Interface and Control Manager for Parcels Sorting

Atanas Atanassov

Abstract: *The paper presents the development of an Interface and Control Manager (ICM) intended to sort automatically parcels in Swiss POST. The main purpose of the ICM is to replace existing IBM sorting system, providing better sorting quality. ICM uses complete or partial address data provided by SIEMENS Parcel Address Reader (PAR). It verifies the data and sends appropriate control information to all existing IBM servers and sorting machines. For parcels dispatching multithreading and scheduling approaches similar to these used in real-time and parallel systems are applied.*

Key words: *Multithreading, RTOS-Scheduling, Parallel Systems, Object- Oriented Programming.*

INTRODUCTION

The Interface and Control Manager (ICM) provides the interfaces of the SIEMENS Parcel Address Reader (PAR) adapted to the IBM parcel processing environment PP2000 used by Swiss POST. The PAR is the replacement of the IBM Automatic Coding System (ACS).

The purpose of ICM is to sort parcels efficiently. ICM communicates with proprietary Siemens OCR algorithms (know how) to determine the receiver's address from the parcel surface. To be able to sort parcels correctly, ICM interacts with external IBM servers (see Figure 1): Alive Monitor (AVM), Management and Database Server (MDS), Operational Control System (BCS), Central Data Server (ZDS), Coding Line Control (CLC), Video Coding System (VCS) and Parcel Data Server (PDS).

To be able to exchange information with external servers, ICM must conform to specific message protocols as given in following paragraphs.

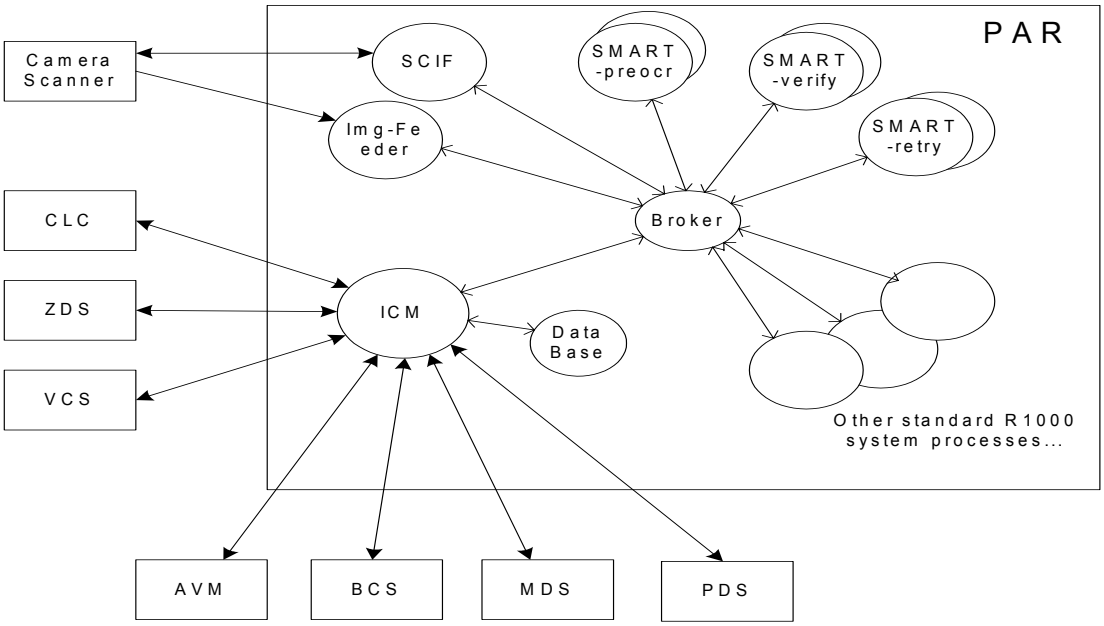

Figure 1: ICM – PAR top-level process view

FUNCTIONAL DESCRIPTION

The Functionality provided by ICM is related to parcel sorting process and includes:

- Control of CLC in order to coordinate the movement of the parcels on the transportation belt (stop, release parcels before scanning point, direct them to cross switch unit directing the parcel to other conveyor belts)
- Communication with PAR barcode and address reading processes in order to receive parcel barcodes, address data and dimensions.

- When an identcode (unique parcel barcode) is available to check for available address data on ZDS and if there is to use it to sort a parcel.
- If there is no address data on ZDS to use address provided by the PAR and to use it to sort a parcel.
- If PAR is not able to provide address data to send VCS tile, VCS Address Block Coding (ABC) or VCS Full Address Coding (FAC) requests.
- To use VCS or MCS address data to sort a parcel, in case PAR is not able to find parcel address.
- To download specific address tables from BCS (on request from MDS) and to use them to refine sorting process.
- To report any changes in the status or errors to MDS
- To report the meaning of some specific internal parameter, related with the parcel sorting process, as well to change them on request from MDS
- Periodically to send alive messages to AVM and ping and to all other systems
- To store and send statistical data (about sorted parcels) to MDS
- To store and provide to PDS parcels' barcode(s) and address images.

DESIGN OVERVIEW

As it told above the main purpose of ICM is to replace the existing ACS. ACS is constructed as parallel machine including 4 independent PC supporting the automated address recognition and one PC responsible for the communication with the external systems.

ICM is developed as a NT process which consists of a set of intercommunicating objects using threads to signal each other for different events. They implement the interfaces to the existing external IBM systems. Communications to the external systems are socket based using messages with different structure [1]. Internal communication and synchronization between the ICM objects are based on events and critical sections.

Before taking a decision to use the multithreading model of ICM some other models were analyzed. It turned out that the model of ICM consisting of one process with some independent threads was more efficient than the model of building ICM as a number of intercommunicating processes because the inter-process communication is more time consuming than inter-thread communication. The same was valid for the model based of a pool of threads some of which serving the external communications and others related with each processed parcel.

Inside ICM process the following objects and threads, listening for their corresponding events exist:

- ICMMain – object that performs the needed tasks to sort parcels.
- AVMTalk – object handling the communication with AVM monitor
- MDSBCSTalk – object handling the communication with MDS and BCS systems
- ClcCon – thread responsible to support the interface to CLC.
- RfCon - object implementing the interface to RF.
- MessageDispacher – dispatches PP2000 messages
- PDSTalk – object handling the communication with PDS server
- ImageHandler – handling and deleting images needed for ABC and FAC requests.

Additional special purpose objects exist:

- SharedMem - shared memory for all ICM objects.
- ParcelProcessingTable - table of parcels that are currently processed by ICM
- Statistic - statistic storage and generation class

IMPLEMENTATION DETAILS

In general ICMMain thread is responsible for:

- Creation of shared memory object that stores common data specific for all ICM objects
- Creation of parcel processing table that records the data for all currently coded parcels
- Creation of statistical object that collects data for already processed parcels
- Creation of the image handler object supporting VCS and PDS systems
- Downloading and updating of tables needed from PAR or for PDS processing
- Buildings of all other objects described below that are responsible for normal parcel processing and for communications with the external systems.

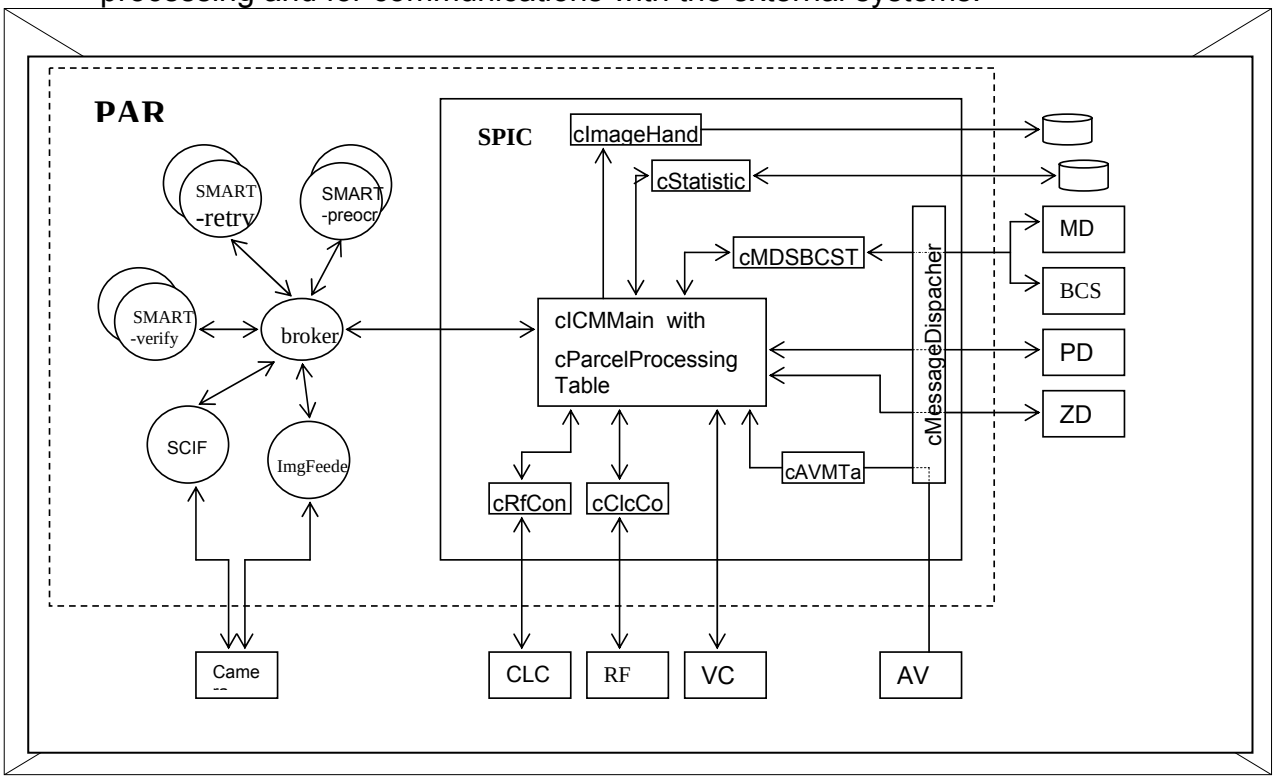

Figure 2: SPICM internal objects view

The following pseudo code of ICMMain shows the basic structure of the implementation:

```
Create and initialize SharedMem, Statistic, ImageHandler
Create sockets to communicate with external systems
while (icm_is_alive) {
   wait for event
   analyse event
   switch (event) {
         BROKER:        analyze OCR results
         ZDS:           SRSP as reply to QSR received
         PDS:           message from PDS arrived(ACK1/ACK2 or PING)
         CLC:           new parcel arrived, create new SingleParcel
         VCS:           VCS result arrived (ABC or FAC)
         AVMTalk:       do nothing (AVM does not send any message)
         MDSBCSTalk:    handle ENA, DISA, Update various tables
         KEYBOARD:      operator pressed a key at the keyboard
    }
} //end of while (icm_is_alive)
Shutdown connections
Free memory for the SharedMem, Statistic, ImageHandler and the sockets
```

The Broker thread / object communicates with the PAR processes from which it obtains complete or partial parcels data (Identcode, address, dimensions, images of the address and other ROIs, etc).

The responsibilities of AVMTalk can be summarized as establishing of TCP/IP connection to AVM using AVM IP address and port number written in ICM.ini file; sending HELO message to initiate message protocol; periodically to send AVE (alive) messages to AVM. If a communication to AVM is disrupted then AVMTalk notifies ICMMain and tries to reestablish the connection.

MDSBCSTalk performs the activities to establish TCP/IP connection to MDS and BCS using MDS/BCS IP addresses and port numbers; it sends HELO message to MDS and BCS to initiate the message protocol. Periodically MDSBCSTalk sends PING messages. It has to answer or to send messages from/to MDS or BCS related with:

- setting of the status of ICM (MDS can enable or disable the normal processing of ICM)
- Reporting of any changes of the status to MDS
- Event signaling
- Parameter handling (answering to queries for parameter values from MDS, or setting new values for some of them or all)
- Supporting the operations with table distribution and activation
- Delivery of statistical data to MDS on request

If communication to MDS or BCS is disrupted then MDSBCSTalk notifies ICMMain and tries to reestablish the connection.

ClcCon handles the communication to CLC. The steps performed by ClcCon are to establish RS232 connection to CLC using the CLC COM number. After that to initiate communication to CLC, to send periodically "echo" messages. To control parcels on CLC belt using the messages as "Parcel at Release Point", "Parcel Stop" and "Parcel Data (ZIAC)" message. If communication to CLC is disrupted ICMMain is notified

PDSTalk handles the communication to PDS by establishing TCP/IP connection to PDS using PDS IP address and port number. It sends HELO message to PDS to initiate message protocol. Periodically sends PING messages. It sends RPP messages to inform PDS where the data related with the big customers is placed (the parcel, barcodes and address images). Mentioned images are prepared by the object ImageHandler and their path is provided to PDS in order to be downloaded. When communication to PDS is disrupted then PDSTalk notifies ICMMain and tries to reestablish the connection.

The ICM process, as already mentioned, consists of a number of communicating objects. SharedMem is the object used to keep information that is accessed from all other objects – parameters, events, status, flags. Thus SharedMem is global memory storage for all objects inside ICM. There exists only one SharedMem object. It uses in an internal reference counting mechanism to keep track of the number of other objects that have access to it. When one needs to use SharedMem he can access the object by calling SharedMem::GetPtrSharedMem(). If the object already exists – pointer to it is returned and number of references is increased. If the object dos not exist – it is created and number of references is set to 1. When the SharedMem object is no longer needed, then
void cSharedMem::ReleasePtrShareMem(SharedMem*& ptr)

ReleasePtrShareMem() only decreases number of references to the current object. If the number of references reaches 0, then SharedMem the object is freed. Because there

exists only one SharedMem object, when one thread changes the value of some SharedMem parameter, then all other threads will use the updated value.

The base class ConnectFlags for SharedMem plays special role. When connection to some system is broken then an event (ConnectFlags:: GetConnectedHandle()) is signaled. For example: If connection to AVM is broken then AVMTalk thread will signal the event in ConnectFlags and ICMMain thread will be notified.

SharedMem:: AreICMConnectionsReady() returns true if all required connections are established. If connections are established and ENA (enable message) from MDS is received then parcel processing is allowed.

The ParcelProcessingTable object contains all required information for currently processed parcels. ParcelProcessingTable is a container of a number of SingleParcel objects.

For each parcel that is being processed ICM creates and maintains a SingleParcel object. The SingleParcel has all the information that was gathered during parcel coding. The information contained within SingleParcel is about the various time stamps as - time when CLC ID was received, time when binary image was received, - time when barcodes data is received or time when parcel geometry information was obtained, as well time when automatic parcel decoding is finished, and QSR to ZDS was sent, etc. Other important parcel data is CLC ID data, PAR ID data, parcel Identcode data, parcel dimensions, zip code, street number, house number, type of the address, etc.

Only ICMMain uses ParcelProcessingTable during processing of parcels according to the postal coding logic. ParcelProcessingTable is organized and can be interpreted as scheduler queue. It is dispatched in the same way as the RTOS dispatcher works with the tasks queue. As soon as one task is finished it is discarded from the queue. The same is with the parcels. When the parcel data is finalized (completed) this data is sent to CLC and ZDS and the parcel object corresponding to this parcel is deleted from the table (queue).

Every object/thread that sends IBM PP2000 messages uses MessageDispacher to send them. Because CLC, Broker and VCS do not conform to PP2000 message formats and protocols they do not need to use MessageDispacher to send messages – they use directly ICMSock and ClcCon objects.

The structure of a typical message is built of a message header and several message tags with fixed or flexible structure of the fields. For example the following message QSR (Query for Transmission Records) has a format:
QSR ^10^0070^**FRA-ACS37^FRA-ZDS** ^001856^IDCOD^0019^9942127694000391 95^
The message header QSR ^10^0070^FRA-ACS37^FRA-ZDS ^001856^, describing the name of the message (**QSR**), the sender station (**ACS37)** and the receiver (**ZDS)**, as well the transaction ID (**001856)**. It has one tag **IDCOD**^0019^9942127694000391 95^ providing the 18 digits identcode number in the field next to the name of the tag (**IDCOD**).

Message that has to be sent is using the MessageDispacher::SendMsg() method. The message dispatcher saves the transaction ID of the message in an internal table. Every object can send simultaneously number of messages to external systems. The object needs to know for which outgoing message, what is the reply from the external system (ACK or NAK). When a reply from external system is received, the object calls IsReply() method to check for which message this is a confirmation. When IsReply() is called it searches its internal table with already sent messages and checks to see if there is message with matching Transaction ID.

For example if we send HELO message to MDS, MessageDispacher keeps the message because it might need to retransmit it.
MessageDispacher performs these functions automatically:

- sends the message through the socket
- if reply is not received (i.e. IsReply() called) in the defined time frame the message is retransmitted again. If the message is retransmitted to N ways (where N is a parameter from SharedMemory table) then the message is marked as :EXPIRED.

MessageDispacher handles special cases with QSR, FTP2 and RPP messages when instead of ACK appropriate confirmation message is used (to QSR an answer is SRSP).

Through the Statistic class the statistics for a parcel are saved and various statistics request are made. The function Statistic::SaveStatistics(SingleParcel* parcel) is called just before the record for the parcel is freed from ParcelProcessingTable. This function inserts into several log files. The data from the statistical log files is required periodically by MDS and can be used on the management level. Statistical information is collected for the period of time defined by parameter and after this period old data is forgotten.

The object ImageHandler is responsible for storing and deleting the parcels image data needed for VCS ABC or FAC coding or from the PDS. It supports mechanism to store image data for period of time defined as a configuration parameter. All images older than this time frame are deleted.

CONCLUSIONS AND FUTURE WORK

The presented ICM module passed the acceptance tests and ensures the needed by the customer quality of parcels sorting. It is able to sort from 83.5% to 99.5% of the input parcels automatically, depending on their shape and surface. Using the accumulated parcels statistical data some online readjustments of the CLC speed can be obtained. This way the CLC speed can be accelerated up to 25% that means more processed parcels per hour. It will be done in the near future as a part of next assignment. At the moment ICM is working in Swiss (Frauenfeld POST sorting center).

REFERENCES

[1] IBM PP2000 Project Message Format Description, IBM Corporation, 2001.
[2] IBM PP2000 Interfaces Communication. IBM Corporation, 2001.
[3] Ruediger A., Using Multithreading and C++ to Generate Live Objects, Microsoft Developer Network Technology Group, 1993

ABOUT THE AUTHOR

Senior Lecturer, Atanas Atanassov, MSc, Department of Computer Sciences, University of Chemical Technology and Metallurgy, Sofia, Phone: +359 02 6254 624, E-mail: naso@uctm.edu.