical set of *plant system states* over **PS**.

(vi) O_h_p is the specification of an order handling programme; its implementation will – in response to an order instance – cause the PS – *representation* to go through a number of states in order to make it deliver a part instance: $O\_h\_p : (\mathbf{O}^{sales})^{\epsilon} \times PS \rightsquigarrow PS \times (\mathbf{P}^{sales})^{\epsilon}$

2. A *Plant System Implementation* PSI for a PSS (as defined before) is a tuple:
   $(PS.rep, O\_h\_p.body)$ with:
   (i) $PS.rep$ is the representation of PS (including an initial marking of PS). It includes a set of cell-instances such that all elements in $\mathbf{P}^{sales}$ can be produced;
   (ii) $O\_h\_p.body$ is the body of the order handling programme; it will handle the instances of $\mathbf{O}^{sales}$; invoke the appropriate instances of $\mathbf{O}$ and of $\mathbf{O}^{purchase}$; and handle the instances of $\mathbf{O}$ such that part instances are "produced" by the cell instances and the plant system.

Furthermore there must be a market system MS and a *order dispatch programme* O_d_p which causes the market system to supply the part instances in response to the instances of $\mathbf{O}^{purchase}$:
$O\_d\_p : (\mathbf{O}^{purchase})^{\epsilon} \times MS \rightsquigarrow MS \times (\mathbf{P}^{purchase})^{\epsilon}$

Remark
"$\rightsquigarrow$" is to be read "leadsto", it expresses our concern with the transit states a plant system or a market system passes through in response to an order instance.

Response to a Single Order Instance
The body of the order handling programme $O\_h\_p.body$ transforms $PS.rep$ according to the order instance received: *(i)* If $(o_i \epsilon)$ $\mathbf{o} \in \mathbf{O}^{sales}$ then $O\_h\_p.body$ will find out which cell instances can produce the ordered part and issue an instance of the relevant $\mathbf{o}' \in \mathbf{O}$. *(ii)* if $\mathbf{o} \in \mathbf{O}$ then $O\_h\_p.body$ (or the cell's autonomous order handling programme) will build a place/transition tree (or P/T tree) according to the creation bundle of the part ordered. A P/T tree is *planted* for each order a cell receives. The tree is *grown* (with empty places) when orders are exploded (in accordance with a cell-order structure and cell capability). Because of auxiliary parts, as required for executing the works, or waste parts the P/T tree should be extended into a P/T net. Places are marked after parts have been created (in phenomena flow, or by a discrete event simulator). After firing a transition, a parent place is marked, and the transition and all its child places can be *pruned*). The P/T net is *uprooted* when the works are executed and the finished part (result) can be delivered. After growing a P/T net, it must be pruned in a Last-In-First-Out manner. *(iii)* if $\mathbf{o} \in \mathbf{O}^{purchase}$ then $O\_h\_p.body$ will invoke an order dispatch programme of the market system.

Terminology
(1) The term *(computational) pulse* (in response to an order received by a cell instance (the *cell pulse*) or a plant system (the *plant pulse*)) denotes the combination of the planting, growing (including auxiliary and waste expansions), pruning and uprooting of the corresponding P/T net). (2) The term *flow* denotes the integration (or combination) of the pulses in response to a number of orders (for various products). (3) Extending a P/T net (auxiliary and waste expanded from a P/T tree) in response to an order is called *pulse explosion.* The activity of reducing the P/T net, typically after accomplishment of work, is called the *pulse implosion.*

Remark
Autonomous cells. For simplicity of presentation all computational response for an order is executed by a single procedure ($O\_h\_p.body$). As the P/T net construction in a cell is completely local, one could have each cell scheduling its own activities.

The Integration of Responses to Order Instances
After identifying the response by a plant system and a cell instance to a single order instance (a pulse), one must investigate the combination (or *integration*) of responses to a number of sequential or concurrent order instances. A *computational flow* is an integration of computational pulses. Planning strategies enrich/refine the order response procedure *O_h_p.body* such that a cell instance, or the plant system as a whole can work for more than one order instance at a time. The work instances in response to different order instances must be interleaved.

A wide variety of planning and scheduling strategies are available according to context and application domain; e.g., the instrument maker: one person does all the work for one order in sequence (marking the places in the P/T net one after one), he may decide to accept a new order only after the result for the previous order has been delivered. If the instrument maker works with apprentices (who can only do some of the activities) he may start working for different order instances at the same time, and try to reduce idle time. Order handling strategies that interleave the work for several order instances are defined on top of sets (*forests*) of P/T nets, as additional constraints, while respecting the precedence constraints expressed in the creation bundle of each part.

4.5 Plant Interflow System

In this step one must account for the interflow of computational flow with the shop floor activities ($\boxed{\aleph \models}$). The requirement of interfacing the computational flow to the operations of a factory implies that the plant system implementation should allow one to distinguish all relevant – according to operations control and monitoring – states of the factory (which is statically modelled by some plant structure). The discrete event simulator which would mark, prune and uproot P/T nets in cell instances is replaced by phenomena flow which changes the net in step with shop-floor operations. Phenomena flow changes include the execution of work during designated time slots, the receipt of parts resulting from earlier work, and the forwarding of parts to be utilized in further work. In defining the interface between a plant system and a shop floor we must ensure that the work, part and order flows are ordered in a time-space-material consistent manner, that the work required can be performed by the cells, that the task descriptions include sufficient details, etc. All these issues must be reflected in the τ attribute of the templates and at the shop floor.

Terminology
(1) The term *inter-pulse* denotes the interactive and synchronized action – in response to an order instance – of a plant system and the shop floor. (2) The term *inter-flow* denotes the combined action – in response to a set of orders – of a plant system and the shop floor.

5 CONCLUSIONS AND FUTURE WORK

A conceptual framework for the definition of information and command services for manufacturing enterprises and industry has been explained. The conceptual and practical validity of the proposed approach must be tested further. Work in the near future should focus on: (a) the derivation of particular models from generic ones; (b) the elaboration of generic models for markets, teams and industries; (c) the development of model execution services; (d) software tool support for the derivation, interfacing, validation, execution and evaluation of particular models; and (e) software tool support for the specification and implementation of interflow mod-

els capable of interflowing with enterprise and industry dynamics.

6 REFERENCES

Alting, L. and Jorgensen, J. (1993) The Life cycle concept as a basis for sustainable industrial production. *Annals of the CIRP*, 42/1, 1993.

ATLAS (1993) *ATLAS; Architecture, methodology and Tools for computer integrated LArge Scale engineering, Public Project Overview*, February 1993, ESPRIT Project 7280, CEC, Brussels.

Atsumi, R. (1983) A social human activity model as an information infrastructure system. In Yoshikawa, H. and Goossenaerts, J., editors, *Information Infrastructure Systems for Manufacturing, IFIP Transactions B-14*, Amsterdam, 1993. Elsevier Science B.V.

Browne, J., Sackett, P. and Wortmann, H. (1995) Industry requirements and associated research issues in the Extended Enterprise. In this volume.

CEN/TC310/WG1 (1994a) CIM systems architecture – enterprise model execution and integration services – statement of requirements. CR1832, CEN/CENELEC, Brussels, Belgium.

CEN/TC310/WG1(1994b) CIM systems architecture – enterprise model execution and integration services – evaluation report. CR1831, CEN/CENELEC, Brussels, Belgium.

CIMOSA (1993) ESPRIT Consortium AMICE, editor. *CIMOSA: Open System Architecture for CIM.* Springer Verlag, Berlin, 2nd, rev. and ext. edition, 1993.

ENV 40 003 (1990) ENV 40 003: Computer integrated manufacturing – systems architecture – framework for enterprise modelling. European prestandard, CEN/CENELEC, Brussels.

Goossenaerts, J. (1993) Enterprise Formulae and Information Infrastructures for Manufacturing. In: Yoshikawa, H. and Goossenaerts, J. (eds.): *Information Infrastructure Systems for Manufacturing*, IFIP Transactions B-14, Elsevier Science B.V. (North Holland), Amsterdam.

Goossenaerts, J. and Bjørner, D. (1994) Generic models for manufacturing industry. Technical report no. 32, UNU/IIST, Macau, December 1994.

Hammer, M. and Champy, J. (1993) *Reengineering the Corporation.* Harper Collins Publishers, Inc., New York, 1993.

Hertz, H.R. (1894) *Die Prinzipien der Mechanik,in neuem Zusammenhange.* Johann Ambrosius Barth, Leipzig, 1894.

Liskov, B. and Guttag, J. (1986) *Abstraction and Specification in Program Development.* The MIT Press, Cambridge, Massachusetts, 1986.

Matthiesen, M. (1994) Synopses of ESPRIT III Project 6706: MS^{2}O - Multi-Supplier/Multi-Site Operations, CEC, Brussels.

The RAISE Language Group (1992). *The RAISE Specification Language.* Prentice Hall, New York 1992.

Scheer, A.-W. (1989) *Enterprise-Wide Data Modelling – Information Systems in Industry.* Springer-Verlag, Berlin - Heidelberg.

Spur, G., Mertins, K. and Jochem, R. (1994). *Integrated Enterprise Modelling.* Beuth Verlag, Berlin.

Williams, T.J. (1993) The Purdue Enterprise Reference Architecture. In: Yoshikawa, H. and Goossenaerts, J.(eds.) *Information Infrastructure Systems for Manufacturing*, IFIP Transactions B-14, Elsevier Science B.V. (North Holland), Amsterdam.

Yoshikawa, H. (1981) General design theory and a CAD system. In T. Sata and E. Warman, editors, *Man-Machine Communication in CAD/CAM*, Amsterdam, 1981. Elsevier Science B.V. (North Holland).