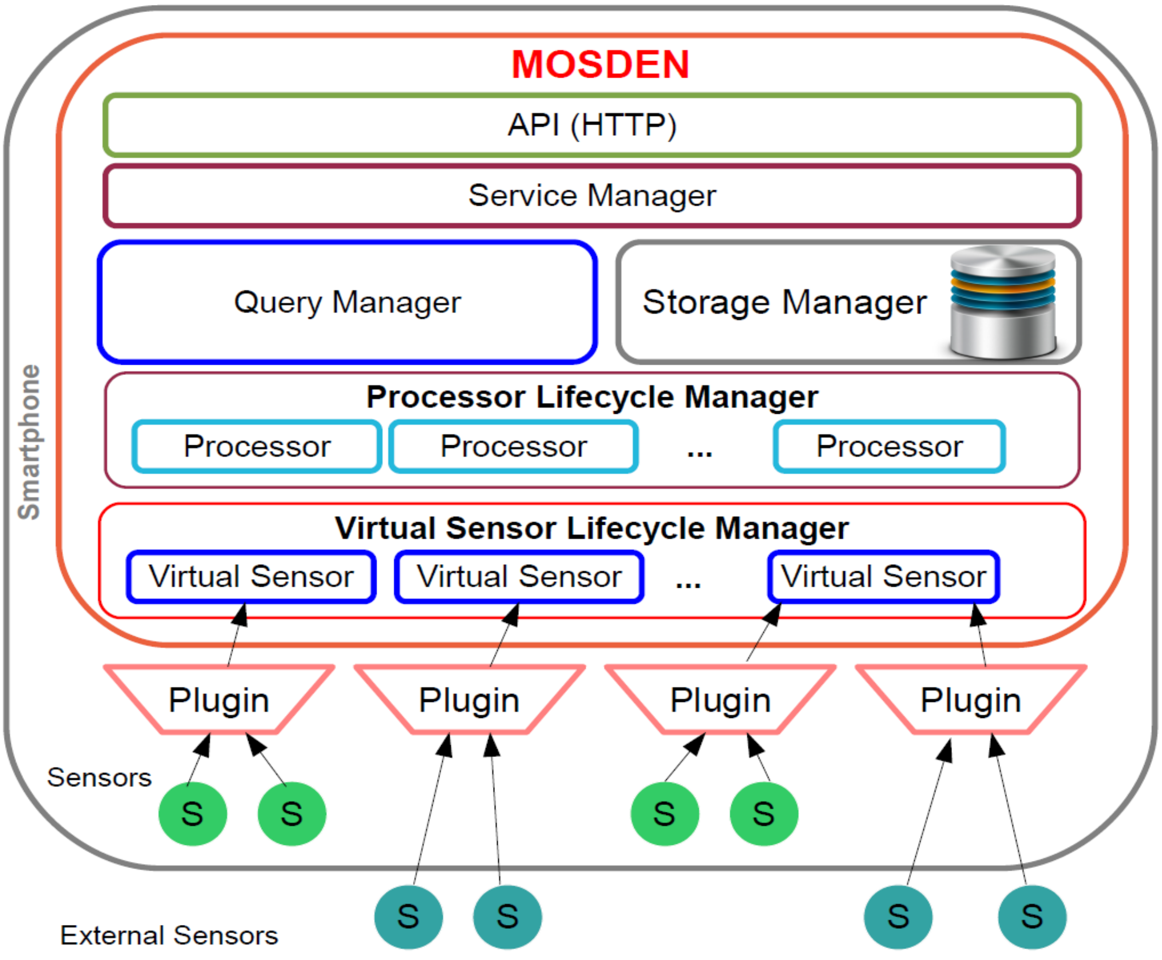
Crowd Sensing Systems

1.Mobile Sensor Data Engine (MOSDEN)

- MOSDEN, a crowd sensing platform built around the following design principles
  - Separation of data collection, processing and storage to application specific logic
  - A distributed collaborative crowd sensing application deployment with relative ease
  - Support for autonomous functioning
  - A component-based system



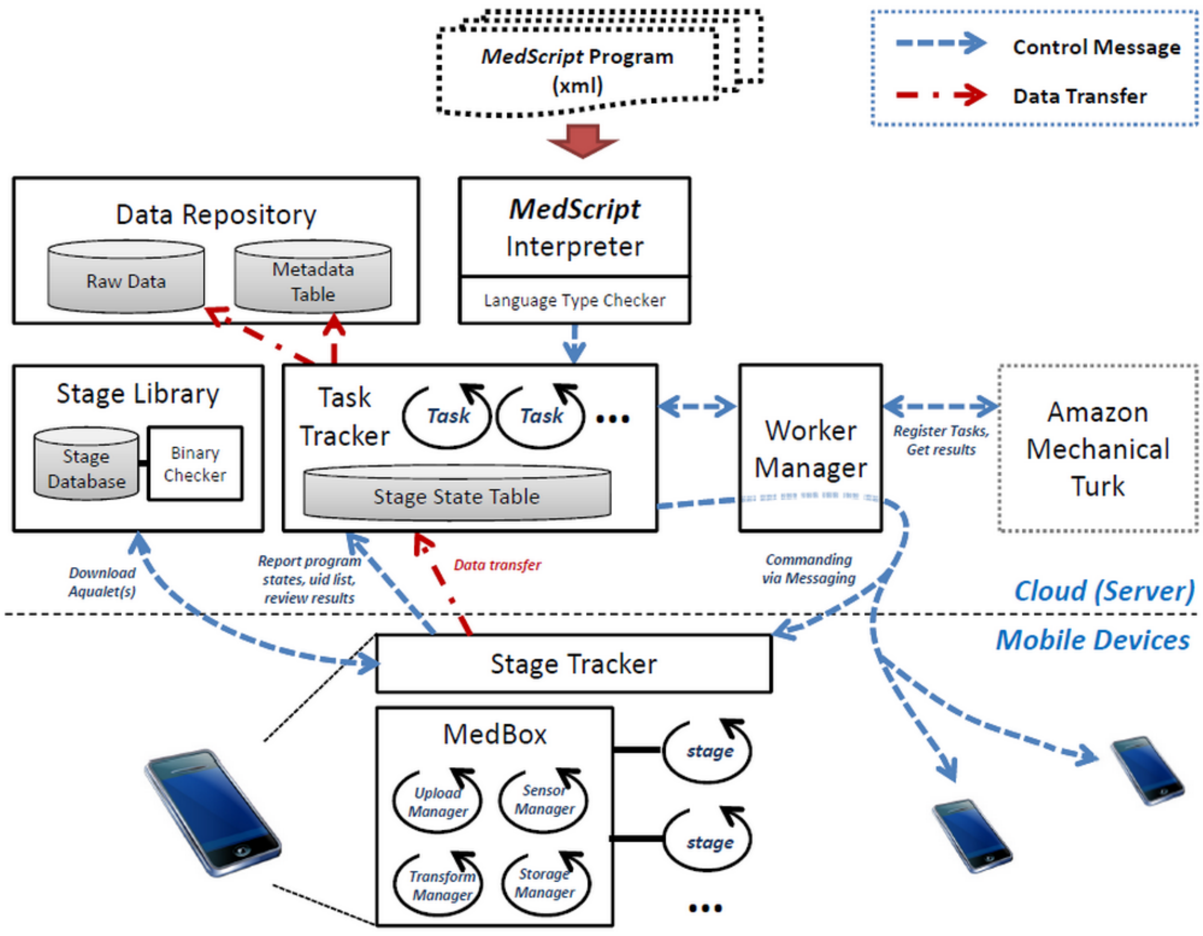
MOSDEN – Platform Architecture

2.Medusa: A Programming Framework for Crowd-Sensing Applications

In Medusa, programmers specify crowd-sensing tasks as a sequence of stages and connections.

Runtime System Architecture

1. Partitioned Service
   A. Uses a collection of services both on the cloud and on worker smartphones.
2. Dumb Smartphone
   A. Medusa should minimize the amount of task execution state that is maintained on smartphones.
3. Opt-in Data Transfer
   A. Medusa should minimize the amount of task execution state that is maintained on smartphones.



Overview of the Medusa Programming Framework

3.SocialFusion

- provides a completely new vision of context-aware world by combining "inputs from mobile social networking services" and
  "sensing data" to create a contextual picture surrounding a user or a group of users.
- provide
  - location information (by using mobile devices)
  - user's action (by using senors such as accelerometer, microphone, camera, and digital compass)
  - user's social behavior (by using online social networking services such as Facebook).
- multistage architecture and distinct classes of data are defined to integrate and extract meaningful contextual information from the raw data.
  - mobile data from smartphone
  - sensor data from sensor networks
  - social networking data from online social network like Facebook

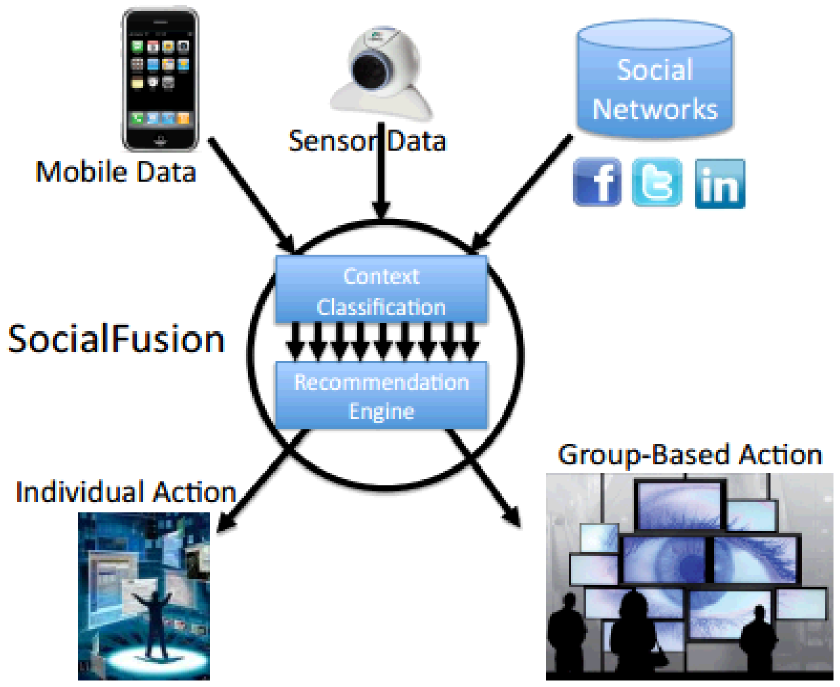


Figure 1: How SocialFusion fuses mobile, sensor, and social data to generate context awareness.

4.MECA: Mobile Edge Capture & Analysis M/W for Social Sensing Application

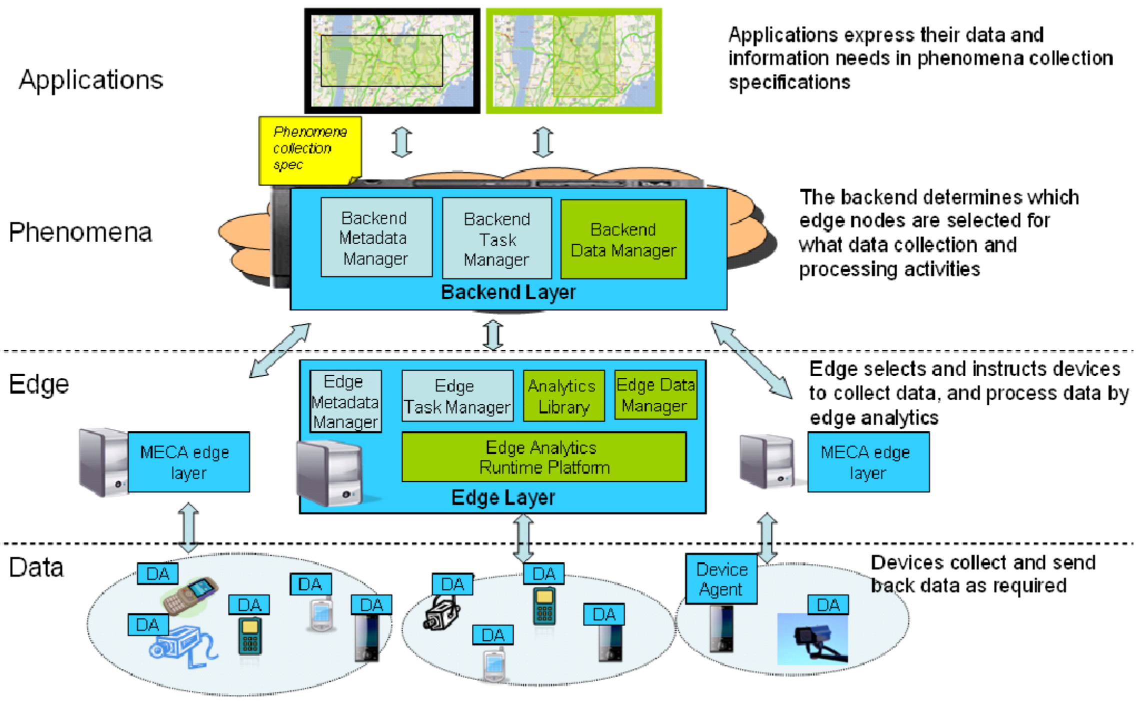
MECA conducts optional primitive processing on the raw data to extract higher level information, and passes back the "half-cooked" data to applications.

The MECA architecture consists of three different logical layers: phenomena, edge and data.

The **phenomena** layer usually resides at the backend (e.g., a data center).

The **edge** layer resides on the network edge (e.g., base stations in cellular networks). Its main function is to receive collection requirements from the phenomena layer, manage the data collection among a subset of local devices, and run edge analytics for primitive data processing.

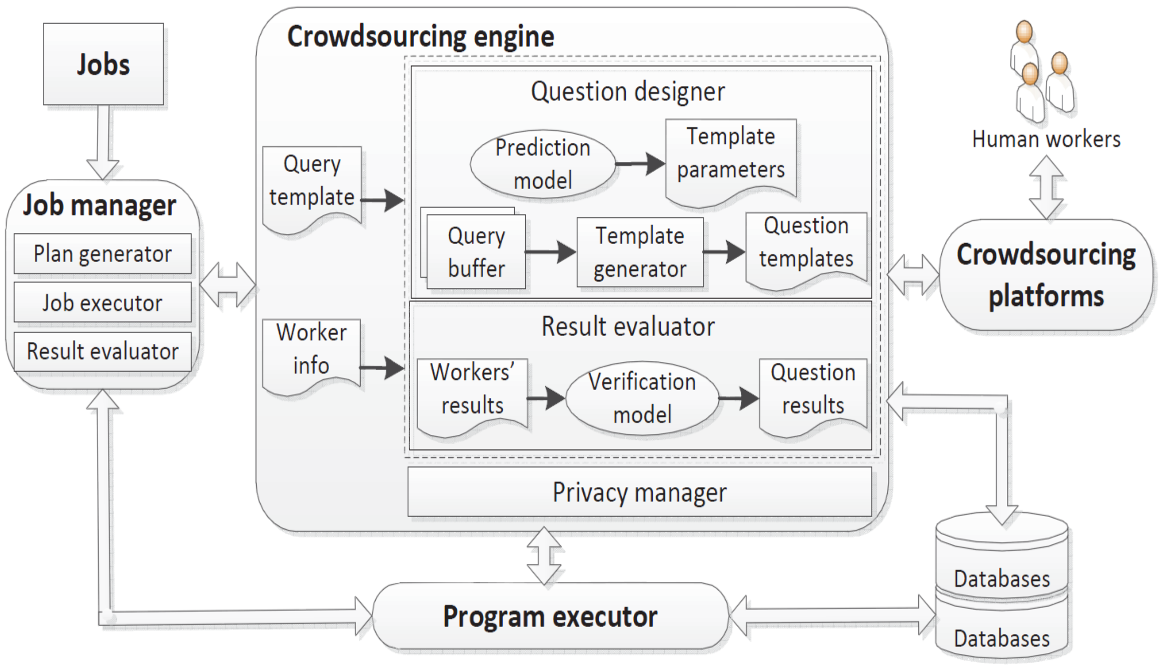
The **data** layer is on devices. It is a software agent running on devices, receiving data collecting instructions from the edge node, producing data and sending them back to the edge.


Applications
Applications express their data and information needs in phenomena collection specifications
Phenomena collection spec
Phenomena
Backend Metadata Manager
Backend Task Manager
Backend Data Manager
Backend Layer
The backend determines which edge nodes are selected for what data collection and processing activities
Edge
Edge Metadata Manager
Edge Task Manager
Analytics Library
Edge Data Manager
Edge Analytics Runtime Platform
Edge Layer
MECA edge layer
MECA edge layer
Edge selects and instructs devices to collect data, and process data by edge analytics
Data
DA
Device Agent
Devices collect and send back data as required

MECA three layer architecture

5.CDAS: A Crowdsourcing Data Analytics System

CDAS exploits the crowd intelligence to improve the performance of different data analytics jobs, such as image tagging and sentiment analysis. CDAS transforms the analytics jobs into human jobs and computer jobs, which are then processed by different modules.
The core difference between CDAS and the conventional analytics systems lies in the processing mechanism. CDAS employs human workers to assist the analytics tasks, while other systems rely solely on computer systems to answer the queries.
CDAS consists of three major components: job manager, crowdsourcing engine and program executor.
The **job manager** accepts the submitted analytics jobs and transforms them into a processing plan, which describes how the other two components (crowdsourcing engine and program executor) should collaborate for the job. In particular, the job manager partitions the job into two parts, one for the computers and one for the human workers. For example, in human-assisted image search, the human workers are responsible for providing the tags for each image, while the image classification and index construction are handled by the computer programs. In most cases, the two parts interact with each other during processing.
The **program executor** summarizes the results of crowdsourcing engine, and the
crowdsourcing engine may change its job schedule due to the requests of program executor.



CDAS Architecture