PhUSE 2006

Paper AD12

The Submission Data File System Automating the Creation of CDISC SDTM and ADaM Datasets

Marcus Bloom, Amgen, Thousand Oaks, USA
David Edwards, Amgen, Thousand Oaks, USA

ABSTRACT

A number of yeas in the making, the SDF System represents the most ambitious software development project ever undertaken within the Biostatistics Department at Amgen. From its inception it was designed to tackle many of the difficulties that have plagued statistical programming ever since the first programs were developed to analyze clinical trial data. The system scope includes the following areas:

- Automating front-end and back-end data security
- Providing rapid access to raw and derived datasets to all sites within the organization
- Implementing large scale systems design and software engineering principles in the clinical trials programming domain to reduce the software development effort required for each statistical analysis
- Modeling the complex relationships inherent within clinical data to provide sophisticated impact analysis tools with the capability to manage and cascade change
- Implementing an object-oriented (OO) approach to the definition and reuse of clinical data standards
- Improving review and quality control processes by coupling the specification of a program together with it's source code to create one logical entity
- Reducing the validation burden (through large-scale reuse) while at the same time increasing overall quality

INTRODUCTION

The Submission Data File System (SDF) was designed to support the development, validation, maintenance and execution of SAS programs required to create Submission Data Files (SDFs).

The system leverages a number of key technologies (SAS, Java and Oracle) to bring the following benefits:

- Enables collaborative working when Study Programmers are located in different sites (and countries)
- Enforces a hierarchical directory structure that is guaranteed to be consistent across development sites, data standards, therapeutic areas, products, indications, studies and analyses
- Accelerates the extraction of study data from the Clinical Data Management System (CDMS) by utilizing parallel processing technology
- Controls data access by automating the management of UNIX security groups
- Encodes data standards in metadata to drive the generation of SAS programs
- Includes a powerful Publish and Subscribe feature, which enables Standard Templates to be set up for use on many analyses
- Resilient to changes in both internal and CDISC standards
- Reduces the time and resources required to produce SDFs
- Automatically generates quality control checks into each program
- Provides a common environment and a consistent technical architecture which facilities the transition of staff from one clinical project to another
- Ensures no lasting dependency between the system itself and the deliverables it produces. Once an analysis has been locked, all the components generated by the system are completely independent from the SDF application itself. In other words, the system is not required to review and/or execute the programs in the future.

A BESPOKE CLINICAL DATA EXTRACT, TRANFORM AND LOAD TOOL (ETL)

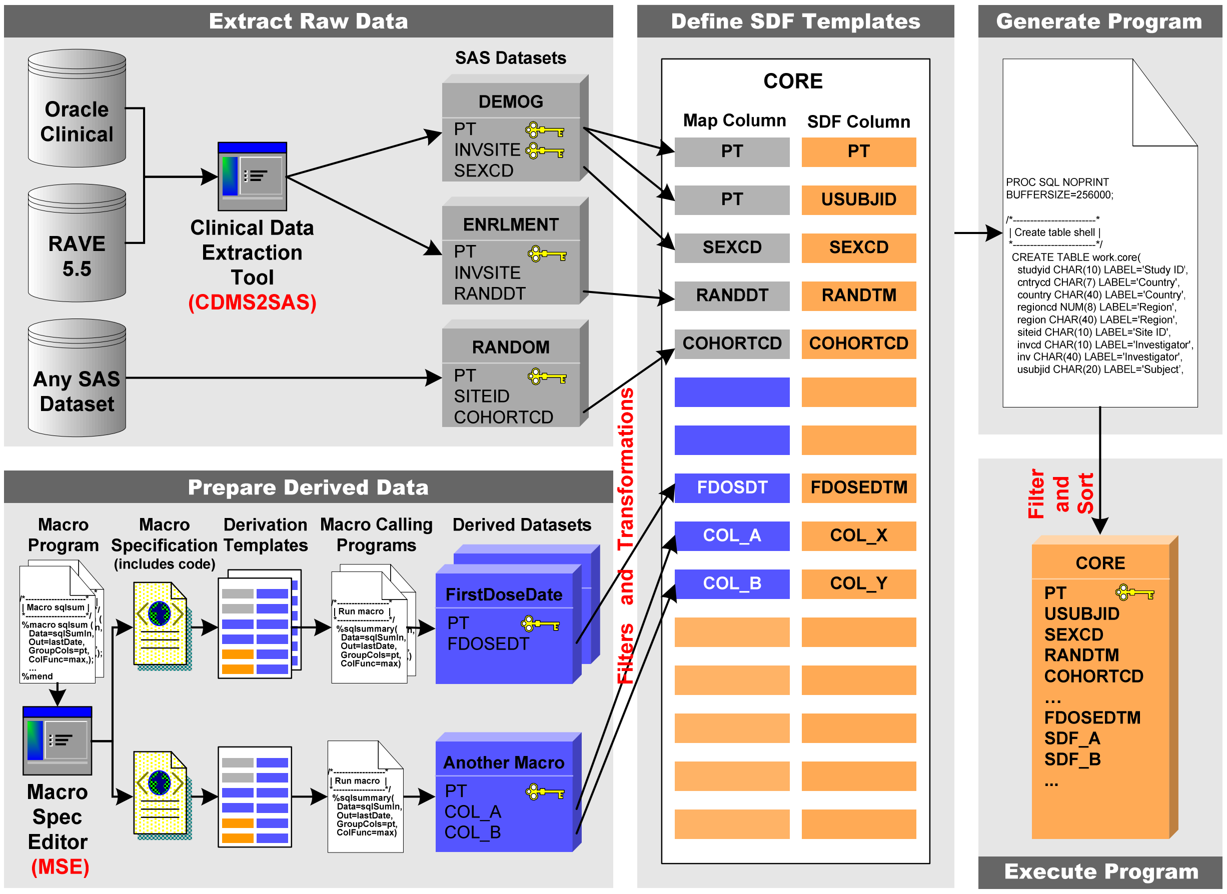
Extract refers to the process of getting the raw data from your CDMS in a form that is accessible by those that are going to use it.

Transform refers to the process of transforming the data into structures that can be sent to the FDA as well as deriving key variables for analysis.

Load refers to the process of loading the raw and transformed datasets into a structure that is suitable for long-term storage and access.

HOW DOES THE SDF SYSTEM ACCOMPLISH THESE 3 TASKS?

Figure 1: Schematic Diagram of the SDF Creation Process



```
/*--------------------*
 | Macro sqlsum |
 *--------------------*/
%macro sqlsum (
   Data=sqlSumIn,
   Out=lastDate,
   GroupCols=pt,
   ColFunc=max,);
...
%mend
```

```
/*------------------*
 | Run macro |
 *------------------*/
 %sqlsummary(
   Data=sqlSumIn,
   Out=lastDate,
   GroupCols=pt,
   ColFunc=max)
```

```
PROC SQL NOPRINT
BUFFERSIZE=256000;

/*----------------------*
 | Create table shell |
 *----------------------*/
  CREATE TABLE work.core(
    studyid CHAR(10) LABEL='Study ID',
    cntrycd CHAR(7) LABEL='Country',
    country CHAR(40) LABEL='Country',
    regioncd NUM(8) LABEL='Region',
    region CHAR(40) LABEL='Region',
    siteid CHAR(10) LABEL='Site ID',
    invcd CHAR(10) LABEL='Investigator',
    inv CHAR(40) LABEL='Investigator',
    usubjid CHAR(20) LABEL='Subject',
```

The **extract** process is handled by the **Clinical Data Extraction Tool (CDMS2SAS)**. Using parallel processing technology, it extracts study data from both Oracle Clinical and Medidata's Rave EDC system and stores it in the SDF System's directory structure. Not only is the directory structure standard across therapeutic areas, products, indications and studies, but the system also automates front end and back end data security using role based security profiles and automatic UNIX security group management. The system is not limited to using raw CDMS data. SAS datasets can also be imported into the SDF directory structure.

The **transform** process is handled by a number of system components:

- The **Macro Specification Editor (MSE)** allows complex data derivations to be specified, developed, tested and released into the SDF System. The MSE creates an XML document that encapsulates the, requirements, design, run time parameters and macro source code. During the development process it is possible to place the macro under source code control and check it in and out as development proceeds. Once the macro has been tested, the Macro Specification file is released into the SDF directory structure. Once released, it can be used to create many Derivation Templates, which essentially generate calling programs to execute the macro for a given analysis. Executing a Derivation Template creates a derived dataset, which can then be used in the creation of SDFs.

- **SDF Templates** are also used to transform data. They contain information that describes both the structure and content of SDFs. Amongst other things SDF Templates contain column mappings, CDISC metadata and row-based transformations. A row-based transformation (called a Map Function) is a transformation that can only operate on the columns in a single row of a dataset. Because SDF Templates can join many datasets together, all the columns, in the datasets which participate in the mappings, can be referenced in Map Functions. Additionally, Map Functions can use any SAS Function that can be called from the Select Clause of a PROC SQL Statement. This makes Map Functions reasonably powerful and a lot can be accomplished without having to resort to macros.

- The system also introduces the concept of **Publish and Subscribe**, which enables Standard Templates to be set up which can be used on many analyses. This gives us the power to control key attributes from a single template, yet at the same time, allow individual studies or analyses to take care of what's different. The SDF Templates are then used to generate SAS programs that are executed within the system to create the physical SDF datasets.

- Code generation techniques allow **quality control checks** to be automatically generated into each program (e.g. data truncation checking, primary and candidate key violation checking and valid values checking), which would otherwise require detailed programming standards to be followed by convention.

The **load** process is handled by the **Program Execution Manager**, which executes all the generated programs in the correct order. It dynamically determines the dependencies between programs and executes programs concurrently whenever possible to ensure maximum performance. The resultant SDF datasets are then loaded into the SDF directory structure becoming both a deliverable for the submission and another data source.

ADVANTAGES OF METADATA AND CODE GENERATION

It is important to realize that just because a program is generated it does not lessen the responsibility for ensuring that it is correct and produces the required results. Exactly that same programming skills are required to construct an SDF using the SDF System as would be required to produce the code by hand.

Given the last statement, a good question would be why bother generating the code if the metadata describes everything anyway; why not just develop it by hand. There are many reasons why code generation makes sense; some of them are listed below:

- In the SDF System the generated code and the program specification and are essentially the same thing. This means that any changes to the program specification are automatically reflected in the generated code.

- The high-level design of every SDF Program is the same. They have the same sections in the same order. The only differences between SDF Programs are the:
  - Structure of the SDF that is being produced
  - Column mappings and map functions required
  - How datasets (raw and derived) are filtered and joined

- Because code is generated it is possible to ensure that certain checks are automatically performed in every SDF program. Currently the system checks for truncation, valid values and key integrity.

- The metadata makes it possible to perform sophisticated impact analysis. The system records exactly where every raw and derived column is used and can therefore display the interrelationships graphically.

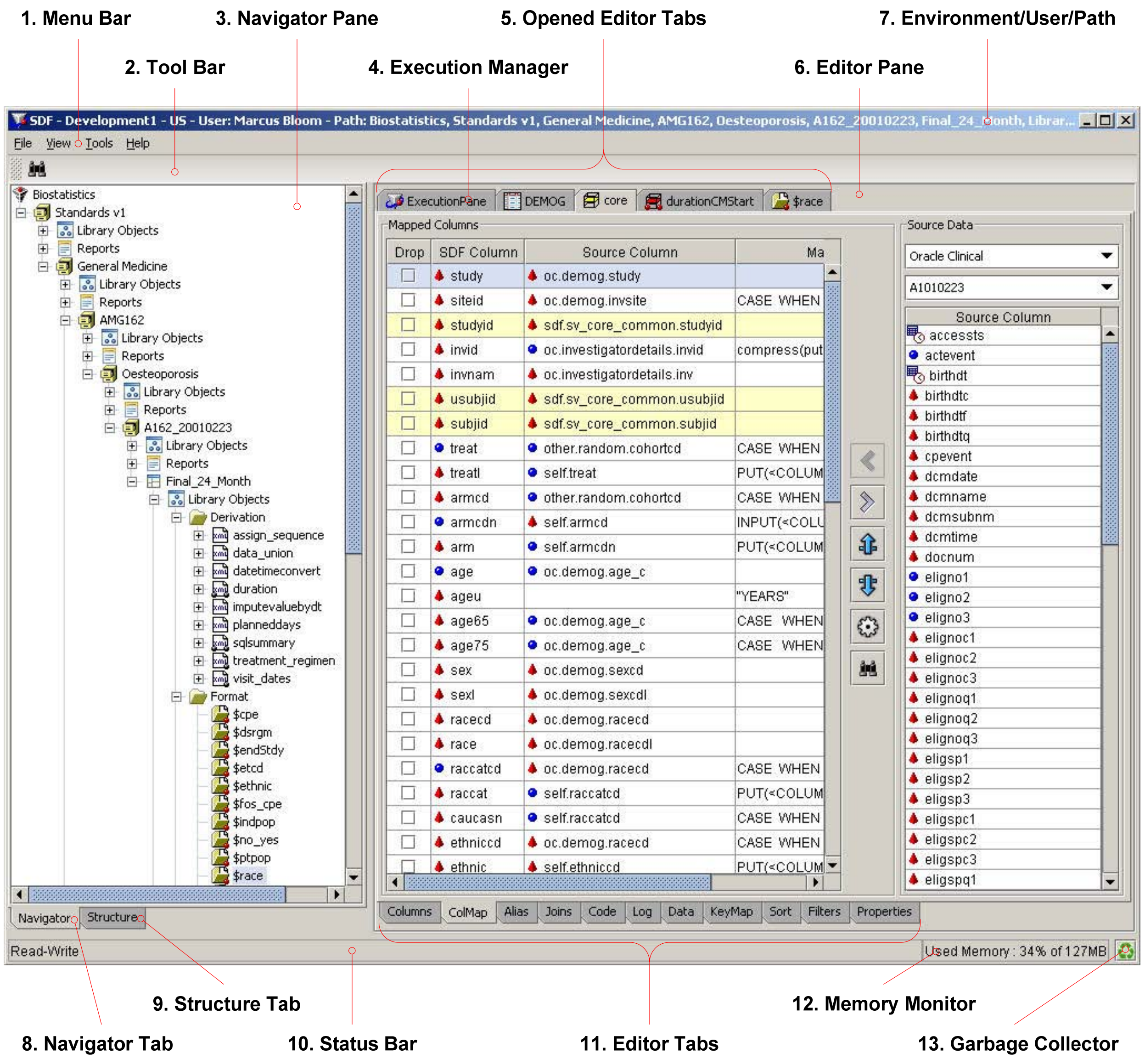
- Using metadata and code generation we are able to better define and share standard data structures. Additionally we can encode some of the implementation details in the standards themselves (e.g. default mappings and map functions) and yet still make it possible to modify any specific implementation where it differs from the norm.

There are limitations to code generation. Firstly, it is not possible to provide all the functionality of a programming language in a system. After all, that is why languages exist. They provide the ultimate interface to program whatever is required. The SDF Template provides mechanisms to implement row-based transformations and to join all the lower level components (raw and derived datasets) together. More complex derived variables are produced using the Macro Specifications and their associated Derivations Templates as described in the previous section.

THE INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)

The SDF System provides a standard IDE that supports the complete Software Development Lifecycle (SDLC). Using the IDE it is possible to specify, develop, maintain and execute SDF programs in a multi-user environment.

Figure 2: The SDF IDE



SDF - Development1 - US - User: Marcus Bloom - Path: Biostatistics, Standards v1, General Medicine, AMG162, Oesteoporosis, A162_20010223, Final_24_Month, Librar...

Biostatistics
- Standards v1
  - Library Objects
  - Reports
  - General Medicine
    - Library Objects
    - Reports
    - AMG162
      - Library Objects
      - Reports
      - Oesteoporosis
        - Library Objects
        - Reports
        - A162_20010223
          - Library Objects
          - Reports
          - Final_24_Month
            - Library Objects
              - Derivation
                - assign_sequence
                - data_union
                - datetimeconvert
                - duration
                - imputevaluebydt
                - planneddays
                - sqlsummary
                - treatment_regimen
                - visit_dates
              - Format
                - $cpe
                - $dsrgm
                - $endStdy
                - $etcd
                - $ethnic
                - $fos_cpe
                - $indpop
                - $no_yes
                - $ptpop
                - $race

Mapped Columns

Drop	SDF Column	Source Column	Ma
☐	study	oc.demog.study	
☐	siteid	oc.demog.invsite	CASE WHEN
☐	studyid	sdf.sv_core_common.studyid	
☐	invid	oc.investigatordetails.invid	compress(put
☐	invnam	oc.investigatordetails.inv	
☐	usubjid	sdf.sv_core_common.usubjid	
☐	subjid	sdf.sv_core_common.subjid	
☐	treat	other.random.cohortcd	CASE WHEN
☐	treatl	self.treat	PUT(<COLUM
☐	armcd	other.random.cohortcd	CASE WHEN
☐	armcdn	self.armcd	INPUT(<COLU
☐	arm	self.armcdn	PUT(<COLUM
☐	age	oc.demog.age_c	
☐	ageu		"YEARS"
☐	age65	oc.demog.age_c	CASE WHEN
☐	age75	oc.demog.age_c	CASE WHEN
☐	sex	oc.demog.sexcd	
☐	sexl	oc.demog.sexcdl	
☐	racecd	oc.demog.racecd	
☐	race	oc.demog.racecdl	
☐	raccatcd	oc.demog.racecd	CASE WHEN
☐	raccat	self.raccatcd	PUT(<COLUM
☐	caucasn	self.raccatcd	CASE WHEN
☐	ethniccd	oc.demog.racecd	CASE WHEN
☐	ethnic	self.ethniccd	PUT(<COLUM

Source Data

Oracle Clinical

A1010223

Source Column
accessts
actevent
birthdt
birthdtc
birthdtf
birthdtq
cpevent
dcmdate
dcmname
dcmsubnm
dcmtime
docnum
eligno1
eligno2
eligno3
elignoc1
elignoc2
elignoc3
elignoq1
elignoq2
elignoq3
eligsp1
eligsp2
eligsp3
eligspc1
eligspc2
eligspc3
eligspq1

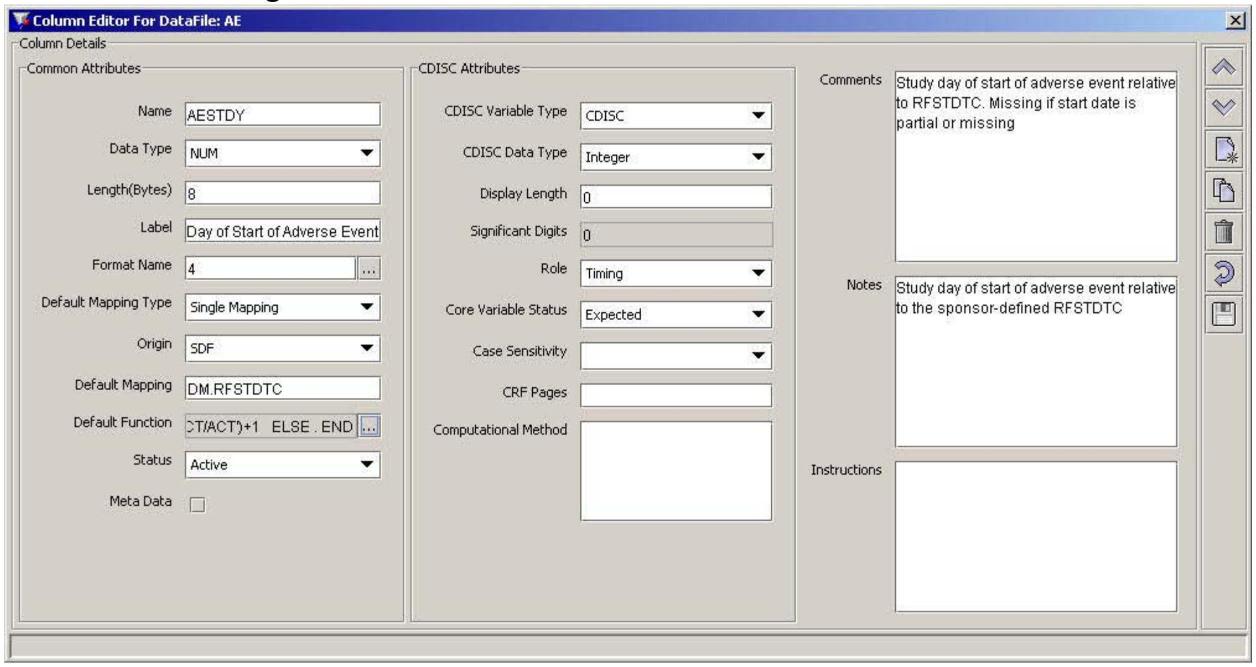
Read-Write

Used Memory : 34% of 127MB

CDISC

Although the SDF System is not limited to any specific data standard, it is proving to be an ideal vehicle with which to deploy CDISC SDTM 3.1.X and ADaM data standards across teams at Amgen. In addition, it also stores all the information necessary to automate the generation of ***define.xml***.

Figure 3: The Column Editor – Includes CDISC Metadata



Name	AESTDY
Data Type	NUM
Length(Bytes)	8
Label	Day of Start of Adverse Event
Format Name	4
Default Mapping Type	Single Mapping
Origin	SDF
Default Mapping	DM.RFSTDTC
Default Function	CT/ACT')+1 ELSE . END
Status	Active
Meta Data	☐

CDISC Variable Type	CDISC
CDISC Data Type	Integer
Display Length	0
Significant Digits	0
Role	Timing
Core Variable Status	Expected
Case Sensitivity	
CRF Pages	
Computational Method	

Comments	Study day of start of adverse event relative to RFSTDTC. Missing if start date is partial or missing
Notes	Study day of start of adverse event relative to the sponsor-defined RFSTDTC
Instructions	

THE DEVELOPMENT OF A CUTTING EDGE INFRASTUCTURE

A system as ambitious as SDF requires the parallel development of a leading edge infrastructure. This is a large undertaking but absolutely essential if the system is to have a solid foundation on which to grow in the future. The SDF project infrastructure includes of the following components:

- **Standard Environment:** Encourage standard SAS, Oracle and UNIX installations globally to aid application supportability
- **Best of Breed Development Tools:** SAS, Java, Oracle, XML
- **Automated Security:** Automated UNIX security group management to reduce reliance on other departments. Application security using LDAP authentication and role based system access.
- **Metadata Repository:** Provide Biostatistics with a centralized Oracle database to hold metadata
- **Define Standards:** Encode standards in metadata whenever possible to support process improvement initiatives. Provide tools for SDF standard directory structure creation and management.
- **Application Deployment:** Implement application deployment technologies to reduce maintenance and support issues
- **Dependencies Management:** Manage system and infrastructure external dependencies to support comprehensive impact analysis when environmental changes are proposed and implemented
- **Maintain Key Skills within the SDF Project Team:** Vision and Strategic Direction, Project Management, Technical Direction, Business Process Knowledge, SAS Programming, Database Design, OO Application Development and Support

- **Validation and Automation:** Use IT Methodologies, Source Code Control tools, Configuration Management tools, Automated Testing tools, and Bug and Enhancements Tracking tools to consistently move the project up the Software Maturity Model

CONCLUSION

The SDF System is poised to transform the way Amgen approaches and implements statistical programming. The major benefits include:

- Rapid software development time scales made possible by maximizing the reuse of data standards which include metadata to create content as well as structure
- Improved quality
- Improved validation documentation
- Improved program design – modular design ensures no hidden or hard to understand code
- Consistency of implementation across individual teams
- The ability to cascade changes across multiple implementations of a given data standard
- Rapid access to CDMS data
- Enforcement of a Standard Directory Structure
- Flexible role-based front-end and back-end security

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Marcus Bloom
Amgen
One Amgen Center Drive
Thousand Oaks, CA, 91320-1799
Work Phone: (805) 313 4477
Email: marcusb@amgen.com

David Edwards
Amgen
One Amgen Center Drive
Thousand Oaks, CA, 91320-1799
Work Phone: (805) 313 4634
Email: edwardsd@amgen.com