Современное представление об архитектуре систем ЧПУ типа PCNC

В.Л. Сосонкин, Г.М. Мартинов

Рассмотрена архитектура системы PCNC, обеспечивающая мобильность и коммуникабельность прикладных модулей, а также масштабируемость системы в целом. Проанализирована структура прикладной компоненты PCNC и введено представление о прикладном интерфейсе отдельных модулей. Приведена структура системной компоненты и ее коммуникационной среды. Представлен механизм взаимодействия модулей прикладной компоненты.

Введение

В среде разработчиков и производителей систем ЧПУ окончательно сложилось понимание того, что современные системы управления должны в максимальной степени использовать достижения компьютерной технологии [1, 2]. В системах ЧПУ нового поколения принято выделять системную платформу PC (Personal Computer) и прикладную компоненту NC (Numerical Control, т.е. ЧПУ); отсюда происходит и общее обозначение класса PCNC [3]. Системная платформа оказывает свои услуги модулям прикладной компоненты через прикладной интерфейс API (Application Program Interface) каждого модуля, причем API скрывает механизм реализации любых услуг. В системе PCNC поддерживается **мобильность** прикладных модулей (т.е. их переносимость на другие системные платформы); **коммуникабельность** модулей (т.е. их способность к взаимодействию через единую коммуникационную среду системной платформы); **масштабируемость** системы в целом (т.е возможность изменять, при необходимости, как функциональность прикладной компоненты, так и вычислительные возможности системной компоненты).

В рамках последующего изложения рассмотрены: структура прикладной компоненты; представление о прикладном интерфейсе API модулей прикладной компоненты; структура системной платформы и ее коммуникационной среды; взаимодействие модулей прикладной компоненты. Статья отражает результаты ряда европейских программ (например, OSACA, Open System Architecture for Controls within Automation systems, см.[4]); а также собственный опыт авторов при практической разработке компьютерных систем управления класса PCNC.

Структура прикладной компоненты PCNC

Для прикладной компоненты можно обозначить три уровня декомпозиции. Первый уровень состоит в выделении «задач управления» (task areas) [5]. В числе подобных задач: геометрическая (motion control), логическая (logic control), терминальная (human machine control) и, возможно, другие. Второй уровень декомпозиции состоит в выделении модулей в составе задач управления; причем каждый отдельный модуль соответствует фазе решения задачи управления. Так, в состав геометрической задачи управления входят: диспетчер режимов (manager); интерпретатор (ISO-процессор [6]); интерполятор; модуль управления следящими приводами (axes control). Часть таких модулей имеет собственный прикладной интерфейс API (рис.1), другие же объединяются в группы с целью построения общего интерфейса API. Примером подобной группы может послужить «канал ЧПУ» (channel),

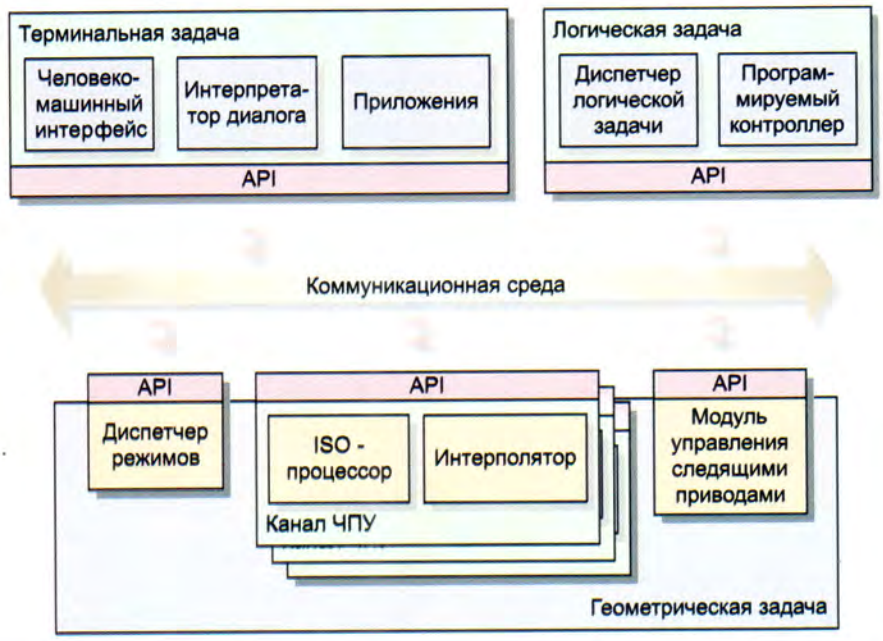

Рис. 1. Задачи и модули в архитектуре PCNC

объединяющий интерпретатор и интерполятор. Выделение канала особенно полезно в многоканальных системах ЧПУ, работающих с несколькими управляющими программами и обслуживающих несколько объектов-станков. Объединение модулей в группы снижает нагрузку на коммуникационную среду, однако при этом уменьшается гибкость системы ЧПУ и ее способность к реконфигурации.

Третий уровень декомпозиции наиболее глубок и, в принципе, не обязателен. Он означает выделение блоков в составе модулей. Блоки не имеют внешних интерфейсов API, а следовательно, и доступа к глобальной коммуникационной среде. При этом внутренняя организация модуля должна носить регулярный характер, а взаимодействие блоков модуля может осуществляться по внутренней локальной программной шине. Примером декомпозиции модуля может послужить реализация интерполятора в виде таких блоков: диспетчера интерполяции, блока опережающего просмотра кадров «look-ahead», блока разгона-торможения, блоков отдельных алгоритмов интерполяции (линейной, круговой, сплайновой).

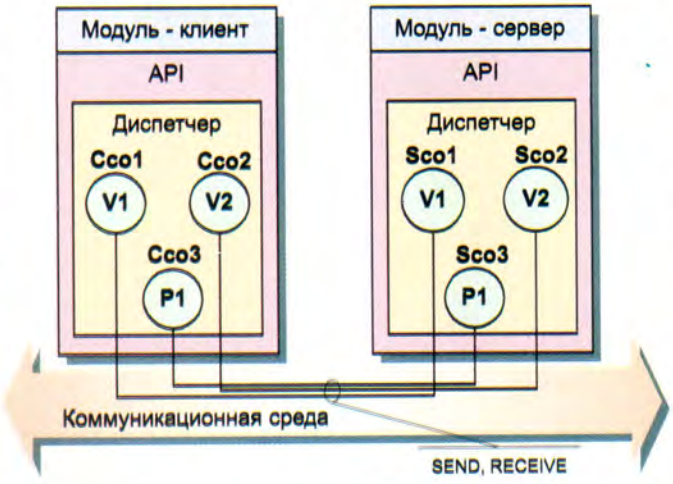

Рис. 2. Взаимодействие модулей посредством прикладного интерфейса API:
Cco (Client Communication Object) – объекты модуля-клиента;
Sco (Server Communication Object) – объекты модуля-сервера;
Cco1, Cco2, Sco1, Sco2 – объекты класса «объект-переменная»;
Cco3, Sco3 – объекты класса «объект-процесс»;
v(variable) – переменная; p(process) – процесс

Представление о прикладном интерфейсе API модулей прикладной компоненты

Определим функции API; при этом будем исходить из того, что каждый модуль реализован на основе объектно-ориентированного подхода с использованием Microsoft Visual C++. Будем полагать, что API модуля (или группы модулей) порождается объектами двух классов: классом **«объект-переменная»** и классом **«объект-процесс»** (рис.2). Каждый объект класса «объект-переменная» сопоставлен одной из переменных модуля (или группы модулей), а значение переменной в объекте прикладного интерфейса может быть предназначено или только для записи, или только для чтения, или для записи и чтения попеременно. Каждый объект класса «объект-процесс» сопоставлен одному конечному автомату, состояния которого есть состояния модуля (группы модулей), а переходы означают смену состояний. Переходы инициируются функциями внешних модулей или органов управления PCNC, а состояния сопоставлены переменным, предназначенным только для чтения.

Из такой структуры API вытекают следующие возможности:

- все модули (и группы модулей) могут взаимодействовать между собой с помощью собственных API через единую коммуникационную среду, представляющую собой по сути виртуальную шину [7];
- между любой парой модулей (включая, возможно, группу модулей) могут возникать клиент-серверные отношения; причем, модуль (или группа) может быть как клиентом, так и сервером;
- используя запросы типов SEND и RECEIVE, модули (или группа) могут посылать новые значения переменных другим модулям (или группам), читать значения переменных у других модулей (или групп);
- одни модули (или группы) могут управлять переходами других модулей (или групп) в новые состояния (например, сброса, готовности, запуска, работы, выхода из работы, ошибки и др.).

Все объекты в составе API имеют имена, структура которых: <аббревиатура модуля, группы или задачи><имя переменной или процесса>. Примеры объектов класса «объект-переменная»:

- mc_active_program_name (mc – motion control, т.е. геометрическая задача управления; active_program_name – имя текущей исполняемой программы; переменная служит только для чтения);
- mc_command_feedrate_override (mc – геометрическая задача; command_feedrate_override – значение коррекции величины подачи; переменная служит для записи и чтения);
- ac_current_axes_values_I (ac – axes control, т.е. модуль управления следящими приводами; current_axes_values – позиционная информация следящего привода подачи; I – номер координатной оси; переменная служит только для чтения).

Примеры объектов класса «объект-процесс»:

- mc_channel (mc – геометрическая задача; channel – имя процесса для группы модулей, включающих интерпретатор и интерполятор; состояния процесса – READY (готов), RUNNING (работает), ABORT (вышел из работы), ERROR (ошибка); переходы процесса – start, reset, abort, error);
- mcm_automatic_mode (mcm – motion control manager, т.е. диспетчер режимов; automatic_mode – имя процесса «режим автоматического управления»; состояния процесса – DESELECTED (режим не выбран), SELECTED (режим выбран), READY (управляющая программа выбрана), ACTIVE (управляющая программа отрабатывается), HOLD (управляющая программа остановлена), ERROR (ошибка); переходы процесса (actions) – select, deselect, reset, prepare, start, clear, stop, resume, abort).

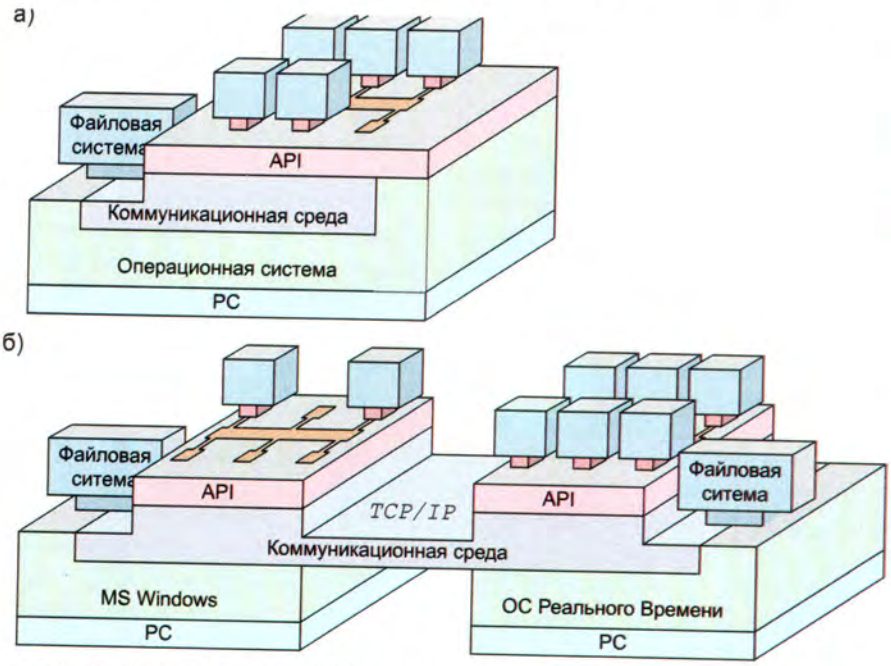

Рис. 3. Однокомпьютерный (а) и двухкомпьютерный (б) архитектурные варианты PCNC

Структура системной платформы и ее коммуникационной среды

В системную платформу PCNC входят: аппаратная часть, операционная система и средства поддержания межмодульной коммуникации.

Стандартную аппаратуру персонального компьютера расширяют за счет платы встроенного программируемого контроллера, а также интерфейсных плат контроллеров следящих приводов и электроавтоматики. Системы PCNC строят на основе одного или двух компьютеров (рис.3) [8]. Во втором случае компьютеры могут быть одинаковыми или разными; причем один из них «обращен» к оператору (терминал), а другой – к объекту управления (машина реального времени). В качестве операционной системы для однокомпьютерного варианта PCNC используют Windows NT с дополнительным расширением реального времени на уровне устройств (devices), к которым относятся контроллеры ввода-вывода. В двухкомпьютерном варианте PCNC применяют две разные операционные системы: Windows NT в терминале; одну из операционных систем реального времени в машине реального времени (UNIX, VxWorks или др.).

Структуру коммуникационной среды PCNC (рис.4) удобно сопоставить с классической многоуровневой моделью архитектуры открытых систем ISO-OSI [9]. В коммуникационной среде принято, согласно OSACA, выделять четыре слоя.

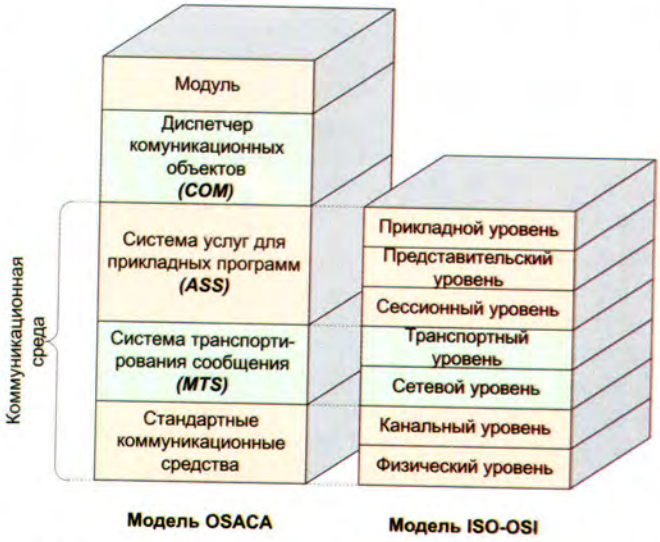

Рис. 4. Сопоставление структур коммуникационной среды PCNC и модели ISO - OSI

- Внизу находятся стандартные коммуникационные средства, которые охватывают физический и канальный уровни модели ISO-OSI. Для двухкомпьютерного варианта подобные средства могут быть реализованы на базе стандартных протоколов Intranet.
- Выше следует уровень MTS (Message Transport System, система транспортирования сообщений); этот уровень выполняет функции сетевого и транспортного уровней модели ISO-OSI. Для двухкомпьютерной системы PCNC в уровень MTS может быть встроен четырехуровневый стек протоколов TCP/IP. Система транспортирования сообщений MTS поддерживает базовые транспортные функции (SEND, RECEIVE) для обмена различными сообщениями между разными парами модулей. MTS скрывает от прикладной части PCNC как стандартные средства коммуникации (такие, как TCP/IP или др.), так и все функции операционной системы, в которых нуждается верхняя часть коммуникационной среды. Модули, расположенные в разных компьютерах, взаимодействуют с помощью MTS так, как если бы они находились в одном компьютере. MTS использует протокол с предварительным установлением соединения. Это означает, что два взаимодействующих MTS-пользователя должны сначала установить соединение, а уже затем пользоваться им.
- Выше расположен уровень ASS (Application Services System, система услуг для прикладных программ); этот уровень соответствует сессионному, представительскому и прикладному уровням модели ISO-OSI. Система ASS предоставляет услуги (с предварительным установлением соединения) прикладным модулям PCNC следующим образом. Если модуль нуждается в удаленной связи, то ASS формирует сообщение, включающее полученную от модуля информацию. Далее вызывается услуга уровня MTS для посылки сообщения. На стороне приема ASS интерпретирует сообщение и передает его модулю-получателю. Перед передачей сообщения ASS подготавливает данные в такой форме, в которой нуждается принимающая система. Если два модуля (или несколько модулей) имеют общую ASS, то они не нуждаются в MTS для поддержания интерактивности.
- Четвертый уровень лишь условно относится к коммуникационной среде – это прикладной интерфейс всех модулей NC-подсистемы, имеющих API. Четвертый уровень реализован как диспетчер коммуникационных объектов COM (Communication Object Manager), который ассоциирован с каждым модулем, имеющим API. Диспетчер COM представляет собой оболочку, наполненную программами, предназначенными для создания, уничтожения и активизации так называемых «коммуникационных объектов».

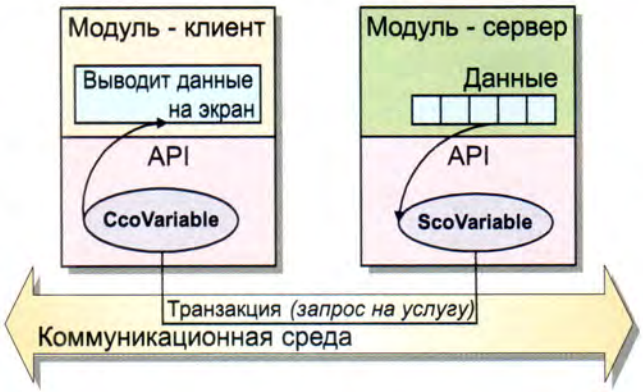

Рис. 5. Пример доступа клиентского приложения к данным сервера

Взаимодействие модулей прикладной компоненты

Внешние интерфейсы прикладных модулей представляют собой коллекции коммуникационных объектов, погруженных в среду диспетчера COM. Коллекции порождаются двумя классами (упомянутыми выше), формирующими API (классы «объект-переменная» и «объект-процесс»); а также и двумя дополнительными классами:

- «объектом-событием» для автоматического уведомления об изменении значений переменных при асинхронных транзакциях (сессиях) между клиентом и сервером;
- «объектом-транзакцией» (только на стороне клиента), содержащим функции обработки данных, получаемых клиентом в свои объекты класса «объект-переменная» (вывод на экран, вычисления и др.).

При таком построении внешних интерфейсов они легко доступны как у клиента, так и у сервера. Если клиент желает получить доступ к информации сервера (прочитать, записать), то должен быть предусмотрен коммуникационный объект Cco (Client communication object) на стороне клиента и этот объект должен соответствовать серверному коммуникационному объекту Sco (Server communication object). Должен быть, по крайней мере, один Cco, запрашивающий услугу у Sco. Cco разных прикладных модулей могут быть одновременно связаны с одним и тем же Sco.

Например, серверный коммуникационный «объект-переменная» ScoVariable хранит данные соответствующей серверной переменной. У каждого клиента имеется другой коммуникационный «объект-переменная» CcoVariable, используя который клиентское приложение имеет прямой доступ к серверным данным в ScoVariable и возможность отображения у себя этих данных (рис.5).

Точно также коммуникационный «объект-процесс» ScoProcess на стороне сервера полностью определяет автомат состояний сервера со всеми серверными переходами (рис.6). Коммуникационный «объект-процесс» CcoProcess на стороне клиента используется для управления автоматом состояний путем вызова предусмотренных действий (actions).

Взаимодействие клиента и сервера носит характер синхронной или

асинхронной транзакции (рис.7). При посылке синхронного запроса, запрашивающий Cco ожидает ответа от Sco и остается заблокированным, пока не поступит подтверждение от запрашиваемой услуги. После отправления асинхронного запроса приложение продолжает свою работу, а ответ от сервера поступает либо по мере его готовности, либо по событию. Последнее означает, что клиент желает быть уведомленным сервером при всяком изменении Sco-Variable или ScoPro-cess. В этом случае на стороне сервера динамически (т.е. в процессе работы) создается «объект-событие» ScoEvent. Его создание инициируется «объектом-событием» CcoEvent, в котором указаны наблюдаемая переменная или наблюдаемый процесс. Бывает так, что во время асинхронной транзакции ответы сервера должны как-то обрабатываться у клиента. В этих случаях создается специальный дополнительный объект – «объект-транзакция» CcoTransaction.

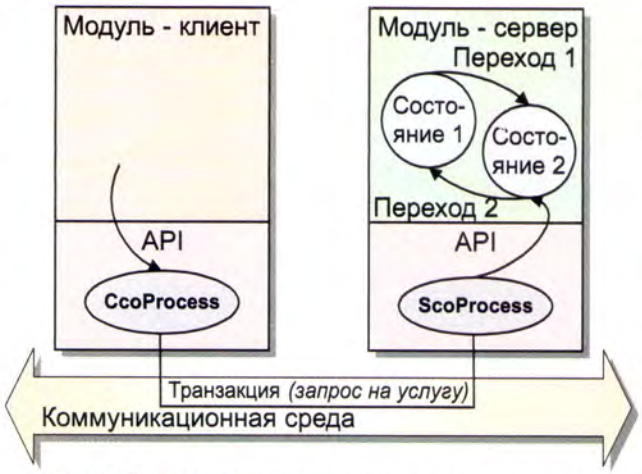

Рис. 6. Пример управления со стороны клиентского приложения переходами между состояниями сервера

Заключение

Изложенный подход к архитектуре систем PCNC позволяет реализовать свойства мобильности, коммуникабельности и масштабируемости. Очень важно, что для каждого модуля NC-подсистемы может быть разработан сколь угодно богатый прикладной интерфейс API, поскольку функциональные возможности систем PCNC в целом в значительной степени определяются полным множеством интерфейсных функций всех прикладных модулей. Очень важно и то, что при встраивании нового модуля в систему PCNC на долю прикладного программиста падает лишь небольшой объем работы, связанный с определением интерфейса (по вполне очевидной методике) и внесением некоторых изменений в систему конфигурации.

Рассмотренный подход был использован авторами при разработке прототипа системы ЧПУ (нового поколения) типа PCNC. Накопленный опыт относится как к разработке большого числа различных прикладных модулей, так и к организации коммуникационной среды и системы клиент-серверных отношений. Имеется и позитивный опыт использования COM-технологии (Component Object Model).

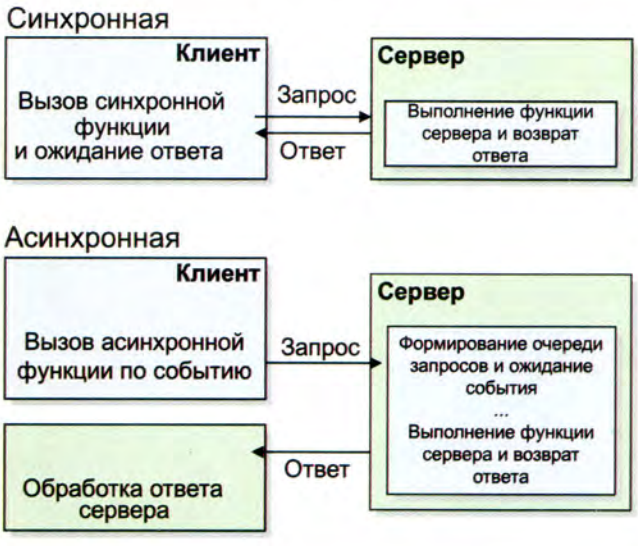

Рис. 7. Синхронная и асинхронная транзакции

Авторы готовы к контактам: тел. 972-9449, факс 9721873, E-mail ncs@postman.ru

Литература

1. ***Соломенцев Ю.М., Сосонкин В.Л., Мартинов Г.М.*** *Построение персональных систем ЧПУ (PCNC) по принципу открытых систем.—Информационные технологии и вычислительные системы. 1997, №3, с. 68-75.*
2. ***Сосонкин В.Л., Мартинов Г.М.*** *Принципы построения систем ЧПУ с открытой архитектурой. – Приборы и системы управления. 1996, №8, с. 18-21.*
3. ***Сосонкин В.Л.*** *Концепция системы ЧПУ на основе персонального компьютера (PCNC). – Станки и инструмент. 1990, №11, с. 9-14.*
4. ***Pritschow G., Spur G., Weck M.*** *Komponenten und Schnittstellen fur offene Steuerungssysteme.Munchen; Wien: Hanser, 1996. p.216.*
5. ***Сосонкин В.Л.*** *Задачи числового программного управления и их архитектурная реализация. – Станки и инструмент. 1988, №10, с. 39-40.*
6. ***Сосонкин В.Л., Мартинов Г.М.*** *Концепция геометрического ISO – процессора для систем ЧПУ. Станки и инструмент 1994, №7, с. 17-20.*
7. ***Сосонкин В.Л.*** *Концепция персональных систем управления в реальном времени. – Приборы и системы управления. 1995, №12, с. 16-18..*
8. ***Сосонкин В.Л.*** *Принципы построения персональных систем ЧПУ с открытой архитектурой. Труды междун. конф. «Информационные средства и технологии, 21-23 окт. 1997 года». М.: Междун. Академия Информатизации. 1997, с. 154-159.*
9. ***Rembold U., Nnaji B.O., Storr A.*** *Computer Integrated Manufacturing and Engineering. First printed in England. Addison-Wesley Publishing Company. 1993. P.P. 640.*